BaSys 4.0:

Metamodell der Komponenten und ihres Aufbaus

Dokument-Version	1st ed.
Datum	09.07.2018
Verbreitungsgrad	public
Projekt	BaSys 4.0
Förderkennzeichen	01 IS 16 022
Laufzeit	1.7.2016 – 30.6.2019

GEFÖRDERT VOM

Bundesministerium
für Bildung
und Forschung

Autoren

Prof. Dr.-Ing. Ulrich Epple (UE), RWTH Aachen University

Julian Grothoff (JG), RWTH Aachen University

Constantin Wagner (CW), RWTH Aachen University

Reviewer

Dr.-Ing. Sten Grüner (SG), ABB AG Forschungszentrum

Dipl.-Ing. Heiko Haase (HH), Festo AG & Co. KG

Diese Arbeit entstand im Rahmen des BMBF geförderten Projekts BaSys 4.0 (Förderkennzeichen 01IS16022). Die Autoren bedanken sich des Weiteren für die Unterstützung des gesamten BaSys-Teams.

DOI: 10.18154/RWTH-2018-225880

Inhaltsverzeichnis

1 Einleitung

Im folgenden Dokument wird ein Meta-Modell für Komponenten zur Steuerung von produzierenden Anlagen in Fertigungs- und Prozessindustrie mit Bezug auf die Geräteebene (PC2) synthetisiert. Von dem Kernmodell einer Komponente ausgehend werden zunächst Eigenschaften technischer Komponenten beschrieben und der Stand der Technik wird umrissen. Im Anschluss wird die BaSys40-Komponete von der allgemeinen technischen Komponente abgeleitet. Wesentlich für Komponenten ist die Definition ihrer Schnittstellen insbesondere für den operativen Betrieb. Weiter werden standardisierbare oder standardisierte Zustandsautomaten für ein einheitliches Verständnis des Komponentenzustands und darauf aufbauende Interaktionsmuster definiert.

Wandlungsfähigkeit in der Produktion erfordert Wandlungsfähigkeit der Steuerungskomponenten im BaSys40-System, weshalb hier der Fokus auf Veränderungsmöglichkeiten, sowohl auf Komponentenebene, als auch in der internen Struktur der Komponenten gelegt wird. Dabei werden verschiedene Mechanismen zur Bildung von BaSys40-Softwarekomponenten beschrieben und Designpattern der internen Komponentenstruktur für operative BaSys40-Komponenten erläutert, welche die Wandelbarkeit ermöglichen. Zusätzlich werden Änderungen durch die Integration von Self-X-Fähigkeiten beschrieben.

Zur einheitlichen Beschreibung wird ein Metamodell von BaSys40-Komponenten eingeführt und es werden grundlegende Anforderungen an die Beziehung zum Komponententyp formuliert. Von diesem Modell können spezialisierte Anwendungen abgeleitet werden. Da in diesem Deliverable Steuerungskomponenten im Fokus stehen, werden Prozessführungskomponenten als generisches Vorbild für operative BaSys40-Komponenten herangezogen. Ein gemeinsames Verständnis des Aufbaus und der Schnittstellen dieser Komponenten ist für die Interoperabilität zwischen Automatisierungssystemen und den Austausch von Automatisierungslösungen unerlässlich. Daher werden diese detailliert als Anwendung in Einzelfunktionseinheiten und Prozessführungskomponenten aufgeteilt, sowie Vorlagen für ihre innere Struktur entworfen. Zuletzt sind exemplarisch weitere Anwendungen für BaSys40-Komponenten beschrieben.

2 Eigenschaften technischer Komponenten

Im folgenden Kapitel wird der Komponentenbegriff diskutiert. Im Umfeld von BaSys spielt insbesondere die technische Komponente eine zentrale Rolle.

Im **Duden (Dudenredaktion)** wird die Komponente allgemein definiert als:

> Komponente (Duden)
> *Def.:"Bestandteil eines Ganzen"*

Diese Definition ist sehr allgemein und lässt eine breite Anwendung zu. Sie enthält jedoch wesentliche Aspekte einer Komponente. Eine Komponente wird nicht nur in sich selbst, sondern auch durch die Rolle die sie als Teil eines Ganzen spielt oder spielen soll definiert. Je nachdem was dieses „Ganze“ ist, kann der Komponentenbegriff in verschiedensten Situationen zur Anwendung kommen. In Bild 2.1 ist der Komponentenbegriff in verschiedene Anwendungsbereiche gegliedert.

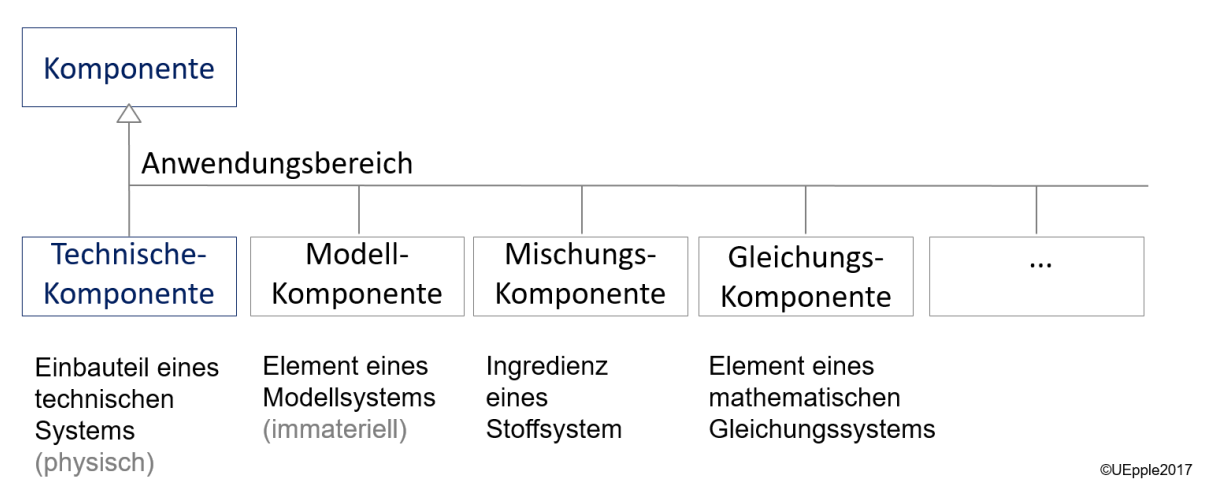

Abbildung 2.1: Klassifizierung von Komponenten nach Anwendungsbereichen.

Wie in Abbildung 2.1 dargestellt, verwendet man den Komponentenbegriff z. B. im Zusammenhang mit Begriffssystemen, mathematischen Gleichungen, stofflichen Zusammensetzungen, Modellen, Softwaresystemen, technischen Geräten und Anlagen.

Im Sprachgebrauch verwendet man den Komponentenbegriff sowohl für das sich bereits als Bestandteil im System befindliche Ding als auch für das zur Verwendung in einem System zur Ausführung einer definierten Rolle vorgesehene Ding. So bezeichnet man z.B. eine Pumpe, die sich noch im Lager befindet, als Komponente, oder ganz allgemein, Antriebe, Getriebe, Scheinwerfer, Autositze usw., die ein Komponentenlieferant als Komponenten zum Einbau in einen PKW anliefert. Entsprechend bezeichnet man z.B. auch eine Stoffmenge die in eine Mischung eingebracht werden soll, als eine Komponente (Zweikomponentenkleber). Entscheidend für den Komponentencharakter ist die intendierte Verwendung als Bestandteil mit definierter Funktionalität in einem bestimmten System.

In den einzelnen Anwendungsbereichen werden mit dem Begriff Komponente zusätzliche Eigenschaften verbunden. Diese zusätzlichen Eigenschaften schwingen bei der Begriffsverwendung implizit mit, sind jedoch nicht explizit festgelegt. Diese Unterschiede kann man auch in spezielleren Definitionen des Komponentenbegriffs zum Ausdruck bringen. So kann man z.B. eine Mischungskomponente definieren als „Substanzielle Ingredienz eines Stoffgemischs“. Die Komponente ist dabei ein in sich unstrukturiertes Mengengut. Sie verliert im Prozess ihre Identität.

2.1 Der Begriff „technische Komponente“

Für BaSys ist insbesondere die technische Komponente von Interesse. In der Literatur findet man verschiedene Definitionen für eine technische Komponente.

So definiert DIN EN 81346-1 (Norm DIN EN 81346-1) eine Komponente vertiefend und mit industriellem Bezug als „Produkt, welches als Bestandteil in einem zusammengesetzten Produkt, System oder in einer Anlage verwendet wird“ (Abschnitt 3.7 und detailliert in 4.6). Weiter wird hier ausgeführt, dass Komponenten „an die tatsächlichen Anforderungen […] angepasst wurde[n]“ und „üblicherweise Produkte von Prozessen in andern technischen Systemen“ sind. Dies unterstreicht die generische Gestaltung von Komponenten und deren Produktcharakter. Besondere Anforderungen stellt dabei das Zusammensetzen in einem Produkt durch verschiedene Beteiligte, eben Produktverantwortliche und Systemverantwortliche, dar. Dafür sind klar definierte Schnittstellen der Komponenten unabdingbar.

Eine Definition von Softwarekomponenten ist bei U. Enste zu finden (Enste 2001, S. 127).Darin wird die Komponente als „Vollständig spezifiziertes Softwaremodul (Struktur und Verhalten), bei dem die Funktionalität abgeschlossen beschrieben wird und nur über vorgegebene Schnittstellen manipuliert werden kann“, definiert. Dabei wird eben der Softwareaspekt von Komponenten beleuchtet. Um die Handhabbarkeit als Entität, die Stabilität und die Erwartungskonformität zu gewährleisten, wird hier die Kapselungsfähigkeit und Abgeschlossenheit besonders betont.

In ISO/IEC 10746-2 (Norm ISO/IEC 10746-2, 9.26) wird eine Komponente wie folgt definiert: "An object that encapsulates its own template, so that the template can be interrogated by interaction with the component. The template and other instantiation parameters are expressed in a form that allows them to be updated during the lifetime of any system of which the component is to form a part, allowing alternative realizations of the component to be substituted." In der Norm steht vor allem der Zusammenhang zwischen System und Komponente im Betrachtungsfeld. Dabei stellen der Wechsel einer Komponente im Lebenszyklus des Systems, möglicherweise auch zur Laufzeit in Softwaresystemen, sowie der Austausch durch alternative Realisierungen besondere Anforderungen an die Flexibilität. Interessant, ist in dieser Definition das die Komponente ihre eigene Vorlage enthält, sodass Ansätze einer Selbstbeschreibungs-Funktionalität zu sehen sind. Weiter soll die Interaktion mit der Komponente selbst erfolgen, sodass die Komponente den Charakter einer Schnittstelle erhält.

In der IT-Welt lässt sich eine weitere Definition der Komponente im UML Standard (Spezifikation formal/2015-03-01) finden, wie in Abbildung 2.2 dargestellt ist. (vgl. Abschnitt 2.4.2)

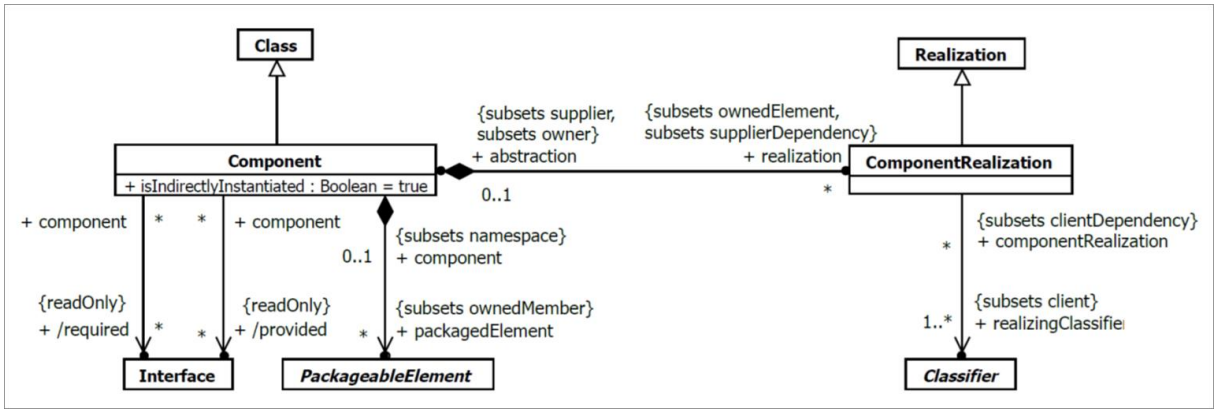

Abbildung 2.2: UML Komponentenmodell aus dem UML Standard (V2.5) (Spezifikation formal/2015-03-01, Figure 11.38).

Hier lässt sich bereits die Idee des Rollenmodells durch „Component“ und „ComponentRealization“, sowie ein Typ/Instanz Konzept durch die Beziehung der „ComponentRealization“ zum „Classifier“ wiederfinden. Dabei haben Komponenten explizite Schnittstellen die durch die Unterscheidung „provided“ und „required“ das Erbringen und das Nutzen von Diensten unterscheiden. An diesen Schnittstellen muss ebenfalls ein Abgleich von Anforderungen und Zusicherungen erfolgen. Weiter wird in UML auch eine Blackbox und Whitebox Modellierung unterschieden. Diese Modellierung findet sich in BaSys durch die verschiedenen Arten der Erzeugung von Komponenten wieder, worauf besonders in Abschnitt 6 eingegangen wird. Das Konzept, Komponenten als Komponentensystem aufzufassen findet sich in der Eigenschaft des „PackageableElement“ wieder. Die Disjunktheit wird hier durch die Kardinalität beim „PackageableElement“ explizit zum Ausdruck gebracht. Eine Darstellung eines Komponentensystems in UML findet sich in Abbildung 2.3. (Vgl. Abschnitt 2.3 zum Begriff des Komponentensystems)

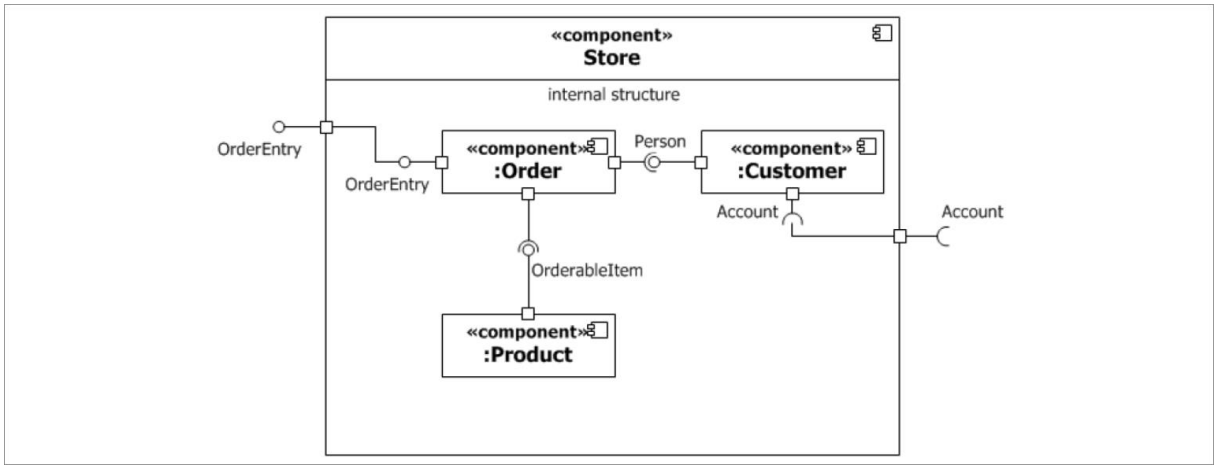

Abbildung 2.3: Darstellung eines Komponentensystems aus dem UML Standard (V2.5)) (Spezifikation formal/2015-03-01, Figure 11.47)

Für BaSys eignet sich insbesondere die Definition aus der DIN SPEC 40912 (Spezifikation DIN SPEC 40912). Diese definiert eine technische Komponente wie folgt.

Technische Komponente (DIN SPEC 40912)

Def: Vorgefertigte, in sich strukturierte und unabhängig hantierbare Einheit, die zur Realisierung einer konkreten Rolle in einem technischen System vorgesehen ist.

Hinter dieser Definition steht implizit eine Reihe von zusätzlichen Eigenschaften, die einer technischen Komponente zugeschrieben werden. Diese werden im nächsten Kapitel diskutiert.

Wenn im Folgenden von einer Komponente gesprochen wird, dann ist immer eine technische Komponente im Sinne der DIN SPEC 40912 gemeint.

2.2 Spezielle Eigenschaften technischer Komponenten

Technische Komponenten umfassen alle Arten von Einbauteilen in einem technischen System. So z.B.: Rohrleitungen, Ventile, Kompressoren, Blechpressen aber auch Programme und implementierte Softwarestrukturen. Technischen Komponenten und damit auch Softwarekomponenten schreibt man spezielle Eigenschaften zu. Für Komponenten findet man in der Literatur (Bronsard et al. 1997, S. 20) beispielsweise Kontextunabhängigkeit, Transparenz des Speicherorts, heterogene Verbindungen und Entwickelbarkeit. Weiter findet man insbesondere für Softwarekomponenten in der Literatur (Sametinger 1997, S. 68) Geschlossenheit, Identifizierbarkeit und Klarheit der Schnittstelle als wesentliche Eigenschaften. Auch in der DIN SPEC zu Kernmodellen finden sich verschiedene Regeln (Spezifikation DIN SPEC 40912, 3.4.2.7) für Komponenten bzw. Komponentenmodelle. In verschiedenen Kontexten sind also diverse Definitionen für Komponenten vorhanden, welche oft einen gleichen Kern enthalten. Daher ist es wichtig die relevanten Eigenschaften zu formulieren, die für den hier verwendeten Komponentenbegriff gelten.

Im Folgenden sind Eigenschaften aufgeführt, die insbesondere alle technischen Komponenten charakterisieren. In Kapitel 6 wird erläutert wie diese Eigenschaften unterstützend als Wandlungsbefähiger wirken können.

- Identifizierbarkeit und Persistenz

  Komponenten sind in allen Phasen ihres Lebenszyklus eindeutig identifizierbar und von anderen Objekten unterscheidbar. Dies gilt auch im eingebauten Zustand. Eine Komponente wird beim Einbau zwar Teil des Ganzen, behält jedoch ihre individuelle Identifizierbarkeit.

- Abgegrenztheit

  Komponenten sind abgegrenzt. Die Abgegrenztheit besagt, dass zu jedem Zeitpunkt eindeutig bestimmt ist, welche Objekte zu einer Komponente gehören und welche nicht.

- Erkennbarkeit externer Abhängigkeiten

  Eine Komponente ist zwar abgegrenzt, besitzt jedoch Schnittstellen über die sie mit ihrer Systemumgebung und anderen Komponenten interagiert. Je nach Art des Komponentensystems kann es sich um ganz unterschiedliche Arten von Schnittstellen (mechanische, elektrische, digitale…) handeln. Für eine Komponente müssen jedoch alle

Schnittstellen explizit bekannt und beschrieben sein. Damit werden die Abhängigkeiten erkennbar und die Auswirkungen bei der Hantierung einer Komponente im Systemverbund berechenbar.

- Disjunktheit

  Komponenten sind disjunkt. Die Disjunktheit besagt, dass ein Element, welches zu einer Komponente gehört, nicht gleichzeitig Element einer anderen Komponente sein kann. Alle Elemente die zu einer Komponente gehören sind Teil von ihr und gehören ihr exklusiv.

- Kompaktheit

  Komponenten sind kompakt. Die Kompaktheit besagt, dass die internen Elemente einer Komponente direkt und nicht über externe Elemente untereinander verbunden sind. Die gesamte Struktur der Komponente ergibt sich aus ihren Elementen und den zwischen diesen Elementen intern bestehenden Beziehungen. Externe Elemente oder externe Beziehung sind am Aufbau der Komponente nicht beteiligt. Die Struktur, Semantik und Funktionalität einer Komponente ist in sich umfassend und vollständig.

- Hantierbarkeit als Einheit

  Auf der Grundlage der dargestellten Eigenschaften ergibt sich insgesamt die Möglichkeit eine Komponente immer als Einheit zu betrachten und auch zu handhaben. Eine Komponente kann mit elementaren Operationen als Einheit

  - verschoben,
  - entfernt,
  - hinzuzugefügt,
  - eingebaut,
  - ausgebaut,
  - in Betrieb genommen und
  - außer Betrieb genommen

  werden. Die Möglichkeit diese Transformationen auf eine Komponente als Gesamteinheit durch eine einfache Operation ausführen zu können, ist die zentrale charakteristische Eigenschaft einer Komponente.

2.3 Komponentensystem und Komponenten-Systemplattform

Im Allgemeinen werden unter dem Begriff „Komponentensystem“ zwei unterschiedliche Begriffsausprägungen zusammengefasst.

In der ersten Ausprägung beschreibt das Komponentensystem, wie ein Ganzes aus seinen Komponenten aufgebaut ist und wie sich die Gesamtfunktionalität des Ganzen aus dem Zusammenwirken seiner installierten Komponenten ergibt.

In der zweiten Ausprägung beschreibt das Komponentensystem die Systemumgebung, in der die Komponenten gehandhabt, also entfernt, hinzugefügt, verschoben, eingebaut, ausgebaut, in Betrieb genommen und außer Betrieb genommen werden. Für jedes Komponentensystem

muss es eine solche Handhabungsumgebung zwingend geben. Sie stellt die Dienste für das Komponentenhandling zur Verfügung.

Um Verwechslungen zu vermeiden wird das System, das aus Komponenten gebildet wird, weiterhin als *Komponentensystem* bezeichnet. Die Systemumgebung, welche die Komponenten verwaltet und ihnen eine Plattform zur Ausführung zur Verfügung stellt, bezeichnen wir als *Komponenten-Systemplattform*.

2.4 Technische Komponenten mit Softwareanteilen

Im Rahmen der BaSys40-Architektur ist es hilfreich, die technischen Komponenten zu spezialisieren in Hardwarekomponenten, Softwarekomponenten, Smarte-Komponenten und Verbundkomponenten (**Bild 3**).

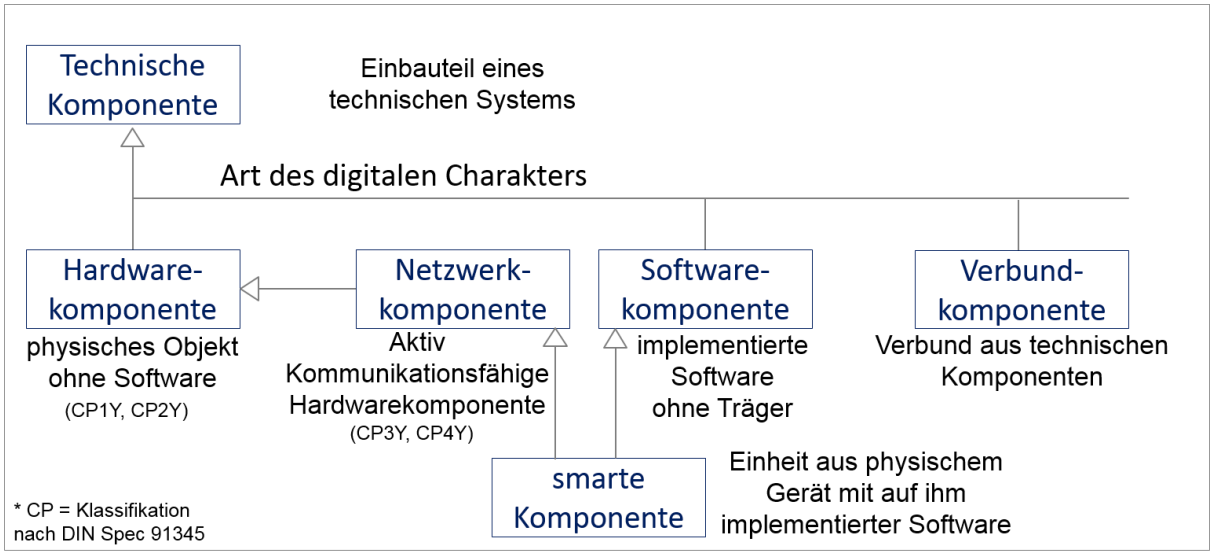

Abbildung 2.4: Spezialisierung der technischen Komponenten nach Art ihres digitalen Charakters.

Hardwarekomponenten bestehen nur aus Hardware, Softwarekomponenten nur aus Software. Hardwarekomponenten, welche mindestens aktiv Kommunikationsfähig im Sinne der CP-Klassifikation nach DIN Spec 91345 sind, werden als Netzwerkkomponenten bezeichnet. Smarte-Komponenten bestehen aus Hard- und Softwareanteilen. Dabei muss der Hardwareanteil mindestens die Fähigkeit zur aktiven Teilnahme an der Netzwerkkommunikation besitzen. Verbundkomponenten sind ein Verbund aus mehreren Einzelkomponenten, die als Verbund wiederum eine Komponente bilden. Grundlage dieser Spezialisierung ist die Bedeutung der Software für die Gestaltung des Komponentensystems. Hardwarekomponenten interessieren im Rahmen von BaSys40 nur am Rande als Hardwarebasis von Datenverarbeitungs-Systemen (DV-Systeme, siehe Abschnitt 2.4.1). Für Softwarekomponenten und Smarte-Komponenten spielt die Bedeutung von Software eine zentrale Rolle. Der nächste Abschnitt enthält eine Erläuterung des diesen Komponenten zugrundeliegenden Softwarebegriffs.

2.4.1 Der Softwarebegriff

Der Begriff Software ist weit verbreitet und enthält viele Interpretationsmöglichkeiten. In der Informatik gibt es eine Reihe von unterschiedlichen Begriffsdefinitionen. Für die hier vorliegende Fragestellung verwenden wir den Softwarebegriff in zwei Ausprägungen: Der üblichen breiten (und ziemlich unscharfen) Bedeutung als Teil der Informationswelt und der für die Komponenten wichtigen engen speziellen Bedeutung als physischem digitalem Abbild. In der ersten Bedeutung tritt der Begriff in den vielfältigsten Bereichen der Entwicklung und Planung auf. Man entwickelt Software, programmiert Software, verkauft Softwareprodukte usw. In diesem Kontext ist Software ein Teil der Informationswelt. Software enthält die Beschreibung von Algorithmen und Informationsmodellen.

Im Kontext der Laufzeitsysteme ist Software physisch zu verstehen. Sie wird definiert als:

> Software (physisch)
>
> *Def: In einem digitalen Datenverarbeitungssystem implementiertes Abbild von Informationen (Daten) und Algorithmen.*

Software (physisch) ist also ein spezieller Teil der physischen Welt. Es sind physisch repräsentierte Bits auf einem physischen Träger. Software (physisch) bedarf eines DV-Systems als Plattform. Das DV-System gehört als solches zur Hardware. Es stellt die erforderliche digitale Speicher-, Rechen- und Kommunikationsfähigkeit zur Verfügung.

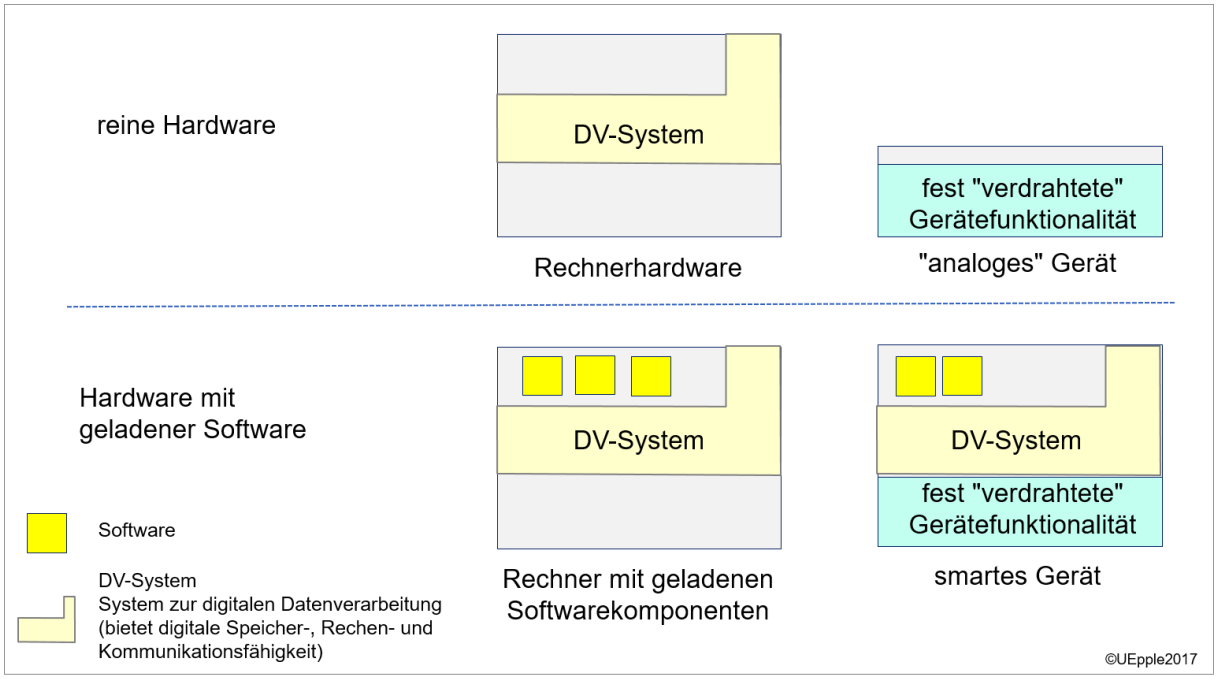

Abbildung 2.5: Erläuterung der Begriffe im Umfeld der Softwarekomponenten und der smarten Komponenten

Dabei gibt es wie in Abbildung 2.5 dargestellt verschiedene Ausprägungen: Rechner, deren Hauptzweck es ist, ein DV-System zur Verfügung zu stellen und smarte Geräte, deren Hauptzweck es eigentlich ist, eine bestimmte technologische Gerätefunktion zu realisieren. Diese

nutzen dazu jedoch softwaretechnisch realisierte Funktionen und müssen dafür intern wieder ein DV-System als Basis zur Verfügung stellen. Das Besondere von software-technisch realisierter Funktionalität gegenüber „fest verdrahteter“ Funktionalität ist die Möglichkeit, Funktionen und Daten elektronisch zu konfigurieren und zu verarbeiten ohne die Hardware zu ändern. Bei der Betrachtung der Verarbeitungsfunktionalität tritt der physische Aspekt der Realisierung in den Hintergrund.

2.4.2 Softwarekomponente

Eine Softwarekomponente wird im Bereich der komponentenbasierten Entwicklung z.B. folgendermaßen definiert:

Softwarekomponente (CBSE)
„Eine Softwarekomponente ist ein Software-Element, das konform zu einem Komponentenmodell ist und gemäß einem Composition-Standard ohne Änderungen mit anderen Komponenten verknüpft und ausgeführt werden kann.“ (Heineman und Councill 2001)
Softwarekomponente (UML)
“A Component represents a modular part of a system that encapsulates its contents and whose manifestation is replaceable within its environment. A Component is a self-contained unit that encapsulates the state and behavior of a number of Classifiers. A Component specifies a formal contract of the services that it provides to its clients and those that it requires from other Components or services in the system in terms of its provided and required Interfaces. A Component is a substitutable unit that can be replaced at design time or run-time by a Component that offers equivalent functionality based on compatibility of its Interfaces. As long as the environment is fully compatible with the provided and required Interfaces of a Component, it will be able to interact with this environment.“ (Spezifikation formal/2015-03-01)

Ein wesentliches Charakteristikum einer Softwarekomponente sind also ihre klar und explizit definierten (Informations-) Schnittstellen und die Möglichkeit sie unter Berücksichtigung dieser definierten Abhängigkeiten unabhängig von anderen Softwareeinheiten handhaben und austauschen zu können. Sie erfüllt alle oben beschriebenen Eigenschaften einer technischen Komponente.

Der Softwarekomponente liegt die physische Interpretation von Software zugrunde. Dies erkennt man z.B. an folgenden Eigenschaften einer Softwarekomponente:

- Man kann eine Softwarekomponente individuell laden, löschen und gegebenenfalls ausführen.
- Eine Softwarekomponente ist immer auf genau einem physischen Träger lokalisiert. Es gibt sie nur einmal. (Bemerkung: der Träger kann auch virtualisiert sein, das ändert nichts am Prinzip)
- Das Erstellen einer Kopie entspricht der Implementierung einer neuen, zweiten Softwarekomponente.

- Die Anwendungsfunktionalität einer Softwarekomponente ist typischerweise unabhängig vom unterlagerten DV-Systems. Die Performance ihrer Speicherung und Ausführung ist jedoch abhängig vom unterlagerten DV-System.

Softwarekomponenten sind zentrale Betrachtungsgegenstände der BaSys40-Architektur.

2.4.3 Smarte-Komponenten

In smarten Komponenten bilden die Hardware und die Software eine Einheit zur Lösung einer bestimmten technischen Aufgabenstellung. Dies kann eine einfache Aufgabe wie z.B. die eines Sensors oder Aktors sein oder auch eine komplexe Aufgabe wie die einer Presse, eines Kompressors, eines Garagentors oder eines Roboters. Wesentliche Abgrenzung zu komplexen Systeminstallationen ist auch hier die Möglichkeit, die Smarte-Komponente insgesamt als Einheit zu hantieren. Ein zentrales Element einer Smarten-Komponente ist ihre Kommunikationsfähigkeit. Eine Smarte-Komponente muss eine digitale Kommunikationsschnittstelle besitzen, über die sie in den Informationsverbund des Gesamtsystems eingebunden werden kann. Im Gegensatz zu einer reinen Softwarekomponente, die auf die Fähigkeiten des unterlagerten Laufzeitsystems aufbauen kann und im Prinzip „nur“ eine Protokollschnittstelle anbieten muss, lebt eine Smarte-Komponente physisch vollständig autark und muss den gesamten Kommunikations-Stack vom Physical Layer aufwärts als Schnittstelle realisieren. Bei ihr spielen also auch elektrische und mechanische Schnittstellen eine signifikante Rolle.

