FMU Best Practice Guide

Best Practices für die Anwendung des FMI-Standards in der virtuellen Inbetriebnahme

Erstellt am: 19.11.2025
Version: 1.0

FESTO

Inhaltsverzeichnis

Vorwort

Die virtuelle Inbetriebnahme beschreibt einen zentralen Prozess zur Absicherung von Steuerungssoftware für Maschinen und Anlagen. Meist ist die Modellierung der einzelnen Komponenten mit großem Aufwand verbunden. Vor diesem Hintergrund wurde im geförderten Projekt DIAMOND der vorliegende Leitfaden entwickelt. Ziel ist es, durch eine einheitliche und wiederverwendbare Modellierungspraxis die Erstellung digitaler Zwillinge zu beschleunigen und deren Qualität zu verbessern. Dieser Leitfaden bietet praxisnahe Empfehlungen für eine erfolgreiche Umsetzung sowie branchenweite Verbreitung. Dabei setzt er auf den Functional Mock-up Interface (FMI) Standard als zentrales Element für die Erstellung und Integration standardisierter Verhaltensmodelle (Functional Mock-up Units (FMUs)).

Der hier vorliegende Leitfaden ist ein Ergebnis des Forschungsprojekts „Digitale Anlagenmodellierung mit neutralen Datenformaten“ (DIAMOND). Das Konsortium besteht aus über 20 Verbundpartnern, darunter Komponenten-Hersteller, Anlagenbauer, Anlagenbetreiber (OEM), Softwareanbieter und Forschungseinrichtungen. DIAMOND wird von der Europäischen Union (EU) finanziert und vom Bundesministerium für Wirtschaft und Energie (BMWE) gefördert. An dieser Stelle gilt besonderer Dank dem Projektträger, der diesen Leitfaden ermöglicht.

Eine zentrale Rolle spielt die DIAMOND Taskforce Simulation, welche sich explizit mit dem Thema standardisierte Verhaltensmodellierung auf Basis des FMI-Standards befasst hat. Experten der Verbundpartner teilten in der Taskforce Simulation ihr Wissen und ihre Erfahrung in Form von Experten-Interviews, welche die Grundlage des Leitfadens legten. Vielen Dank an jeden einzelnen Interviewpartner, der dazu beigetragen hat, dass der Leitfaden das Thema aus verschiedenen Blickwinkeln beleuchtet. Nur so wurde eine breite Sichtweise auf verschiedene Prozesse geschaffen, welche die Akzeptanz dieses Leitfadens in der Industrie erhöht.

Zu erwähnen ist auch die Modelica Association, die unter dem „Modelica Association Project Functional Mock-up Interface“ den Standard weiter verbessert und für den ausgezeichneten Reifegrad verantwortlich ist. Vielen Dank für den fachlichen Austausch im Rahmen des DIAMOND-Projektes.

Schließlich sind die unmittelbar am Leitfaden beteiligten Personen zu nennen. Diese sind am Ende des Dokuments unter Editoren und Mitwirkende gesondert aufgeführt. An dieser Stelle ist die hervorragende, firmenübergreifende Zusammenarbeit hervorzuheben. Vielen Dank für die vielen Diskussionen, die Anregungen und die Erfahrungen, welche in den Leitfaden eingeflossen sind.

1. Executive Summary

Der vorliegende Leitfaden wurde im Rahmen des Förderprojekt DIAMOND entwickelt und fokussiert auf die Anwendung des FMI-Standards zur Erstellung sowie Integration von standardisierten Verhaltensmodellen durch FMUs für die virtuelle Inbetriebnahme von Produktionsanlagen. Ziel ist es, einen Prozess zu schaffen, der die Wiederverwendbarkeit, Qualität sowie die Effizienz bei der Verhaltensmodellierung für digitale Zwillinge steigert. Dies wird durch die Zusammenarbeit zwischen Komponentenherstellern, Anlagenbauern, Anlagenbetreiber (OEMs) und Tool-Herstellern realisiert.

Problemstellung

Die virtuelle Inbetriebnahme hat sich als zentraler Prozess zur Absicherung von Steuerungssoftware und der Inbetriebnahme für Maschinen und Anlagen in der Industrieautomatisierung etabliert. Gerade im Hinblick auf komplexer werdende Automatisierungskomponenten liegt eine große Herausforderung in der Erstellung von dazugehörigen Verhaltensmodellen zur Simulation. Aktuell entwickeln Tool-Hersteller, Anlagenbauer, Anlagenbetreiber (OEMs) und teilweise Komponenten-Hersteller eigene Modelle, was zu einem hohen Aufwand, Redundanzen und einer eingeschränkter Wiederverwendbarkeit führt. Aufgrund der Vielzahl an verwendeten Engineering-Tools bedarf es eines einheitlichen Standards, um eine durchgängige, firmenübergreifende Digitalisierung zu ermöglichen.

Lösung und damit verbundene Herausforderungen

Der FMI-Standard bietet eine toolunabhängige Schnittstelle zur Modellierung und Simulation von Verhaltensmodellen in Form sogenannter FMUs. Allerdings ist in einem unternehmen- und branchenübergreifenden Standardisierungs-Prozess die Integration von FMUs mit Herausforderungen, Fragen und Bedenken verbunden. Aus diesem Grund beschreibt der vorliegende Leitfaden praxisnah, wie FMUs im Kontext der virtuellen Inbetriebnahme eingesetzt werden können – von der Modellierung über die Validierung sowie der Auslieferung bis hin zur Integration in die bisherige Toollandschaft. Der Leitfaden gibt konkrete Empfehlungen zum Aufbau und Inhalt der FMU, zur Nutzung von Flags, zur Dokumentation sowie zur Integration in gängige VIBN-Prozesse. Dabei soll die flächendeckende Einführung und Nutzung des FMI-Standards beschleunigt werden.

Mehrwert für die Industrie

Die Anwendung des FMI-Standards ermöglicht:

- ✓ Reduktion des Modellierungsaufwands durch Wiederverwendbarkeit und Standardisierung.
- ✓ Höhere Flexibilität in der Tool-Auswahl und bei einem Tool-Wechsel durch die Integration standardisierter Modelle
- ✓ Höhere Modellqualität, da Komponenten-Hersteller durch die Bereitstellung von FMUs ihr Wissen direkt einbringen können.
- ✓ Neue Einsatzmöglichkeiten durch realitätsnahe Modellierung und Bereitstellung verschiedener Detaillierungsgrade aufgrund der Integration des Komponenten-Herstellers.
- ✓ Geringerer Aufwand beim Aufbau und der Pflege von Bibliotheken durch eine einheitliche Bereitstellung und Versionierung.
- ✓ Langfristige Akzeptanz durch eine mögliche Anbindung an Standards wie AAS, AML und SSP, sowie Einsatzmöglichkeiten in Cloud-Umgebungen und der Nutzung von künstlicher Intelligenz.

Fazit & Bedeutung des Leitfadens

Der FMI-Standard stellt einen entscheidenden Baustein für die Digitalisierung im Maschinen- und Anlagenbau dar. Hierfür liefert der FMU-Leitfaden eine praxisorientierte Grundlage und adressiert wesentliche beteiligten Bereiche im digitalen Engineering-Prozess. Langfristig wird erwartet, dass FMUs analog zu den CAD-Daten fester Bestandteil der Komponentenauslieferung werden. Nur durch standardisierte Verhaltensmodelle kann die steigende Komplexität in der Automatisierung beherrscht und die Wettbewerbsfähigkeit der Industrie gesichert werden.

2. Begriffsdefinition und Stand der Technik

2.1 Original Equipment Manufacturer (OEM)

Da dieser Leitfaden im Kontext der Automobilindustrie entstanden ist, wird unter dem Begriff OEM der Fahrzeug-Hersteller verstanden. Dieser integriert Produkte von Modul-, Komponenten- und Teilelieferanten und wird somit als Erstausrüster bezeichnet. Die vom Anlagenbauer gelieferten Maschinen und Anlagen dienen zur Produktion von Fahrzeugen und werden vom OEM betrieben. Aus diesem Grund wird der Begriff OEM und Anlagenbetreiber im Folgenden gleichgestellt.

2.2 Verhaltensmodell einer Automatisierungskomponente

Unter einer Automatisierungskomponente wird ein Gerät verstanden, welches in irgendeiner Art und Weise mit einem Steuerungssystem interagiert. Dabei wird grundsätzlich zwischen Aktoren (z.B. Antriebe, Ventile, Heizstäbe, Signalleuchten) und Sensoren (z.B. Taster, Schalter, RFID-Leser, Lichtschranken) unterschieden. Ein Verhaltensmodell beschreibt dabei jeweils das virtuelle Verhalten dieser Geräte bzw. Komponenten im Kontext eines Anlagenmodels [16]. Die virtuelle Komponente verhält sich analog der realen Komponente [20-24]. Die Modelle in der virtuellen Inbetriebnahme (VIBN) sind aktuell zumeist auf abstrakter Kommunikationsebene angesiedelt: Das Eingangs-/Ausgangs-(EA-) Verhalten der Steuerungen wird abgebildet, selten jedoch physikalische Effekte. [16] Eine detaillierte Modellierung wird bewusst vermieden, da diese bei komplexen Komponenten zu hohen Performanceanforderungen an die Software-Tools sowie die verwendete Hardware führt. Zudem fehlt den Drittanbietern bei der Erstellung von Verhaltensmodellen häufig das detaillierte Wissen über die Komponente.

2.3 Virtuelle Inbetriebnahme (VIBN)

Ein elementarer Aspekt des Engineering-Prozesses ist die VIBN. Bei der VIBN werden gewünschte Funktionen und Abläufe zunächst in einem virtuellen Anlagenmodell schrittweise getestet. Dadurch sollen die realen speicherprogrammierbaren Steuerungs- (SPS) und Roboterprogramme sowie die Kommunikation zu übergeordneten IT-Systemen im Ablauf, Zusammenspiel und in Fehlerfällen evaluiert werden, bevor die reale Produktionsanlage existiert [16]. Dies ermöglicht parallele Entwicklungen der verschiedenen Gewerke und eine frühe Absicherung des Steuerungscodes. Um diesem Ziel nachzukommen, werden Verhaltensmodelle der einzelnen Anlagenkomponenten benötigt, welche mit virtuellen oder realen SPS- bzw. Roboter-Steuerungen (RC) gekoppelt werden (vgl. Abbildung 1).

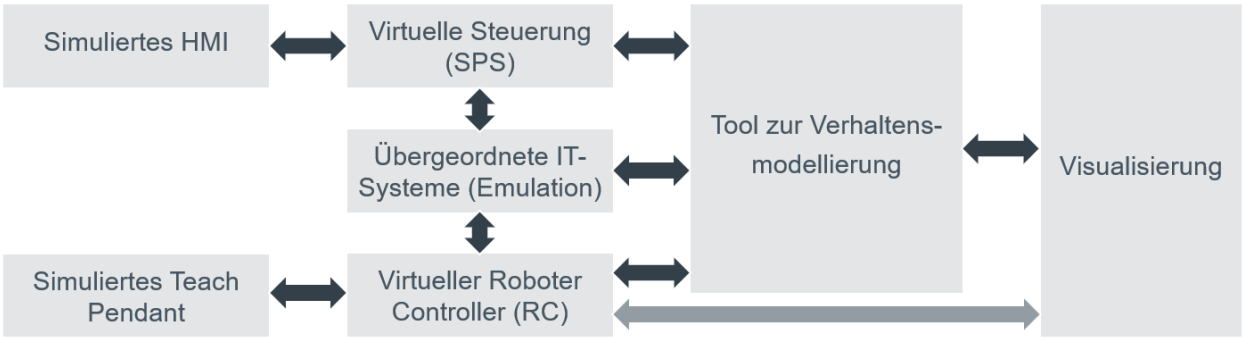

Abbildung 1: Klassischer Aufbau eines VIBN-Setups. Die Verbindung des virtuellen Roboter Controllers kann auch über das Tool zur Verhaltensmodellierung erfolgen

Die Schnittstellen zum Anlagen-Bediener, wie beispielswiese das Human-Machine Interface (HMI) oder das Teach Pendant, werden an die reale oder virtuelle Steuerung angebunden. Die Verhaltensmodelle stehen in Interaktion zu einer Visualisierung, um z.B. eine Sensor-Rückmeldung sowie einen Materialfluss zu ermöglichen. Beispiele für übergeordnete IT-Systeme bilden Leitsysteme, die Betriebsdaten- (BDE) und Maschinendatenerfassung (MDE) sowie IoT-Server. Zur weiteren Auseinandersetzung mit dem Thema VIBN ist auf die Literatur [31], [32] sowie [33] zu verweisen.

2.4 Datengrundlage

Das Simulationsmodell der VIBN besteht aus Modellen der einzelnen Teilkomponenten. Diese sind z.B. SPS, HMI, Peripherie und Roboter. Für SPS, HMI und Roboter bieten die meisten Hersteller Emulatoren an. Somit können die originalen Steuercodes genutzt werden. Zur 3D-Visualisierung werden Daten aus der mechanischen Konstruktion verwendet, die üblicherweise von den Herstellern ebenso bereitgestellt werden. Für die notwendigen Verhaltensmodelle der real verbauten Komponenten, wie Antriebe, Sensoren, etc., gab es keine einheitlichen Datenformate. Verhaltensmodelle müssen aktuell in Eigenregie oder durch Dritte aufwändig nachgebildet werden. Dies führt zwangsläufig zu Abweichungen im Verhalten. In einigen Fällen können daher einzelne Tests nur an der realen Anlage durchgeführt werden. Die Tool-Hersteller entwickeln häufig ihre eigenen Bibliotheken, was in der Vergangenheit zu einer Vielzahl unterschiedlicher Varianten für ein und dieselbe Komponente geführt hat [16].

2.5 Digitaler Zwilling

Für den Begriff „digitaler Zwilling“ existiert eine Vielzahl an verschiedenen Definitionen. Je nach Anwendungsgebiet und Organisation unterschieden sich die Auffassungen teils erheblich. Im Kontext dieses Leitfadens wird unter dem Begriff „digitaler Zwilling“ das Abbild der später existierenden, realen

Produktionsanlage verstanden. Der Umfang und Detaillierungsgrad des digitalen Zwillings orientieren sich an den Anforderungen für den Einsatz in der VIBN. Dabei wird vor allem die Automobilbranche betrachtet. Wichtige Bestandteile des digitalen Zwillings sind Verhaltensmodelle, ein kinematisiertes CAD-Modell, der Materialfluss, virtuelle Steuerungen sowie eine Bedienoberfläche zur Durchführung der VIBN.

2.6 VIBN-Tool

Unter VIBN-Tool werden in diesem Leitfaden kommerzielle Software-Lösungen bezeichnet, welche zur Modellierung des digitalen Zwillings für die virtuelle Inbetriebnahme zum Einsatz kommen. Darunter fallen vor allem Tools für die Verhaltensmodellierung von Automatisierungskomponenten. Beispiele bilden SIMIT (*Siemens AG*), RF::ViPer (*EKS InTec GmbH*) oder WinMOD (*WINMOD GmbH*). Software-Lösung für die Visualisierung über 3D-Geometrien werden meist gesondert über „Visualisierung“ erwähnt, obwohl sie auch zur Modellierung des digitalen Zwillings dienen.