2.4.4 Verbundkomponente

Eine Verbundkomponente ist eine Komponente, die selbst aus mehreren Komponenten besteht. Eine Verbundkomponente kann aus Hardwarekomponenten, Softwarekomponenten, smarten Komponenten, Verbundkomponenten oder irgendeiner Mischung aus diesen bestehen. So gibt es z.B. reine Hardware-Verbundkomponenten, reine Software-Verbundkomponenten und z.B. Smarte Verbundkomponenten die aus Hardwarekomponenten und Softwarekomponenten bestehen.

Um den Komponentenanforderungen zu genügen müssen auch Verbundkomponenten die Forderungen nach Identifizierbarkeit, Kompaktheit und einheitlicher Hantierbarkeit erfüllen. Dies ist bei dezentral realisierten, gemischten Hard- bzw. Softwarekomponenten eine Herausforderung an die Komponenten-Systemplattform.

3 BaSys40-Komponenten

BaSys40-Komponenten sind Smarte bzw. reine Softwarekomponenten, die bestimmte Anforderungen erfüllen. BaSys40-Komponenten sind durch folgende grundlegende Eigenschaften charakterisiert:

3.1 Teilnehmer im BaSys40-Dienstsystem

Eine BaSys40-Komponente muss sich im Kommunikationsnetz als BaSys40-Dienstsystemteilnehmer darstellen und alle mit dieser Rolle verbundenen Standards erfüllen.

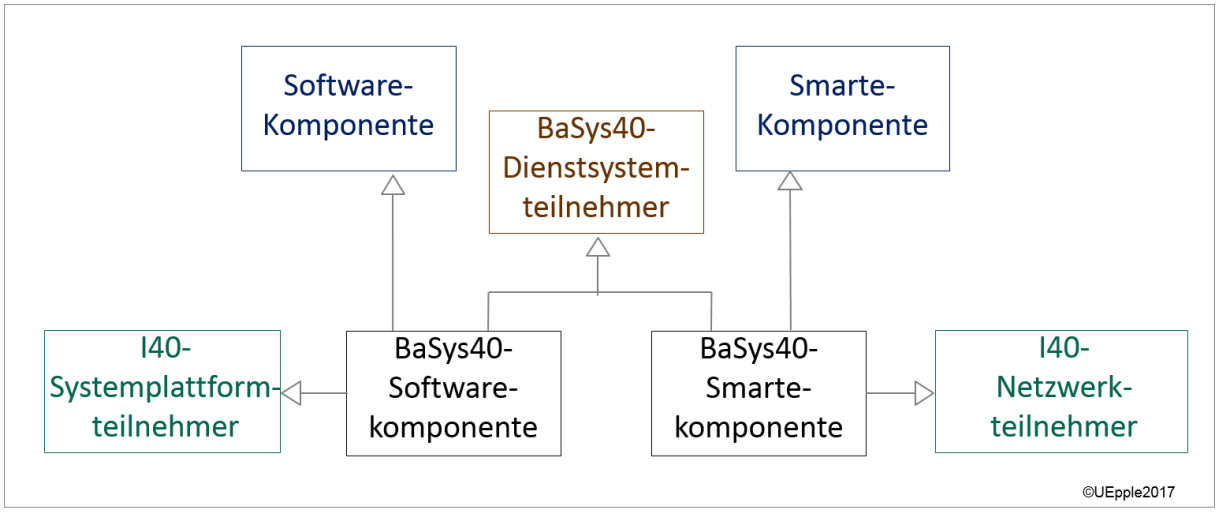

Abbildung 3.1: Ableitung der BaSys40-Komponente aus Softwarekomponenten und smarten Komponenten

Eine BaSys40-Komponente erbt also wie in Abbildung 3.1 dargestellt alle Eigenschaften einer technischen Komponente und alle Eigenschaften eines BaSys40-Dienstsystemteilnehmers. Softwarekomponenten, Smarte-Komponenten und Verbundkomponenten können solche Schnittstellen besitzen und damit zu BaSys40-Komponenten werden. Reine Hardwarekomponenten können keine BaSys40-Komponenten darstellen. Sie können jedoch als Teil einer Verbundkomponente Teil einer BaSys40-Komponente sein.

Für die Smarte-Komponente müssen weitere Festlegungen gemacht werden. In diesem Fall muss die Interoperabilität zwischen der smarten Komponente und dem Systemverbund durch eine Standardisierung des gesamten Kommunikations-Stacks sichergestellt werden. Hier kann auf Technologiestandards aus der Plattform I40 zurückgegriffen werden. Daher erbt die BaSys40-Smarte-Komponente zusätzlich von “I40-Netzwerkteilnehmer“.

3.2 Komponenten der BaSys40-Komponenten-Systemplattform

Jede BaSys40-Komponente lebt in einer Laufzeit- und Verwaltungsumgebung die wir als Komponenten-Systemplattform bezeichnen (vgl. Abschnitt 2.3). BaSys40 definiert standardisierte Software-Systemplattformen für Softwarekomponenten, Netzwerk-Systemplattformen für Smarte-Komponenten und Verbund-Systemplattformen für Verbundkomponenten.

Es kann jeweils sowohl konzeptionell, als auch technologisch verschiedene Standardlösung geben. Diese sind Teil der BaSys40-Systemspezifikation. Soweit verfügbar können die Standards der Plattform I40 übernommen werden.

3.3 Kontextneutralität

Die Kontextneutralität verlangt:

BaSys40-Festlegung:
Die Funktionalität jeder BaSys40-Komponente muss kontextneutral formuliert sein.

Festlegung 3.1

Damit ist gemeint, dass in der Funktionalität der Komponente selbst keinerlei konkrete Relationen zu externen Instanzen existieren dürfen. Die Einbindung in den jeweiligen Kontext erfolgt über das Anlegen von Signalverbindungen und im Falle der Dienstschnittstelle durch die Zuweisung der jeweiligen Dienstpartner, entweder über Signal-Eingangsparameter oder im Rahmen eines Auftrags. Dieses Prinzip gilt grundsätzlich für alle BaSys40-Komponenten, also auch für Komponenten, die für genau eine konkrete Anwendung erstellt wurden und nur in diesem Kontext verwendet werden. Als Konsequenz muss zum Beispiel die Auftragsausgabe auf Rollen parametrierbar sein, wie in Abschnitt 4.4 beschrieben wird.

Weiter gilt die Festlegung unabhängig davon ob es sich um eine typisierte Komponente handelt oder ob die Komponente als Makro aus Instanzen gebildet wurde. Man vergleiche dazu den Aufbau und die Erzeugung von Komponenten wie in Kapitel 6 und 7 beschrieben.

3.4 Typisierung

BaSys40-Softwarekomponenten können sowohl typisiert sein als auch als Makro aus Instanzen gebildet werden. Diese Thematik wird in Kapitel 6 erläutert.

3.5 Ablaufverhalten

Alle BaSys40-Komponenten besitzen unabhängig von ihrer technologischen Aufgabe gemeinsame Verhaltensmuster, die sich durch standardisierte Dienstaufrufe steuern und verwalten lassen. Diese Verhaltensmuster und entsprechende Dienstprotokolle beruhen auf gemeinsamen Zustandsautomaten. Eine Auswahl relevanter Zustandsautomaten findet sich in Kapitel 5. Welche Zustandsautomaten für die hier fokussierte Anwendung der Prozessführung obligatorisch sind, wird in Abschnitt 9 beschrieben.

Insbesondere BaSys40-Komponenten zur Prozessführung besitzen ein gemeinsames Grundschema ihrer Ablaufsteuerung, welches in Abschnitt 9.2 beschrieben wird. Diese interne Strukturierung der BaSys40-Prozessführungskomponenten ermöglicht eine einheitliche Prozessführung und ist ein wichtiger Wandlungsbefähiger, wie in Abschnitt 7 erläutert wird.

4 Schnittstellen einer BaSys40-Komponente

BaSys40-Komponenten dienen z.B. der Prozessführung, der Anlagensteuerung, der Asset-Verwaltung und der Verwaltung und Steuerung der Systeminfrastruktur. Sie besitzen digitale Dienst- und Signalschnittstellen und als Smarte-Komponenten Netzwerkschnittstellen.

Smarte-Komponenten besitzen auch elektrische, hydraulische und mechanische Schnittstellen. Im Kontext von BaSys40 sind jedoch insbesondere die digitalen Schnittstellen und die Netzwerkschnittstellen von Interesse.

In Abbildung 4.1 ist das allgemeine Referenzmodell einer BaSys40-Softwarekomponente mit seinen Schnittstellen aus operativer Sicht dargestellt. Detailliertere Darstellungen und ein UML Modell dazu ist für die Außensicht in Abschnitt 8.2 und für die Innensicht von Prozessführungskomponenten in Abschnitt 9.2.1 zu finden. Wesentlich für die Definition von Komponenten sind deren Schnittstellen, da diese beispielsweise bei der Wiederverwendung oder Ersetzung durch eine andere Realisierung gleichbleiben. Dahingegen kann der innere Aufbau variieren solange die Schnittstellenanforderungen erfüllt werden.

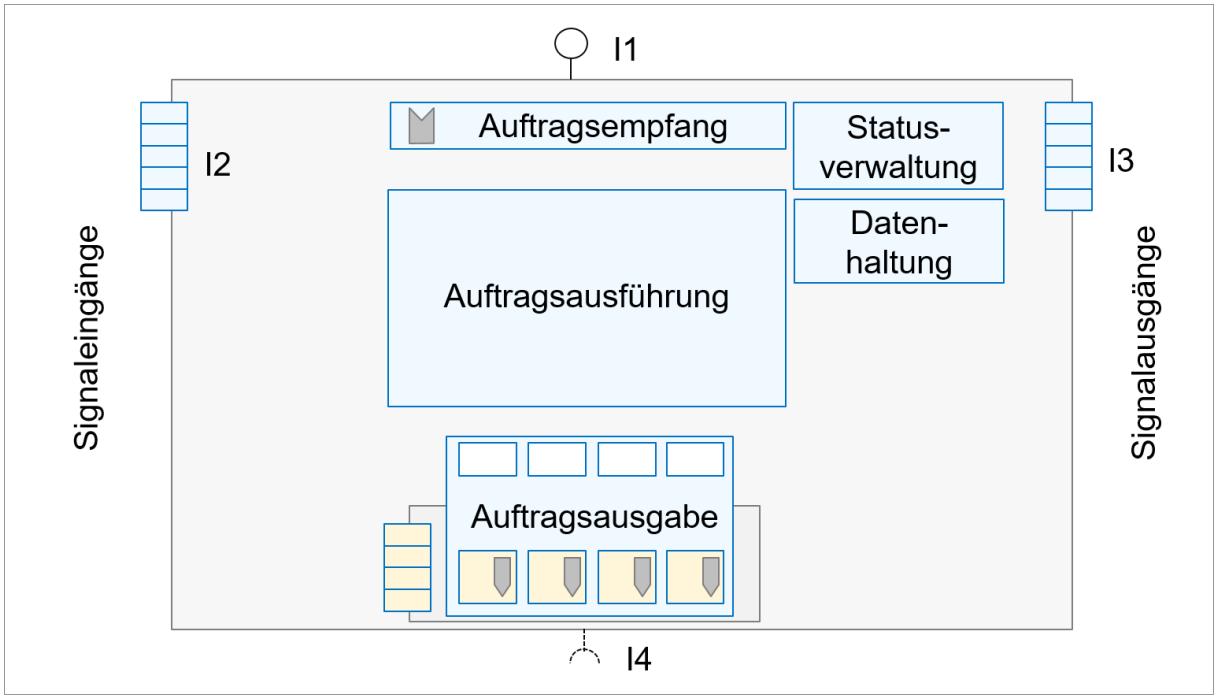

Abbildung 4.1: Allgemeines Referenzmodell einer BaSys40-Softwarekomponente mit ihren Schnittstellen.

Jede BaSys40-Komponente hat die Eigenschaften einer BaSys40-Softwarekomponente.

Sie besitzt folgende Schnittstellen (Interfaces):

I1 Komponenten-Dienstschnittstelle

I2 Signaleingänge

I3 Signalausgänge

I4 Auftragsausgabe-Schnittstelle

4.1 Komponenten-Dienstschnittstelle:

Jede BaSys40-Komponente besitzt obligatorisch eine Komponenten-Dienstschnittstelle (siehe Bild 4.1 Schnittstelle (I1)). Über sie bietet die Komponente das gesamte Spektrum ihrer Fähigkeiten an. Dazu gehören z.B. Fähigkeiten zur Ausführung operativer Aufgaben, Funktionen zur Selbstverwaltung und Konfiguration, Funktionen zur Selbstdiagnose und Instandhaltungsunterstützung, Funktionen zur Erkundung, etc.. Die Komponenten-Dienstschnittstelle entspricht den allgemeinen Anforderungen an eine Dienstschnittstelle des BaSys40-Systems. Dazu gehört die Selbstbeschreibung. Jede Komponenten-Dienstschnittstelle bietet einen Erkundungsservice an, über den alle ihre angebotenen Fähigkeiten erkundet werden können.

4.1.1 Design der BaSys40-Komponenten-Dienstschnittstelle

In BaSys40 wird die Komponenten-Dienstschnittstelle als ein Kommunikationsendpunkt mit einem Zugangspunkt angesehen, der in verschiedenen Technologien realisiert werden kann. Als Beispiel kann ein Kommunikationsendpunkt eine URL für einen OPC-UA oder Webserver gesehen werden: opc.tcp://192.168.0.1:4840 oder http://hostname:7509. Ein Zugangspunkt könnte beispielhaft dann eine OPC-UA node id oder der Pfad Anteil einer URL sein; letztendlich ist der Zugangspunkt das adressierte Objekt hinter dem Kommunikationsendpunkt.

BaSys40-Festlegung:

Jede BaSys40-Komponente stellt an ihrer Dienstschnittstelle **Dienste** zur Verfügung. Die verschiedenen **Fähigkeiten** einer Komponente werden durch aufrufbare **Operationen** dieser Dienste abgebildet.

Festlegung 4.1

Damit stellt die Dienstschnittstelle verschiedene aufrufbare Operationen zur Verfügung, über die die einzelnen Fähigkeiten aufgerufen werden können.

BaSys40-Festlegung:

Der Aufruf einer Operation wird in BaSys40 als **ORDER** bezeichnet.

Festlegung 4.2

4.1.2 Vergabe von ORDER-Namen

Die einheitliche Adressierung aller Funktionalitäten und Fähigkeiten einer Komponente über die Operationen bedingt, dass sämtliche ORDER einer Komponente zu einem gemeinsamen Namensraum gehören und voneinander verschieden sein müssen.

Aus Gründen der Echtzeitanforderungen müssen die ORDER-Namen in ihrer Länge prinzipiell begrenzt sein. In modernen leistungsfähigen Systemen kann diese Grenze großzügig gewählt werden.

BaSys40-Festlegung:

Die maximale Länge eines ORDER Namens beträgt **64 Zeichen**.

Festlegung 4.3

Zu beachten ist allerdings, dass die ORDER Namen in der Beschreibung von Anwendungen und in den Bedienoberflächen explizit erscheinen. Sowohl für den Ersteller eines Rezeptablaufs, als auch für den Bediener (Operator) erscheinen zu lange Namen unhandlich und fehleranfällig. Dazu ergeben sich Schwierigkeiten in der Darstellung auf Bedienoberflächen. Daher wird empfohlen:

BaSys40-Empfehlung:

Die maximale Länge eines ORDER-Namen beträgt **8 Zeichen.**

Empfehlung 4.1

Dazu kann optional zu jeder Operation eine Description bereitgestellt werden, die das Wesen der Operation näher erläutert.

BaSys40-Systeme sollen international anwendbar sein. Dies bedeutet u.a., dass Komponententypen einheitlich weltweit eingesetzt werden können. Um dies zu gewährleisten wird gefordert:

BaSys40-Empfehlung:

ORDER-Namen sollen möglichst einfachen englischen **Kurznamen** entsprechen, nur **alphanumerische** Zeichen (A-Z, a-z, 0-9) beinhalten und mit einem Buchstaben beginnen.

Empfehlung 4.2

Da ORDER-Namen jedoch nicht nur systemintern auftreten, sondern auch an der Produktions- und Bedienoberfläche erscheinen, werden die englischen Namenskürzel in vielen Fällen zu Akzeptanzschwierigkeiten führen. Um dem entgegenzuwirken, gilt auch für ORDER-Namen das allgemeine BaSys-Aliasprinzip:

BaSys40-Empfehlung:

Für ORDER-Namen können zusätzlich zu den Standard-Namen **lokale Alias-Namen** vergeben werden. In diesem Fall werden für alle HMI-Aktivitäten die Alias-Namen verwendet. Die Komponenten akzeptieren sowohl die Alias-Namen als auch die Standard-Namen als Eingabe.

Empfehlung 4.3

Um die Konfliktfreiheit sicherzustellen, darf ein Alias-Name zwar dem entsprechenden Standard-Namen entsprechen, muss sich jedoch von allen anderen Alias- und Standard-Namen unterscheiden.

4.1.3 Auftragsempfang

Zu jeder Komponenten-Dienstschnittstelle gehört obligatorisch eine Auftragsempfangs-Funktion. Aufgabe des Auftragsempfangs ist die Organisation der Zugriffsberechtigung, der Belegung und die Entgegennahme und Prüfung von Aufträgen. Angenommene Aufträge werden der Auftragsausführung zur Ausführung übergeben.

Zugriffsberechtigung

Das BaSys40-Konzept geht von einem mehrstufigen Konzept zur Sicherstellung der Zugriffssicherheit aus. In den Zielkomponenten werden dabei keine, oder nur sehr einfache, Zugriffssicherungen realisiert. Das Konzept ist durch folgende Festlegungen geprägt:

BaSys40-Festlegung:

Der Auftragsempfang vertraut, dass die zu einer ORDER angegebene **Absender-Identität** und **Absender-Rolle** korrekt sind.

Festlegung 4.4

Dahinter steht die Vorstellung, dass diese Korrektheit für Absender aus dem engen lokalen Umfeld (Trusted Environment) der Komponente vorausgesetzt werden kann. Die Identität von Absendern die nicht aus diesem Umfeld kommen, müssen bei ihrem Eintritt in den Bereich Trusted Environment gesichert werden. Dafür sind spezielle Infrastrukturkomponenten zuständig. Die empfangende Anwendungskomponente kann immer von einer gesicherten Identität des Absenders ausgehen und vertraut den Angaben des Absenders.

Das Konzept ist in Abbildung 4.2 dargestellt.

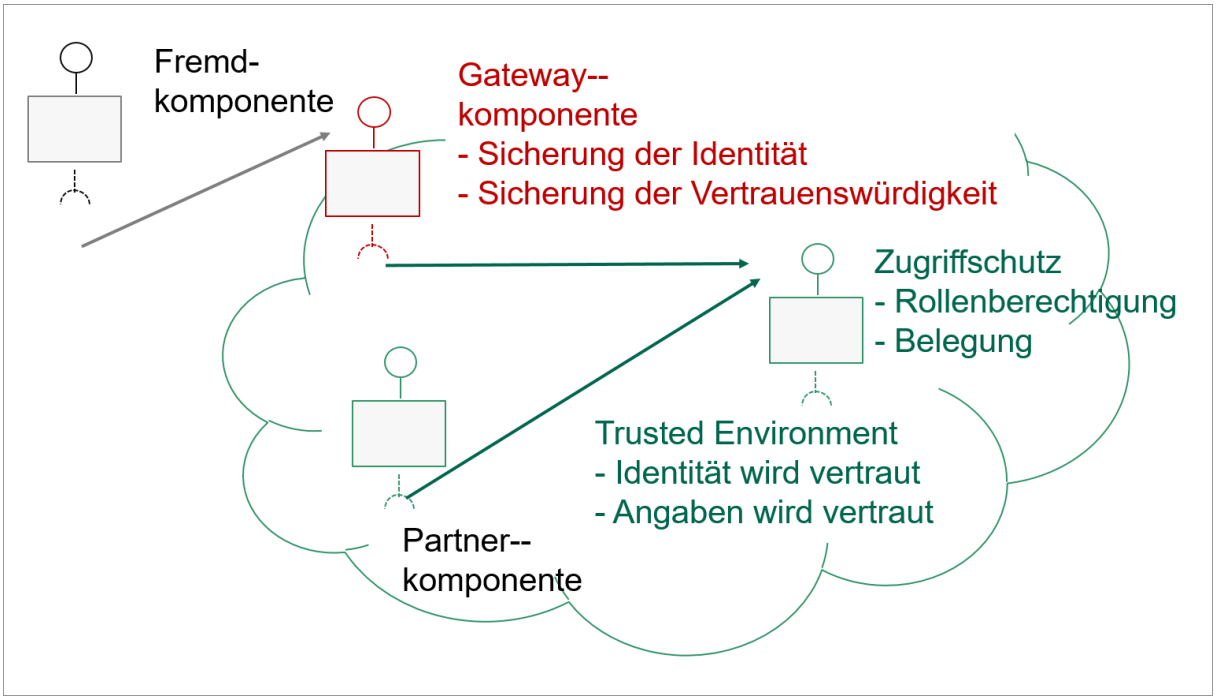

Abbildung 4.2: Gestuftes Konzept der Zugriffsicherung

Um Fehler im Zugriff auszuschließen und unberechtigten Nutzern den Zugriff zu verweigern, besitzen die Anwendungskomponenten für jede Operation eine Liste der Rollen, die auf diese Operation zugreifen dürfen. Nur Nutzer, die sich unter dieser Rolle anmelden bekommen Zugriff auf diese Operationen.

BaSys40-Festlegung:

Der **Auftragsempfang** prüft, ob ein Nutzer mit der von ihm angegebenen Rolle die von ihm gewünschte Operation ausführen darf.

Festlegung 4.5

Nutzer mit nicht berechtigten Rollen werden abgewiesen.

4.1.3.1 Allokation von Komponenten und deren Ressourcen

Es gibt Komponenten, die eine Vielzahl von Aufträgen unterschiedlicher Auftraggeber gleichzeitig ausführen können. Zur Organisation der Ausführung dienen z.B. Warteschlangen für die Einzelaufträge. In der Automatisierungstechnik sind die Aufträge jedoch oft an physikalische Prozesse gekoppelt und ein Nutzer muss sicherstellen, dass ihm die dafür erforderlichen Ressourcen (Reaktor, Werkzeugmaschine, …) während des gesamten Prozessablaufs uneingeschränkt zur Verfügung stehen. In diesem Fall müssen die Komponenten einen Belegungsmechanismus unterstützen. Sämtliche BaSys40-Komponenten der Prozessführung (Einzelsteuereinheiten, Gruppensteuereinheiten, Prozessführungskomponenten, Maßnahmen, Rezeptsteuereinheiten, Produktionssteuereinheiten usw.) besitzen einen Belegungsmechanis-

mus für ihre operativen Funktionen. Allgemein kann jede BaSys40-Komponente einen Belegungsmechanismus besitzen. Dazu gehört ein Satz von Operationen, die nur von dem belegenden Nutzer ausgeführt werden können. Daneben kann es weitere Operationen (z.B. zur Erkundung, ...) geben, die auch von nicht belegenden Nutzern ausgeführt werden können.

BaSys40-Festlegung:

Jede BaSys40-Komponente zur Prozessführung besitzt einen **Belegungsmechanismus**. Dazu gehört ein Satz von Operationen die nur im BELEGT-Zustand durch den belegenden Nutzer ausgeführt werden können. Der Belegungsautomat stellt sicher, dass die Komponente zu jedem Zeitpunkt durch höchstens einen Benutzer belegt ist.

Festlegung 4.6

Auch im Bereich der Informatik gibt es verschiedene Mechanismen, um geteilte Ressourcen zu allozieren. So kann beispielsweise über Locking-, Semaphore- oder Mutex-Konzepte sichergestellt werden, dass ein Prozess eine Ressource unter seiner Kontrolle hat. Dadurch können beispielsweise Race Conditions verhindert werden. Die BaSys40-Softwarekomponenten können hier als Nutzer und Anbieter von Ressourcen betrachtet werden. Dabei entstehen durch die Belegung unterlagerter Einheiten Führungshierarchien. Um den aus der Informatik bekannten Ressourcenkonflikten bei BaSys40-Softwarekomponenten entgegen zu wirken, wird hier also ein einfacher Belegungsmechanismus definiert.

Der BaSys40 Zustandsautomat für die Belegung von Komponenten wird detailliert in Abschnitt 5.2 definiert.

4.2 Signaleingangsschnittstellen

Der Besitz von Signaleingangsschnittstellen ist optional. Z.B. besitzen Einzelsteuereinheiten und bestimmte Prozessführungseinheiten fest verdrahtete Signaleingangsschnittstellen zur Verriegelung oder zum Erfassen von Steuerzuständen für die Regelung.

4.2.1 Namenskonventionen

Signaleingangsschnittstellen unterliegen folgender Standard-Namenskonvention.

BaSys40-Empfehlung:

Die maximale Länge eines Signaleingangs-Namen beträgt **8 Zeichen.**

Der Name soll möglichst einem einfachen englischen **Kurznamen** entsprechen, nur **alphanumerische** Zeichen (A-Z, a-z, 0-9) beinhalten und mit einem Buchstaben beginnen.

Empfehlung 4.4

Für Signalnamen sind keine Alias-Namen vorgesehen.

4.2.2 Zuweisungsregeln

Die signalorientierte Datenübertragung ist eine Grundlage für die akzeptierten und verbreiteten Automatisierungssprachen, wie IEC 61131 und IEC 61499. Für Signaleingänge gelten folgende Regeln:

- Signaleingänge können verschaltet oder nicht verschaltet sein.
- Es ist je Eingang immer nur eine Verschaltung auf ein anderes Signal (Ausgang) möglich.
- Nicht verschaltete Eingänge können von extern oder durch die Komponente selbst parametriert werden.
- Ein Eingang führt immer einen aktuellen Wert.
- Der aktuelle Wert eines Eingangs kann durch einen Lesedienst (READ) abgefragt werden.

So wird zum Beispiel die spätere Rückverfolgbarkeit und Rückwirkungsfreiheit sichergestellt. Im Sinne der IEC 61499 gelten diese Regeln auch für Eventsignale. Wie der Eingang über eine Signalverbindung seinen Wert erhält, ist Angelegenheit der Realisierung. In jedem Fall gilt jedoch:

- Während der Abarbeitung der Komponentenfunktion bleiben die Eingangswerte unveränderlich.

4.2.3 Standardeingänge

Für Technologien, in denen keine Aufrufsemantik für Operationen zur Verfügung steht oder deren Implementierung ungünstig ist, kann statt der aufrufbaren Operationen ein Kommandosignaleingang genutzt werden. Der Name des Kommando Signaleingangs ist CMD. Der Datentyp des Kommando Signaleingangs ist String. Der Auftrag selbst und eventuell zum Auftrag gehörende Parameter werden als String-Parameter in den Kommandoeingang geschrieben und von der Komponente eingelesen. Die Syntax der Kommandos lautet dabei wie folgt:

"[SENDERID];[ORDER];[PARAMETER1=VALUE1,PARAMETER2=VALUE2,…]"

Neben dem Kommandoeingang ist es sinnvoll einen zweiten Signaleingang oder Signalausgang zu definieren, welcher die möglichen ORDERs auflistet. Dieser Signal-Eingang oder Ausgang hat den Namen ORDERLIST und ist ein String array.

Weitere Ausführungen zu sprechenden Kommandos als Grundlage für Prozessführungsschnittstellen finden sich in einem Kongressbeitrag von Wagner (Wagner und Epple 2015).

Weitere BaSys Standardeingänge für einzelne Anwendungsklassen können in Kapitel 9 definiert werden.

4.3 Signalausgangsschnittstellen

Der Besitz von Signalausgangsschnittstellen ist optional. Sie dienen z.B. der Bereitstellung von Ergebnissen der Komponentenfunktionalität (Stellwerte, Kaskadenwerte, …), der standar-

disierten Darstellung der Komponentenzustände (Belegungszustand, Betriebsart, ...) und anderer für die Signalverknüpfung oder die Modellerkundung interessanter Zustände. Die Namen der Standardausgänge für die in Kapitel 5 definierten Zustandsautomaten finden sich in Abschnitt 0.

Signalausgänge ermöglichen nicht nur die Verknüpfung dieser Ausgänge mit anderen Eingängen, sondern stellen auch eine explizite Modellschnittstelle dar, auf diemit allgemeinen READ-Diensten zugegriffen werden kann, ohne die Komponentendienstschnittstelle zu tangieren. Dies ist gerade bei der Erfassung der Zustände benachbarter oder unterlagerter Komponenten in der Prozessführung von erheblichem Vorteil. So können z.B. die Zustandswerte die zur Berechnung der Transitionsbedingungen benötigt werden auf einfache und allgemeine Weise und – falls die Zustandsautomaten normiert sind – in standardisierter Form abgefragt werden.

4.3.1 Namenskonventionen

Signalausgangsschnittstellen unterliegen folgenden Standard-Namenskonvention.

BaSys40-Empfehlung:

Die maximale Länge eines Signalausgangs-Namen beträgt **8 Zeichen.**

Der Name soll möglichst einem einfachen englischen **Kurznamen** entsprechen, nur **alphanumerische** Zeichen (A-Z, a-z, 0-9) beinhalten und mit einem Buchstaben beginnen.

Empfehlung 4.5

Für Signalnamen sind keine Alias-Namen vorgesehen. Für Signalausgänge gelten folgende Regeln:

- Signalausgänge können verschaltet oder nicht verschaltet sein.
- Es sind je Ausgang beliebig viele Verschaltungen möglich.
- Ein Ausgang führt immer einen aktuellen Wert.
- Der aktuelle Wert eines Ausgangs kann durch einen Lesedienst (READ) abgefragt werden.

Wie die Komponente ihre Ausgangswerte beschreibt ist Angelegenheit der Realisierung. In jedem Fall gilt jedoch:

- Der Ausgangswert kann nur durch die Komponente selbst geschrieben werden. Ein externer Lesevorgang ändert den Wert des Ausgangs nicht.
- Während der Bearbeitung einer Komponente können sich die Ausgänge jederzeit ändern.

4.4 Auftragsausgabe

BaSys40-Komponenten besitzen optional Auftragsausgabeschnittstellen. Diese können als Klienten betrachtet werden, mittels derer die Komponente Dienstanfragen versenden (siehe

Bild 4.1 Schnittstelle (I4)) und Antworten empfangen können. Dabei kann es sich um die verschiedensten Arten von Diensten handeln. Sie können z.B.

- über eine READ-Abfrage sich benötigte Werte verschaffen,
- Kommandos (ORDER) an unterlagerte Prozesssteuereinheiten absetzen oder
- sich für den Bezug bestimmter Nachrichten bei einem Broker anmelden.

Nicht jede BaSys40-Komponente besitzt eine Auftragsausgabe. Z.B. besitzen konventionelle Einzelsteuereinheiten keine Auftragsausgabe. Sie übergeben Ihre Stellwerte an ihre Signalausgangsschnittstellen von denen sie dann über die I/O-Ebene verdrahtet an die Aktoren weitergeleitet werden. (Vgl. dazu Abschnitt 9.1)

4.4.1 Design der BaSys40-Auftragsausgabe-Schnittstelle

In BaSys40 ist vorgesehen, dass in einer Komponente für jeden Partner, den sie ansprechen möchte, eine eigene Auftragsausgabeeinheit existiert. Die Auftragsausgabeeinheit ist in der Kommunikation der eigentliche Klient. Sie versendet den Request, und empfängt und verarbeitet die Antwort. Dies kann über verschiedene Kommunikationsmechanismen, wie beispielsweise Publisher-Subscriber, geschehen. Es gibt Aufträge mit und ohne Antwort. Gerade in der operativen Prozessführung verwendet man in vielen Fällen Aufträge ohne Antwort, wie in Abschnitt 5.1.3 beschrieben wird. Rückgemeldete Probleme werden als separate Nachrichten direkt an das Störungssystem gesandt und nicht an die Applikation. Die Applikation vergewissert sich über die korrekte Ausführung des Auftrags z.B. über eine spätere Abfrage des angestrebten Steuerungs- oder Anlagenzustands. Diese Rückfrage erfolgt erst, wenn die Komponente diese Information z.B. für eine Transition benötigt. Die Abfrage selbst, z.B. durch einen READ-Request, bekommt eine Antwort, die die READ-Auftragsausgabeeinheit an ihrem internen Ausgang der Komponentenhauptfunktion zur Verfügung stellt.

Zurzeit sind die Auftragsausgabeeinheiten noch implementierungsspezifisch realisiert. Im Verlauf des Projekts sollen allgemeine Standards zur Ankopplung der Auftragsausgabe an das Kommunikationssystem entwickelt werden.

BaSys40-Festlegung:

Jede BaSys40-Komponente besitzt für jeden Partner (Rolle) dem sie einen Request senden möchte eine **Auftragsausgabeeinheit**.

Die Auftragsausgabeeinheit vermittelt die internen Dienstaufrufe der Komponente an das Kommunikationssystem, sie empfängt die Antwort und stellt diese intern der Komponente zur Verfügung.

Festlegung 4.7

Typischerweise arbeitet die Auftragsausgabeeinheit asynchron zu den anderen Abläufen in der Komponente. Nach dem Anstoß wickelt sie das gesamte Austauschprotokoll selbständig ab. Die Hauptfunktion in der Komponente prüft zu dem Zeitpunkt an dem sie die Antwort benötigt, ob diese vorliegt und wertet diese dann aus. Liegt die Antwort zu diesem Zeitpunkt noch nicht vor, dann obliegt es der Strategie der Anwendung was weiter geschieht (warten, abbrechen, wiederholen, …).