2.7 Grundlagen zum FMI-Standard

Der Functional Mock-up Interface (FMI) Standard [6] wurde im Jahr 2008 während eines Projekts namens ITEA 2 MODELISAR begonnen, welches als Zielsetzung die Entwicklung eines toolunabhängigen, offenen Schnittstellen-Standards hatte [11,14]. Hierüber sollten verschiedene Simulationstools miteinander verknüpft werden [11]. Die Wurzeln des FMI-Standards liegen somit im Bereich der Interaktion verschiedener Steuergeräte (ECUs) in der Fahrzeugtechnik [15]. Inzwischen unterstützen fast 250 Tools (Modellierungs-, Simulations-, Testtools, …) den Standard in verschiedenen Ausführungen. Das letzte Update des Standards fand im November 2024 mit der Version 3.0.2 statt [11].

Analogien zum Anlagenbau, wie die zyklische Abarbeitung von Schrittfolgen, die Kapselung von einzelnen Komponenten und die Zusammenführung über verschiedene Software-Tools, führten im Jahr 2018 im Projekt ENTOC bereits zu Untersuchungen im Kontext der VIBN. Inzwischen unterstützen gängige VIBN-Tools, wie SIMIT, RF::ViPer oder WinMOD die Simulation und Integration von im FMI-Standard erstellten Modellen (Functional Mock-up Units (FMUs)).

Im FMI-Standard können Modelle gekapselt über die Co-Simulation oder den ModelExchange (vgl. Abschnitt 5.2) an beliebige Simulations-Tools angebunden werden. Dabei werden verschiedene Dateien über einen gezippten Container bereitgestellt. Das Modellverhalten wird über eine im C-Code kompilierte Binärdatei beschrieben. Dies ermöglicht einen Know-how-Schutz und schafft Akzeptanz beim FMU-Ersteller. Über eine offene XML-Datei werden allgemeine Informationen sowie die Schnittstellen-Signale mitsamt ihren Metadaten, wie die Bezeichnungen, Datentypen, Default-Werten, usw., bereitgestellt. Diese XML kann der Co-Simulator auslesen und für die Integration nutzen. Bei der Simulation werden die Eingänge eingelesen, ein Berechnungsschritt

durchgeführt und anschließen die Ausgänge geschrieben. Dieses Vorgehen wird mit einer vorgegebenen Zeit zyklisch wiederholt (vgl. Abbildung 2).

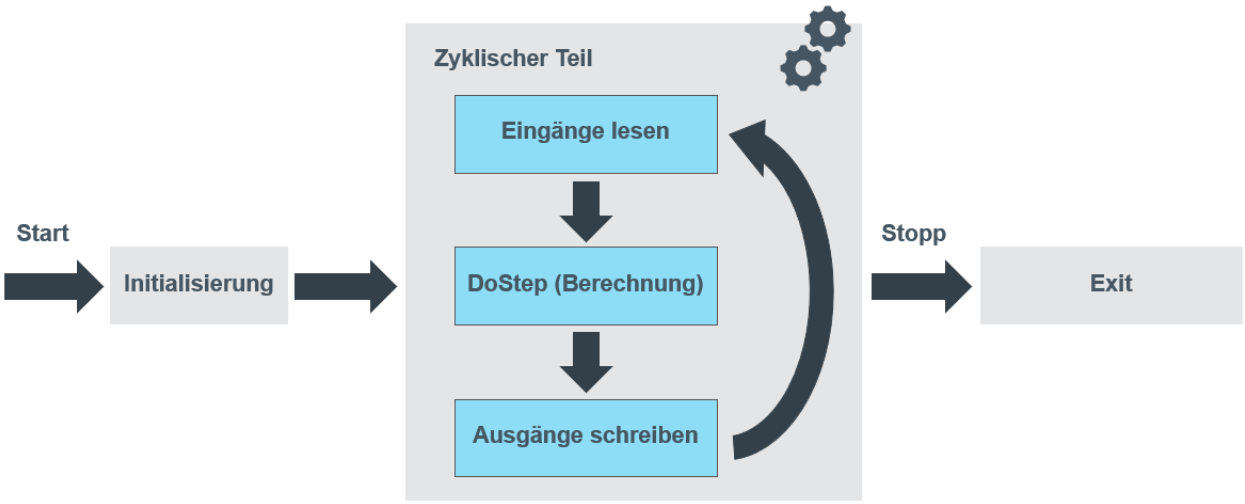

Abbildung 2: Abarbeitung einer FMU

Über so genannte „Layered Standards“ können zusätzliche Konventionen und Funktionalitäten beschrieben werden, sodass der Standard auf den jeweiligen Anwendungsfall angepasst werden kann.

3. Einführung und Motivation

Der FMI-Standard bietet im Kontext der virtuellen Inbetriebnahme ein großes Verbesserungs-potential. Aktuell fließt bei der Modellerstellung ein großer Anteil des Aufwandes in die Implementierung von Verhaltensmodellen. Es sind große, toolspezifische Bibliotheken nötig, um verschiedene Anlagen simulieren zu können. Gleichzeitig müssen diese für jedes Simulationstool spezifisch erstellt werden. Für den Komponenten-Hersteller ist es nicht realisierbar, bei der Vielzahl an Tools für jede Simulationsumgebung Verhaltensmodelle bereitzustellen. Somit existieren derzeit beim Original Equipment Manufacturer (OEM) verschiedene toolspezifische Bibliotheken, welche Verhaltensmodelle für dieselbe Komponente beinhalten. Die Modelle gilt es gesondert zu erstellen und zu pflegen, was einen großen Aufwand mit sich bringt (vgl. Abbildung 4). Je nach verwendetem Tool kann sich somit das Verhalten der gleichen Komponente für jedes Projekt unterscheiden.

Abbildung 3: Logo des FMI-Standards [11]

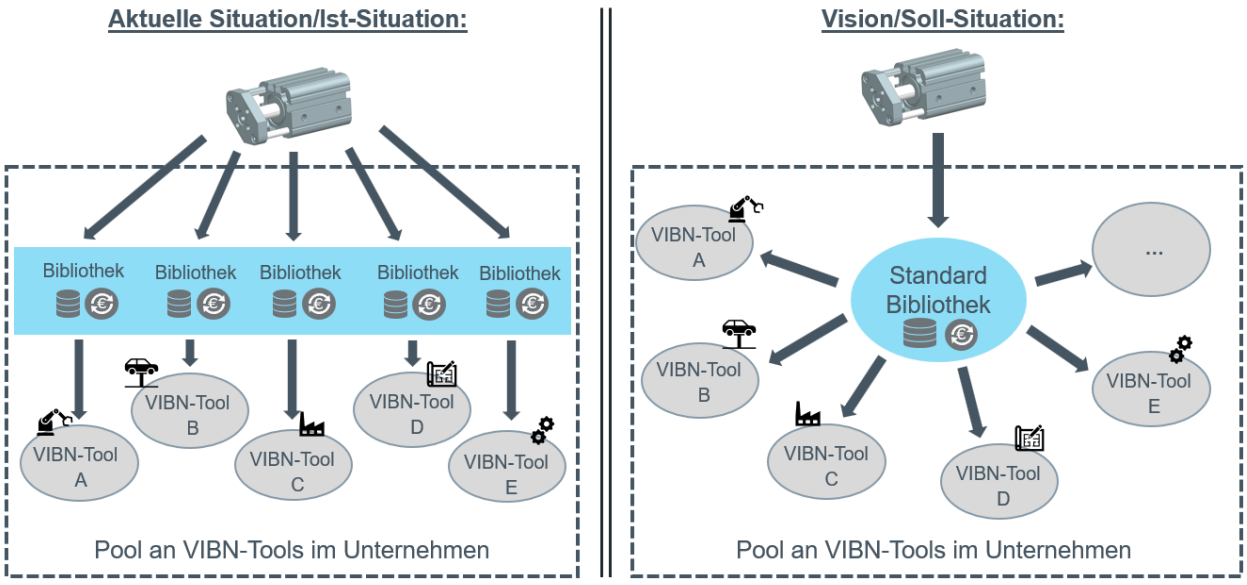

Abbildung 4: Gegenüberstellung von aktueller Ist-Situation beim OEM und der Soll-Situation (Vision)

Mit Hilfe von FMUs kann der Komponenten-Hersteller neben seinen realen Komponenten auch standardisierte Simulationsmodelle bereitstellen, welche von verschiedensten Tools angebunden werden können. Aktuell werden Verhaltensmodelle häufig auf Grundlage von Datenblättern, Handbüchern oder den SPS-Bausteinen erstellt. Somit ist lediglich eine Annäherung an das Verhalten der realen Komponente möglich. Da der Komponenten-Hersteller über seine Produkte am besten informiert ist, steigt folglich auch die Modellqualität und

Fehler im Modell werden reduziert. (vgl. Abbildung 5) Dadurch können die Verhaltensmodelle, neben der Nutzung in der VIBN, auch zur Entwicklung von SPS-Bausteinen verwendet werden, wofür aktuell meist die realen Komponenten genutzt werden.

Durch den FMI-Standard kann der Komponenten-Hersteller toolunabhängige Modelle (FMUs) bereitstellen.

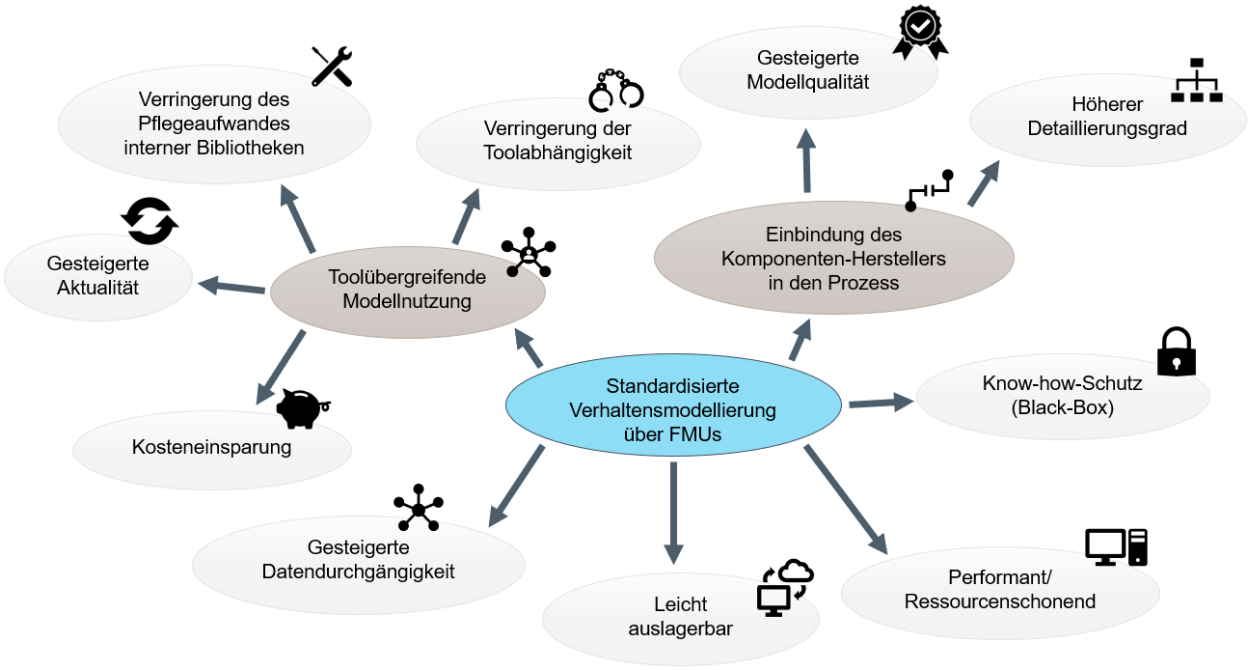

Abbildung 5: Vorteile bei der Nutzung von FMUs im Kontext der VIBN

Neben den Verhaltensmodellen für Einzelkomponenten ist es häufig auch nötig, Teilsysteme abzubilden. Beispiele hierfür bilden die Subtechnologie von Robotern sowie Messsysteme, welche teils eine eigene Steuerung beinhalten und über eine Schnittstelle mit der restlichen Anlage kommunizieren. Die Dokumentation beschränkt sich dabei meist auf die Ansteuerung. Der interne Aufbau und relevante Logiken werden nicht beschrieben, wodurch die Modellierung durch Drittanbieter meist nur oberflächlich geschieht. In diesem Fall bietet der FMI-Standard dem Systemlieferanten die Möglichkeit, neben seinem realen Modul auch ein Verhaltensmodell bereitzustellen, was die Qualität der VIBN erheblich verbessert.

Um eine realitätsnahe VIBN zu ermöglichen, werden in bestimmten Bereichen auch die Prozesse mitsimuliert. Diese können ebenfalls als standardisierte FMUs modelliert werden. Generell ist zwischen Geräte/Steuerungstechnik- und Prozess-FMUs zu unterscheiden. Auch in diesem Kontext bietet der FMI-Standard die Möglichkeit, toolunabhängige Modelle zu erstellen.

Da der FMI-Standard in verschiedenen Einsatzbereichen und Branchen Anwendung findet, sind die Funktionalitäten sehr vielfältig und die Betriebsarten unterschiedlich. Der Einstieg in die Erstellung von geeigneten FMUs ist häufig zeitaufwendig und fordert eine gewisse Einarbeitung in den Standard. Zudem ist für die FMU-Erstellung, nach jeweiliger Präferenz, ein geeignetes Tool zu finden. Hierfür sind Untersuchungen nötig, damit eigene Erfahrungen gesammelt werden können. Mit der Erstellung von standardisierten Verhaltensmodellen geht ein zusätzlicher Aufwand einher. Aus diesem Grund bieten die Komponenten-Hersteller aktuell nur vereinzelt Verhaltensmodelle als FMUs an. Auch die Integration in die jeweiligen VIBN-Tools gestaltete sich in den vergangenen Jahren sehr zögerlich. Der Bedarf eines FMU-Co-Simulators war im Produktiv-Betrieb meistens nicht gegeben. Die Anbindung von FMUs fand häufig nur im Rahmen von Forschungsprojekten oder für Demonstratoren statt.
Für die Nutzung von FMUs als Verhaltensmodelle stellen sich beim Komponenten-Hersteller somit unter anderem folgende Fragen [16]:

- Welche Arten von FMUs sind zu simulieren?
- Welche Funktionen/welcher Umfang des Standards ist im Kontext der VIBN zu implementieren?
- Welches Tool ist im jeweiligen Anwendungsfall für die FMU-Erstellung geeignet?
- Welcher Detaillierungsgrad muss für das Modell abgebildet werden?

Der flächendeckende Einsatz von FMUs in der VIBN wird nur durch die Zusammenarbeit verschiedener Rollen ermöglich. So liegt die Erstellung von FMUs bei den **Komponenten-Herstellern**. Die Verhaltensmodelle werden an den **Anlagenbauer** weitergegeben, welcher die Modelle dann zu einem digitalen Zwilling der Anlage zusammenführt (vgl. Abbildung 6). Die VIBN wird dann vom Anlagenbauer im Beisein des **Anlagenbetreibers (OEM)** durchgeführt. Um die Anbindung der FMUs im digitalen Zwilling zu ermöglichen, bedarf es einer Integration in die Simulationsumgebung durch den **Tool-Hersteller**. Der Leitfaden adressiert jede der vier genannten Rollen.