4.4.2 Zuordnung von Kommunikationspartnern

Für Komponenten gilt das Gebot der Kontextneutralität (siehe Abschnitt 3.3). Für die Auftragsausgänge bedeutet dies, dass die internen Komponentenfunktionen die Aufrufe nicht an konkrete Partner, sondern an Rollen adressieren. Dies gilt grundsätzlich für alle Arten von Komponenten. Die Auftragsausgangseinheiten repräsentieren also intern Rollen an die sich die Funktionen wenden. Die Festlegung welche Realisierungseinheit diese Rolle zurzeit ausfüllt erfolgt dynamisch durch eine Parametrierung der Auftragsausgabeeinheiten. Diese Parametrierung kann entweder über die Signaleingänge erfolgen oder durch einen Konfigurationsauftrag an die Komponente. In jedem Fall muss die Parametrierung dynamisch möglich sein.

BaSys40-Festlegung:

Jede Auftragsausgabeeinheit repräsentiert eine **Rolle**, an die die Komponentenfunktionen ihre Aufträge richten. Die Zuordnung der konkreten Realisierungseinheit erfolgt durch Parametrierung der Auftragsausgabeeinheiten.

Diese **Parametrierung** muss in einem BaSys40-Kontext zwingend dynamisch realisiert werden können.

Festlegung 4.8

In Abbildung 4.3 und Abbildung 4.4 sind die Verhältnisse dargestellt.

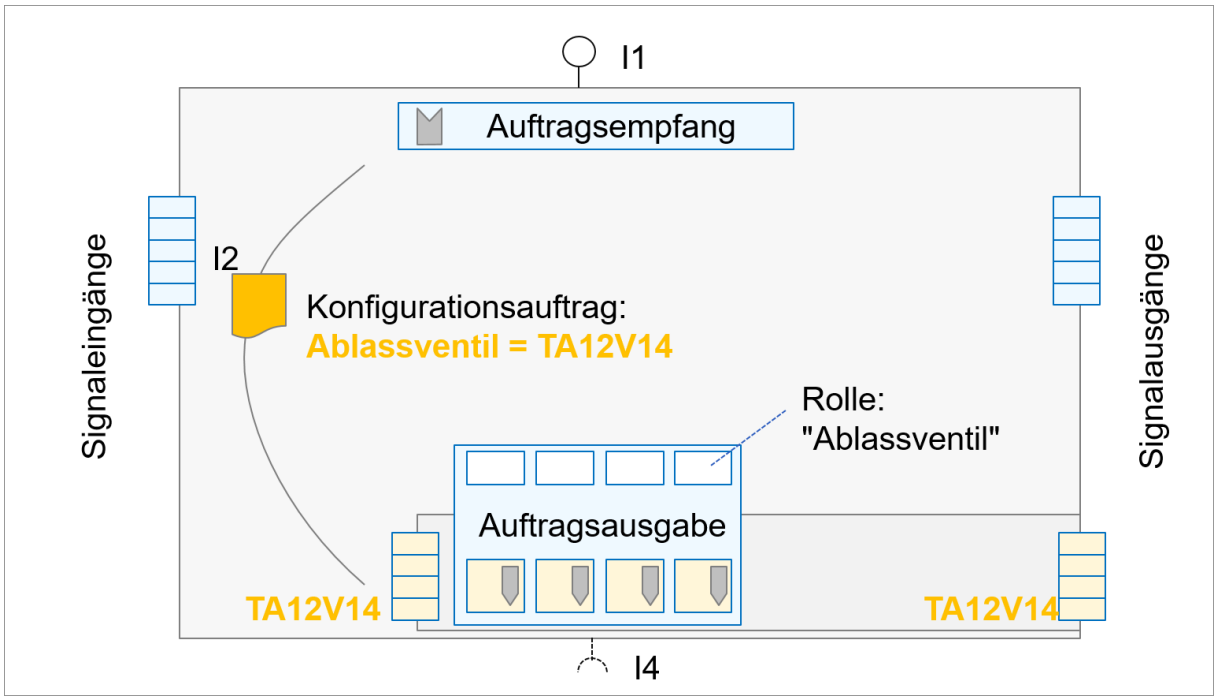

Abbildung 4.3: Dynamische Zuordnung der aktuellen Realisierungseinheiten.

Abbildung 4.3 zeigt den Vorgang der dynamischen Zuordnung der Realisierungseinheit durch einen Konfigurationsauftrag. Die Konfiguration wird als Parameter an der Auftragsausgabeeinheit abgelegt und – falls existent – an den Signalausgängen erkennbar angezeigt.

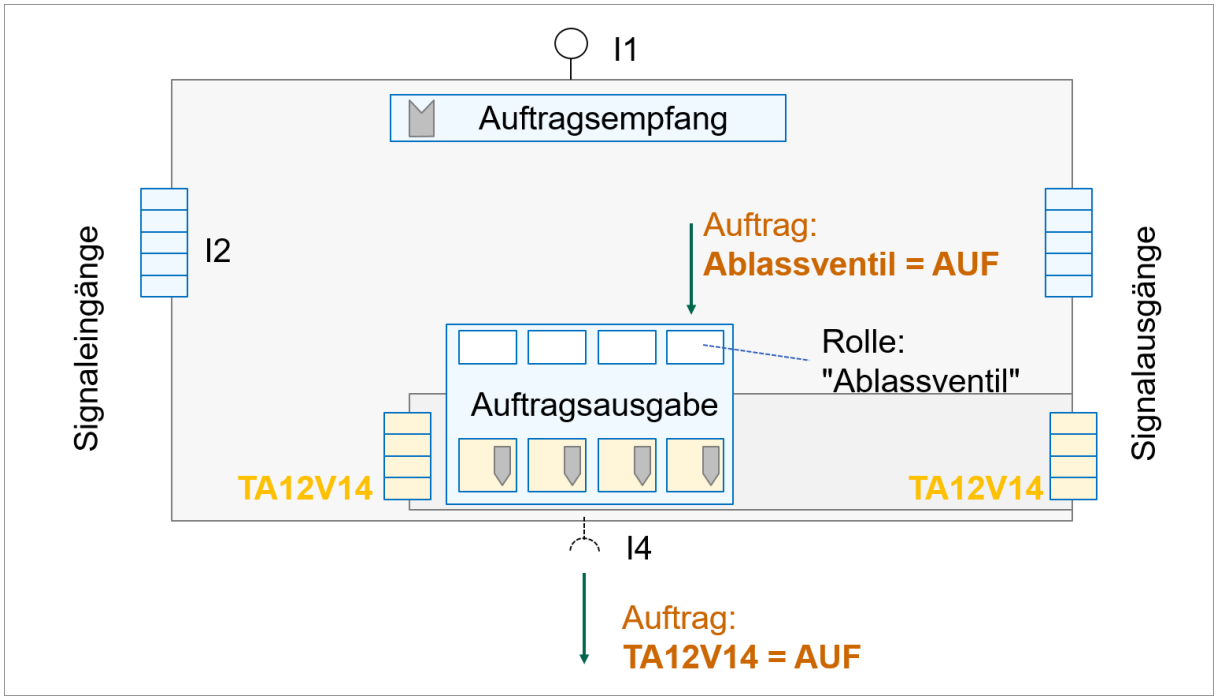

Abbildung 4.4: Automatisches Routing der Auftragsausgabe von der Rolle an die parametrierte Realisierungseinheit.

Abbildung 4.4 zeigt die Situation im operativen Betrieb. Zur Ausführung ihrer Aufgabe will die Komponente das Ablassventil öffnen. Die Komponentenfunktion kennt die reale Anlagenumgebung nicht und weiß nicht welches Ventil das zu öffnende Ablassventil ist. Die Auftragsausgabe routet den Befehl „Ablassventil = AUF“ an die parametrierte Realisierungseinheit TA12 V14.

Für den Informationshaushalt ist es sinnvoll die aktuelle Parametrierung als Ausgangssignale sichtbar zu machen.

BaSys40-Empfehlung:
Die Parametrierung der den Rollen zugeordneten Realisierungseinheiten ist an den Signalleisten sichtbar. BaSys40 stellt dafür **Standardausgänge** zur Verfügung.

Empfehlung 4.6

Bemerkung: Die Sichtbarmachung kann auch durch Herausführung der Parameter an die Eingangssignalleisten erfolgen.

5 Zustandsautomaten einer BaSys40-Komponente

Der Zustand einer Komponente muss unter gewährten Zugriffsrechten zu jederzeit abfragbar sein. Nur so können die Dienstnutzer der BaSys40-Komponente operative Entscheidungen treffen und der aktuelle Zustand der Anlage bleibt erfassbar. Dies kann über eigene Zustandssignalausgänge oder durch das BaSys40-Dienstsystems bzw. Dienste der BaSys40-Komponente selber sichergestellt werden. Die BaSys40-Komponente kann nach Innen durch mehrere, teils orthogonale, teils abhängige Zustandsautomaten dargestellt werden. Da vielen BaSys40-Komponenten bestimmte Zustandsautomaten gemein sind, können diese einheitlich beschrieben und möglicherweise standardisiert werden. Dies erleichtert gleichfalls einen einheitlichen Zugriff über das BaSys40-Dienstsystem.

5.1 Konventionen

Je nach Anwendung der BaSys40-Komponente sind verschiedene Zustandsautomaten für die BaSys40-Komponente relevant. Die notwendigen Zustandsautomaten werden in Abschnitt 9 für die einzelnen Anwendungen spezifiziert. Zur Definition der Automaten gilt folgende Empfehlung, welche beispielhaft in Abschnitt 5.2 für den Belegungszustand umgesetzt ist.

BaSys40-Empfehlung:

Mehrere, orthogonale Zustandsautomaten mit sinnvoller technischer bzw. funktionaler Abgrenzung sind größeren gemeinsamen Zustandsautomaten vorzuziehen.

Eine Vollständige **Definition eines Zustandsautomaten** einer BaSys40-Komponente umfasst:

- Definition der Zustände durch Auflistung von Nummer, Name und einer Beschreibung.
- Definition der ORDER Namen die von dem Zustandsautomaten verarbeitet werden.
- Standardisierte Beschreibung des Zustandsautomaten mit Transitionsbedingungen.

Empfehlung 5.1

5.1.1 Zustandssignalausgänge

In BaSys wird der Ansatz verfolgt, Zustände neben der Abfrage durch BaSys40-Dienste auch als Ausgangssignale zur Verfügung zu stellen. Um die Zustände möglichst einheitlich darzustellen, gilt folgende Empfehlung:

BaSys40-Empfehlung:

Jedem Zustand eines Zustandsautomaten ist eine **natürliche Zahl** (inkl. 0) zugeordnet.

Jedem Zustand eines Zustandsautomaten ist ein Name zugeordnet. Dieser soll einem möglichst einfachen englischen Kurznamen entsprechen, nur **alphanumerische** Zeichen (A-Z, a-z, 0-9) beinhalten und mit einem Buchstaben beginnen.

Jeder Zustandsautomat der Komponente besitzt zwei Signalausgänge: Einen als natürliche Zahl inkl. 0 und einen als Zeichenkette.

Empfehlung 5.2

Ein Beispiel für eine Zustandsdefinition könnte sein: 0:"IDLE". So müssen maschinenlesbar nur die vorzeichenlosen Signalausgänge als Zahlen verglichen werden. Diese haben zusätzlich den Vorteil, dass Sie eine Ordnung besitzen und somit Zahlenräume (Zustandsbereiche) verglichen werden können. Beispielsweise könnte der Zustandsbereich 0-9 eines Automaten guten Zuständen entsprechen, 10-19 sind Zustände mit Warnungen, 20-29 sind Fehler Zustände, die quittiert werden müssen. Auch entsprechen diese Zahlen oft den Nummern von Enumerationen in entsprechenden Implementierungen. Dies erleichtert das Zurückverfolgen und Beheben von Fehlern. Weiter ist der Vergleich von Zahlen einfacherer, da keine Probleme mit Groß- und Kleinschreibung sowie Sonderzeichen und deren Codierung entstehen. Nicht zuletzt benötigen Zahlen eine Definierte Speichergröße, was für deterministische Zugriffe und Systeme mit sehr begrenztem Speicher wichtig ist.

Gleichzeitig kann ein menschlicher Bediener über den Kurznamen verstehen, um welchen Zustand es sich handelt und dessen Semantik interpretieren, ohne zusätzliche Maßnahmen zur Dekodierung der Zahl vorzunehmen. Weiter kann der Name direkt für Meldungen oder Anzeigen in Bedienoberflächen genutzt werden.

Für die in diesem Kapitel definierten Zustandsautomaten werden als Konvention folgende Zustandssignalausgänge nach Empfehlung 4.5 definiert. Die Werte, welche die Zustände annehmen können, werden in den entsprechenden Unterkapiteln definiert.

Zustandsautomat	Eigenschaft	Name des Signalausgangs
Belegung	Aktueller Belegungszustand	OCCST
	Aktuelle Beleger ID	OCCUPIED
	Vorherige Beleger ID	OCCLAST
Betriebsart	Aktive Betriebsart	EXMODE
Betriebszustand	Aktueller Betriebszustand	EXST
Fahrweise	Aktive Fahrweise	OPMODE
Arbeitszustand	Aktueller Arbeitszustand	WORKST
Fehlerzustand	Aktueller Fehlerzustand	ER
	Vorheriger Fehlerzustand	ERLAST

Tabelle 5.1: Namen der Standardausgänge für die verschiedenen Zustandsautomaten.

5.1.2 Komponententyp-spezifische Zustandsautomaten

In der Regel erfolgt neben verschiedenen Anwendungsklassen, eine technologische Typisierung, welche sich in verschiedenen Komponententypen wiederspiegelt. Für die Anwendung als Prozessführungskomponente mag es beispielsweise Komponenten zur Steuerung einer Pumpe, eines Ventils, einer Pick and Place Einheit, eines Rollgangs, etc. geben. Um Komponententyp-spezifische Zustandsautomaten für alle BaSys40-Komponenten einheitlich zu gestalten gilt es Empfehlung 5.1 und Empfehlung 5.2 zu beachten.

Als ein Beispiel für einen Komponententyp-spezifischen Zustandsautomaten kann der Beladungszustand eines Rollgangs betrachtet werden. Alle Rollgänge können unbeladen, einseitig vorne, einseitig hinten oder beidseitig (eine Palette zentral oder eine vorne und eine hinten) beladen sein. Dies ergäbe also vier Zustände die sich aus einer Verknüpfung des Prozessabbildes ergeben. Ein weiteres Beispiel könnte eine Pick & Place Einheit oder ein Greifer sein. Hier könnte zum Beispiel der Zustand „0:EMPTY“ und „1:PICKED“ sein.

5.1.3 Auftragszustand

Der Auftragszustand ist nicht explizit modelliert, da die BaSys40-Architektur eine rückmeldungsfreie Nachrichtenkommunikation erlaubt. Dies ist in Abschnitt 4.4.1 bereits beschreiben worden. Überlagerte Einheiten müssen daher selber (beispielsweise über die folgenden Zustandsautomaten) feststellen, dass der Auftrag die gewünschten Prozesse in der Anlage hervorruft. Damit überprüft die überlagerte Einheit nicht den Ablauf der Prozedur selber, sondern den Ablauf des Prozesses. Dies erhöht die Flexibilität und stellt sicher, dass der Prozess korrekt ausgeführt wird. Beispielsweise können Störungen so zwischenzeitlich durch andere Einheiten, wie etwa von Hand, behoben werden. Man vergleiche dazu auch das Führungsmodell der Kernmodelle DIN Spec (Spezifikation DIN SPEC 40912). Zusätzlich kann aber über den Arbeitszustand (siehe Abschnitt 5.5) auf den Auftragszustand zurückgeschlossen werden.

5.2 Belegungszustand

In der Automatisierungstechnik ist es üblich, dass die Nutzer eines Betriebsmittels klassifiziert werden in Operator (Manual oder Hand), Automatikfunktion (Auto) oder Vor-Ort-Service (Local oder Vor Ort). In Abbildung 5.1 ist der zu dieser Klassifizierung gehörende Automat dargestellt.

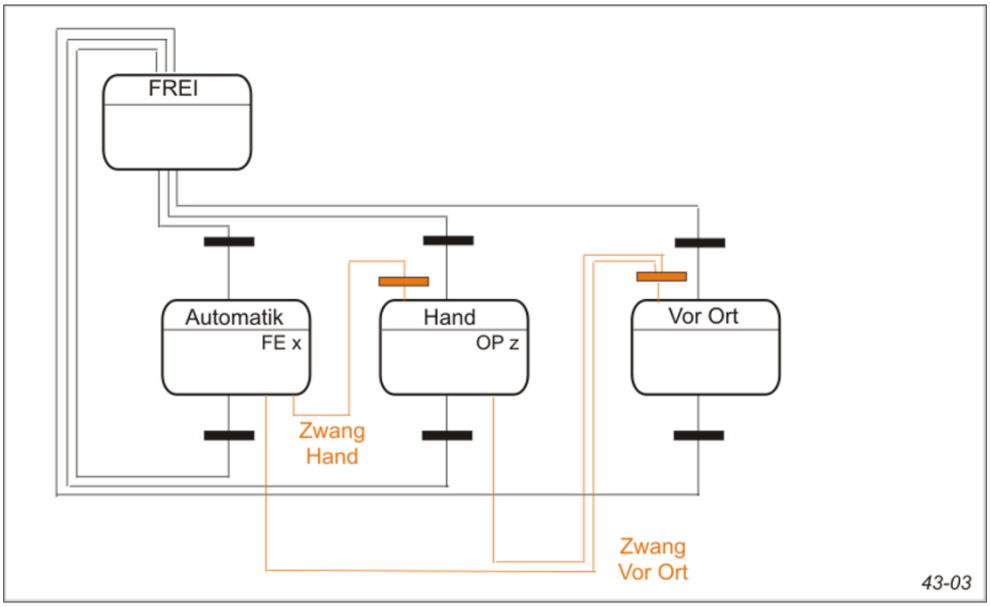

Abbildung 5.1: Belegungsautomaten in der Automatisierungstechnik. (Abel 2008, 4.3.2)

Prinzipiell muss ein Nutzer erst die Komponente freigeben, bevor sie durch einen anderen Nutzer belegt werden kann. Aus Sicherheitsgründen kann im Notfall jedoch ein Operator (Leitwarte oder vor Ort) die Belegung durch einen „Zwang-Hand“-Eingriff (z.B.: durch Zweihandbedienung) erzwingen.

Der Belegungszustandsautomat einer Komponente legt analog fest, wer gerade die Komponente bedient und von wem Aufträge angenommen werden. Für den Hand- und vor-Ort-Betrieb existieren passende Überschreibungsmöglichkeiten, da diese eine höhere Priorität besitzen. Zur besseren Abgrenzung von Betriebsarten und Betriebszuständen, wurden statt Automatik und Hand die englischen Kurznamen OCCUPIED und PRIORITY nach Empfehlung 5.2 gewählt. Weiter unterstützt dies die Trennung in orthogonale Automaten. Außerdem betont dies die allgemeinere Bedeutung des Belegungszustandes für die Allokation der Komponenten und ihrer Ressourcen, wie in Abschnitt 4.1.3.1 beschrieben. Entsprechend ergeben sich folgende Zustände:

#	Name	Beschreibung
0	FREE	Nicht belegt.
1	OCCUPIED	Eine andere Einheit (BaSys40-Komponente) übernimmt gerade die Steuerung.
2	PRIORITY	Ein Operator übernimmt Steuerung von Hand. Zum Beispiel zum Einrichten eines Betriebsmittels oder zum Testen von Teilfunktionalitäten.
3	LOCAL	Die Komponente ist vor Ort belegt. Zum Beispiel durch einen Handschalter oder ein Bedienpanel, welches vor Ort die Steuerung einer Zelle übernimmt, ohne über eine Einzelsteuereinheit zu agieren. Dieser Zustand ist optional.

Tabelle 5.2: Belegungszustände der BaSys40-Komponente

Der Belegungszustandsautomat kann als Ablaufgraph innerhalb der BaSys40-Komponente hinterlegt sein. Eine Vorlage dafür ist in folgender Abbildung gezeigt:

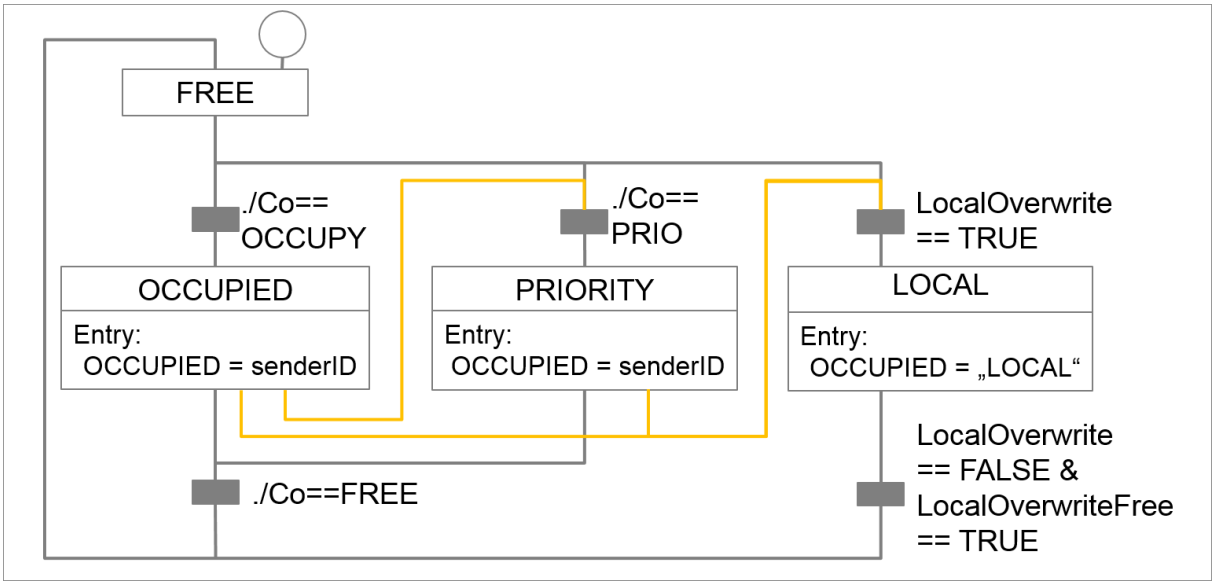

Abbildung 5.2: Vorlage für einen Ablaufgraph des Belegungszustandsautomaten mit optionalem LOCAL Zustand.

Folgende Ports werden vom Ablaufgraph des Belegungszustandsautomaten benötigt:

Signalname	Eingang / Ausgang	Datentyp	Kommentar
Co	E	String	Kommandoeingang.
SenderID	E	String	Angeforderter Beleger.
LocalOverwrite	E	Bool	Belegung durch Zwang vor Ort angefordert.
LocalOverwriteFree	E	Bool	Belegung durch Zwang vor Ort freigegeben.
OCCUPIED	A	String	Aktueller Beleger.
OCCST	A	String	Aktueller Belegungszustand / Schritt im Ablaufgraph. Z.B.: „FREE“

Tabelle 5.3: Ports des Belegungszustandsautomaten

Folgende ORDER werden vom Belegungszustandsautomaten verarbeitet:

ORDER	Parameter	Kommentar
FREE		Selbst belegte Komponenten freigeben.
OCCUPY		Komponente belegen.
PRIO		Komponente mit Priorität übernehmen. (Zwang von Hand)

Tabelle 5.4: ORDER des Belegungszustandsautomaten

5.3 Betriebsarten und Betriebszustände

Nachdem durch den Belegungszustand festgelegt ist, wer die Komponente gerade bedient, entscheidet die Betriebsart der Komponente darüber, wie sie auf Kommandos reagiert und wie beispielsweise die Transitionen in Ablaufprogrammen geschaltet werden. Dabei ist die Komponente zu jedem Zeitpunkt exakt in einer Betriebsart.

BaSys40-Festlegung:

Jede BaSys40-Komponente, die verschiedene Betriebsarten unterstützt, befindet sich zu jedem Zeitpunkt in exakt einer **Betriebsart**.

Festlegung 5.1

Je nach Betriebsart schreitet der Prozedurablauf unterschiedlich voran. Beispielsweise könnten so in einem Einricht- oder Wartungsbetrieb Prozedurabschnitte schrittweise durchlaufen werden oder Randbedingungen an Grenzwerte wie Maximalgeschwindigkeiten formuliert werden. Gegebenenfalls ist es in solch einer Betriebsart möglich Schritte zu überspringen.

Für automatische Werkzeugmaschinen eigenen sich beispielsweise Sicherheitsnormen (Norm DIN EN 12417; Norm DIN EN 13128) zur Beschreibung von Betriebsarten. Hierin finden sich insbesondere folgende vier Betriebsarten wieder:

- Automatikbetrieb
- Einrichtbetrieb
- Prozessbeobachtung in der Fertigung
- Prozessbeobachtung in der Fertigung ohne Zustimmeinrichtung

Es kann vorkommen, dass bestimmte Betriebsarten nur in bestimmten Belegungszuständen erreicht werden können. So ist ein schrittweiser Hand-Betrieb nur im Belegungszustand PRIO oder LOCAL sinnvoll, da bei Belegung durch eine andere Komponente möglicherweise keine manuelle Zustimmung zur Aktivierung von Transitionsschritten vorhanden ist. Auch ist so sichergestellt, dass niemand anderes die Komponente durch einen Zwangeingriff übernehmen kann. Da der Belegungszustandsautomat jedoch weitestgehend Domänenübergreifend bzw. generisch gültig ist, ist es sinnvoll diesen von den Betriebsarten zu trennen. Dies entspricht Empfehlung 5.1.

Ein Wechsel der Betriebsart erfolgt, wenn alle Bedingungen dafür erfüllt sind oder ein entsprechendes Kommando erhalten wird. Dabei kann ein Wechsel in einer Komponente dazu führen, dass weitere, beispielsweise unterlagerte, Komponenten ebenfalls einen Wechsel der Betriebsart durchführen. Die Regeln zum Propagieren des Betriebsartwechsels werden hier nicht betrachtet und sind im Einzelfall festzulegen.

In der Norm ISA-88 werden Modi (Norm ISA-88.00.01, 7.3 Table 1) für Batch-Fahrweisen vorgestellt, welche als Betriebsart für BaSys40-Komponenten herangezogen werden können. Dabei wird unterschieden in Prozedurale Einheiten und Einheiten für Betriebsmittel. Dies entspricht in etwa der Unterscheidung in Einzelsteuereinheiten und Gruppensteuereinheiten in BaSys40. Beispielhaft nennt und erläutert die ISA-88 drei verschiedene Modi:

- Automatic
- Semi-automatic (Nur Prozedurale Elemente)
- Manual

In ISA-88 Teil 2 (Norm ISA-88.00.02), im Folgenden als PackML bezeichnet, werden diese Betriebsarten angewendet und mit Nummern versehen, was als Grundlage für BaSys40-Komponenten gelten kann. Folgende Modi werden definiert:

- 0: Undefined
- 1: Producing
- 2: Maintenance
- 3: Manual
- 4-15: <future reserve>
- 16-n: User Defined <1-n>

Damit BaSys40-Komponenten domänenübergreifend eingesetzt werden können, sollten Sie verschiedene Betriebsartmodelle unterstützen. Dafür können die Betriebsarten des domänenspezifischen Betriebsartenmodells auf die Betriebsarten von PackML abgebildet werden, oder benutzerdefinierte Betriebsarten definiert werden. So lassen sich die Betriebsarten für automatische Werkzeugmaschinen beispielsweise wie folgt auf die zuvor gezeigte Liste abbilden:

#	ISA-TR88.00.02 „PackML“	ISA-88.00.01-2010 „ISA-88“	DIN EN 12417 und 13128	BaSys40
1	Producing	Automatic	Automatikbetrieb	AUTO
2	Maintenance	Semi-automatic	Einrichtbetrieb	SEMIAUTO
3	Manual	Manual	Prozessbeobachtung in der Fertigung	MANUAL
4	future reserve		Prozessbeobachtung in der Fertigung ohne Zustimmung	

Tabelle 5.5: Zuordnung der Betriebsarten von PackML, ISA-88 und DIN EN 12417

Als Beispiel für eine „User Defined“-Betriebsart wird ein Reinigungsmodus für Verpackungsmaschinen genannt. Dies könnte jedoch auch durch eine Fahrweise realisiert werden, siehe dazu Abschnitt 5.4. Damit die Zustände zugeordnet werden können, sollte zusätzlich das Betriebsartmodell referenziert werden. Auch hier hilft die Nummer der Zustände zur eindeutigen Zuordnung, wie in Empfehlung 5.2 beschrieben ist.

Um den Übergang zwischen den verschiedenen Betriebsarten zu definieren führt PackML ein sogenanntes Mode-Management ein. Der Wechsel einer Betriebsart kann nur aus bestimmten Betriebszuständen erfolgen, wie in folgender Abbildung exemplarisch gezeigt:

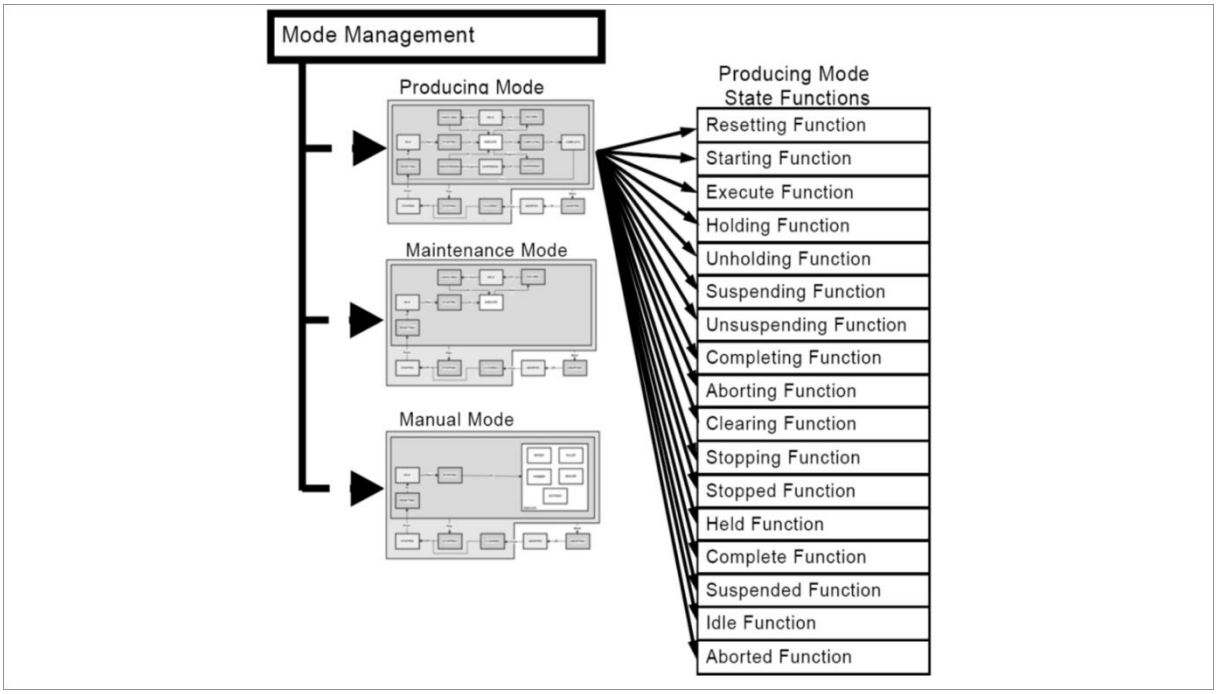

Abbildung 5.3: Betriebsarten Wechsel in PackML (Norm ISA-88.00.02, Figure 5)

Als Auslöser für einen Betriebsartwechsel kommen eine ORDER oder interne Zustandswechsel in Frage. Interne Zustandswechsel können beispielsweise bei einem Signal von einem

Schlüsselschalter oder einem Fehler auftreten. Dabei sind gewisse Sicherheitsaspekte zu beachten, sodass beispielsweise eine Quittierung oder Zustimmung beim Wechsel von MANUAL in AUTO notwendig ist. Folgende ORDER können vom Betriebsartautomat in BaSys40-Komponenten verarbeitet werden:

ORDER	Parameter	Kommentar
AUTO		Wechsel in Betriebsart AUTO
SEMIMANUAL		Wechsel in Betriebsart SEMIMANU
MANUAL		Wechsel in Betriebsart MANUAL

Tabelle 5.6: ORDER des Betriebsartautomaten

Wird die Betriebsart von der BaSys40-Komponente nicht ausgegeben, beispielsweise weil die Komponente sehr einfach ist und keine Betriebsarten benötigt werden, ist davon auszugehen, dass die Komponente sich wie in der Betriebsart AUTO verhält.

5.3.1 Betriebszustand

In jeder Betriebsart existieren verschiedene Betriebszustände für die Komponente. Um den Betriebszustand der BaSys40-Komponente zu beschreiben, eignet sich ebenfalls das Zustandsmodell aus der ISA-88-2. PackML beschreibt die Anwendung der „Modes“ (Betriebsarten) und „States“ (Betriebszustände) für die Verpackungsmaschinen-Domäne und kann daher als Beispiel herangezogen werden.

BaSys40-Festlegung:

Jede BaSys40-Komponente befindet sich zu jedem Zeitpunkt in exakt einem **Betriebszustand** einer Betriebsart.

Alle Betriebszustände der BaSys40-Komponente müssen auf die Betriebszustände aus Deliverable D-PC2.3 Abschnitt 3.4.1 (PackML) **abgebildet** werden.

Festlegung 5.2

Wenn in einer BaSys40-Komponente mehrere, parallele BaSys40-Dienste ablaufen, muss im Einzelfall definiert werden, wie sich der gesamt Betriebszustand der Komponente aus den einzelnen Betriebszuständen der Dienste zusammensetzt. Für BaSys40-Komponenten zur Prozessführung ist es daher sinnvoll einen Fahrweisenautomat (vgl. Abschnitt 5.4) ohne Nebenläufigkeit einzuführen. Dies wir bei den Anwendungen der BaSys40-Komponente (vgl. Abschnitt 9.2) genauer behandelt.

Wie sich die Betriebszustandsautomaten auf Funktionsbauseine und Ablaufsprachen (CFC/SSC) abbilden lassen, ist in Deliverable D-PC2.3 in Abschnitt 3.4.2 beschrieben. Zuvor werden die relevanten Zustandsautomaten darin in Abschnitt 2.5 beschrieben.