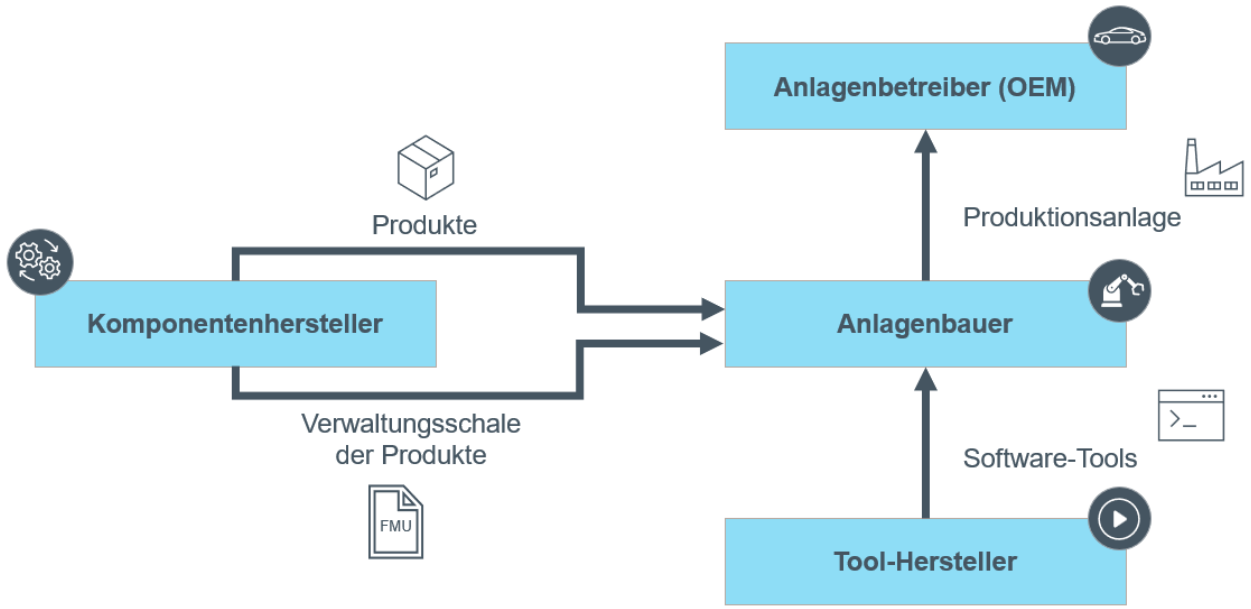

Abbildung 6: Zusammenspiel der vier definierten Rollen

Eine umfassende Integration des FMI-Standards ist nur möglich, wenn die Komponentenhersteller, Anlagenbauer, Anlagenbetreiber (OEM) sowie Tool-Hersteller mitwirken.

Aktuell ist der Standard im täglichen, produktiven VIBN-Geschäft kaum vertreten und findet vor allem Aufmerksamkeit durch verschiedene Forschungsaktivitäten und Pilot-Projekte. Die Marktdurchdringung ist entsprechend gering [16]. Die Integration des FMI-Standards muss Firmen-übergreifend erfolgen und von allen beteiligten Parteien akzeptiert werden. Das Förderprojekt DIAMOND gibt mit diesem FMU-Leitfaden einen Wegweiser, um die Integration von FMUs in den VIBN-Prozess zu beschleunigen. Dieses DIAMOND-Konsortium ist gerade dazu prädestiniert, eine Einführung des FMI-Standards im Kontext der VIBN in der Automobilindustrie bestmöglich vorzubereiten. Die Verbundpartner vertreten ein breites Spektrum an Sichtweisen auf das Thema.
Wichtig ist eine durchgängige Akzeptanz des Dokuments in der Industrie. Jede Rolle muss sich mit den Inhalten des Leitfadens identifizieren können. Aus diesem Grund wurden im Rahmen des Forschungsprojektes DIAMOND die verschiedenen Rollen getrennt voneinander über Experten-Interviews befragt. Insgesamt nahmen neun Partner aus dem DIAMOND-Konsortium aus allen vier Rollen teil. Die Interview-Partner haben bereits erste Erfahrungen mit dem FMI-Standard gesammelt und waren in der Lage, aussagekräftige Antworten und Aspekte zu liefern.
Über dieses Vorgehen wurden bisherige Erkenntnisse, Herausforderungen sowie Bedenken identifiziert und gesammelt. Diese Informationen dienten zum Ableiten von funktionalen sowie nichtfunktionalen Anforderungen, welche an den Standard gestellt werden. Der Leitfaden beschäftigt sich mit der Modell-Erstellung sowie der

Simulation von FMUs und gibt Antworten auf die Umsetzung und Integration. Des Weiteren werden Workflows beschrieben, wie der Standard in den VIBN-Prozess integriert werden kann. Da auf einen bereits bestehenden Standard gesetzt wird, gilt es, bestimmte Funktionen und Bereiche auszuklammern, welche nicht im Kontext der virtuellen Inbetriebnahme benötigt werden. Andere wichtige Themen und Stolpersteine bei der Einführung sind hervorzuheben.

4. Reading Guide

Dieses Kapitel dient dazu, dem Leser einen schnellen Einblick in den Umfang dieses Leitfadens zu gewähren. Dabei wird, abhängig vom Standpunkt und von der Zielsetzung des Lesers auf bestimmte Kapitel verwiesen.

4.1 Ein Leitfaden für verschiedene Rollen und Sichtweisen

Wie im Kapitel 3 erläutert, lassen sich 4 Rollen abgrenzen (vgl. Abbildung 7), welche in diesem Leitfaden adressiert werden. Abhängig von der jeweiligen Rolle sind einzelne Kapitel von besonderer Relevanz:

- **Komponenten-Hersteller:**
  Kapitel *5;* Kapitel *6;* Kapitel *7;* Kapitel *10;* Kapitel *11*
- **Anlagenbauer:**
  Kapitel *6;* Kapitel *8;* Kapitel *9;* Kapitel *10;* Kapitel *11*
- **Anlagenbetreiber (OEM):**
  Kapitel *6;* Kapitel *8;* Kapitel *9;* Kapitel *10;* Kapitel *11*
- **Tool-Hersteller:**
  Kapitel *5;* Kapitel *8;* Kapitel *9;* Kapitel *10*

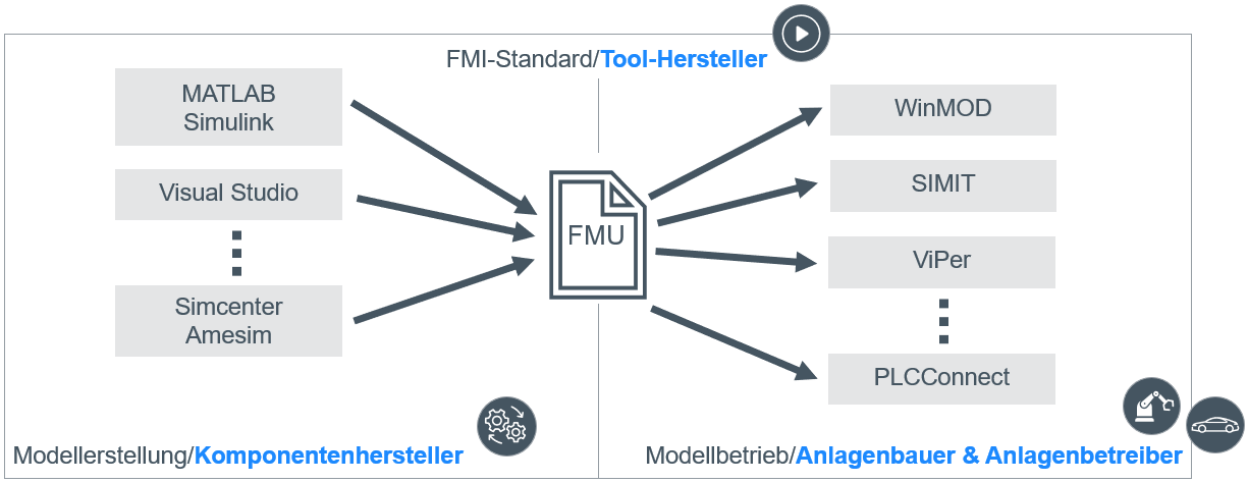

Abbildung 7: Einordnung der vier verschiedenen Rollen in den Gesamt-Kontext

4.2 Ein Leitfaden zur Diskussion von verschiedenen Anforderungen

In Verbindung mit dem FMI-Standard im Kontext der VIBN werden häufig Herausforderungen und Anforderungen am Standard geäußert, welche die flächendeckende Einführung und Nutzung von FMUs erschweren. Auch das im

Kontext von DIAMOND veröffentliche Paper „FMI in der VIBN: Ein Leitfaden zur Umsetzung“ [16] beschäftigt sich diesen Anforderungen, welche im Folgenden aufgeführt und mit den entsprechenden Abschnitten im Leitfaden verlinkt werden. Dies ermöglicht dem Leser ein sehr punktuelles und effizientes Lesen:

- **Echtzeitfähigkeit, Zykluszeit und Performance:** *Abschnitt 9.11; Abschnitt 7.4*
  Die FMU-Simulation ist von der Realzeit entkoppelt, sodass die simulierte Zeit und die Simulationszeit nicht zwingend voneinander abhängig sind. Im Kontext der VIBN sind aufgrund der geforderten „Echtzeit“ sehr kleine Zykluszeiten nötig [16].

- **Visualisierung und Nutzerintegration:** *Abschnitt 5.5; Abschnitt 7.3, Abschnitt 9.6*
  Für die VIBN werden FMUs in Realzeit betrieben. Die Simulation muss dadurch beeinflussbar sein. Somit sind Schnittstellen und die Interaktion zum Anwender nötig. FMI selbst bietet keine Visualisierung in Form eines Bedienfeldes bzw. einer Benutzerschnittstelle [16].

- **FMI-VIBN-Bibliotheken:** *Kapitel* 6; *Abschnitt 7.3; Kapitel 8; Abschnitt 9.1*
  Generalisierbare toolübergreifende Bibliotheken bringen gewisse Herausforderungen mit sich: Die Komponenten-Hersteller müssen einen geeigneten Modell-Detaillierungsgrad wählen. Das Verhalten liegt in Form einer Blackbox vor. Die VIBN-Tools müssen die FMU-Anbindung entsprechend integrieren [16].

- **Kompatibilität & Abhängigkeiten:** *Abschnitt 5.7;* Abschnitt 6.4; *Abschnitt 7.6*
  FMUs sind teilweise eigenständig nicht lauffähig und referenzieren externe Abhängigkeiten oder Lizenzmechanismen. FMUs von verschiedenen Herstellern müssen miteinander kombiniert werden [16].

- **Security:** *Abschnitt 9.9*
  Da die FMU eine gekapselte, kompilierte Binärdatei beinhaltet, gehen mit der Ausführung der FMU gewisse Sicherheits-Risiken einher. Gerade wenn die FMUs über das Internet bereitgestellt werden [16].

Die im Leitfaden genannten Software-Lösungen und Tools stellen nur Beispiele dar. Es wurden überwiegend Produkte von Konsortialpartnern aufgeführt. Auf dem Markt existieren weitere Lösungen, welche analog zu betrachten sind. Gleiches gilt auch für die genannten Feldbussysteme. PROFINET und EtherCAT werden hier nur exemplarisch erwähnt.

Der Leitfaden soll auch als solcher verstanden werden. So werden ausschließlich Empfehlungen und keine strikten Vorgaben gegeben. Das Dokument soll gerade beim Einstig in das Thema FMI unterstützen und die Einarbeitung beschleunigen. Ohne Einschränkung der Allgemeinheit ist hier der Fokus auf den Einsatz für die VIBN in der Automobilindustrie gerichtet. Die standardisierten Verhaltensmodelle sind analog in anderen Branchen nutzbar, unterscheiden sich jedoch möglicherweise im geforderten Detaillierungsgrad. Der Leitfaden fokussiert Verhaltensmodelle von Automatisierung-Komponenten. Die Kernaussagen sind jedoch auch auf Prozess-, Produkt-, Gebäude- und Modul-FMUs zu übertragen. Der Leitfaden beschreibt die Momentaufnahme zur Zeit der Veröffentlichung. Zukünftige Entwicklungen, Gegebenheiten sowie die Versionierung des Standards sind dringend zu berücksichtigen.

5. FMI-Standard in der VIBN

Der FMI-Standard ist über die Homepage der Modelica Association offen verfügbar. Die aktuelle Version 3.0.2 ist über folgenden Link einsehbar: https://fmi-standard.org/docs/3.0.2/ [11].
In diesem Leitfaden werden ausschließlich die Versionen 2.0 und 3.0 betrachtet. Im Allgemeinen beschreibt eine FMU ein im FMI-Standard erstelltes Modell. Diese wird in Form eines gezippten Containers (.ZIP File Format Specification in der Version: 6.3.10 [7]) dem Anwender bereitgestellt. Die Struktur des Containers unterscheidet sich für die verschiedenen FMI-Versionen (vgl. Abschnitt 5.1) und ist nachfolgender Abbildung 8 zu entnehmen. Einen der wichtigsten Bestandteile der FMU bildet die modelDescription.xml. Hierbei handelt es sich um eine standardisierte Modell- und Schnittstellen-Beschreibung. Die XML dient als Verknüpfung zwischen dem Anwender und dem hinterlegten Modell. So werden die Schnittstellen-Signale (Inputs, Outputs, Parameter) hier aufgeführt und mit Attributen wie Name, Beschreibung, Default-Wert oder der Einheit angereichert. Es werden allgemeine Informationen zum Modell und zur Simulation gegeben. Je nach Betriebsart der FMU liegt das Modell-Verhalten als offengelegter Source-Code im Ordner „resources“ oder als gekapselte kompilierte Datei unter „binaries“ ab [11]. Die restlichen Dateien und Informationen sind als optional zu betrachten und werden in diesem Kapitel 5 zum Teil näher betrachtet. Ebenso wird auf wichtige Bestandteile und Inhalte in der modelDescription.xml näher eingegangen.