In der folgenden Grafik ist der Zustandsautomat für den „Automatic Mode“ nach PackML V3 zu sehen. Eine Tabellarische Beschreibung der Zustände und Transitionen ist in Tabelle 5.7 zu finden.

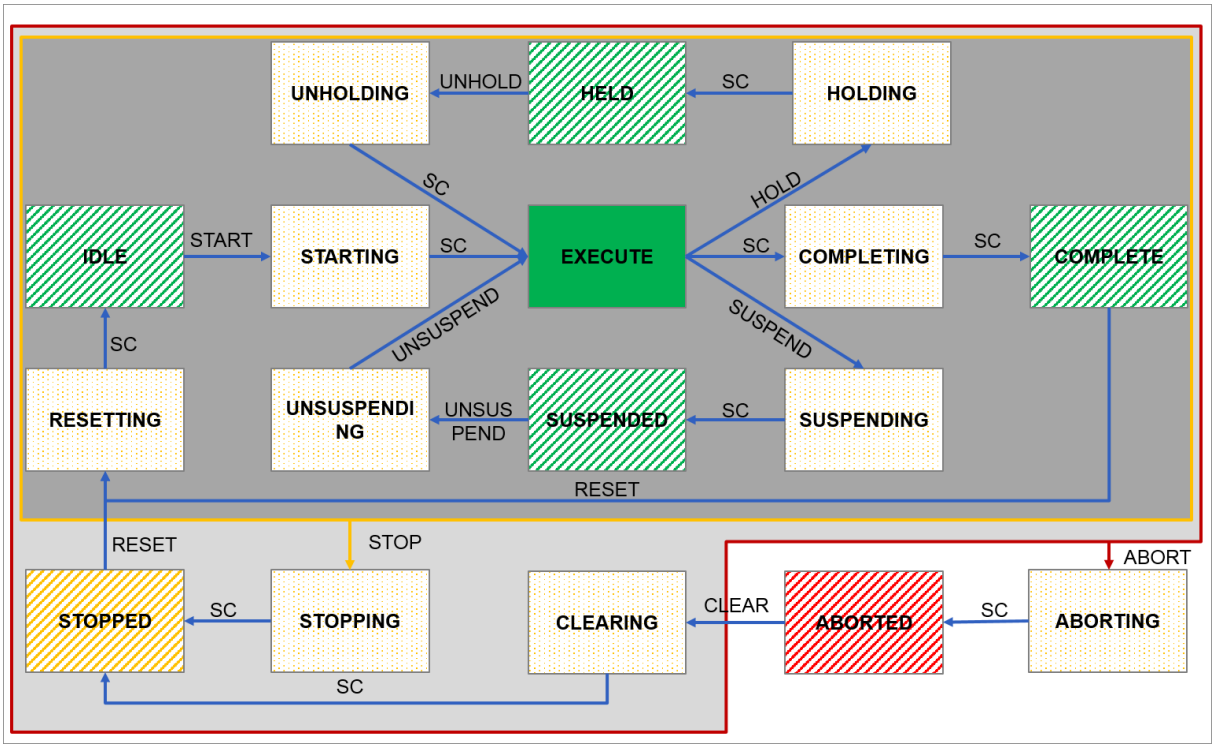

Abbildung 5.4: Betriebszustände nach PackML im Automatic Mode (Norm ISA-88.00.02, Figure 3)

Der Wechsel eines Betriebszustandes kann verschiedene Auslöser haben. Man unterscheidet grundsätzlich nach Steuereingriff durch eine ORDER, oder Zustandswechsel (StateChange, SC). Der interne Zustandswechsel kann dabei durch Änderungen des Prozessabbildes, bzw. durch Zustandswechsel zu steuernder Betriebsmittels oder unterlagerter Komponenten hervorgerufen werden. Auch Änderungen des Zustandes durch interne Logiken bzw. Arbeitsabläufe sind möglich.

Die Modellierung des Betriebszustandes durch Teilzustandsautomaten ist ebenfalls möglich. Oft können einfache Systeme auch mit einer Teilmenge von Zuständen auskommen. Dann ist es wichtig, dass diese auf den in Deliverable D-PC2.3 beschriebenen Betriebszustandsautomat abgebildet werden, um einheitliches maschinelles Handling des Automaten zu ermöglichen. Beispiele für die Reduktion des Zustandsautomaten finden sich im Deliverable D-PC2.3 in Abschnitt 3.4.4. In Zukunft lassen sich domänenspezifische Betriebszustandsautomaten gegebenenfalls auch als eigene „User Defined“ Betriebsarten standardisieren, sodass ein Mapping nur noch einmal nötig ist.

Sollte von einer BaSys40-Komponente kein Betriebszustand angezeigt werden, weil es sich beispielsweise um eine sehr einfache Komponente handelt, ist davon auszugehen, dass sich die Komponente im Minimalen Betriebszustandsmodell, wie in D-PC2.3 Figure 18 beschrieben, im Zustand EXECUTE befindet.

Es existieren folgende Zustände und Übergänge mit den zugehörigen Kommandos im vollen Betriebszustandsautomaten in der AUTO Betriebsart:

Current State	**Übergangsmatrix mit Kommandos** Command										**State Complete (SC)**
	START	**COM-PLETE**	**RESET**	**HOLD**	**UN-HOLD**	**SUS-PEND**	**UNSUSPEND**	**CLEAR**	**STOP**	**ABORT**	**SC**
IDLE	STARTING								STOPPING	ABORTING	
STARTING									STOPPING	ABORTING	EXECUTE
EXECUTE		COMPLETING		HOLDING		SUSPENDING			STOPPING	ABORTING	
COMPLETING									STOPPING	ABORTING	COMPLETE
COMPLETE			RESETTING						STOPPING	ABORTING	
RESETTING									STOPPING	ABORTING	IDLE
HOLDING									STOPPING	ABORTING	HELD
HELD					UNHOLDING				STOPPING	ABORTING	
UNHOLDING									STOPPING	ABORTING	EXECUTE
SUSPENDING									STOPPING	ABORTING	SUSPENDED
SUSPENDED							UNSUSPENDING		STOPPING	ABORTING	
UNSUSPEND									STOPPING	ABORTING	EXECUTE
STOPPING										ABORTING	STOPPED
STOPPED			RESETTING							ABORTING	IDLE
ABORTING											ABORTED
ABORTED								CLEARING			
CLEARING										ABORTING	STOPPED

Tabelle 5.7: Betriebszustände und Kommandos nach PackML im Automatic Mode

5.4 Fahrweisen

Die Gliederung der Funktionalität von operativen Komponenten kann durch Fahrweisen realisiert werden. Die Komponente befindet sich dabei immer exakt in einer Fahrweise. Während die Betriebsart festlegt, wie der Ablauf von Programmen oder Schrittketten vonstatten geht und diesen somit überlagert ist, legt die Fahrweise fest was passiert. Fahrweisen können dazu beispielsweise als Ablaufprogramme oder Funktionsblocknetzwerke realisiert werden, die durch das BaSys40-Metamodell in Deliverable D-PC2.3 beschrieben sind. In jeder Fahrweise kann eine BaSys40-Komponente in verschiedenen Betriebszuständen sein. Im regulären Produktionsbetrieb befindet sich die Komponente innerhalb einer Fahrweise meist im Betriebszustand „EXECUTE“.

BaSys40-Festlegung:

Jede BaSys40-Komponente, die ihre Funktionalität durch verschiedene Fahrweisen darstellt, befindet sich zu jedem Zeitpunkt in **exakt einer Fahrweise** oder im Grundzustand.

Festlegung 5.3

Ein wesentlicher Vorteil dieser Festlegung ist, dass keine Nebenläufigkeit für operative Aufgaben zugelassen wird. Damit ergibt sich Beispielsweise der gesamt Betriebszustand aus dem Ablauf der aktiven Fahrweise. Es müssen also auch keine Querverriegelungen und gegenseitige Beeinflussungen zwischen den Fahrweisen definiert werden. Wie sich ein Fahrweisenautomat in die Architektur einer BaSys40-Komponenten zur Prozessführung einbetten lässt, ist in Abschnitt 9.2 beschrieben. Im Rahmen der Wandelbarkeit kann die parallele Nutzung der Basisfähigkeiten einer Komponenten durch spezielle Fahrweisen erreicht werden. Dies wird in Abschnitt 7.1 und Abbildung 7.2 genauer beschrieben. Die Nebenläufigkeit wird dabei durch eine Aufteilung in parallel arbeitende Komponenten erreicht, sodass pro Komponente weiterhin nur eine Fahrweise aktiv ist.

Die Fahrweisen selber sind dabei Komponenten- bzw. Komponententyp-spezifisch und können daher nicht allgemein definiert werden. Die Definition einer Fahrweise enthält ihren Kurznamen, einen Langnamen, eine Beschreibung, eine Liste von Parametern und Kommandos, welche die Fahrweise beeinflussen. Der Entsprechende Zustandsautomat enthält daher die Kurznamen und den Grundzustand „BSTATE“. Der Wechsel zwischen Fahrweisen findet grundsätzlich über den Grundzustand statt, wie in Abbildung 5.5 gezeigt ist. Der Wechsel einer Fahrweise ohne den Grundzustand zu durchlaufen (vgl. Abbildung 5.5 Wechsel von M1 zu M2) ist Fahrweisen spezifisch zu definieren. Ein solcher Wechsel hängt in der Regel von weiteren Zuständen, wie dem Arbeitszustand der Komponente, ab.

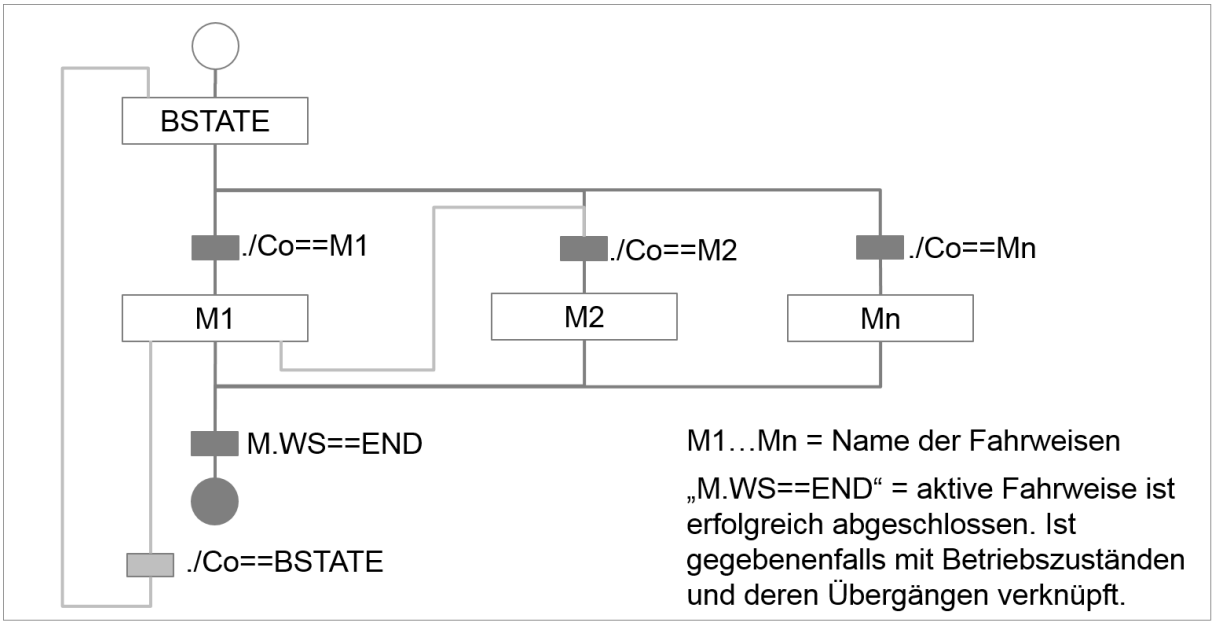

Abbildung 5.5: Vorlage für einen Ablaufgraphen des Automaten zur Fahrweisenauswahl

Das Verlassen einer Fahrweise kann dabei nicht nur durch ein spezifisches Kommando von außen erreicht werden. Beispielsweise können die Fahrweisen verlassen werden, wenn der Betriebszustand zu „COMPLETED“, „ABORTED“ oder „STOPPED“ gewechselt ist. Somit sind die Betriebszustandsautomaten und Fahrweisen nicht unabhängig voneinander. Über Kommandos zur Steuerung des Betriebszustandes können beispielsweise Abläufe innerhalb einer Fahrweise angehalten und wieder gestartet werden. Da diese Betriebszustände der Fahrweisen und die dazugehörigen Kommandos aber allen Fahrweisen gemein sind, können Sie unabhängig im Betriebszustandsautomaten definiert werden. Dies entspricht dem Konzept der „Separation of Concerns“ bzw. der Vereinfachung durch orthogonale Automaten, wie in Empfehlung 5.1 gefordert.

5.5 Arbeitszustand

BaSys40-Komponenten sind gemäß der Definition der technischen Komponente in Abschnitt 2.1 zur Erfüllung einer Rolle in einem System geschaffen. Um ihre Arbeit zu erfüllen, durchlaufen sie in der Regel bestimmte Prozeduren. Der aktuelle Arbeitszustand beschreibt, in welchem aktuellen Schritt sich die Komponente gerade befindet. Dazu kann beispielsweise der Name des aktuellen Schrittes nach außen geleitet werden. Bei verschachtelten Schritten kann ein hierarchisches Namenskonzept eingeführt werden, indem die Namen der verschachtelten Schritte durch Trennzeichen, beispielsweise „/“, zusammengeführt werden. Bei parallel ablaufenden Schrittketten kann eine Liste der aktuellen Schrittnamen eingeführt werden, um den Arbeitszustand zu definieren.

Das Konzept, den aktuellen Arbeitszustand zur Laufzeit für das Komponenten-, BaSys40- oder I40-System zur Verfügung zu stellen, ermöglicht den auf dieser Komponentenarchitekturen aufbauenden Dienstteilnehmern bessere Entscheidungen zu treffen. Damit ist die explizite Darstellung des Arbeitszustands für eine Erhöhung der Flexibilität wichtig und ein wichtiger Wandlungsbefähiger. Der Arbeitszustand ist auch zu Diagnose und Anzeige Zwecken wichtig.

5.6 Fehlerzustand

Falls eine Störung innerhalb der Komponenten oder bei unterlagerten Einheiten auftritt, muss diese nach außen darstellbar sein. BaSys40-Komponenten sollten dafür exakt einen Fehler nach außen geben. Da die Anwendungen für BaSys40-Komponenten sehr vielfältig seien können, kann kein allgemeiner Fehlerzustandsautomat festgelegt werden. Der Konvention für Zustandssignalausgängen folgend, wird jedoch folgendes festgehalten:

BaSys40-Festlegung:

Befindet sich die Komponente in keiner Störung hat sie den **Fehlerzustand 0**. Dieser trägt den Kurznamen „OK“.

Festlegung 5.4

Weiter ist es in der Automatisierung üblich, dass getrennte Diagnose und Alarmsysteme diese Fehlerzustände überwachen oder über die Auftragsausgabe der Fehlerhaften Komponente informiert werden.

5.7 Asset-Zustand

Bei BaSys40-Komponenten, die technische Assets (VDI/VDE-Gesellschaft Mess- und Automatisierungstechnik (GMA) 2015) in der Informationswelt vertreten oder diese nach außen Kapseln, kann es sinnvoll sein, Zustände des Assets nach außen hin darzustellen. Dies ist beispielsweise für Smart Devices sinnvoll, da die Einzelfunktionseinheit die unterlagerte Hardware nach außen kapselt. Vergleiche dazu Abschnitt 9.1. Für diese ist es beispielsweise wichtig den Betriebszustand der Einzelsteuerungseinheit von dem Betriebszustand der anlagentechnischen Hardwarekomponente (Asset) zu unterscheiden.

Als Beispiel könnte eine Einzelsteuereinheit und damit die Einzelfunktionseinheit als Ganzes im Betriebszustand „ABORTED“ verharren, weil die Hardwarekomponente selber stromlos ist, momentan keine Verbindung zum Leitsystem hat, neustartet und so weiter. Oder umgekehrt kann die Hardwarekomponente in einem „IDLE“ Zustand sein, während die Einzelsteuereinheit und damit die Einzelfunktionseinheit als ganzes im „STOPPED“ Zustand ist.

Dabei muss der Asset-Zustand nicht zwangsläufig in der BaSys40-Komponente abgebildet werden, sondern kann durch Verwaltungskomponenten zur Verfügung gestellt werden.

Insgesamt handelt es sich beim Asset-Zustand auch um eine Sammlung mehrerer Zustandsautomaten, die vom jeweiligen Asset-Typen abhängen.

6 Wandelbarkeit auf Komponentenebene

Im Rahmen des BaSys40-Konzepts wird die Flexibilität und Wandelbarkeit primär durch die dynamische Handhabbarkeit der Komponenten realisiert. Die BaSys40-Komponenten-Systemplattform erlaubt ein Hinzufügen, Wegnehmen oder Ersetzen von Komponenten zur Laufzeit. Ein Kern von BaSys40 ist die Festlegung von Standards, um diese Flexibilität in einem heterogenen und dezentralen Systemumfeld zu gewährleisten.

Für die Definition und Abgrenzung des Begriffs der Wandelbarkeit und Flexibilität wird auf das Deliverable D-FT2.1 Abschnitt 4.3.1 verwiesen. Darin beschreibt „Wandlungsfähigkeit die […] Eigenschaft eines Objekts, sich an Veränderungen anzupassen“ (Heger 2007, S. 25). Dagegen wird „Flexiblität […] dann erreicht, wenn eine Anlage bereits während ihrer Planungsphase auf bestimmte Eventualitäten vorbereitet wird“ (Nyhuis et al. 2008, S. 14). In D-FT2.1 werden verschiedene Wandlungsbefähiger vorgestellt. Diese sind kurz in Abbildung 6.1 dargestellt. Die BaSys40-Architektur wird komponentenorientiert auf Grundlage der in Abschnitt 2.2 definierten speziellen Komponenteneigenschaften gestaltet. Diese Eigenschaften unterstützen dabei bereits die verschiedenen Wandlungsbefähiger:

- **Modularität**: Komponenten können zu Verbünden zusammengefasst werden und selber als Komponentensystem aus Komponenten aufgebaut sein. Damit ist die Modularität ein Grundprinzip von Komponentenarchitekturen. Daneben kann der Aufbau von Komponenten in BaSys bspw. durch definierte Zustandsautomaten vereinheitlicht werden, was zusätzlich in den nächsten Abschnitten 7.1 und 8.2 beschrieben ist.
- **Neutralität**: BaSys40-Komponenten müssen die Anforderung der Kontextneutralität (Abschnitt 3.3) erfüllen. Zusätzlich wird durch die explizite *Erkennbarkeit externer Abhängigkeiten* und durch die weiteren speziellen Eigenschaften der Komponenten sichergestellt, dass zumindest die Einflüsse auf andere Objekte bekannt sind. Im Sinne der Prozessführung verhindern insbesondere die hierarchischen Belegungsstrukturen ungewollte Einflüsse.
- **Kompatibilität**: Die Kompatibilität wird bei Komponenten durch die explizite Definition der Schnittstellen (Abschnitt 4) sichergestellt. Dabei ist vor allem die Eigenschaft „*Erkennbarkeit externer Abhängigkeiten*“ eine wichtige Voraussetzung.
- **Universalität**: Die Universalität wird nicht direkt unterstützt, da Komponenten zu einem bestimmten Zweck (Rolle im System) geschaffen und typisiert sind. Die Universalität kann verbessert werden, indem die Komponente Modular aufgebaut ist, wie im nächsten Abschnitt 7.1 beschrieben.
- **Skalierbarkeit**: Die Skalierbarkeit wird in erster Linie in diskreten Schritten von dieser Komponentenarchitektur unterstützt. Dabei können weitere Komponenten hinzugefügt oder entfernt (bzw. in oder außer Betrieb genommen) werden, um eine skalierte Lösung zu erstellen. Auch hier kann auf Ebene der internen Komponentenstruktur die Skalierbarkeit verbessert werden, wie im nächsten Kapitel 7 beschrieben ist.
- **Standardisierung**: Dieses Dokument dient der Vereinheitlichung von Komponenten insbesondere zur Prozessführung und trägt somit zur Standardisierung bei. Gerade bei den Zustandsautomaten wurde dabei versucht auf bereits standardisierte Elemente zurückzugreifen.
- **Mobilität**: Die Mobilität wird vor allem durch die *Kompaktheit*, *Abgegrenztheit* und *Disjunktheit* der Komponenten sichergestellt. Versteht man unter Mobilität neben der

räumlichen Bewegbarkeit auch das Verschieben von Software auf einen anderen physischen Träger, so gelten diese Eigenschaften für diese Komponentenart genauso.

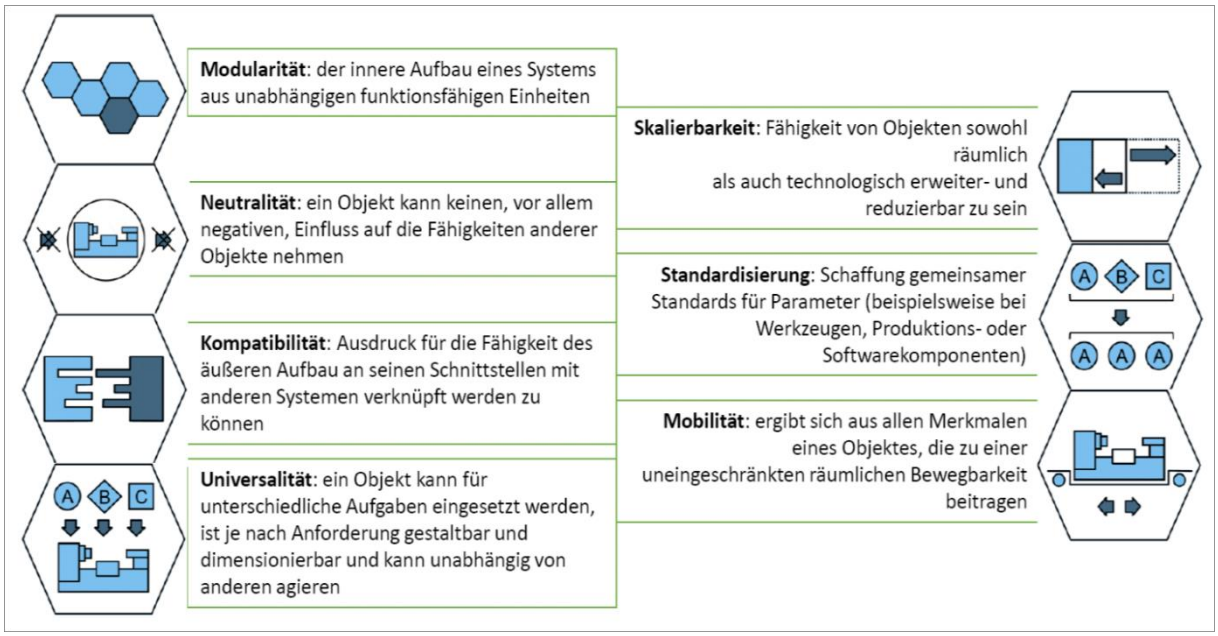

Abbildung 6.1: Wandlungsbefähiger. Aus Präsentation beim ersten BaSys40 Projekttreffen im Mai 2017.

Insgesamt trägt die Eigenschaft *Hantierbarkeit als Einheit* der technischen Komponenten dazu bei Wandelbarkeit zu ermöglichen. Die wesentliche Aufgabe bei der Wandlung auf Komponentenebene ist das Hinzufügen einer neuen Komponente. Das Hinzufügen einer neuen Komponente besteht aus zwei Teilen:

1. Die Bereitstellung der neuen Komponente
2. Das Einbinden der Komponente in den Systemverbund

Je nach Art der Komponente und ihrem Hardware-/Softwarecharakter erfolgt die Bereitstellung einer neuen Komponente nach unterschiedlichen Mechanismen.

BaSys40-Festlegung:

BaSys40-Komponenten können auf folgende Weise **bereitgestellt** werden:

1. die Bildung einer Instanz aus einer Klasse,
2. das Kopieren einer vorhandenen Instanz,
3. das Konstruieren aus einzelnen Teilelementen,
4. die Auswahl einer geeigneten Komponente aus einem Pool von verfügbaren Komponenten

Festlegung 6.1

Die ersten beiden Mechanismen spielen sich vollständig im Softwarebereich ab und sollen in BaSys durch die Komponenten-Systemplattform selbst organisiert werden. Dies gilt auch für den dritten Mechanismus, solange es sich bei den Teilelementen um reine Softwareelemente

handelt. Die Erstellung bzw. Beschaffung von Hardware ist außerhalb des Scopes der Komponenten-Systemplattform. Hardwarekomponenten müssen vorab beschafft und in einem Pool verfügbarer Komponenten dem System zur Verfügung gestellt werden. Dies gilt für Smarte-Komponenten als Ganzes, aber auch für die Hardwareteile von Verbundkomponenten. Das Prinzip, eine neue Komponente durch Auswahl aus einem Pool vorhandener Komponenten hinzuzufügen, kann auch in bestimmten Softwareumgebungen angewendet werden. In diesem Fall stellt die Komponenten-Systemplattform eine feststehende Anzahl von Komponentenressourcen in einem Pool zur Verfügung. Aus diesem können dann einzelne Komponenten bei Bedarf für bestimmte Aufgaben genutzt werden.

Nach dem Bereitstellen einer neuen Komponente muss diese in den Systemverbund eingebunden werden. Bei reinen Softwarekomponenten bedeutet dies die Einbindung und Aktivierung in die jeweilige lokale Laufzeitumgebung und die Einbindung in das übergeordnete Dienstsystem. So muss beispielsweise auch eine Registrierung der Komponente erfolgen. Smarte-Komponenten besitzen ihr eigenes Laufzeitsystem. Sie müssen jedoch als neuer Teilnehmer in die Netzwerkinfrastruktur, beispielsweise durch Infrastrukturkomponenten, integriert werden.

Für die ersten drei Mechanismen und die dynamische Einbindung in den Systemverbund stellt BaSys40 Referenzmodelle als gemeinsamen Standard zur Verfügung. Diese Referenzmodelle sind Grundlage der BaSys40-Komponentenverwaltungsdienste (BaSys Deliverable D-PC2.15) der BaSys40-Komponenten-Systemplattform.

6.1 Bildung von BaSys40-Softwarekomponenten durch Instanziierung

Die BaSys40-Komponenten-Systemplattform unterstützt ein Typ-/Instanzkonzept. Das Typ-/Instanzkonzept ist Grundlage der automatisierungstechnischen Bausteinarchitektur. Die Funktionsbausteinsprachen der Normen IEC61131 und IEC61499 basieren auf diesem Konzept.

6.1.1 Typentwicklung

Bei typisierten Softwarekomponenten wird in einem ersten Schritt der Komponententyp entwickelt und realisiert. Dies kann mit beliebigen Sprachen und Entwicklungstools erfolgen. Im Rahmen von BaSys wird der Prozess der Typentwicklung nicht betrachtet. Er erfolgt in der Regel vorab und offline. Im Fokus von BaSys steht das Ergebnis der Typentwicklung. Ergebnis der Typentwicklung sind auf die Laufzeitsysteme der BaSys40-Komponenten-Systemplattform ladbare Elemente, die alle Anforderungen qualitativer und funktionaler Art, die an eine BaSys40-Komponente gestellt werden, erfüllen. Zur Gruppierung von Komponententypen können dabei Bibliotheken genutzt werden. Im BaSys40-Konzept wird diesen Bibliothekselementen vertraut, d.h. es wird angenommen, dass sie die Eigenschaften die sie zusichern auch tatsächlich erfüllen. Prozesse zur Sicherung dieser Eigenschaften sind außerhalb des Scopes von BaSys40.

BaSys40-Festlegung:

Ergebnis der Typentwicklung sind verifizierte und auf die Laufzeitsysteme der BaSys40-Komponenten-Systemplattform ladbare **Bibliothekselemente**.

Der **Integrität** der Bibliothekselemente wird prinzipiell vertraut.

Festlegung 6.2

Der Ladevorgang kann bei einfachen Laufzeitsystemen nur offline, in flexibleren Laufzeitsystemen auch online erfolgen (siehe nächster Abschnitt).

6.1.2 Das Typ-/Instanzkonzept in der Laufzeitumgebung

Das Typ-/Instanzkonzept gibt es sowohl als reines offline Engineering-Paradigma, als auch als online-Konzept zur dynamischen Verwaltung von Funktionsbausteinen im Zielsystem. Als Zielsystem wird die Laufzeitumgebung bezeichnet, welche die BaSys40-Komponenten-Systemplattform für diese Komponenten bzw. Komponententypen verwaltet. Zur dynamischen Verwaltung von Komponenten wird das Typ-/Instanzkonzept in der online-Variante benötigt. Im Folgenden ist mit dem Typ-/Instanzkonzept daher immer die online-Variante gemeint.

BaSys40-Festlegung:

Die BaSys40-Komponenten-Systemplattform unterstützt die **Verwaltung** von Komponententypobjekten und Komponenteninstanzen nach dem Typ-/Instanzkonzept im Zielsystem.

Festlegung 6.3

Das Typ-/Instanzkonzept sieht die Existenz von im Zielsystem geladenen Typobjekten vor, aus denen sich dynamisch Instanzen bilden lassen. Bei der Instanziierung bleiben die Metadaten und die Methoden Teil des Typs. Die Instanzen referenzieren ihren Typ und benötigen zur Laufzeit ständig Zugriff, sowohl auf die Methoden, als auch auf die Metadaten. In einem verteilten Echtzeitsystem ist es daher nicht möglich, ein zentrales Typobjekt im System zu verwalten, auf das dann alle Instanzen systemweit zugreifen. Vielmehr muss jede Instanz in ihrer lokalen Laufzeitumgebung direkt auf ihr Typobjekt zugreifen können.

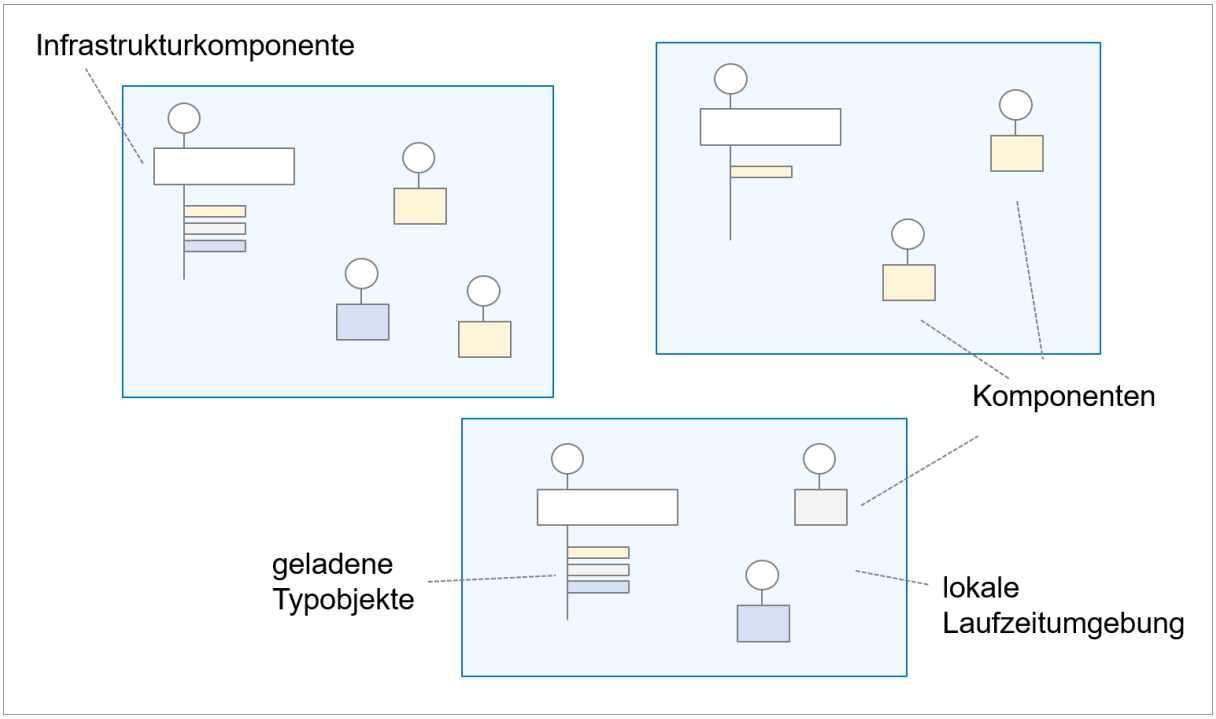

Abbildung 6.2: Lokale Laufzeitumgebungen mit geladenen Typobjekten und Komponenteninstanzen

Dies bedingt, dass ein Typobjekt in jeder Laufzeitumgebung in der sich eine seiner Instanzen befindet geladen sein muss (Abbildung 6.2).

BaSys40-Festlegung:

Jede Komponente muss auf ein **Typobjekt** ihres Typs in ihrer lokalen Umgebung direkt zugreifen können.

Von einem Typ muss in jeder lokalen Umgebung in der sich eine seiner Instanzen befindet ein Typobjekt geladen sein.

Festlegung 6.4

Die Anforderung an die lokale Verfügbarkeit der Typobjekte hat strukturelle Konsequenzen:

- Jeder Typ im System wird nicht durch ein Typobjekt, sondern durch eine Vielzahl von Typobjekten in den verschiedenen lokalen Laufzeitumgebungen repräsentiert.
- Für verschiedene Laufzeitumgebungen sind die Typobjekte eines Typs verschieden (z.B. abhängig vom zugrundeliegenden Betriebssystem)

BaSys40-Festlegung:

Aus Sicht des Komponentenhandlings können die **Fähigkeiten eines Laufzeitsystems** zur Unterstützung des Typ-/Instanzkonzepts in folgende Variabilitätsstufen klassifiziert werden:

0. Keine Variabilität:
   das dynamische Anlegen und Löschen von Komponenten ist nicht möglich.
1. Variabilität mit fest vorgegebener Typbibliothek:
   im Zielsystem sind bestimmte Typobjekte fest geladen. Diese geladene Bibliothek ist nicht veränderbar.
   Von geladenen Typobjekten können dynamisch Komponenten instanziiert werden.
2. Variabilität mit dynamisch ladbarer Typbibliothek:
   das Zielsystem erlaubt das dynamische laden von Typobjekten.
   Von geladenen Typobjekten können dynamisch Komponenten instanziiert werden.

Festlegung 6.5

In BaSys40 werden die Fähigkeiten eines Laufzeitsystems durch eine Infrastrukturkomponente abgebildet. Diese Infrastrukturkomponente ist Teil der Komponenten-Systemplattform und stellt die entsprechenden Dienste zum Anlegen und Löschen von Komponenten zur Verfügung.

In der Variabilitätsstufe 2 muss die Infrastrukturkomponente zusätzlich Dienste zur Verwaltung der geladenen Typobjekte bereitstellen.

BaSys40-Festlegung:

In BaSys40 werden die Fähigkeiten jedes lokalen Laufzeitsystems durch eine eigene **Infrastrukturkomponente** abgebildet.

Die Infrastrukturkomponenten sind Teil der Komponenten-Systemplattform und stellen Dienste zum

- Anlegen und Löschen von Komponenten, sowie deren Verbindungen

und in Variabilitätsstufe 2 zum

- Laden und Entladen von Typobjekten

zur Verfügung.

Festlegung 6.6

6.2 Bildung von BaSys40-Softwarekomponenten durch Kopieren

Das Typ-/Instanzkonzept ist ein zentrales Konzept der BaSys40-Komponentenverwaltung. Es ist aber nicht das einzige Konzept. In BaSys können auch Komponenten verwaltet werden, die nicht diesem Konzept unterliegen. So können auch Komponenten verwaltet werden, die nicht auf ein externes Typobjekt angewiesen sind, sondern sämtliche Ablauffähigkeiten in der Instanz in sich enthalten. Sie können ihren Code z.B. in interpretierbarer Form besitzen und sämtliche erforderlichen Metadaten selbst verwalten. Das heißt, zu diesen Komponenten gibt es im System kein separat verwaltetes Typobjekt. Solche Komponenten werden nicht instanziiert, sondern von einer Vorlage kopiert. Dazu wird eine bereits vorhandene Komponente als Vorlage genommen und mit allen Parametern und Elementen komplett 1:1 kopiert und nur mit

einer neuen ID versehen. Diese Art der Komponentenbildung stellt dabei besondere Anforderungen an das Variantenmanagement. Als Beispiel könnte ein Funktionsbausteinnetzwerk kopiert und der Name des kapselnden Funktionsblocks angepasst werden.

6.3 Bildung von BaSys40-Softwarekomponenten durch Konstruktion nach Vorlage

BaSys40-Komponenten können aus einer vorliegenden formalen Beschreibung in einer Interpreter-Sprache, die der Komponenten-Systemplattform bekannt ist, generiert werden. Grundlage sind Standardbibliotheken elementarer Baustein- und Funktionstypen, die den Laufzeitumgebungen des Systems bekannt sind und die von den Laufzeitumgebungen dynamisch instanziiert werden können. Typische Vertreter solcher Bibliotheken sind z.B. die IEC 61131-Bibliothek (Norm IEC 61131-3) oder die VDI/VDE-Bibliothek (Regelwerk VDI, VDI-Richtlinien VDI/VDE 3696 Blatt 2). Als Interpretersprache bietet sich z.B. die Sprache ST an. Eine weitere Möglichkeit ist das von der PLCopen vorgeschlagene XML-Schema (PLCopen Technical Committee 6 2009) bzw. das XML Austauschformat der IEC 61131-10. Für IEC 61499 konforme Automatisierungssysteme bietet sich das entsprechende XML Format der IEC 61499-2 an.

BaSys40-Festlegung:

Aufbauend auf den **Sprachen** der Automatisierungstechnik IEC61131 bzw. IEC 61499 werden für BaSys40 einheitliche Sprachen entwickelt, mit denen sich alle Aspekte einer BaSys40-Komponente zur Prozessführung formal beschreiben lassen. (D-PC2.3)

Dazu gehört auch die Festlegung einer **Standardbibliothek** von elementaren Bausteinen und Funktionen.

Festlegung 6.7

Durch die Konstruktion entsteht eine White-Box Komponente, deren interner Aufbau im Laufzeitsystem explizit zu sehen ist und prinzipiell auch dynamisch geändert werden kann. Damit ergibt sich die Möglichkeit der Realisierung einer komponenteninternen Variabilität, wie sie im nächsten Kapitel diskutiert wird.

Für die Ausführung der Konstruktion gibt es verschiedene Möglichkeiten. Entweder wird die Ausführung durch einen externen Ausführungsagenten (Z.B. Infrastrukturkomponente, Factory) abgewickelt und kontrolliert - in diesem Fall erfolgt die Implementierung durch einzelne Dienstaufrufe zum Anlegen einzelner Elemente an die Infrastrukturkomponente -, oder die Laufzeitumgebung ist selbst in der Lage die formale Beschreibung auszuwerten und die entsprechende Struktur zu generieren.

In jedem Fall muss sichergestellt sein, dass die Komponente vollständig und in sich konsistent generiert wurde.

7 Wandelbarkeit der internen Komponentenfunktionalität

Die flexible Handhabung von Komponenten ist der Hauptmechanismus der Wandelbarkeit in BaSys40. Zusätzlich besteht die Möglichkeit, die Funktionalität der Komponenten selbst an neue Aufgabenstellungen anzupassen. Dies ist eine zusätzliche Fähigkeit einer Komponente, die nicht jede BaSys40-Komponente und nicht jede BaSys40-Komponenten-Systemplattform unterstützen muss. Primär ist eine BaSys40-Komponente eine gekapselte Einheit mit definiertem äußerem Verhalten. Ihr interner Aufbau ist im Allgemeinen nicht sichtbar. Um die innere Funktionalität einer Komponente wandelbar zu gestalten muss

- die Funktionalität der Komponente in formaler Form beschrieben sein, oder sich aus ihrer White-Box Struktur explizit erkennen lassen,
- die Möglichkeit bestehen, diese Struktur zur Laufzeit zu verändern,
- ein Konzept existieren, wie bestimmte äußere Eigenschaften angepasst werden können, ohne andere zu verändern.

Prinzipiell sind auch typisierte Komponenten wandelbar. In diesem Fall ist die Änderung außerhalb des Scopes von BaSys40. Für BaSys40 wäre ein gewandelter Typ einfach ein neuer Typ, oder zumindest eine neue Version eines bestehenden Typs.

Im Rahmen von BaSys40 ist jedoch insbesondere die Wandelbarkeit von konstruierten White-Box-Komponenten von Interesse. Diese erlauben eine Änderung der Funktionalität in der Laufzeit unter Kontrolle der BaSys40-Komponenten-Systemplattform.

7.1 Design-Patterns der internen Komponentenstruktur

Um Änderungen der Funktionalität so gestalten zu können, dass gezielt bestimmte Eigenschaften und Fähigkeiten einer Komponente ergänzt oder verändert werden und andere unberührt bleiben, ist es hilfreich die Komponente intern modular zu strukturieren. Dies kann nach verschiedenen Entwurfsmustern geschehen.

Für BaSys40 wird das in Abbildung 7.1 dargestellte Grunddesign vorgeschlagen. Die Funktionalität der Komponente ist in dem blauen Rahmen dargestellt. Dieses Grunddesign wird in Abschnitt 8.2 und 9 detaillierter ausgeprägt.

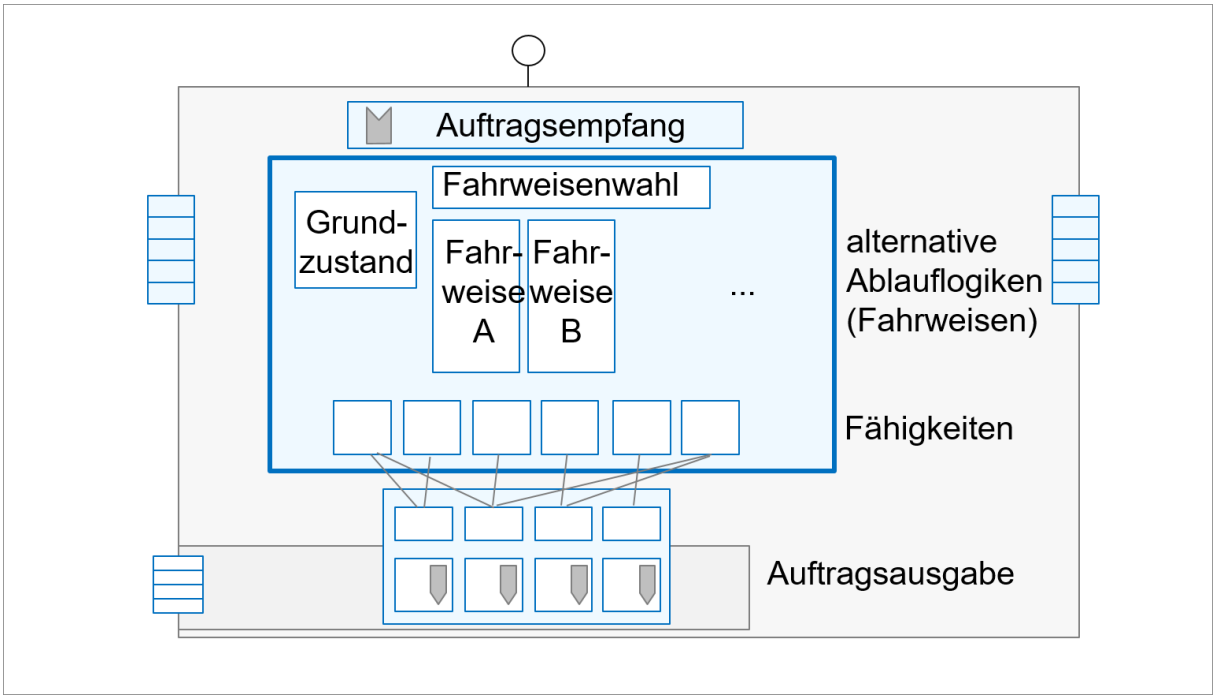

Abbildung 7.1: Beispiel für eine modular strukturierte Komponentenlogik

Die unterhalb der Fahrweisen aufgeführten Elemente repräsentieren die (Basis-)fähigkeiten, die diese Komponente auf Grund ihrer Einwirkungsmöglichkeit auf unterlagerte Einheiten oder Aktoren besitzt. Im einfachsten Fall besteht die Fähigkeit darin, eine Stellgröße auszugeben und damit auf den Prozess einzuwirken. Es können jedoch auch komplexere Fähigkeiten definiert werden, die z.B. zwei Aktoren synchron ansteuern. Fähigkeiten stehen den überlagerten Ablaufstrukturen (Fahrweisen) im Prinzip parallel zur Verfügung und können auch prinzipiell parallel genutzt werden. Das geht dann nicht, wenn zwei Fähigkeiten das gleiche Ausgabeelement benötigen. Die dazu notwendige gegenseitige Blockierung wird unter den Fähigkeiten lokal geregelt. Dazu können Verriegelungs-Logiken zum Einsatz kommen, wie dies in Abschnitt 9.2.1 beispielsweise für die Prozessführungskomponente vorgesehen ist.

Die über den Fähigkeiten operierenden Ablauflogiken werden als Fahrweisen bezeichnet. Sie können als orchestrierte bzw. höherwertige Fähigkeiten angesehen werden, welche die Komponenten den Nutzern des BaSys40-Dienstsystems zur Verfügung stellen. Zur besseren Abgrenzung, wird im Folgenden nicht von höherwertigen Fähigkeiten, sondern von Fahrweisen bzw. Ablauflogiken gesprochen. Diese werden in BaSys40 immer alternativ betrieben. Das heißt, zu jedem Zeitpunkt ist eine Komponente in genau einer Fahrweise oder im Grundzustand. Neben einer Fahrweise können, wie in Abschnitt 4.1.3.1 beschrieben „parallel“ BaSys40-Dienste bzw. Operationen von Komponenten angeboten und ausgeführt werden. Diese BaSys40-Dienste könnten auch dazu genutzt werden parallele Ablauflogiken zu realisieren. Der wesentliche Nachteil besteht darin, dass dann der Zugriff auf deren Ressourcen (Basisfähigkeiten) geregelt und Querverriegelungen eingebracht werden müssen. Weiter erschwert dies die Bestimmung des gesamt Zustandes des Komponente, z. Bsp. was den Fehler oder Betriebszustand betrifft. Unter dem Gesichtspunkt der Wandelbarkeit erschwert dies die Änderung der internen Komponentenstruktur. Daher wird hier kein weiteres Grunddesign mit parallelen Ablauflogiken betrachtet.

Je nach Anwendungsszenario können Ablauflogiken auch zwischen verschiedenen Komponenten aufgeteilt werden.

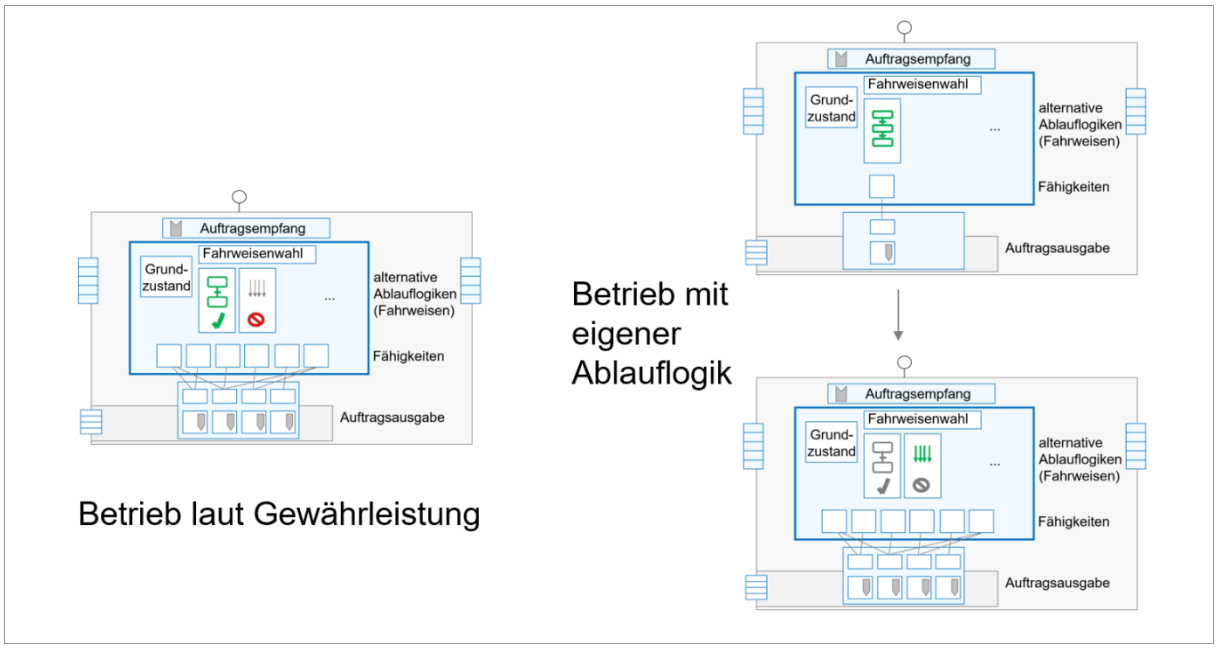

Abbildung 7.2: Beispiel für eine Aufteilung der Ablauflogik in zwei Komponenten

In Abbildung 7.2 ist beispielhaft dargestellt, wie die Ablauflogiken flexibel auf verschiedene Komponenten aufgeteilt werden können. In diesem Anwendungsbeispiel liefert der Hersteller eine Maschine mit einer geprüften und von ihm garantiert funktionierenden Ablauflogik aus. Diese Situation ist auf der linken Seite dargestellt. Der Betrieb mit dieser ausgelieferten Ablauflogik erfolgt im Rahmen der Gewährleistung. Zusätzlich stellt der Hersteller dem Anwender eine zweite Fahrweise zur Verfügung, mit der er direkt die Fähigkeiten und Ausgänge und damit die Aktoren der Maschine ansteuern kann. Der Betrieb dieser Fahrweise liegt außerhalb der Gewährleistung. Hier könnte z.B. eine unzulässige Aktor Konstellation angefahren werden. Die Verhinderung unzulässiger Zustände obliegt nun dem Anwender. Dieser kann in dieser Fahrweise nun jedoch seinen eigenen, für ihn optimaleren Ablauf realisieren. Diesen hat er in einer eigenen überlagerten Komponente hinterlegt die nun in der unterlagerten Komponente nur noch über die „leere“ Fahrweise die Aktoren ansteuert.

Bis zu diesem Punkt ist die Komponente selbst unverändert. Die Nutzung der alternativen Fahrweise war in der Komponente von vornherein vorgesehen und implementiert. Dieser Mechanismus, um Basisfähigkeiten für das BaSys-Dienstsystem verfügbar zu machen, verbessert die Universalität der Komponenten und ist somit ein wesentlicher Wandlungsbefähiger.

7.2 Wandelbarkeit

Die Wandelbarkeit einer Komponente selbst ist eine zusätzliche Anforderung. Im Allgemeinen besitzen Komponenten diese Eigenschaft nicht. Bei einer wandelbaren Komponente muss es möglich sein, einer installierten Komponente neue Funktionalitäten hinzuzufügen und die interne Struktur der Komponente dynamisch zu rekonfigurieren. Dies stellt erhebliche Anforderungen an den inneren Aufbau einer Komponente und an deren Verwaltung im Laufzeitsystem.

In der dargestellten modularen Struktur ist es nun jedoch relativ einfach möglich, zusätzliche Fähigkeiten und Fahrweisen dynamisch in die Komponente zu integrieren. Dies ist in Abbildung 7.3 dargestellt.

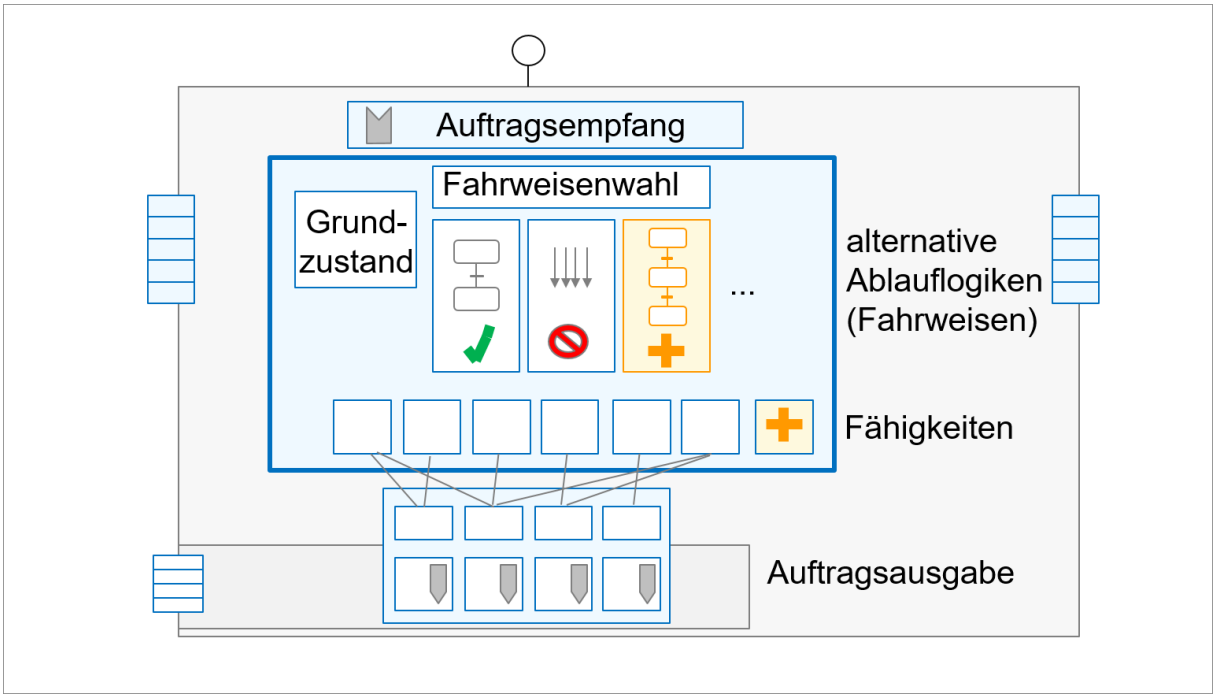

Abbildung 7.3: Integration von zusätzlichen Fähigkeiten und Fahrweisen in eine Komponente.

Durch die Erweiterungen werden die bestehenden Fahrweisen und das bestehende Verhalten der Komponente nicht verändert, die Komponente gewinnt nur zusätzliche Funktionalität.

Genauso ist es möglich einzelne bestehende Fahrweisen an neue Aufgabenstellungen anzupassen. Diese können gezielt durch neue Versionen ersetzt werden. Dies stellt natürlich Anforderungen an das Laufzeitsystem, die mittels der Variabilitätsstufen in Festlegung 6.5 klassifiziert werden können. Der Vorteil des modularen Designs ist die strukturelle Sicherstellung, dass durch die Änderung ausschließlich die betroffene Fahrweise geändert wird und alle anderen Fahrweisen unverändert weiter funktionieren. Weiter können zur Klassifizierung von BaSys40-Komponenten, welche diesem Grunddesign folgen, folgende Festlegungen zur Klassifizierung getroffen werden.

BaSys40-Festlegung:

Auf dem Grunddesign der internen Komponentenstruktur basierende BaSys40-Komponenten können verschiedene Variabilitätsstufen für die dynamische **Rekonfiguration von Fahrweisen** erfüllen:

0. Keine Variabilität:
   das dynamische Rekonfigurieren von Fahrweisen ist nicht möglich.
1. Parametrierung möglich:
   Fahrweisen können zur Laufzeit parametriert und damit auf eine andere Aufgabenstellung angepasst werden.
2. Hinzufügen möglich:
   Fahrweisen können zur Laufzeit hinzugefügt werden.
3. Änderung möglich:
   Fahrweisen können zur Laufzeit verändert werden.

Festlegung 7.1

BaSys40-Festlegung:

Auf dem Grunddesign der internen Komponentenstruktur basierende BaSys40-Komponenten können verschiedene Variabilitätsstufen für die dynamische **Rekonfiguration von Fähigkeiten** erfüllen:

0. Keine Variabilität:
   das dynamische Rekonfigurieren von Fähigkeiten ist nicht möglich.
1. Parametrierung möglich:
   Fähigkeiten können zur Laufzeit parametriert und damit auf eine andere Aufgabenstellung angepasst werden.
2. Hinzufügen möglich:
   Fähigkeiten können zur Laufzeit hinzugefügt werden.
3. Änderung möglich:
   Fähigkeiten können zur Laufzeit verändert werden.

Festlegung 7.2

Die Variabilitätsstufe der Komponente stellt dabei Anforderungen an das Zielsystem, also die Laufzeitumgebung in der sie ausgeführt wird. So können bestimmte Variabilitätsstufen der Komponente nur erreicht werden, wenn das Zielsystem die notwendigen Vorrausetzungen erfüllt.

7.3 Self-X-Fähigkeit

Unter dem Begriff Self-X fasst man alle Arten von Selbstverwaltungsfähigkeiten einer Komponente zusammen. Dazu gehören z.B.

- Selbstüberwachung
- Selbstheilung
- Selbstoptimierung
- Selbstkonfiguration
- Selbstadaption

Im klassischen Sinne handelt es sich bei einer Self-X-Fähigkeit um eine Fähigkeit, die eine Komponente selbst in sich besitzt (Abbildung 7.4). Im Sinne eines virtualisierten Plattformkonzepts kann man in Zukunft jedoch auch dann von Self-X-Fähigkeiten sprechen, wenn die Self-X-Funktionalität im Netz (z.B. in der Cloud) bereitsteht und von der Komponentenverwaltung z.B. über ihre Verwaltungsschale selbst aktivierbar sind (Abbildung 7.5).

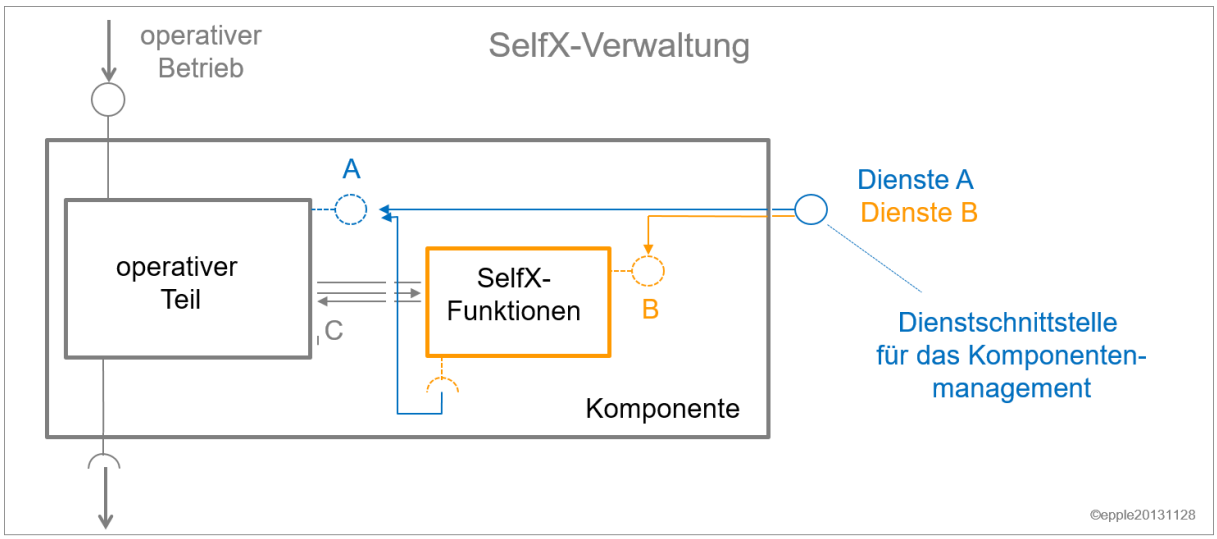

Abbildung 7.4: In Komponente integrierte Self-X-Funktionalität

Self-X-Fähigkeiten spielen in allen Arten von Komponenten eine Rolle. Sie sind insbesondere in den operativen Komponenten, wie z.B. den Prozessführungskomponenten von besonderem Interesse. Wie in den Bildern dargestellt kann man eine Self-X Funktionalität als selbständige, von der operativen Führungsfunktionalität getrennte Funktionalität verstehen. Die eigentliche operative Funktionalität ist auch ohne die Self-X-Funktionalität gegeben. Die Self-X-Funktionalität passt die Konfiguration und Parametrierung der operativen Funktionalität an die sich ändernden Gegebenheiten an. Dies kann über eine vorhandene Dienstschnittstelle für das Komponentenmanagement erfolgen (A) oder über eine interne spezielle API (Abbildung 7.4, C). Self-X-Funktionalitäten bieten selbst eine Schnittstelle an (B) über die sie aktiviert, gesteuert und konfiguriert werden. Diese Schnittstelle kann durch BaSys40-Dienste realisiert werden.

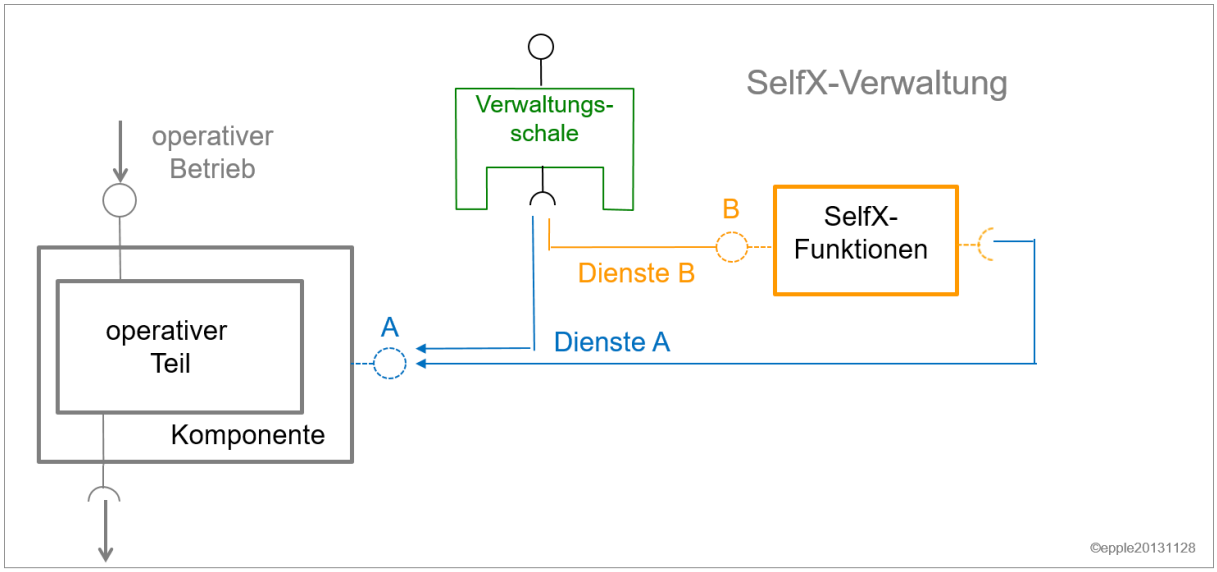

Abbildung 7.5: Im Netz realisierte Self-X-Funktionalität

8 Beschreibung von Komponenten

Um eine Komponente im Systemverbund sinnvoll einsetzen zu können, muss bekannt sein, welche Fähigkeiten diese Komponente besitzt und welche Restriktionen und Ressourcenbedarfe mit ihrem Einsatz verbunden sind. Die Beschreibung der Fähigkeiten und Dienste einer BaSys40-Komponente sind nicht im Scope dieses Deliverables. Dies wird in BaSys innerhalb der Fähigkeiten Gruppe behandelt. Das Metamodell der Komponente und ihres inneren Aufbaus werden jedoch hier bzw. in den Anwendungen in Abschnitt 9 beschrieben.

Dazu bedarf es einer entsprechenden Beschreibung der Eigenschaften einer Komponente. Auch wenn sich bestimmte Eigenschaften im Betrieb von einer aktiven Komponente selbst erkunden lassen, ist eine separate Beschreibung doch unverzichtbar. Auf der einen Seite reichen die Selbstbeschreibungsfähigkeiten im Allgemeinen nicht und sind auf der anderen Seite ja auch erst verfügbar wenn die Komponente schon eingebaut bzw. instanziiert ist.

8.1 Beschreibung des Komponententyps

Für die semantische Beschreibung betrachtet BaSys40 alle Komponenten als streng typisiert, das heißt, jede Komponente leitet sich mit ihren Eigenschaften von einem Typ ab. Der Typ bestimmt die funktionalen Eigenschaften der Komponente vollständig. In der Komponente selbst sind nur noch bestimmte Parameter und Zustände individuell ausgeprägt.

Für die semantische Beschreibung betrachten wir auch singuläre Komponenten, die es in ihrer Art nur einmal gibt, als typisiert und formulieren ihre Beschreibung als Typbeschreibung.

Die Typbeschreibung enthält Informationen zu allen wesentlichen Eigenschaften ihrer Komponenten wie Schnittstellen, Fahrweisen und Fähigkeiten. Dies gilt für alle Arten von Komponenten: Hardwarekomponenten, Softwarekomponenten, Smarte-Komponenten und Verbundkomponenten. Die Beschreibung selbst kann beispielsweise über eine Verwaltungsschale abgefragt werden.

BaSys40-Festlegung:

Jede BaSys40-Komponente ist von mindestens einem definierten Typ.

Zu jedem Komponententyp gibt es eine Komponententypbeschreibung.

Die Beschreibung von BaSys40-Komponententypen erfolgt nach einem standardisierten Metamodell.

Festlegung 8.1

8.2 Metamodell der Komponentenbeschreibung

Zur formalen Beschreibung von Klassenmodellen zeichnet sich UML als bekannte Modellierungssprache aus. Aufgrund des hohen Abstraktionsgrades des Modells ist eine Beschränkung auf Essential Meta Object Facility (EMOF) konformes UML (Spezifikation formal/2015-03-01) ausreichend. Ein Beispiel für die Anwendung und Erweiterung des Metamodels auf Prozessführungskomponenten findet sich in Abschnitt 9.2.

Ein Hauptaspekt der Komponente ist der Systemgedanke. Das Komponenten-Metamodell orientiert sich deshalb am Systemelement-Interface-Connection-Systemmodell (SIC-Modell (Spezifikation DIN SPEC 40912)), wie in Abbildung 8.1 und Abbildung 8.3 zu erkennen ist. Jede Komponente kann wiederum als Komponentensystem aufgefasst werden, sodass komplexe technische Problemstellungen durch Schachtelung der Komponenten gelöst werden können (Vergleiche Abschnitt 2.3 und Abschnitt 8.2.3). maximal einem Komponentensystem zugeordnet. Damit wird die Disjunktheit, Abgegrenztheit und Kompaktheit sichergestellt (Vergleiche Abschnitt 2.2).

Technische Komponentensysteme werden im Gegensatz zu natürlichen Systemen immer zu einem bestimmten Zweck geschaffen. Die Erreichung der Soll-Funktionalität wird dabei durch Pläne bestimmt. Das Kernmodell für Rollensysteme (Spezifikation DIN SPEC 40912) beschreibt diesen Kontext adäquat. Um eine hohe Flexibilität zu erreichen und Wandlungsfähigkeit zu ermöglichen, muss die Beziehung zwischen der Rolle und ihrer Realisierungseinheit, der Komponente, daher explizit modelliert werden. Die Rollenbeschreibung sollte möglichst zu dem Komponentensystem gehören, oder von der Komponenten-Systemplattform mitverwaltet werden.