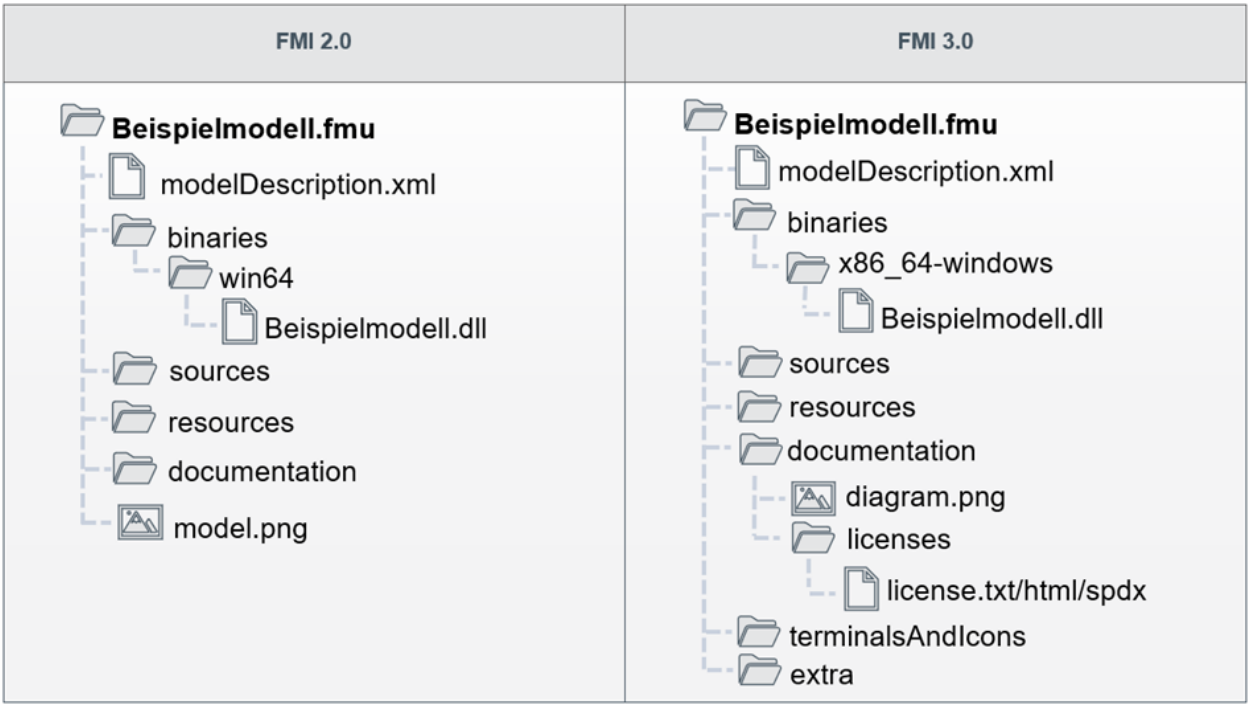

Abbildung 8: Struktur eines FMU-Containers im FMI 2.0 (links) und FMI 3.0 Standard (rechts)

5.1 Wahl der FMI-Version

Da es sich um einen Standard handelt, der stetig weiterentwickelt wird, ist es schwierig, eine bestimmte Version zu empfehlen. Es ist ratsam, möglichst neue Versionen zu nutzen. Die Anwendung im Kontext der VIBN hat gezeigt, dass ab der Version 2.0 ein ausreichender Umfang für die Modellierung und Anbindung an das VIBN-Tool im Standard enthalten ist. Ab der Version 2.0 können somit FMUs produktiv für VIBN-Projekte genutzt werden. Die Integration in die VIBN-Toollandschaft hat bereits stattgefunden. In der Version 3.0 werden zusätzliche, unter anderem komplexe Datentypen, wie z.B. Arrays, (vgl. Abschnitt 9.4) sowie weitere Funktionen unterstützt, wodurch diese Version für die Erstellung von Verhaltensmodellen im Kontext der VIBN zu bevorzugen ist. Gerade diese zusätzlichen Datentypen erlauben es, die Schnittstelle analog zur realen Komponente abzubilden. Beispielsweise kann über Arrays die PROFINET-Schnittstelle in ihrer Gesamtheit als ein einzelnes Signal abgebildet werden. Ein Umweg über aufgespaltene Einzelsignale, wie im 2.0er Standard, ist nicht nötig. Der 3.0-Standard wird bereits von vielen Erstellungs-Tools sowie Co-Simulatoren unterstützt.

Wichtig ist, dass bei einem Versionsupgrade auch Rückwärtskompatibilität gewahrt wird. Dies betrifft den Co-Simulator, welcher zusätzlich zu neuen auch alte FMU-Versionen (ab 2.0) weiterhin unterstützen muss. Zudem ist im Co-Simulator auch ein hybrider Betrieb von FMUs verschiedener FMI-Versionen (2.0 und 3.0) zu ermöglichen. Prinzipiell kann der Komponenten-Hersteller auch FMUs in verschiedenen Standard-Versionen bereitstellen, sofern er in diesem Zusammenhang Vorteile sieht.

Die Verwendung des FMI-Standards in der Version 3.0 ist im Kontext der VIBN zu bevorzugen.

5.2 Wahl des FMI-Modus

Der FMI-Standard unterstützt verschiedene Modi, wie ein Simulationsmodell standardisiert ausgetauscht werden kann. So existieren der so genannte „ModelExchange“, die „Co-Simulation“, und in der Version 3.0 der Modus „Scheduled Execution“.

Vielen Komponenten-Herstellern ist es wichtig, ihr Know-how zu schützen. Hier bietet der FMI-Standard die Möglichkeit, das Modellverhalten über kompilierte Binärdateien auszutauschen. Entscheidend ist dies vor allem, sobald für die Modellierung Software bzw. Source-Code aus der realen Komponente integriert wird. Einfache Logikmodelle sind hierbei weniger kritisch. Durch die gekapselten, kompilierten Dateien wird das Modell in Form einer Blackbox übergeben. Unter Windows wird dies über eine DLL- und unter Linux über die so-Datei realisiert. Dies ist für den Anwender nicht optimal, da er keine Einblicke in das interne Verhalten hat und Anpassungen am Modell nicht möglich sind. Bei ausreichend validierten Modellen entfallen diese Bedarfe, sodass die Akzeptanz dieser Blackbox-Modelle steigt.

Alternativ oder optional kann das Verhalten in der FMU über einen standardisierten C-Code hinterlegt und dem Zielsystem zur Verfügung gestellt werden. Der Inhalt wird somit über die Tool-Grenzen hinweg offengelegt. Eine Vertraulichkeitsvereinbarung (NDA) zum Know-how-Schutz zwischen Ersteller und Anwender ist dringend nötig, sofern schützenswerter Source-Code mit in der FMU abgelegt wird.

5.2.1 ModelExchange

Beim ModelExchange wird das Model vom Solver des Zieltools berechnet (vgl. Abbildung 9). Somit wird eine gute Modell-Performance, parallele Berechnung sowie eine hohe Genauigkeit erreicht. Da die verschiedenen Ziel-Tools unterschiedliche Solver verwenden, kann das Modellverhalten möglicherweise im jeweiligen Anwendungsfall voneinander abweichen.

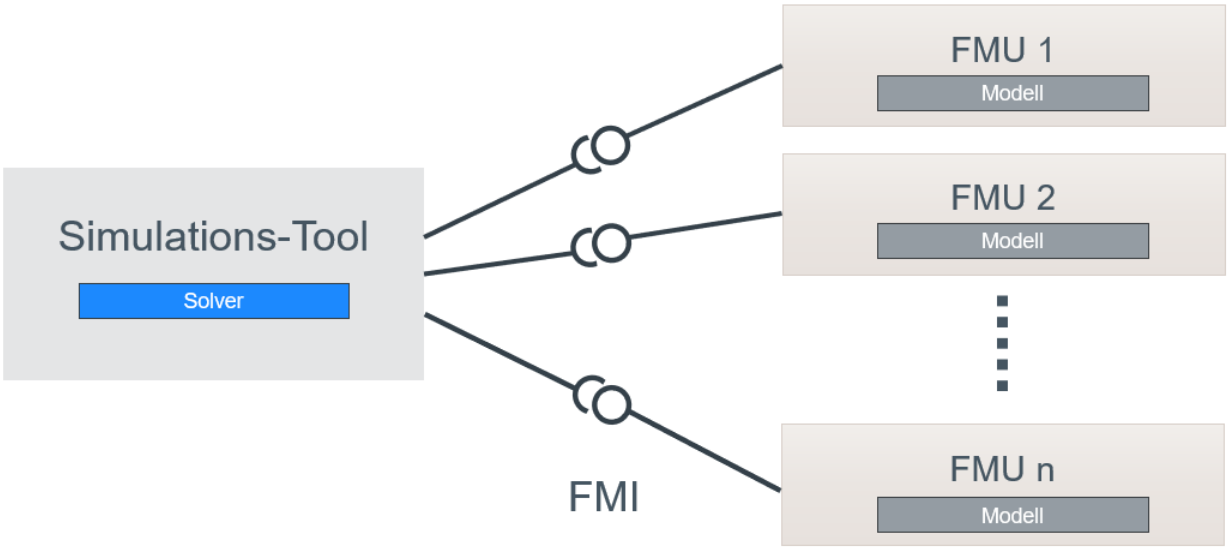

Abbildung 9: Skizzierte Funktionsweise einer FMU im Modus ModelExchange nach [34].