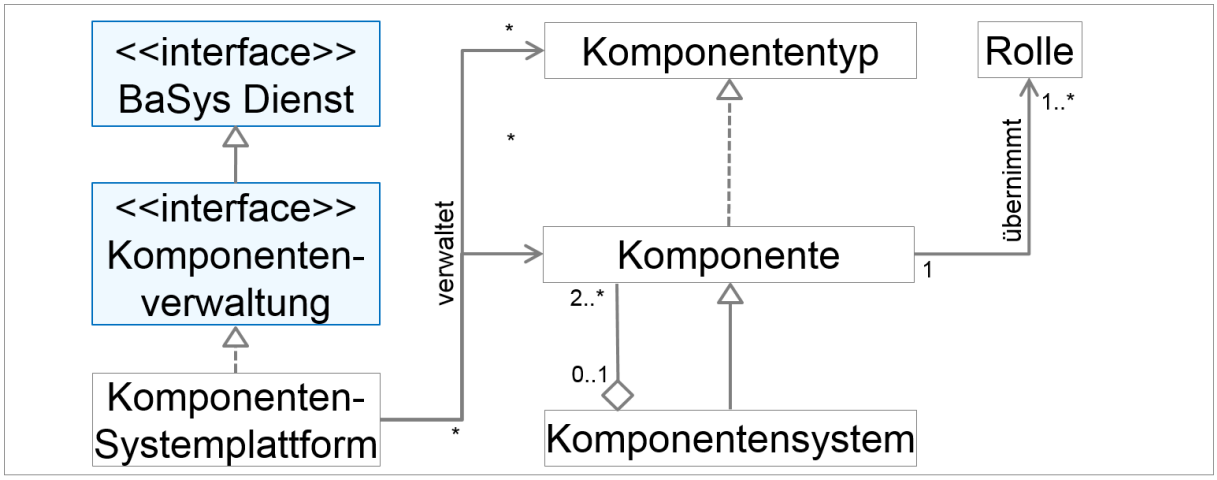

Abbildung 8.1: Komponente als Systemelement im Komponentensystem mit Verwaltung durch eine Komponenten-Systemplattform.

Die Spezialisierung in Hardware-, Software-, Smarte- und Verbund- Komponente nach „Art des digitalen Charakters“ kann durch eine zusätzliche Typisierung erreicht werden. Dies ist in Abbildung 2.1 bereits beschrieben. So kann beispielsweise eine Prozessführungskomponente sowohl als Software- oder Smarte-Komponente realisiert werden, während für beide die gleichen Entwurfsmuster und Steuerungsstrategien entwickelt werden können.

Um die Komponente als Einheit hantieren zu können, muss mindestens eine Komponenten-Systemplattform existieren. Diese kann selber eine Komponente sein oder eine Komponente zu diesem Zweck besitzen, was hier als Infrastrukturkomponente (vgl. Abschnitt 9.4) bezeichnet wird. Neben der Handhabung der Komponenten selber, übernimmt diese auch die Handhabung der Komponententypen. In Festlegung 6.6 wird für BaSys explizit die Existenz einer Infrastrukturkomponente gefordert. Innerhalb des BaSys40-Dienstsystems realisiert die Komponenten-Systemplattform über die Infrastrukturkomponente somit die BaSys40-Komponentenverwaltungsdienste. Dafür muss eine explizite Beschreibung der Komponentenverwal-

tungsdienste (genauer deren Signatur) in BaSys formuliert werden. Dies wird in BaSys in Deliverable D-PC2.15 abgehandelt. Die Komponenten-Systemplattform kann ebenfalls ein Startpunkt für die Verwaltung von Zugriffsrechten, Qualitätsmerkmalen, Discovery-Diensten, Identifikations-Diensten und weiterer komponentenbezogene Dienste sein.

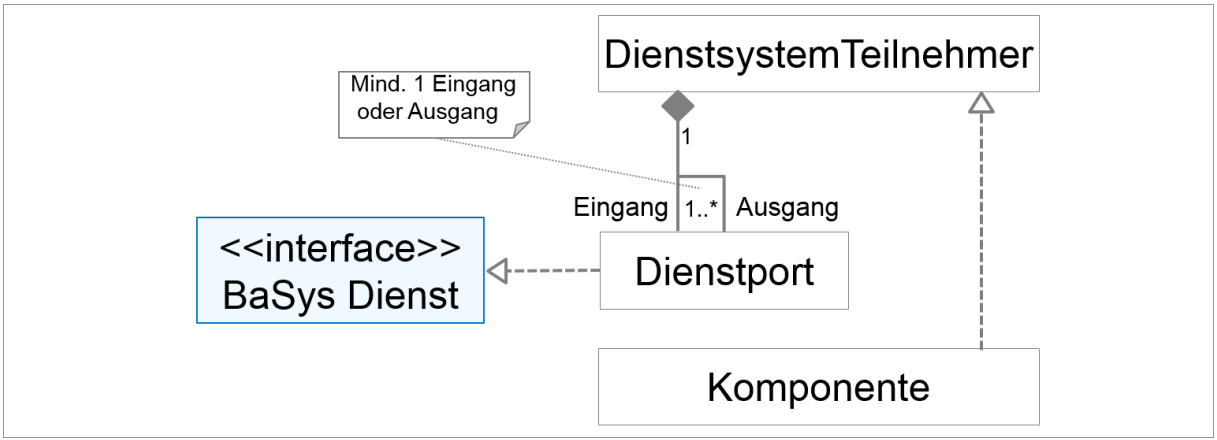

Abbildung 8.2: Die Komponente realisiert einen Dienstsystem-Teilnehmer im BaSys40-Dienstsystem.

Im Allgemeinen realisiert die Komponente einen Dienstsystem-Teilnehmer des BaSys40-Dienstsystems, wie in Abbildung 8.2 dargestellt ist. Dem SIC-Modell folgend, erbt die Komponente so die Möglichkeit Ein- und Ausgänge als dedizierte Dienstports zu besitzen. Diese Dienstports realisieren wiederum eine BaSys40-Dienstschnittstelle, welche die Dienstsignaturen festlegt. An den Dienstports können zukünftig beispielsweise auch QoS-Policies oder aktuelle Verbindungsdetails verwaltet werden.

Smarte-Komponenten bieten darüber hinaus die Möglichkeit, direkt über das I40-Netzwerk angesprochen zu werden.

8.2.1 Überblick und Einordnung des Komponenten-Metamodells

Im Wesentlichen können Komponenten zu BaSys-konformen Komponenten werden, wenn sie als BaSys40-Dienstsystemteilnehmer auftreten und alle dafür notwendigen Anforderungen erfüllen.

Dabei handelt es sich hauptsächlich um Schnittstellen-Definitionen. Dies ist in Abbildung 3.1 vereinfacht in einer Vererbungshierarchie dargestellt. Weiter ist die Anforderung an Smarte-Komponenten auch im I40-Netzwerk teilnehmen zu können. Dies entspricht der Anforderung CP-4X in der I40-Klassifikation (Spezifikation DIN SPEC 91345) von Assets und kann durch die Nutzung von Plattform I40-Standards erfüllt werden.

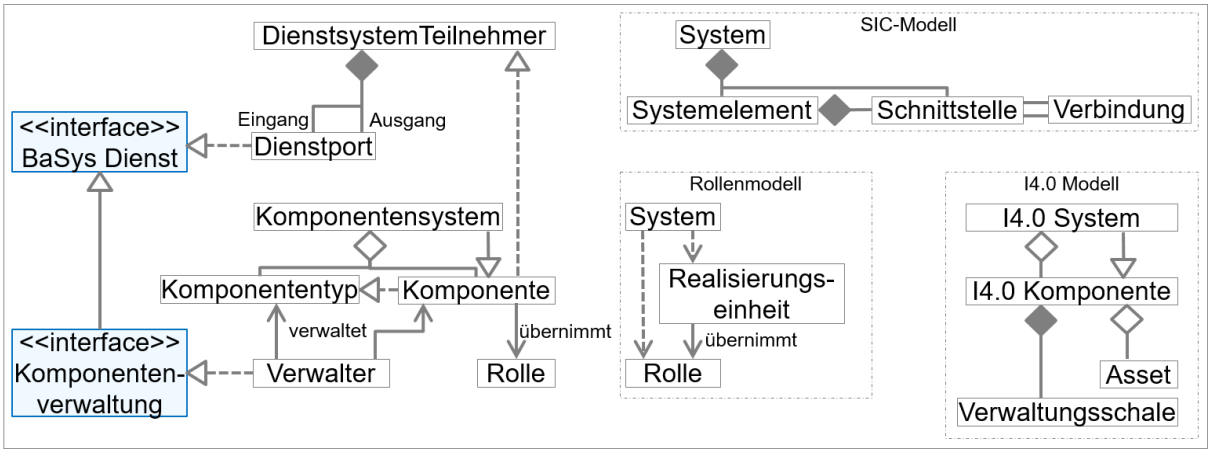

Abbildung 8.3: Gesamtüberblick der Komponente mit verwandten Modellen.

In Abbildung 8.3 sind nochmal die beiden wesentlichen Aspekte der Komponente gezeigt und zusammen mit den zugrunde gelegten, vereinfachten Strukturmodellen SIC-Modell und Rollenmodell abgebildet. Zusätzlich ist als Vergleich ein Modell der I40-Komponente abgebildet (Plattform Industrie 4.0 2016c, Abbildung 2). Weiter ist die Ähnlichkeit zum UML-Komponentenmodell und die Herleitung aus der Literatur in Abschnitt 2.1 beschrieben.

8.2.2 Einordnung in die BaSys-Metamodelllandschaft und -Referenzarchitektur

BaSys40-Komponenten können mit den einschlägigen Engineering Sprachen, welche in D-PC2.3 beschrieben und homogenisiert sind, entwickelt werden. Dafür baut das Metamodell der Komponente genauso, wie das Metamodell in D-PC2.3 auf dem SIC-Modell auf. So lassen sich die in der Industrie bewährten Elemente der IEC 61131 und IEC 61499 verwenden, was eine Grundvoraussetzung für den industriellen Einsatz ist. Die Abbildung von Komponentenverwaltungsdiensten und BaSys40-Diensten in beiden Sprachen ist daher ein weiterer wichtiger Schritt zur Umsetzung, welcher Beispielsweise durch das BaSys Sprint Team erfolgen und in den Demonstratoren validiert werden kann.

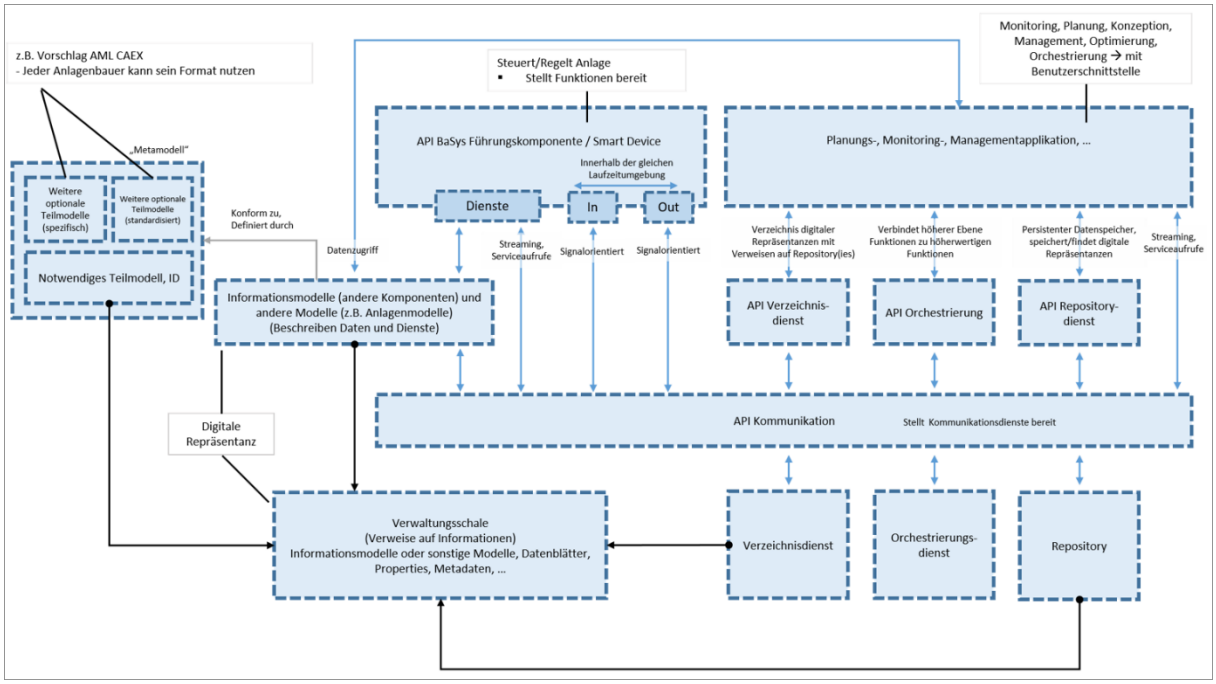

Abbildung 8.4: Einordnung in die BaSys40-Referenzarchitektur.

Weiter wird durch das BaSys-Fähigkeiten-Team sichergestellt, dass BaSys40-Komponenten ihre Fähigkeiten durch Dienste BaSys- bzw. Industrie 4.0-konform darstellen können. Dabei stehen vor allem die in Abschnitt 9.1 und 9.2 beschriebenen Einzelfunktionseinheiten und Prozessführungskomponenten im Fokus der Entwicklung, da diese operative Aufgaben in der Anlage übernehmen. Diese operativen BaSys40-Komponenten für Führungsaufgaben finden sich als „Führungskomponente“ im aktuellen Arbeitszustand der Referenzarchitektur des BaSys-Architekturteams wieder, wie in Abbildung 8.4 zu erkennen ist. Auch in einer erweiterten eher Funktionalen Sicht der BaSys40-Architektur, die in Abbildung 8.5 zu sehen ist, erscheinen die Prozessführungskomponenten als „Basic Device“ und „Group Device“ in der Geräteebene.

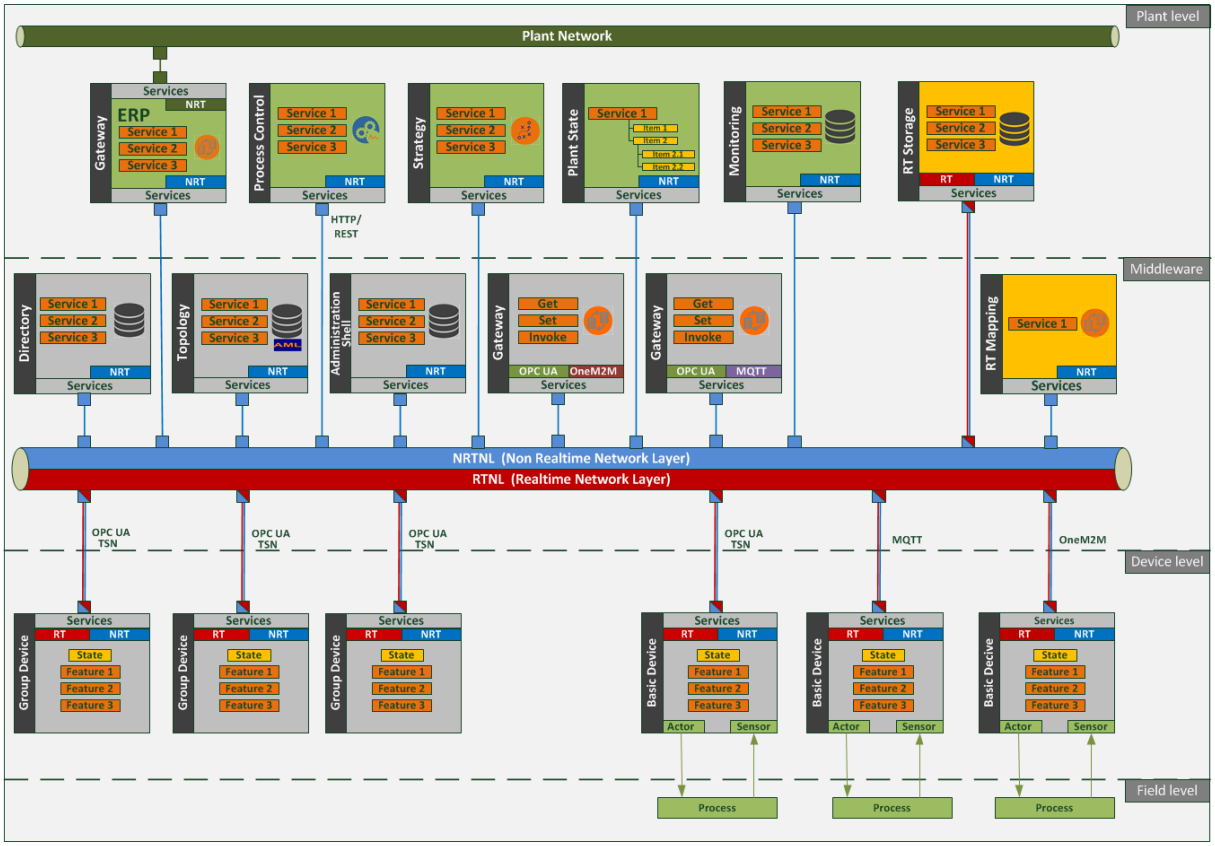

Abbildung 8.5: Sicht auf die BaSys40-Architektur.

8.2.3 Aggregation von BaSys40-Komponenten

Gerade wenn operative Aufgabenstellungen mit der hier beschriebenen Komponentenarchitektur gelöst werden sollen, kommt immer wieder die Fragestellung der Aggregations- bzw. Orchestrierungsmöglichkeiten von Komponenten auf. Dabei wird die Orchestrierung im Metamodell explizit durch die Teilnahme am BaSys40-Dienstsystem ermöglicht. Das Komponentendesign mit Auftragsstrukturen und den verschiedenen Zustandsautomaten unterstützt dabei die Lösung komplexer Führungsaufgaben. Je nach Aggregationsmuster kann die Komponentenarchitektur flexibler oder robuster entworfen werden.

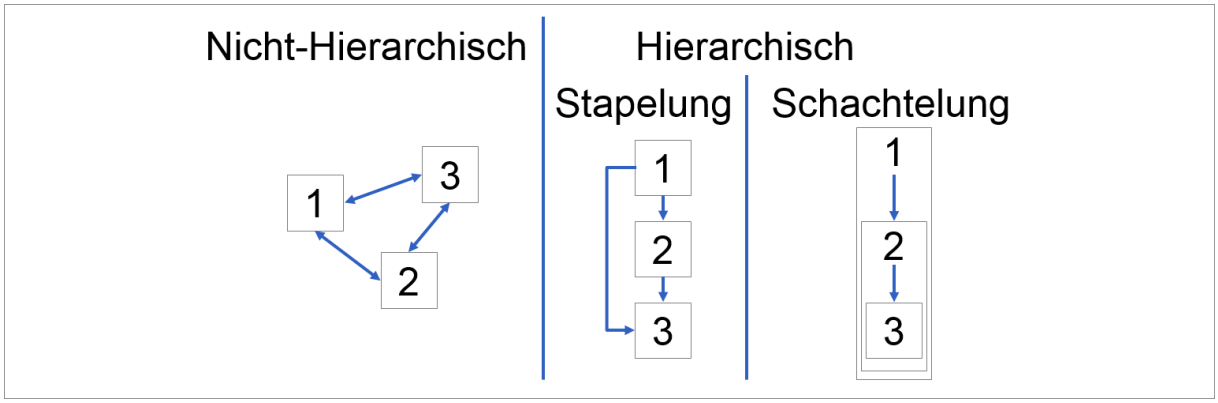

Abbildung 8.6: Aggregationsmuster für Komponenten

Für die Aggregation von Gegenständen finden sich grundsätzlich drei verschiedene Muster. Diese sind in Abbildung 8.6 schematisch gezeigt. Diese Muster werden wie folgt durch BaSys40-Komponenten unterstützt:

- Nicht Hierarchisch: verteilte Komponenten ermöglichen durch die einheitliche Dienstschnittstelle eine lose Kopplung verschiedener Komponenten, z.B. bei Agentensystemen.
- Hierarchisch
  - Stapelung: Über Belegungsautomaten kann sichergestellt werden, dass auch ohne Schachtelung robuste hierarchische Strukturen entstehen.
  - Schachtelung / Kapselung: Die Aggregation von Komponenten zu Komponentensystemen erlaubt deren Schachtelung.

8.3 Abbildung der Komponenten-Typbeschreibung in die Laufzeitumgebung

Die Typbeschreibungen sind modellmäßig in standardisierter Form im System so abgelegt, dass die Komponenten-Systemplattform auf sie jederzeit zugreifen kann. Die Typbeschreibungen bilden die Grundlage zur Verwaltung der Typmethoden und Komponenteninstanzen durch die Komponenten-Systemplattform. Sie ermöglicht ihr, z.B. die Fähigkeiten, Eigenschaften und Restriktionen einer neuen smarten Komponente zu erkennen und diese korrekt in den Systemverbund einzubinden.

Typbeschreibungen sind als Assets anzusehen. Sie können z.B. im Rahmen von Asset-Verwaltungsschalen im System realisiert werden. Das Beschreibungsmodell von BaSys40-Komponententypen ist standardisiert.

Gibt es zu einem Komponententyp mehrere Versionen, dann gibt es sowohl für den Typ selbst als auch zu jeder Version eine eigene Beschreibung.

BaSys40-Festlegung:

Die Verwaltung der Komponententyp-Beschreibungen im System erfolgt in eigenen Asset-Verwaltungsschalen.

Die Beschreibung erfolgt nach einem einheitlichen Muster für BaSys40-Komponententypen.

Festlegung 8.2

Die Verwaltung der Beschreibungen von Komponententypen und weiterer Asset bezogener Informationen kann selber auch durch Komponenten erfolgen. Diese werden als Verwaltungs-komponenten bezeichnet und in Abschnitt 9.3 genauer beschrieben.

9 Anwendungstypen der BaSys40-Komponenten

Zur einheitlichen Realisierung von Komponenten nach dem vorgestellten Metamodell und seinen Randbedingungen, werden in diesem Kapitel verschiedene exemplarische Anwendungen für Komponenten vorgestellt. Ein Beispiel für ein System, welches alle vorgestellten Anwendungstypen enthält ist in Abbildung 9.1 gezeigt.

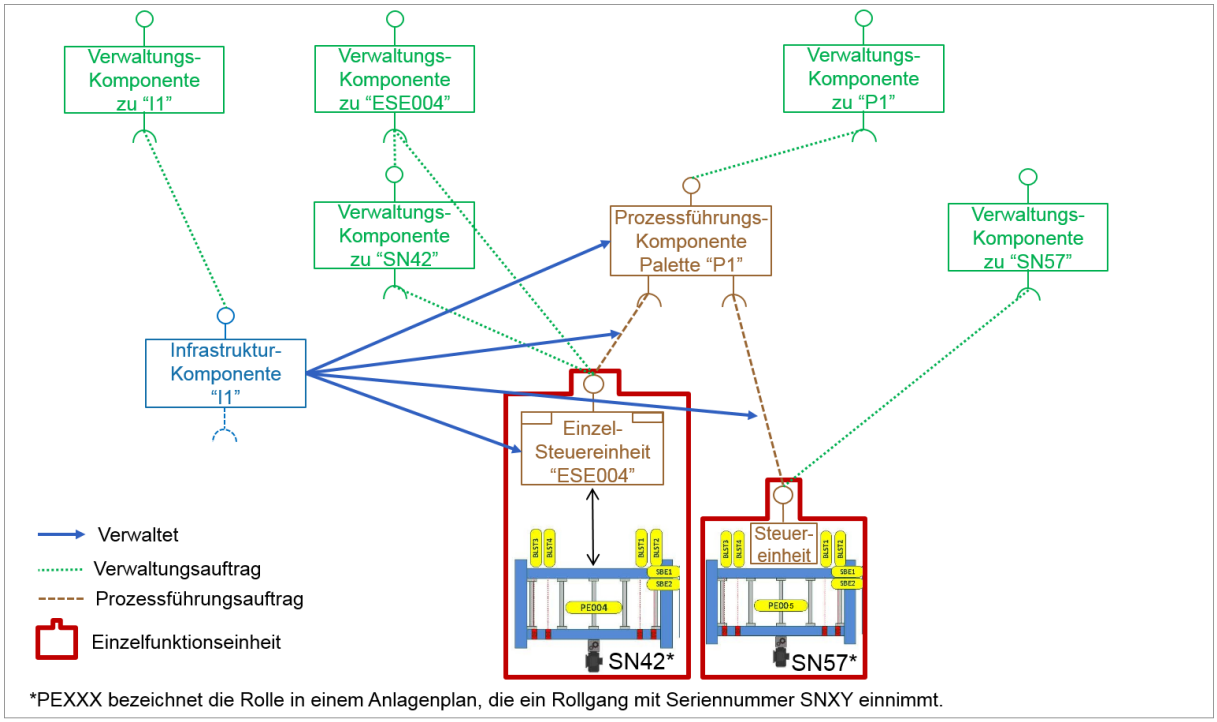

Abbildung 9.1: Anwendungstypen von BaSys40-Komponenten in einem System.

Im Beispiel sind zwei Module, eines als klassische und eines als smarte Hardware-Komponente gezeigt. Das klassische Hardware-Modul ist ein Rollgang mit Seriennummer „SN42“. Dieser Hardware-Komponente ist eine Einzelsteuereinheit „ESE004“ überstellt, welche gegen die Rolle im Plan „PE004“ arbeitet. Die Hardware-Komponente stellt, zusammen mit ihrer Einzelsteuereinheit, eine Einzelfunktionseinheit im BaSys40-Dienstesystem dar. Analog stellt die smarte Hardware-Komponente mit einer BaSys40-konformen Dienstschnittstelle eine Einzelfunktionseinheit im BaSys40-Dienstesystem dar. Dies ist der erste Anwendungstyp von BaSys40-Komponenten, welcher in Abschnitt 9.1 genauer beschrieben ist. Zur hierarchischen Prozessführung können diesen Einzelfunktionseinheiten Prozessführungskomponenten überlagert werden. Man bezeichnet diese daher auch als Gruppenführungseinheit. Im Beispiel ist eine Palettensteuerung „P1“ als Prozessführungskomponente gezeigt. Da Prozessführungskomponenten und Einzelfunktionseinheiten beide hauptsächlich der Prozessführung dienen, sind sie sehr ähnlich und können gleich strukturiert werden. Daher kann die Einzelfunktionseinheit als Spezialisierung der Prozessführungskomponente betrachtet werden. Auf Prozessführungskomponenten wird in Abschnitt 9.2 eingegangen.

Um alle notwendigen Informationen im BaSys40-Dienstsystem einheitlich zur Verfügung zu stellen, können Verwaltungskomponenten genutzt werden. Dabei können andere BaSys40-

Komponenten ebenfalls als Asset betrachtet und verwaltet werden. Die Frage welche Infrastrukturkomponente und damit welche Komponenten-Systemplattform für die Ressourcen zuständig ist, ist dabei eine Frage des Deployments. In Abschnitt 9.3 wird die Anwendung als Infrastrukturkomponente genauer beleuchtet.

Die Verwaltung und Bereitstellung der Komponenten, Kommunikationsmechanismen und weiterer Ressourcen in einer BaSys40-Komponenten-Systemplattform kann ebenfalls durch eine eigene BaSys40-Komponente übernommen werden. Dadurch werden die Dienste der Infrastruktur explizit im BaSys40-Dienstsystem verfügbar. Dieser Anwendungstyp wird als Infrastrukturkomponente bezeichnet und ist in Abschnitt 9.4 beschrieben.

Dabei werden folgende Zustandsautomaten als optional oder obligatorisch für die verschiedenen Anwendungstypen angesehen:

	Belegung	Betriebsart	Betriebs-zustand	Fahrweisen	Arbeits-zustand	Fehler-zustand	Asset-Zustand
Einzelfunktionseinheiten	+	0	+	+	0	0	0
Prozessführungskomponenten	+	0	+	+	0	0	-
Verwaltungskomponenten	-	-	-	-	-	-	+
Infrastrukturkomponenten	-	-	-	-	-	0	-

Tabelle 9.1: Obligatorische (+) , empfohlene (0) und optionale (-) Zustandsautomaten verschiedener Anwendungstypen.

9.1 Einzelfunktionseinheiten

Eine Einzelfunktionseinheit inklusive ihrer BaSys40-Dienstschnittstelle bildet insgesamt eine BaSys40-Komponente. Sie realisiert die Funktionalität einer anlagentechnischen Komponente. Es gibt verschiedene Ausführungsformen von Einzelfunktionseinheiten.

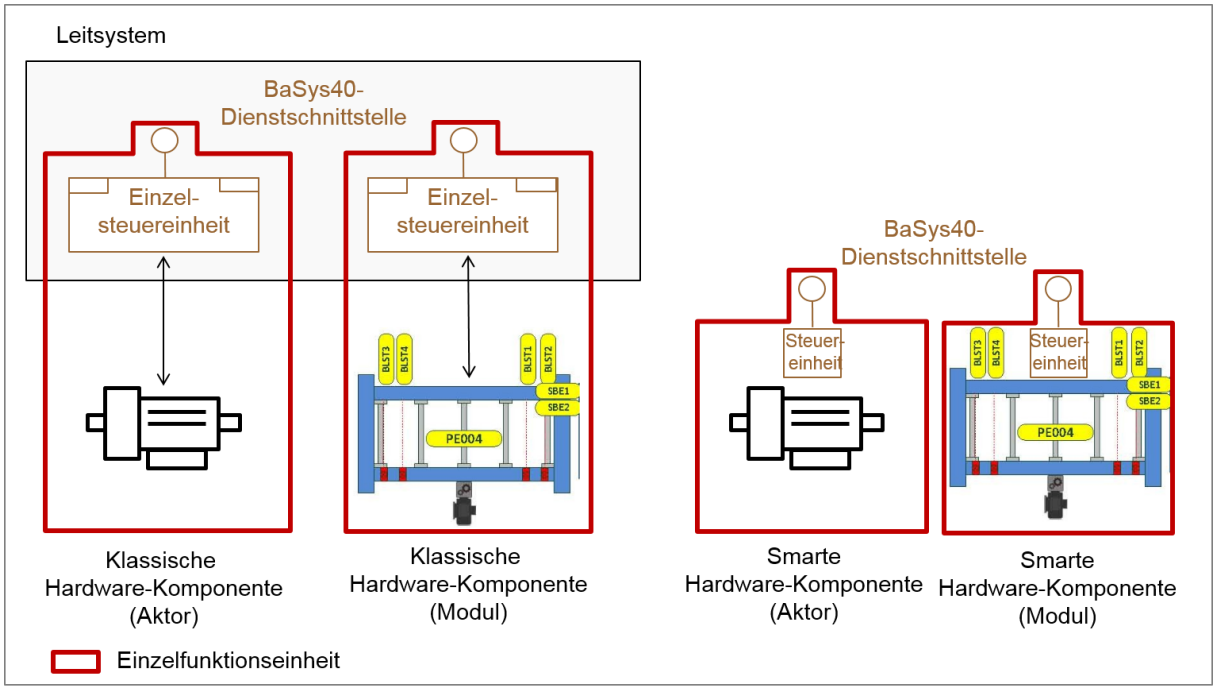

Abbildung 9.2 Ausführungsformen von Einzelfunktionseinheiten und Abgrenzung zu Einzelsteuereinheiten.

Eine Einzelfunktionseinheit umfasst immer sowohl die anlagentechnische Hardwarekomponente, als auch die zum Betrieb dieser Komponente notwendige Steuer-Software. Diese Steuerungskomponente bezeichnen wir als Einzelsteuereinheit. Als BaSys40-Komponente hat jede Einzelfunktionseinheit eine BaSys40-Dienstschnittstelle, die von der Einzelsteuerung zur Verfügung gestellt werden muss. Diese BaSys40-Dienstschnittstelle muss die, für BaSys40 standardisierten, Prozessführungsprotokolle unterstützen. Wie in Abbildung 9.2 dargestellt kann die Einzelsteuereinheit entweder auf der Hardwarekomponente selbst realisiert werden oder als getrennte Softwarekomponente im Leitsystem. Einzelfunktionseinheiten sind also typischerweise Verbundkomponenten aus einer oder mehreren Hardwarekomponenten und der Einzelsteuereinheit als Softwarekomponente.

Zu beachten ist, dass auch komplexe Devices, (Apparate, Anlagenmodule, ...) mit mehreren Aktoren und Sensoren als Einzelfunktionseinheiten anzusehen sind, solange Sie über eine gemeinsame Prozessführungs-Dienstschnittstelle verfügen. Einzelfunktionseinheiten können also sowohl einfachste Aktoren, als auch komplexe, aber kompakte Anlagenmodule sein.

Bei klassischen Devices, stellt die Handhabung als Einheit der Einzelsteuereinheit in Verbindung mit ihrem Device, eine besondere Herausforderung dar. Das Device könnte beispielsweise zur Wartung ausgebaut und dort in einer Wartungsbetriebsart getestet werden. Dabei muss die Einzelsteuereinheit entsprechend mit verschoben oder angepasst werden. Beispielsweise müssten nun andere Signale, Busadressen oder IP-Adressen verwendet werden. Hier sind Smart Devices im Vorteil, da die Einzelsteuereinheit physisch und logisch mitbewegt wird.

Einzelsteuereinheiten können analog zu Prozessführungskomponenten aufgebaut werden, welche im Abschnitt 9.2 genauer beschrieben sind. Die Anforderungen an Zustandsautomaten sind ansonsten gleich. Einzelfunktionseinheiten unterscheiden sich hauptsächlich dadurch, dass sie in der Regel keine Auftragsausgabe zur Steuerung des Devices sondern Signale nutzen. Signale können durch die Komponenten-Systemplattform aus verschiedenen Quellen

bereitgestellt werden. Als Beispiel sei die Ankopplung an einen Feldbus genannt. Wichtig ist auch hier, dass die Steuerung gegen die Rollen des Devices arbeitet und somit eine Trennung von funktionalen und technologischen Signalen entsteht. Diese Trennung kann einerseits durch das nach außen Führen der funktionalen Signale erreicht werden, was besonders für Black-Box Komponenten sinnvoll ist (Abbildung 9.3 links). Anderseits kann in einer Komponente ein Element (partielles Prozessabbild) zum Mapping der Signale und Zugriff auf die entsprechenden Ressourcen vorhanden sein (Abbildung 9.3 rechts). Dies ist bei der Auftragsausgabe mit den Rollen zu vergleichen.

Die Komponenten-Systemplattform, bzw. der Bediener muss durch die Parametrierung (R) sicherstellen, dass der Zugriff auf die Ressourcen gesichert ist. Beispielsweise dürfen nicht zwei Einzelsteuereinheiten dieselbe Rolle erfüllen. Die Einzelsteuereinheit bzw. das partielle Prozessabbild muss dabei rückwirkungsfrei von der Ressource lesen. Beide Varianten sind in folgendem Beispiel dargestellt:

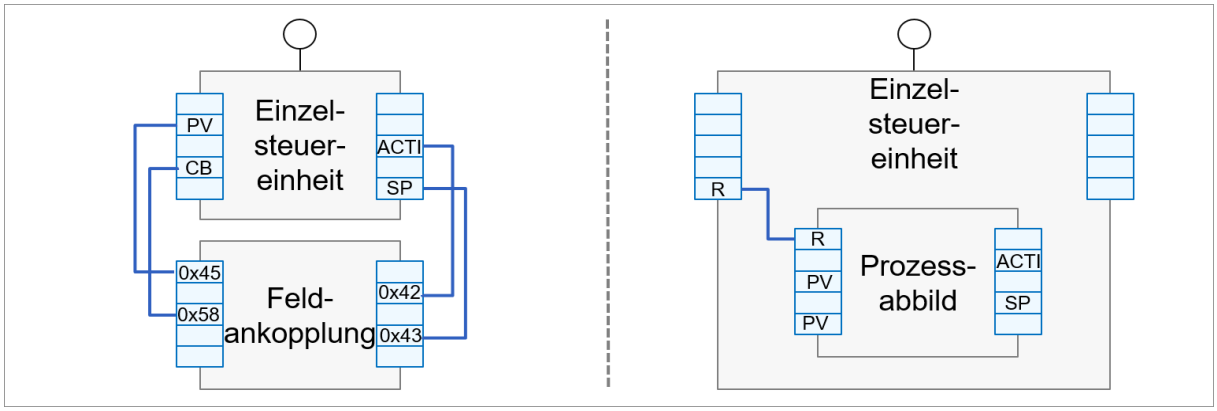

Abbildung 9.3 Beispiel für die Trennung von funktionalen und technologischen Signalen.

Im Beispiel stellen PV(ProcessValue), CB(CheckBack), ACTI(Activ), SP(SetPoint) die funktionalen Signale dar. Die Signale 0xXX stellen technologische Signale dar. Instanz-spezifisch ist im rechten Beispiel nur der Parameter R, welcher die Zuordnung zur Rolle darstellt. Diese Zuordnung kann natürlich auch, wie bei der Auftragsausgabe durch Konfigurationsaufträge über die Dienstschnittstelle geschehen.

Während die Beschreibung der Steuerung auf funktionaler Ebene in den Typen der Komponente gehalten wird, ist das technologische Mapping Instanz-spezifisch. So wird gleichzeitig die Kontextneutralität der Komponente sichergestellt (siehe Abschnitt 3.3). Bei Prozessführungskomponenten wird dieses Mapping durch die Auftragsausgabe realisiert (Vergleiche Abschnitt 4.4.2).

9.2 Prozessführungskomponenten

In der Automatisierungstechnik ist es üblich und verbreitet Softwarelösungen in speziellen Komponenten, so genannten Funktionsbausteinen, zu organisieren. Dieses Vorgehen stammt aus der Gerätetechnik, bei der Lösungen durch das zusammenschalten von einzelnen Geräten aufgebaut wurden. Die Geräte selbst realisierten einzelne Funktionalitäten und wurden mit elektrischen Verbindungen (Leitungen) rückwirkungsfrei verbunden. Im Zuge der voranschreitenden technischen Entwicklung wurden die Einzelgeräte als Softwarekomponenten realisiert

(Funktionsbausteine) und diese werden im Engineering mit Leitungen (Signalverbindungen) verknüpft.

Das weitere Zusammenwachsen und die immer komplexer werdenden Softwarearchitekturen, führten zur Einführung von Hierarchieprinzipien. Diese sind nötig, um die steigende Anzahl an Softwarekomponenten beherrschbar zu halten. Ein Typ der in diesem Zusammenhang entstandenen Komponenten sind Prozessführungskomponenten (Enste und Epple 1998; Enste 2001). Einzeln und im Verbund stellen diese Komponenten sicher, dass der Prozess geführt wird. Nach (Krämer et al. 2008) bedeutet Prozessführung, dass der Prozess innerhalb der technischen und wirtschaftlichen Randbedingungen gehalten wird.

Prozessführungskomponenten im Sinne von BaSys sind Softwarekomponenten. Sie steuern den Ablauf von Prozessen. Dazu greifen sie auf unterlagerte Komponenten zurück. Eine Prozessführungseinheit stellt eine Dienstschnittstelle zur Verfügung, über die sie die Ausführung von Aufträgen anbietet. Sie wickelt die Aufträge selbständig ab. Zur Ausführung vergibt sie Unteraufträge an andere Komponenten. In der Führungshierarchie sind dies unterlagerte Prozessführungseinheiten und letztendlich die Einzelfunktionseinheiten. Im Allgemeinen kann es sich aber auch z.B. um Verwaltungseinheiten oder Diagnoseeinheiten handeln.

9.2.1 Metamodell der Prozessführungskomponente

Viele Komponenten übernehmen innerhalb des Komponentensystems operative Aufgaben. Diese bestehen oftmals darin, andere Komponenten zu führen, oder die in ihnen gekapselten Fähigkeiten auszuführen. Um diese Aufgaben einheitlich innerhalb des BaSys40-Dienstsystems bedienen, entwickeln und beschreiben zu können, wird hier die Prozessführungskomponente als spezialisierte Form der Komponente definiert.

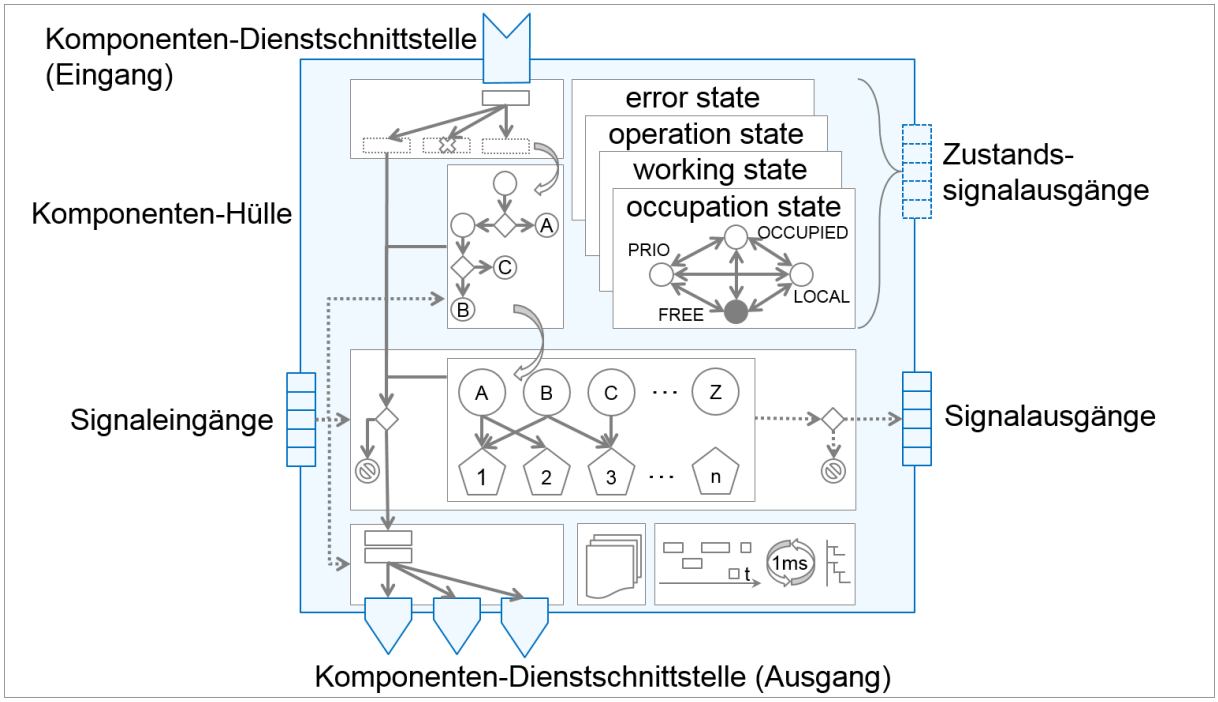

Abbildung 9.4: Schematische Darstellung des Aufbaus einer Prozessführungskomponente mit Erklärung der Außensicht

Als Vorlage dafür dient das Referenzmodell aus Abbildung 4.1 und die Design Pattern aus Abschnitt 7.1. Weitere Abbildungen der Struktur von Prozessführungskomponenten sind in der Literatur beispielsweise in (Enste und Epple 1998, Abbildung 3) und Enste 2001, Bild 7.29 zu finden. Diese wurden durch einen Steuerkopf erweitert, um den Anforderungen in der Fertigungsindustrie gerecht zu werden. So kann bei entsprechender Umsetzung, nicht nur eine Fahrweise ausgewählt und parametriert werden, wie es in der Prozessindustrie üblich ist. Vielmehr können verschiedene Fahrweisen kombiniert und angesteuert werden, um einen bestimmten Auftrag abzuarbeiten. Diese verallgemeinerte Form der Prozessführungskomponente ist schematisch in Abbildung 9.4 gezeigt.

Damit die Vielfalt der Realisierungen von Prozessführungskomponenten so wenig wie möglich eingeschränkt wird, kann in eine Außensicht und eine Innensicht der Prozessführungskomponenten unterschieden werden. Dies unterstützt den Ansatz von „Separation of Concerns“ und wurde bereits bei den Betrachtungen zur Wandelbarkeit in Abschnitt 6 und 7 so umgesetzt. Zusätzlich lässt sich durch die explizite Definition, der nach außen hin sichtbaren Eigenschaften der Prozessführungskomponente, die Disjunktheit und Abgegrenztheit als wesentliche Charakteristik sicherstellen. Somit tritt jede BaSys40-Komponente, die operative Funktionen ausführt nach außen als Prozessführungskomponente auf, wenn folgende Schnittstellendefinitionen erfüllt sind.

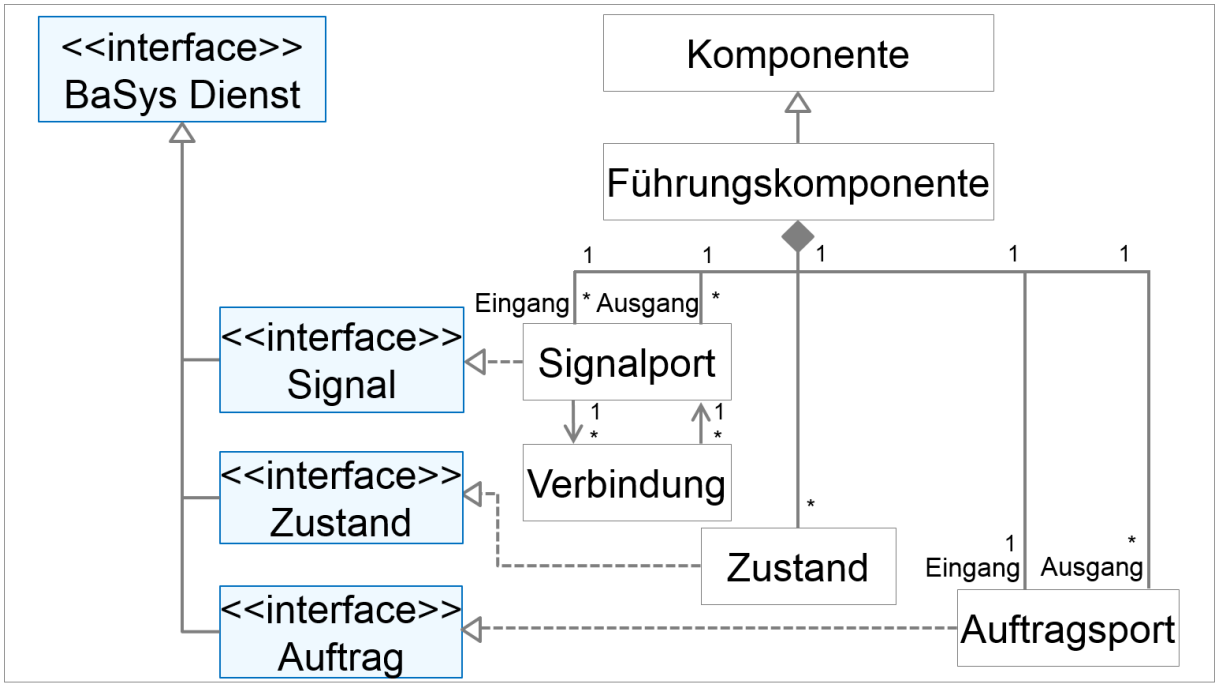

Abbildung 9.5: UML Modell der Außensicht der Prozessführungskomponente.

Zunächst folgen Prozessführungskomponenten einem auftragsgesteuerten Verhalten. Somit muss neben den von der Komponente vererbten Dienstports genau ein Auftragsport als Eingang definiert sein. Dieser realisiert die Auftrags-Dienste des BaSys40-Dienstsystems. Wenn es sich um eine Gruppensteuereinheit handelt, welche weiteren Prozessführungskomponenten Aufträge erteilt, benötigt diese ebenfalls Auftragsports als Ausgänge zum Verschicken dieser Aufträge.

Um ein ganzheitliches Prozessabbild für beispielsweise Optimierungsmaßnahmen oder überlagerte Steuerungen zu ermöglichen, muss der aktuelle Zustand der Prozessführungskomponente ebenfalls über das BaSys40-Dienstsystem oder über Signale zur Verfügung stehen. Vergleiche dazu Abschnitt 5. Die Zustände können dabei über Zustand-Dienste des BaSys40-Dienstsystems abgefragt oder beeinflusst werden. Zusätzlich können die Zustände, wie in Abschnitt 0 beschrieben, als Signale nach außen dargestellt werden.

Weiter können signalorientierte Ein- und Ausgänge definiert werden. Dies ist vor allem aus Kompatibilitätsgründen zu bestehenden Lösungen und für hoch performante, besonders sichere oder statische Verbindungen sinnvoll. Diese Art der Verbindung wird oft bei besonders prozessnahen Komponenten eingesetzt. Grundsätzlich werden die Komponenten dadurch stärker vernetzt, weshalb die Anzahl der Signale möglichst gering zu halten ist. Weiter dürfen die Signalverbindungen die Hülle des Komponentensystems nicht verlassen, da sonst die Abgegrenztheit und Kompaktheit des Komponentensystems oder der Komponente verletzt werden kann.

Zur Vereinheitlichung des Aufbaus von Prozessführungskomponenten ist die Innensicht der Prozessführungskomponente schematisch in Abbildung 9.6 und als UML Diagramm in Abbildung 9.7 gezeigt und im Folgenden beschrieben.

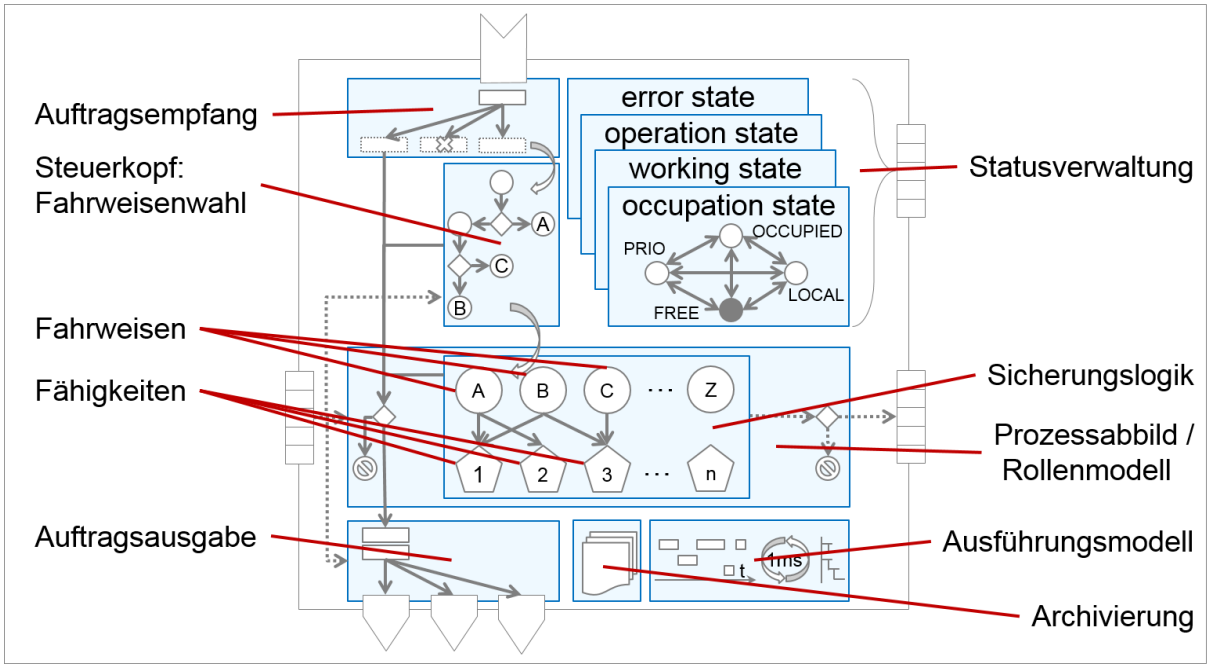

Abbildung 9.6: Schematische Darstellung des Aufbaus einer Prozessführungskomponente mit Erklärung der Innensicht.

Um im industriellen Einsatz operative Aufgaben zu übernehmen, benötigt eine Prozessführungskomponente verschiedene Zustandsautomaten. Damit wird festgelegt, wer wie welche Fähigkeiten bzw. Fahrweisen nutzt und in welchem Arbeitszustand sich die Komponente gerade befindet. Wie in Abschnitt 4.1.3 beschrieben benötigt jede Komponente zur Prozessführung einen Belegungszustandsautomaten. Dieser wurde bereits in Abschnitt 5.2 weiter ausgeführt. Damit benötigt eine Prozessführungskomponente Zustandsautomaten für folgende Zustände:

- Belegungszustand
- Betriebsart und Betriebszustand
- Fahrweisen
- Arbeitszustand
- Fehlerzustand

Jede Prozessführungskomponente muss über eine gewisse Belegungslogik verfügen, um die Auftragsannahme und Umsetzung zu realisieren. Dies stellt zudem sicher, dass immer nur eine überlagerte Steuerung gerade diese Komponente befehligt und eine hierarchische Prozessführung möglich wird. Diese Belegungslogik findet sich im Metamodell im Auftragsempfang wieder.

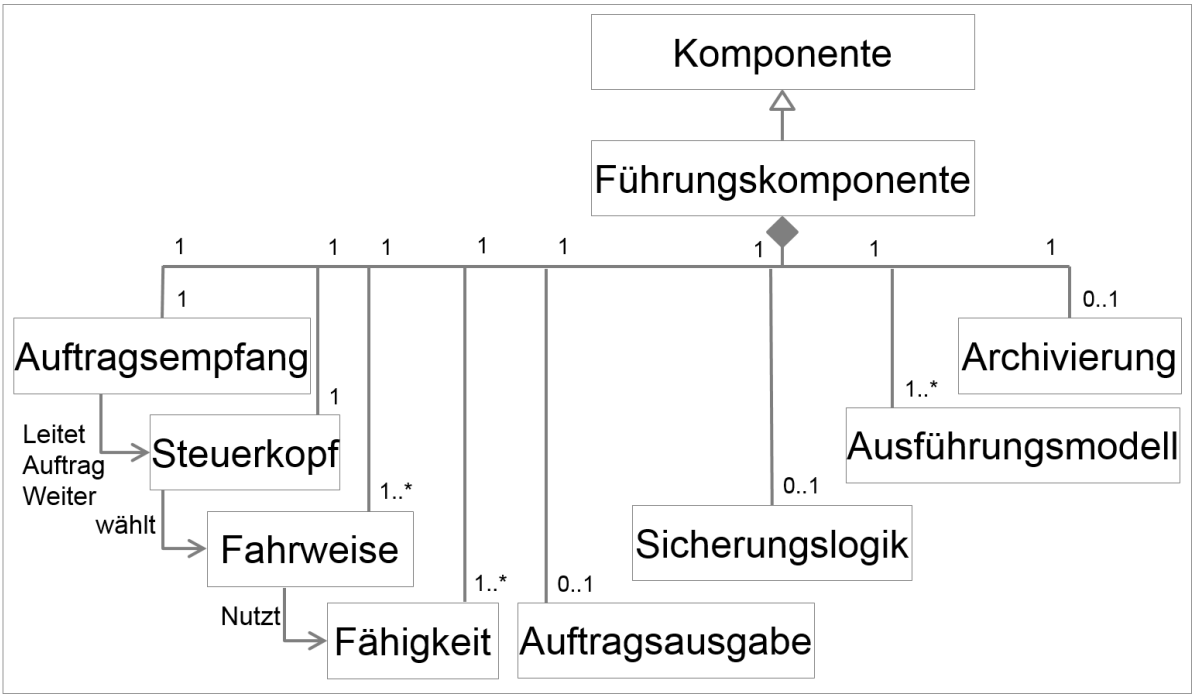

Abbildung 9.7: UML Darstellung der Innensicht der Prozessführungskomponente.

Im Steuerkopf der Prozessführungskomponente sind die Ablauf- bzw. Verknüpfungslogik hinterlegt. Im Falle der Prozessindustrie entspricht dieser Teil der Auswahl von Fahrweisen. Der Steuerkopf setzt somit die Aufträge um, die von der Belegungslogik zugelassen werden und die der Wahl der Fahrweise, oder der Anfrage zur Erfüllung von Fähigkeiten entsprechen. Dazu ist Steuerungscode in der Komponente hinterlegt, der die Fähigkeiten in ausführbarer Form realisiert. Die Fähigkeiten können durch Fahrweisen orchestriert bzw. zu höherwertigen Fähigkeiten zusammengefasst werden, was genauer in Abschnitt 5.4 und 7.1 beschrieben wird. Dabei ist zu betonen, dass die Komponente sich in exakt einer Fahrweise befindet, aber diese parallel verschiedene Fähigkeiten nutzen kann.

Um Signale und Aufträge an unterlagerte Einheiten verriegeln zu können, ist der Auftragsausgabe und den Signalausgängen eine Sicherungslogik zwischengeschaltet. Diese arbeitet mit den funktionalen Signalen und adressiert ebenso Rollen und nicht Realisierungseinheiten. Dies wird in Abschnitt 4.1.3 und für Einzelfunktionseinheiten in Abschnitt 9.1 detaillierter beleuchtet. So operieren die Fahrweisen und Fähigkeiten in einer kontextfreien und abgesicherten Umgebung, was die Wandlungsfähigkeit, Robustheit und Wiederverwendbarkeit wesentlich steigert.

Zur Weitergabe von Aufträgen an andere Komponenten, welche einen Auftragsport als Eingang besitzen, benötigen Prozessführungskomponenten eine Auftragsausgabe. Dieses legt beispielsweise Aufträge für unterlagerte Komponenten in einem Auftragsport ab. Dabei übernimmt die Auftragsausgabe auch die Zuordnung von Rollen zu Realisierungseinheiten, wie in Abschnitt 4.4 beschrieben ist.

Für viele Prozessführungskomponenten ist es sinnvoll Aufträge, Lebenszyklusereignisse und weitere komponentenbezogene Informationen zu archivieren, weshalb eine Archivierung opti-

onal vorgesehen ist. Hier gibt es in der Industrie, beispielsweise in der Lebensmittelverarbeitung, besondere Anforderungen, die durch eine dezentrale Archivierung oder Pufferung der Dokumentationsinformationen vereinfacht werden kann.

Weiter besteht eine Prozessführungskomponente aus einem Ausführungsmodell, in welchem festgelegt ist, wie die Bestandteile der Komponente ausgeführt werden. So könnten beispielsweise zyklische (Bsp. Tasklisten in IEC 61131) oder ereignisbasierte (Bsp. ECCs in IEC 61499) Ausführungen, Zeitpartitionierungen und Prioritäten festgelegt werden. Es können auch verschiedene Ausführungsmodelle hinterlegt werden, um die Flexibilität über Komponenten-Systemplattformen zu erhöhen. Man beachte dazu auch die Vereinheitlichung der Engineeringsprachen aus Deliverable D-PC2.3.

Zusammenfassend sind relevante Signal- und Ereignisflüsse in Abbildung 9.8 dargestellt. Zunächst entscheidet der Auftragsempfang anhand der Zustände der Komponente, wie der Belegungslogik, ob ein Auftrag abgelehnt wird. Falls nicht, kann dieser zur Konfiguration oder zum Zustandswechsel an die entsprechenden Bestandteile der Prozessführungskomponente weitergeleitet werden. Auch ist es möglich, dass bei Komponentensystemen beispielsweise der Auftrag an unterlagerte Einheiten weitergeleitet wird. Falls es sich um einen Auftrag zum Fahrweisenwechsel oder direkten Aufruf der intern realisierten Dienste zur Ausführung einer Fähigkeit handelt, wird dieser an den Steuerkopf weitergegeben.

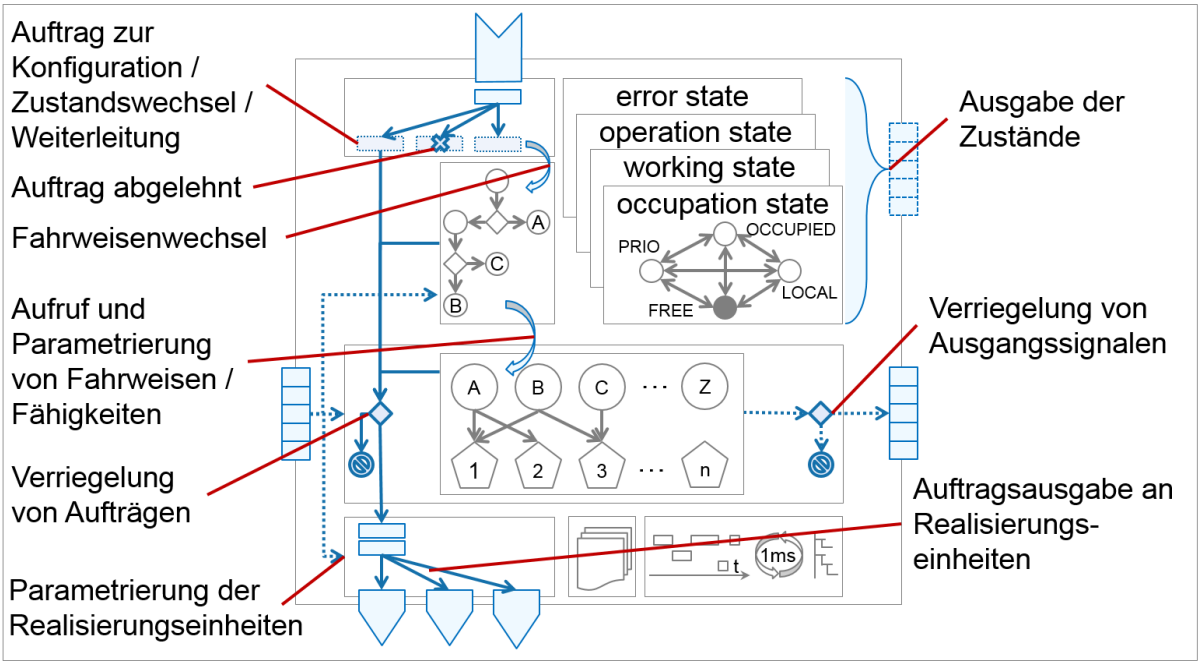

Abbildung 9.8: Schematische Darstellung des Aufbaus einer Prozessführungskomponente mit Erklärung des Signal- und Auftragsflusses.

Der Steuerkopf ruft dann je nach Zustand der Komponente die gewünschte Fahrweise bzw. den internen Dienst einer Fähigkeit auf, oder parametriert diese nach einer hinterlegten Logik. Der Steuerkopf, sowie Fahrweisen und Fähigkeiten können dabei ebenfalls neue Aufträge zur Weitergabe an unterlagerte Einheiten erzeugen. Diese, sowie die von den Fahrweisen und ausführbaren Fähigkeiten ausgegebenen Signale können durch die Sicherungslogik verriegelt

werden. Letztendlich setzt die Auftragsausgabe die genehmigten Aufträge auf die Realisierungseinheiten um und sorgt für die Zustellung der Aufträge. Über Signaleingänge oder Konfigurationsaufträge können die Rollen mit den Realisierungseinheiten parametriert werden.

9.2.2 Beispiel für eine Komponentenbeschreibung

Entwurfsmuster unterstützen den Planungs- und Instanziierungsprozess einer Komponente aus dem Metamodell heraus. Dazu sollten die Regeln und Zustände der inneren Bestandteile der Komponente bestimmt werden. Weiter sollten die möglichen Aufträge, die von der Komponente bearbeitet werden können festgelegt werden. Daher dient die folgende Tabelle als Beschreibung einer Prozessführungskomponente und kann als Entwurfsmuster für diese Anwendung von Komponenten genutzt werden.

Metamodell Bestandteil	Notwendige Festlegungen
Auftragsempfang	• Belegungszustände und Übergänge ○ Berechtigungen/Bedingungen für Belegungszustandswechsel • Liste der Aufträge mit Ihren Parametern • Auftragsschlange: Puffergröße, Überlaufverhalten, Prioritäten

Zustandsautomaten	• Liste der extern sichtbaren Zustände und Zustands-Signalausgänge
	  ○ Fehlerzustände
	  ○ Belegungszustände
	  ○ Betriebszustände
	  ○ Betriebsarten
	  ○ Arbeitszustände
	• Existieren Komponententyp-spezifische Zustandsautomaten? → Zustände und Übergänge definieren

Steuerkopf	• Ablauflogik definieren: Zuordnung der ORDER bzw. Dienste zu Fahrweisen.

Fahrweisen	• Liste der Fahrweisen:
	  ○ Parametrierungsmöglichkeiten
	  ○ Benötigte Fähigkeiten
	  ○ Ablaufgraphen
	  ○ Signaldefinitionen

Fähigkeiten	• Liste der Fähigkeiten • I40-konforme Darstellung der durch die Komponente zu bedienenden Fähigkeiten (Bspw. mit Merkmalen)

Ausführungsmodell	• Ausführstrategie der Komponentenbestandteile • Tasking, Scheduling, Prioritäten, Interrupt-/ Ereignisbehandlung, …
Auftragsausgabe	• Welche Rollen führt die Komponente? → Anzahl der Auftragsausgabelemente (statisch/dynamisch)
Signalschnittstellen	• Anzahl und Datentypen der Signalein-/ausgänge
Archivierung	• Archivierungsstrategie für Abläufe in der Komponente • Protokollierung und Auslagerung
Sicherungslogik	• Verriegelungslogik für die zuvor definierten Rollen und Signale

Tabelle 9.2

Eine spätere Maschinenlesbarkeit kann durch eine Abbildung der Tabelle in einem XML Schema, einem OPC UA Informationsmodell, oder dem benötigten Format erreicht werden. Da dies sehr spezifisch für die technologische Umsetzung ist, wird hier nur das menschenlesbare Format der Tabelle genutzt.

Die Informationen der Tabelle sind im Wesentlichen durch die Definition mehrerer Zustandsautomaten für Belegungszustände, Betriebszustände, Arbeitszustände und Fehlerzustände ausfüllbar. Daher eignen sich auch graphische Ablaufsprachen zu deren Definition. In der Automatisierungstechnik sind dies insbesondere Sequential Function Charts (SFCs) und verwandte Formen.

Wenn die Tabelle vollständig ausgefüllt ist, können technologische Zuordnungen und spezifische Teile zu dem Typ dieser Komponente hinzugefügt werden. Danach können die entsprechenden Komponenten erzeugt werden.

Im dynamischsten oder generischen Fall können Prozessführungskomponenten auch erst zur Laufzeit des Systems zusammengestellt werden. Dafür sollte mindestens der Auftragsempfang durch eine rudimentäre Belegungslogik und die extern abfragbaren Zustände realisiert sein. Damit ist die Komponente im System sicht- und ansprechbar. Ihre innere Struktur kann durch verschiedene Instanzen zu verschiedenen Zeiten gefüllt werden. (Vergleiche dazu Kapitel 6)

9.3 Verwaltungskomponenten

In jeder technischen Anlage existiert eine große Menge von technischen Gegenständen, die einen Wert für verschiedene, mit Gegenständen verknüpfte Organisationseinheiten hat. Diese technischen Gegenstände werden mit Bezug zu Ihrer Organisationseinheit im Folgenden als Asset bezeichnet (VDI/VDE-Gesellschaft Mess- und Automatisierungstechnik (GMA) 2015). Die digitale Verwaltung dieser Assets kann für die Organisationseinheit von wesentlichem Vorteil sein. So können beispielsweise Stammdaten, Dokumentationen, Verschaltung, Parametrierung, usw. erfasst werden. Da die Assets einen Wert für die Organisationseinheit bereitstellen, ist im Besonderen die Erfassung des Lebenszyklusses der Komponente von Relevanz.

Diese Aufgabe kann in der BaSys40-Komponenten-Systemplattform durch Verwaltungskomponenten erfüllt werden. Da diese Art von Komponenten Dienstteilnehmer im BaSys-Dienstsystem und gegebenenfalls im I40-Netzwerk sind, haben sie insbesondere Zugriff auf andere BaSys40-Komponenten. Wie in Festlegung 8.2 beschrieben, könnten Komponententyp-Beschreibungen über solche Verwaltungskomponenten zur Verfügung gestellt werden.

9.4 Infrastrukturkomponenten

In der BaSys40-Komponenten-Systemplattform existieren verschiedenste Ressourcen die von mehreren BaSys40-Dienstsystemteilnehmern benötigt werden. Um die Infrastruktur des Laufzeitsystems verfügbar zu machen und zu verwalten, können Infrastrukturkomponenten als BaSys40-Komponenten realisiert werden.

Nach Festlegung 6.6 obliegt den Infrastrukturkomponenten insbesondere die Verwaltung der Komponenten-Systemplattform selbst. Dazu gehört unter anderem die Verwaltung der BaSys40-Komponenten und BaSys40-Komponententypen, was in Abbildung 6.2 zusätzlich dargestellt ist und im BaSys40-Komponentenmetamodell in Abschnitt 8.2 aufgegriffen wird.

Weitere Anwendungen für Infrastrukturkomponenten sind die Verwaltung des Kommunikationssystems, die Realisierung von Nachrichtensystemen, sowie die Bewältigung von Discovery- und Identifikations-Aufgaben.

10 Anhang

10.1 Stand der Technik zur Industrie 4.0 Komponente

Im Umfeld der Industrie 4.0 Initiative ist der Begriff der Industrie 4.0 Komponente (I40-Komponente) entstanden. Einen Literaturüberblick einiger zum Thema Industrie 4.0 erschienen Papiere, welcher die Aktualität der Thematik wiederspiegelt, zeigt der Fortschrittsbericht der Plattform Industrie 4.0 (Plattform Industrie 4.0 2016a). Die I40-Komponente wird dabei zu meist als Aggregation aus Verwaltungsschale und Asset dargestellt, wie in Abbildung 10.1 gezeigt. Zusätzlich findet sich eine Kurzbeschreibung der I40-Kompenten im ZVEI-Papier über die I40-Komponente (Dr. Michael Hoffmeister 2015-04-00).

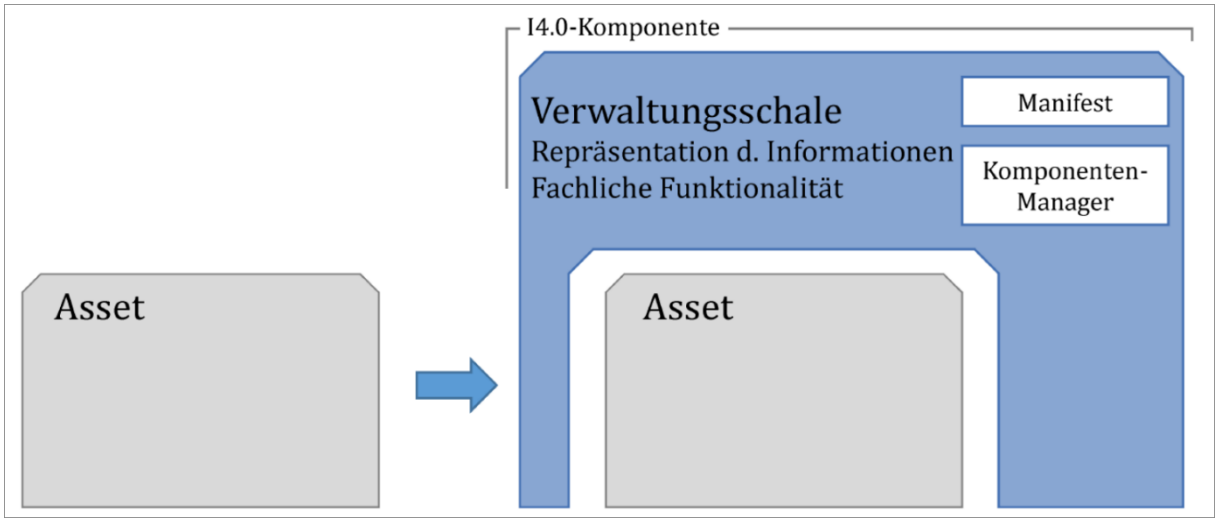

Abbildung 10.1: 5 I40-Komponente als Aggregation von Asset und Verwaltungsschale. (Spezifikation DIN SPEC 91345, Bild 10)]

Während für die Verwaltungsschalte schon erste prototypische Implementierungen vorhanden sind (OpenAAS, (Florian Palm)), ist die I40-Komponente bis dato nur auf dem Papier definiert. BaSys40-Komponenten können daher als Grundlage oder Erweiterung für die Konzepte der I40-Komponente dienen, da sie erste Implementierungen und Validierungen der Konzepte enthalten. Darauf wird in Abschnitt 3.1 Bezug genommen, indem Smarte-Komponenten als Netzwerkteilnehmer im I40-Netzwerk charakterisiert werden. Weiter wird das Metamodell der BaSys40-Komponente an das Referenzmodell der I40-Komponente angelehnt, wie in Abbildung 8.3 deutlich wird.

10.1.1 Technische Assets als Grundlage für I40-Komponenten

Der Statusreport „Industrie 4.0 – Technical Assets“ (VDI/VDE-Gesellschaft Mess- und Automatisierungstechnik (GMA) 2015) beschreibt grundlegende Begriffe und Konzepte auf denen viele weitere Veröffentlichungen der Plattform I40 aufbauen. Hierin wird der Begriff des Assets,

welches immer als technischer Gegenstand zu verstehen ist, definiert. Assets bilden dabei die unterste Schicht (Layer) des Referenzarchitekturmodell Industrie 4.0 (RAMI).

Begreift man die I40-Komponente als Aggregation von Asset und Verwaltungsschalte, so bildet der Asset-Begriff eine Grundlage für die Definition der Verwaltungsschale („Asset Administration Shell“, AAS) und der I40-Komponente. Dabei verwaltet die AAS das Asset in der Informationswelt für eine Organisationseinheit. Entsprechend ist die AAS Eigentum der Organisationseinheit und wird als solche auch nicht mit einer anderen Organisationseinheit ausgetauscht oder an diese weitergegeben. Es werden lediglich Informationen zwischen AASs verschiedener Organisationseinheiten ausgetauscht. Anders formuliert verknüpft eine andere Organisationseinheit andere Werte mit dem selben technischen Gegenstand, sodass dieser von verschiedenen Organisationseinheiten als verschiedenes Asset betrachtet werden kann. Ein abstraktes Beispiel für den Wechsel eines Assets von einem Hersteller zum Anwender ist in Abbildung 10.2 gezeigt.

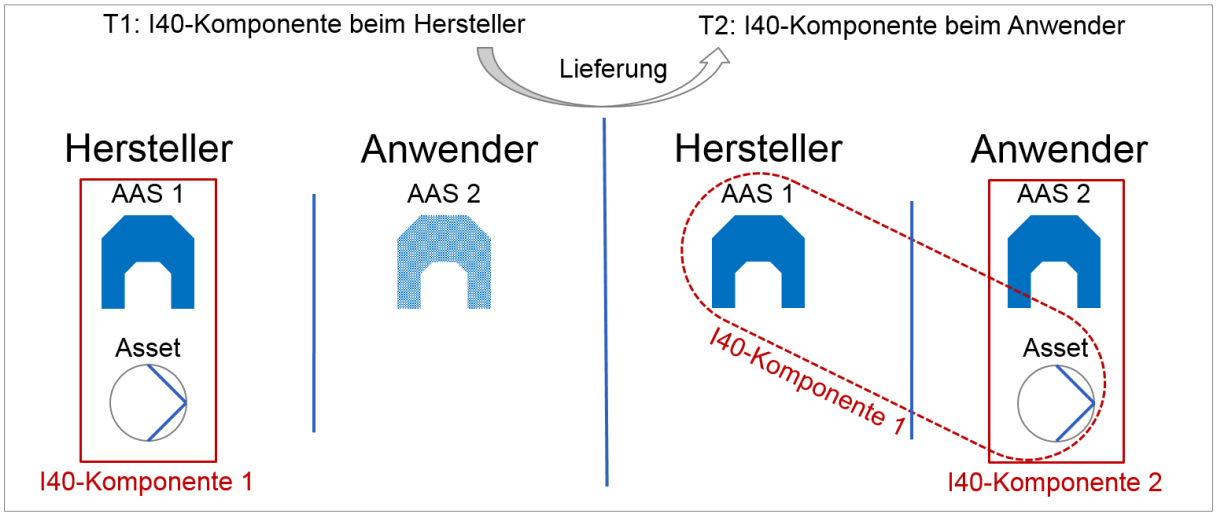

Abbildung 10.2: I40-Komponente vor und nach Lieferung des technischen Gegenstandes vom Hersteller zum Anwender.

Zur Klassifikation von Assets in Bezug auf die Kommunikationsfähigkeit (Communication) und den Bekanntheitsgrad im Informationssystem (Presentation/Publicity) wird dabei die sogenannte CP-Klassifikation eingeführt. Demnach ist die Voraussetzung für eine I40-Komponente ein Asset, welches mindestens passiv Kommunikationsfähig ist und als Entität im Informationssystem verwaltet wird.

Eine darauf aufbauende allgemeine Einführung in die I40-Komponente inklusive CP-Klassifikation findet sich in einem weiteren Statusbericht der Plattform I40 zu „Gegenstände[n], Entitäten, Komponenten“ (VDI/VDE-Gesellschaft Mess- und Automatisierungstechnik (GMA) 2014).

10.1.2 Verwaltungsschale als Grundlage der I40-Komponente

Die Fortwährende Nennung der I40-Komponente in der Literatur zeigt auf, dass ein Komponentenbasierter Ansatz für die I40-Architektur gewünscht ist. Die Einordung von I40 Komponenten im RAMI ergibt sich dabei aus ihrer Zusammensetzung aus Asset und Verwaltungs-

schale. Sie ist beispielsweise in DIN Spec 93145 wiederzufinden. Hierin wird die I40-Komponente definiert als „weltweit eindeutig identifizierbarer kommunikationsfähiger Teilnehmer bestehend aus Verwaltungsschale und Asset mit digitaler Verbindung (entspricht CP24, CP34 oder CP44) eines I40-Systems, der dort Dienste mit definierten QoS (Quality of Service)-Eigenschaften anbietet “(Spezifikation DIN SPEC 91345). Eine aktuelle Definition im Industrie 4.0 Glossar (VDI/VDE-Gesellschaft Mess- und Automatisierungstechnik (GMA)) bezieht die I40-Komponente direkt mit in die Definition der AAS ein: „virtuelle digitale und aktive Repräsentanz einer I4.0 Komponente im I4.0 System“ (VDI/VDE-Gesellschaft Mess- und Automatisierungstechnik (GMA) 2017). Somit bildet die Struktur der Verwaltungsschale (Plattform Industrie 4.0 2016b) die Grundlage für den Aufbau von I40-Komponenten. Dabei sind die an UML angelehnten Modelle im ZVEI Diskussionspapier (Plattform Industrie 4.0 2016c) die konkretesten Darstellungen des Aufbaus der Industrie 4.0 Komponente, wie in Abbildung 10.3 zu sehen ist.

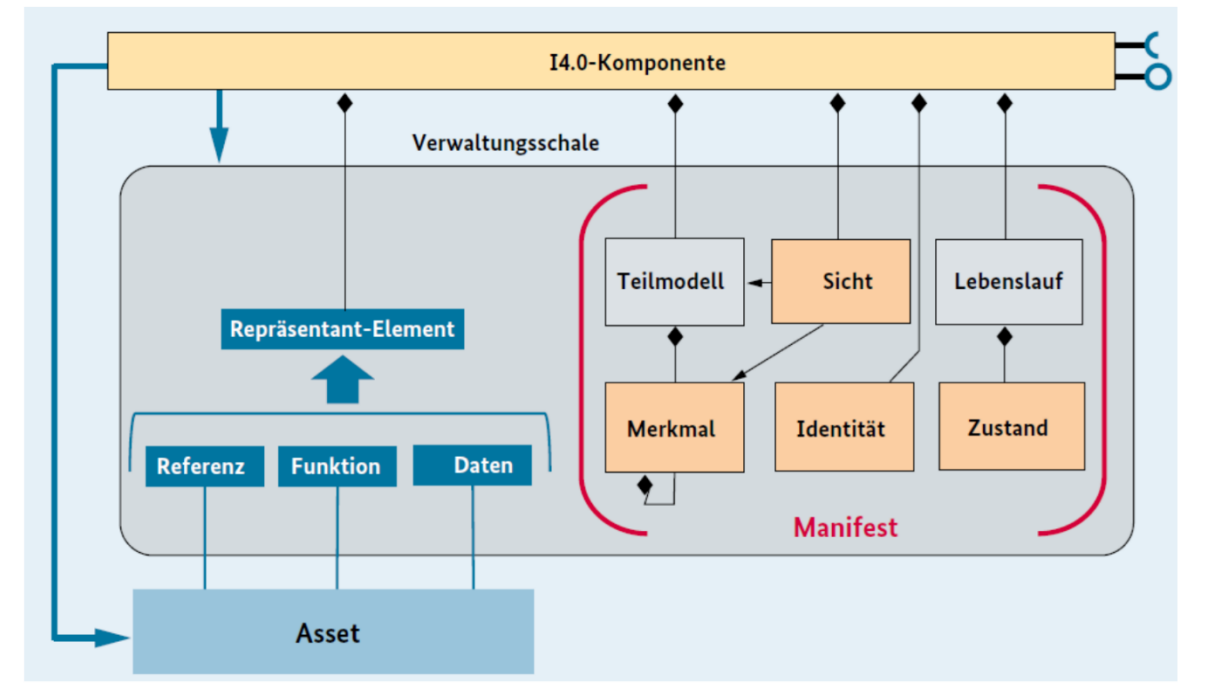

Abbildung 10.3: Referenzmodell der Industrie 4.0 Komponente. [19] Abbildung 1

Die Aktualität der Diskussion manifestiert sich in der Vielzahl der Berichte und der laufenden Definition der Begriffe im Glossar online. Hier hat die Komponente als zentrales Architekturelement sogar eine eigene Seite (VDI/VDE-Gesellschaft Mess- und Automatisierungstechnik (GMA)) zu ihren Eigenschaften erhalten. Ein umfassender Bericht zur „Umsetzungsstrategie Industrie 4.0“ beschreibt dabei die Grundlagen für die DIN Spec 91345 und insbesondere für das Referenzmodell der I40-Komponente (BITKOM e.V. et al. 2015-04-00, 6.3).

Modellierungsbeispiele für I40-Komponenten finden sich in einem weiteren Statusbericht. (VDI/VDE-Gesellschaft Mess- und Automatisierungstechnik (GMA) 2016) Weiter wurde schon die Interpretation und Beziehung der Verwaltungsschale zum Begriff des Digitalen Zwillings aus BaSys heraus beleuchtet. (Drath et al. 2017) Außerdem wurde innerhalb von BaSys ebenfalls ein konsolidiertes Modell der Verwaltungsschale aufbauend auf der Definition von Designräumen für Verwaltungsschalen in Deliverable D-FT2.1 erstellt.

10.1.3 Aggregation von I40-Komponenten

Der Begriff der I40-Komponente ist eng mit dem Begriff des I40-Systems verknüpft. So kann ein I40-System als Aggregation von I40-Komponenten, also als I40-Komponentensystem (vgl. Abschnitt 2.3), aufgefasst werden. Dies findet sich beispielsweise im Bericht der Plattform zur Weiterentwicklung des Interaktionsmodells für I40-Komponenten wieder. (Plattform Industrie 4.0 2016c, Abbildung 2)

Eine weitere Art der Aggregation ist das Konzept von Verbundkomponenten (Plattform Industrie 4.0 2017-06-00) aus der Plattform I4.0 SG Modelle und Standards. Dabei fokussiert der Bericht auf das in-Beziehung-setzen von I40-Komponenten und deren Teilelemente mit dem Ziel zweckgerichtete Verbünde zu organisieren.

Zur Strukturierung von I40-Komponenten können sogenannte Teilmodelle genutzt werden. Damit kann die I40-Komponente als Aggregation von Teilmodellen angesehen werden. Die Teilmodelle erfüllen allerdings nicht zwangsläufig die hier geforderten Komponenteneigenschaften. Beispiele für Teilmodelle finden sich in unter anderem im Bericht vom ZVEI zu Beispielen der Verwaltungsschale (ZVEI e.V. 2016-11-00). Nach aktuellem Stand der Forschung findet sich die in den Veröffentlichungen vorgesehene „fachliche Funktionalität“ in den Teilmodellen der Verwaltungsschale wieder. Somit bringen Teilmodelle die operative Funktionalität standardisiert in Verwaltungsschalen und damit in I40-Komponenten. Dabei sind aktuell nur Service Aufrufe vorgesehen, ohne dass Aussagen über Ablaufverhalten, Signalschnittstellen, Zustandsmaschinen, etc. gemacht werden. Daher bildet die BaSys40-Komponente in Kombination mit der AAS eine Grundlage oder Erweiterung für zukünftige Konzepte der operativen I40-Komponente.

10.1.4 Eigenschaften der Industrie 4.0 Komponente

I40-Komponenten lassen sich ebenfalls durch verschiedene Eigenschaften charakterisieren. Dazu findet man zunächst die CP-Klassifikation, wie in Abschnitt 10.1.1 erläutert. Weitere Eigenschaften wie Identifizierbarkeit, Zustand im Lebenslauf, I4.0-konforme Kommunikation, Schachtelbarkeit und Kapselfähigkeit, finden sich in der DIN Spec zum RAMI (Spezifikation DIN SPEC 91345, 6.1). Neben diesen Eigenschaften können Anregungen auf der Seite „Eigenschaften Industrie 4.0-Komponente“ des I40-Glossars gefunden werden (VDI/VDE-Gesellschaft Mess- und Automatisierungstechnik (GMA)).

11 Glossar

Im Folgenden werden relevante, allgemeine und BaSys-spezifische Begriffe im Kontext des Deliverables erläutert. Für die Begriffe aus dem Industrie 4.0 Kontext sei auf die Arbeiten der VDI GMA FA 7.21 UAG Begriffe und das globale BaSys Glossar verwiesen. Weitere Begriffsdefinitionen im Kontext von Fähigkeiten sind dem BaSys „Fähigkeiten Arbeitsdokument“ entnommen oder können dort nachgeschlagen werden.

11.1 Allgemeine Begriffe

11.1.1 Einzelfunktionseinheit

Eine Einzelfunktionseinheit ist eine Prozessführungskomponente. Sie umfasst zusätzlich immer sowohl die anlagentechnische Hardwarekomponente, als auch die zum Betrieb dieser Komponente notwendige Steuer-Software.

11.1.2 Einzelsteuereinheit

Notwendige Steuer-Software zum Betrieb einer Einzelfunktionseinheit.

11.1.3 (technische) Komponente

Als Komponente wird hier, wenn nicht weiter ausgeführt, stets eine technische Komponente im Sinne der DIN Spec 40912 verstanden: Eine Komponente ist eine vorgefertigte, in sich strukturierte und unabhängig hantierbare Einheit, die zur Realisierung einer konkreten Rolle in einem System vorgesehen ist.

11.1.4 Komponentensystem

Ein Komponentensystem beschreibt, wie ein Ganzes aus seinen Komponenten aufgebaut ist und wie sich die Gesamtfunktionalität des Ganzen aus dem Zusammenwirken seiner Komponenten ergibt.

11.1.5 Komponenten-Systemplattform

Eine Komponenten-Systemplattform beschreibt die Systemumgebung, in der die Komponenten gehandhabt, also entfernt, hinzugefügt, verschoben, eingebaut, ausgebaut, in Betrieb genommen und außer Betrieb genommen werden.

11.1.6 Laufzeitsystem

Ein Laufzeitsystem beschreibt die Laufzeitumgebung mit den auf ihr geladenen Programmen bzw. Bibliotheken.

11.1.7 Laufzeitumgebung

Eine Laufzeitumgebung, auch Ausführungsumgebung oder seltener Ablaufumgebung, beschreibt die zur Laufzeit von Computerprogrammen verfügbaren und festgelegten Voraussetzungen eines bestimmten Laufzeitsystems. Eine Laufzeitumgebung lädt Programme oder Bibliotheken und lässt diese auf einer Plattform ablaufen, für welche die Laufzeitumgebung gemacht worden ist.

11.1.8 Netzwerkschnittstelle

Eine Netzwerkschnittstelle umfasst den gesamten Kommunikations-Stack vom Physical Layer aufwärts bis zur Applikationsschicht (ISO/OSI Modell).

11.1.9 Prozessführung

Prozessführung bedeutet, dass der Prozess innerhalb der technischen und wirtschaftlichen Randbedingungen gehalten wird.

11.1.10 Prozessführungskomponente

Einzeln und im Verbund stellen diese Komponenten sicher, dass ein Prozess geführt wird.

11.1.11 Software

In einem digitalen Datenverarbeitungssystem implementiertes Abbild von Informationen (Daten) und Algorithmen.

11.1.12 Softwarekomponente

Eine Softwarekomponente ist ein Software-Element, das konform zu einem Komponentenmodell ist und gemäß einem Composition-Standard ohne Änderungen mit anderen Komponenten verknüpft und ausgeführt werden kann.

11.1.13 Verbundkomponente

Eine Verbundkomponente ist eine Komponente, die selbst aus mehreren Komponenten besteht. Eine Verbundkomponente kann aus Hardwarekomponenten, Softwarekomponenten, smarten Komponenten, Verbundkomponenten oder irgendeiner Mischung aus diesen bestehen.

11.2 BaSys 4.0 Begriffe

11.2.1 (BaSys40-) Dienst

Ein Dienst ist grundsätzlich ein abgegrenzter Funktionsumfang, der von einer Entität (einem Gegenstand, er in der Informationswelt verwaltet wird) oder Organisation über Schnittstellen angeboten wird (Quelle: Plattform Industrie 4.0 – Glossar).

Im Kontext von BaSys4.0 repräsentiert dieser „abgegrenzte Funktionsumfang“ unter ande-rem die Ausführung einer Fähigkeit von Ressourcen/Realisierungseinheiten. Folglich kann ein Dienst eine ihm zugehörige ausführbare Fähigkeit darstellen bzw. das Potential diese Fä-higkeit nutzbar zu machen. Ein Dienst abstrahiert weiterhin die konkreten Software- und Hardwareelemente, welche benötigt werden, um die Fähigkeit letztlich zu realisieren.

11.2.2 (BaSys40-) Dienstschnittstelle

Eine Dienstschnittstelle im Sinne von BaSys ist eine Schnittstelle, die das Ausführen und Beeinflussen von BaSys40-Dienstens erlaubt.

11.2.3 (BaSys40-) Dienstsystemteilnehmer

Ein Dienstsystemteilnehmer besitzt die nötigen Schnittstellen, um Dienste des Systems zu nutzen, zu erbringen oder beides.

11.2.4 (BaSys40-) Fähigkeit

Eine Fähigkeit (engl. „Skill“ oder teilweise auch „Capability“) ist das Potenzial eines Assets (bzw. einer I4.0-Komponente), eine Wirkung zu erzielen. Sie ist unabhängig von ihrer tatsächlichen Ausgestaltung in Hard- und Software.

11.2.5 (BaSys40-) Komponente

BaSys40-Komponenten sind Smarte-Komponenten bzw. reine Softwarekomponenten, die bestimmte Anforderungen erfüllen. Eine BaSys40-Komponente muss sich im Kommunikationsnetz als BaSys40-Dienstsystemteilnehmer darstellen und alle mit dieser Rolle verbundenen Standards bzw. Schnittstellen erfüllen. Weitere Anforderungen sind die Kontextneutralität und die Implementierung bestimmter standardisierter Zustandsautomaten.

12 Index

13 Literaturverzeichnis

Abel, Dirk (Hg.) (2008): Integration von Advanced control in der Prozessindustrie. Rapid control prototyping. [Online-Ausg.]. Weinheim: Wiley.

Norm ISA-88.00.01, 2010-00-00: Batch control - Part 1: Models and terminology.

Norm ISA-88.00.02, 2001-00-00: Batch Control - Part 2: Data Structures and Guidelines for Languages. Online verfügbar unter http://sesam-world.com/_pdf/make2pack/mode/2010-11-29/Materiale/TR_880002.pdf, zuletzt geprüft am 21.12.2017.

BITKOM e.V.; VDMA e.V.; ZVEI e.V. (Hg.) (2015-04-00): Umsetzungsstrategie Industrie 4.0. Ergebnisbericht der Plattform Industrie 4.0.

Bronsard, F.; Bryan, D.; Kozaczynski, W. (1997): Toward software plug-and-play SSR'97 Proceedings of the 1997 symposium on Software reusability Pages 19–29. In: *ACM New York, NY, USA*.

Dr. Michael Hoffmeister (2015-04-00): Industrie 4.0: Die Industrie 4.0-Komponente. 1.0. Aufl. Hg. v. ZVEI e.V. Online verfügbar unter https://www.zvei.org/fileadmin/user_upload/Themen/Industrie_4.0/Das_Referenzarchitekturmodell_RAMI_4.0_und_die_Industrie_4.0-Komponente/pdf/Industrie_4.0_Komponente_Download.pdf.

Drath, Rainer; Malakuti, Somayeh; Grüner, Sten; Grothoff, Julian Alexander; Wagner, Constantin August; Epple, Ulrich et al. (2017): Die Rolle der Industrie 4.0 „Verwaltungsschale“ und des „digitalen Zwillings“ im Lebenszyklus einer Anlage. In: [Automation 2017, 2017-06-27 - 2017-06-28, Baden-Baden, Germany]. Automation 2017, Baden-Baden (Germany), 27 Jun 2017 - 28 Jun 2017. Online verfügbar unter https://publications.rwth-aachen.de/record/692031.

Dudenredaktion, bearb von der (Hg.): Deutsches Wörterbuch. Auf der Grundlage der neuen amtlichen Rechtschreibregeln ; [rund 45000 Stichwörter aus allen Bereichen des täglichen Lebens]: Dudenverl.

Enste, Udo (2001): Generische Entwurfsmuster in der Funktionsbausteintechnik und deren Anwendung in der operativen Prozeßführung. Als Ms. gedr. Düsseldorf: VDI-Verl. (Fortschritt-Berichte VDI Reihe 8, Meß-, Steuerungs- und Regelungstechnik, 884).

Enste, Udo; Epple, Ulrich (1998): Standardisierte Prozeßführungsbausteine. die Basis für Applikationsmodelle zur operativen Führung von verfahrenstechnischen Produktionsanlagen. In: Mess- und Automatisierungstechnik. Neue Entwicklungen, Technologie, Anwendungen ; Tagung Ludwigsburg 18. und 19. Juni 1998. Düsseldorf: VDI-Verl. (VDI-Berichte / Verein Dt. Ingenieure, 1397).

Florian Palm: openAAS. Development Repository for open Asset Administration Shell. Hg. v. ZVEI. Lehrstuhl für Prozessleittechnik der RWTH Aachen University. Online verfügbar unter https://github.com/acplt/openAAS.

Heger, Christoph Lutz (2007): Bewertung der Wandlungsfähigkeit von Fabrikobjekten. Garbsen: PZH, Produktionstechn. Zentrum (Berichte aus dem IFA, 2007,1).

Heineman, George T.; Councill, William T. (2001): Component-based software engineering. Putting the pieces together. Boston, Mass.: Addison-Wesley.

Regelwerk VDI, VDI-Richtlinien VDI/VDE 3696 Blatt 2, 1995-10-00: Herstellerneutrale Konfigurierung von Prozeßleitsystemen - Standard-Funktionsbausteine.

Norm DIN EN 81346-1, 01.05.2010: Industrielle Systeme, Anlagen und Ausrüstungen und Industrieprodukte - Strukturierungsprinzipien und Referenzkennzeichnung - Teil 1: Allgemeine Regeln (IEC 81346-1:2009); Deutsche Fassung EN 81346-1:2009.

Norm ISO/IEC 10746-2, 15.12.2009: Informationstechnik - Verteilte Verarbeitung in Offenen Systemen - Referenzmodell - Teil 2: Grundlagen.

Spezifikation DIN SPEC 40912, 01.11.2014: Kernmodelle - Beschreibung und Beispiele.

Krämer, S.; Bamberg, A.; Dünnebier, G.; Hagenmeyer, V.; Piechottka, U.; Schmitz, S. (2008): Prozessführung. Beispiele, Erfahrung und Entwicklung. In: *Chemie Ingenieur Technik* 80 (9), S. 1341–1342. DOI: 10.1002/cite.200750564.

Nyhuis, Peter; Reinhart, Gunther; Abele, Eberhard (Hg.) (2008): Wandlungsfähige Produktionssysteme. Heute die Industrie von morgen gestalten. Garbsen: PZH Produktionstechnisches Zentrum.

Spezifikation formal/2015-03-01, 01.03.2015: OMG Unified Modeling Language. Online verfügbar unter http://www.omg.org/spec/UML/2.5.

Plattform Industrie 4.0 (2016a): Digitalisierung der Industrie – Die plattform Industrie 4.0. Fortschrittsbericht. Hg. v. Bundesministerium für Wirtschaft und Energie (BMWi).

Plattform Industrie 4.0 (2016b): Struktur der Verwaltungsschale. Fortentwicklung des Referenzmodells für die Industrie 4.0-Komponente. Ergebnispapier. Hg. v. Bundesministerium für Wirtschaft und Energie (BMWi). AG Referenzarchitekturen, Standards und Normung in Kooperation mit dem ZVEI Elektronikindustrie e.V.

Plattform Industrie 4.0 (2016c): Weiterentwicklung des Interaktionsmodells für Industrie 4.0-Komponenten. Diskussionspapier. Hg. v. Bundesministerium für Wirtschaft und Energie (BMWi). AG "Ontologie und Grammatik für I4.0-Komponenten" der AG1 „Referenzarchitektur, Standards und Normen“. Online verfügbar unter https://www.plattform-i40.de/I40/Redaktion/DE/Downloads/Publikation/interaktionsmodell-i40-komponenten-it-gipfel.pdf?__blob=publicationFile&v=13, zuletzt geprüft am 21.12.2017.

Plattform Industrie 4.0 (2017-06-00): Beziehungen zwischen I4.0-Komponenten – Verbundkomponenten und intelligente Produktion. Fortentwicklung des Referenzmodells für die Industrie 4.0–Komponente SG Modelle und Standards. Ergebnispapier. Hg. v. Bundesministerium für Wirtschaft und Energie (BMWi). AG „Modelle und Standards“ des ZVEI in Kooperation mit der AG Arbeitsgruppe „Referenzarchitekturen, Standards und Normung“.

PLCopen Technical Committee 6 (2009): XML Formats for IEC 61131-3. Technical Paper. Online verfügbar unter http://www.plcopen.org/pages/tc6_xml, zuletzt geprüft am 01.03.2018.

Spezifikation DIN SPEC 91345, 2016-04-00: Referenzarchitekturmodell Industrie 4.0 (RAMI4.0).

Sametinger, Johannes (1997): Software engineering with reusable components. With 26 tables. Berlin: Springer.

Norm DIN EN 13128, 2009-09-00: Sicherheit von Werkzeugmaschinen - Fräsmaschinen (einschließlich Bohr-Fräsmaschinen); Deutsche Fassung EN 13128:2001+A2:2009.

Norm IEC 61131-3, 2013-02-00: Speicherprogrammierbare Steuerungen - Teil 3: Programmiersprachen.

VDI/VDE-Gesellschaft Mess- und Automatisierungstechnik (GMA): FA7.21 Begriffe - Eigenschaften Industrie 4.0-Komponente. Online verfügbar unter http://i40.iosb.fraunhofer.de/FA7.21%20Begriffe%20-%20Eigenschaften%20Industrie%204.0-Komponente, zuletzt geprüft am 22.01.2018.

VDI/VDE-Gesellschaft Mess- und Automatisierungstechnik (GMA): FA7.21 Begriffe - Industrie 4.0. Online verfügbar unter http://i40.iosb.fraunhofer.de/FA7.21%20Begriffe, zuletzt geprüft am 22.01.2018.

VDI/VDE-Gesellschaft Mess- und Automatisierungstechnik (GMA) (2014): Industrie 4.0 – Gegenstände, Entitäten, Komponenten. Hg. v. Verein Deutscher Ingenieure e.V. GMA Gesellschaft Mess- und Automatisierungstechnik (GMA).

VDI/VDE-Gesellschaft Mess- und Automatisierungstechnik (GMA) (2015): Industrie 4.0 – Technical Assets. Grundlegende Begriffe, Konzepte, Lebenszyklen und Verwaltung. Hg. v. Verein Deutscher Ingenieure e.V. GMA Gesellschaft Mess- und Automatisierungstechnik (GMA) (ISBN 978-3-931384-83-8).

VDI/VDE-Gesellschaft Mess- und Automatisierungstechnik (GMA) (2016): Industrie 4.0 Components – Modeling Examples. Hg. v. Verein Deutscher Ingenieure e.V. GMA Gesellschaft Mess- und Automatisierungstechnik (GMA).

VDI/VDE-Gesellschaft Mess- und Automatisierungstechnik (GMA) (2017): Industrie 4.0 – Begriffe/Terms. Hg. v. Verein Deutscher Ingenieure e.V. GMA Gesellschaft Mess- und Automatisierungstechnik (GMA).

Wagner, Constantin; Epple, Ulrich (2015): Sprechende Kommandos als Grundlage moderner Prozessführungsschnittstellen. In: [AUTOMATION 2015 : 11.-12.Juni 2015, Baden-Baden / VDI]. AUTOMATION 2015, Baden Baden (Germany), 11 Jun 2015 - 12 Jun 2015. Düsseldorf: VDI-Verl. Online verfügbar unter http://publications.rwth-aachen.de/record/479208.

Norm DIN EN 12417, 2009-07-00: Werkzeugmaschinen - Sicherheit - Bearbeitungszentren; Deutsche Fassung EN 12417:2001+A2:2009.

ZVEI e.V. (Hg.) (2016-11-00): Beispiele zur Verwaltungsschale der Industrie 4.0-Komponente – Basisteil. Fortentwicklung des Referenzmodells für die Industrie 4.0-Komponente. White Paper.