Eine Bereitstellung von FMUs im Modus ModelExchange ist für den Komponenten-Hersteller gerade bei komplexen Modellen in Ausnahmesituationen denkbar. Gerade im Austausch mit anderen Modellen oder FMUs bietet eine direkte Integration ins Zielsystem eine bessere numerische Stabilität, da der gleiche Solver verwendet wird. Dazu werden Informationen wie die kontinuierlichen Zustände und ihre Ableitungen (oft auch als Ausgangssignale) offengelegt. Die FMUs können dann zusammengefasst und über einen Solver als Co-Simulation extern oder im Ziel-Tool simuliert werden. Im Kontext der VIBN stellt der ModelExchange nicht den präferierten und gängigen Weg dar. Somit sind bei derartigen Sonderfällen spezielle Beauftragungen nötig.

5.2.2 Co-Simulation

Im Gegensatz zum ModelExchange steht der Modus Co-Simulation, wobei hier eine kompilierte Datei mitsamt dem entsprechenden Solver übergeben wird. Die Co-Simulation ist die robusteste Art der Anwendung und liefert auch bei verschiedensten Tool-Setups ein gleiches, reproduzierbares Verhalten, da der Solver direkt in die FMU integriert ist. Im Falle der Co-Simulation werden die einzelnen FMUs über einen Co-Simulations-Master (ohne Solver) simuliert. Die FMU beinhaltet dabei sowohl das Modell als auch den entsprechenden Solver (vgl. Abbildung 10). Aus diesem Grund ist dieser Modus für den Einsatz in der VIBN besonders geeignet.
Zu beachten ist allerdings, dass aufgrund der gekapselten und unabhängigen Berechnung der einzelnen Teil-Modellen bei stark gekoppelten Komponenten (z.B. Regler, physikbasierten Rückmeldungen) Probleme auftreten können. Die Kopplung der FMUs geschieht ausschließlich über die Ein- und Ausgänge, wodurch numerische Fehler auftreten können. In diesem Kontext wird auch die Abarbeitungsreihenfolge der einzelnen Modelle (FMUs) von Bedeutung.

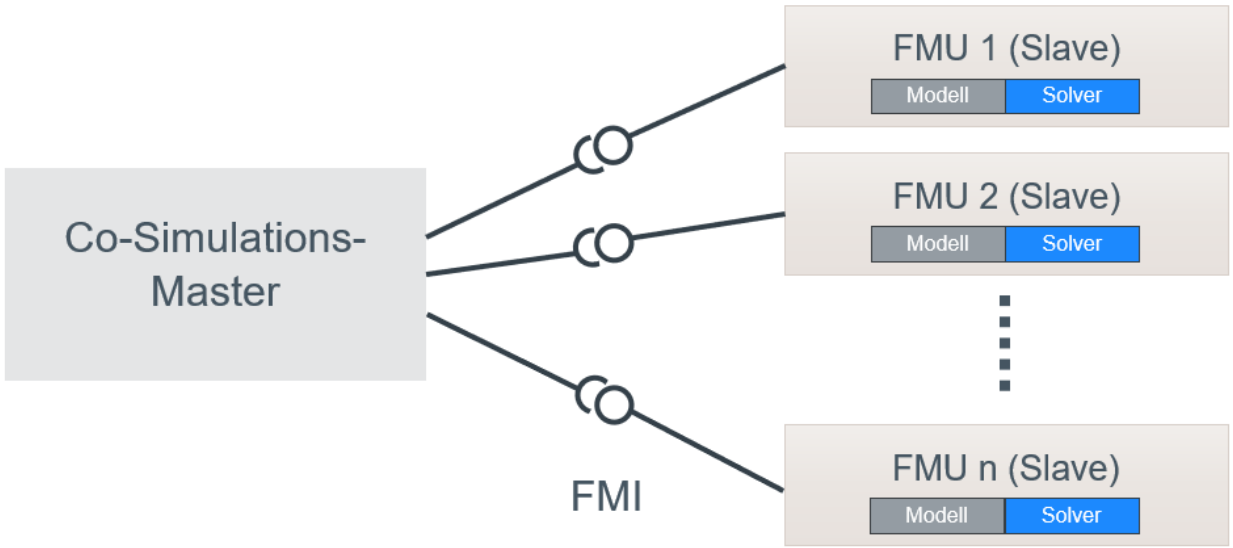

Abbildung 10: Skizzierte Funktionsweise einer FMU im Modus Co-Simulation in Anlehnung an [11].

5.2.3 Scheduled Execution

Neben den beiden genannten Modi unterstützt der Standard in der Version 3.0 die so genannte „Scheduled Execution“. Hier können verschiedene Modell-Partitionen gezielt getaktet und nacheinander ausgeführt werden. Für komplexe Verhaltensmodelle könnte dieser Modus in Zukunft relevant werden.

FMUs sind über die Co-Simulation zu erstellen und anzubinden.

5.3 Wahl des Zielsystems

Da ein Großteil der VIBN-Tools in **Windows**-Umgebung läuft, ist die Variante mit der **64Bit-DLL** zu wählen. Die 32Bit-Version wird nur noch vereinzelt verwendet, sodass eine klare Empfehlung auf die 64Bit-DLL ausgesprochen wird. Der Standard erlaubt es, zusätzlich auch Linux-Dateien zu hinterlegen. Dies wird vor allem im Kontext Edge Computing, Cloud oder bei Hochleistungsrechnern in Zukunft von immer größer werdender Bedeutung und bringt einen stabileren Betrieb von FMUs mit sich. Für einen flexiblen Einsatz kann die FMU mittel- bis langfristig sowohl Windows als auch Linux Binärdateien enthalten.

FMUs sind aktuell als 64Bitt-Variante über Windows zu entwickeln und anzubinden.

5.4 Einträge in der modelDescription.xml

Die modelDescription.xml definiert die Schnittstelle zwischen der Blackbox (DLL) und dem Co-Simulator. Dabei werden Informationen und Einstellungen bei der FMU-Erstellung hinterlegt. Der Anwender bzw. der Co-Simulator konsumiert diese Inhalte. Eine nachträgliche Änderung oder Anpassung der modelDescription.xml (z.B. vom FMU-Anwender) ist nicht vorgesehen.

5.4.1 DefaultExperiment

In der modelDescription.xml existiert ein Bereich namens „DefaultExperiment“. Hierunter werden Angaben zur Simulation gemacht. Beispiele sind die insgesamte Simulationsdauer für das Modell, welche über die Start- und Stopp-Zeit definiert wird, oder eine mögliche (meaningful) Zykluszeit (vgl. Abbildung 11). Der Co-Simulator kann die hier gemachten Angaben für die Simulation nutzen und berücksichtigen. Die Informationen unter „DefaultExperiment“ sind optional zu hinterlegen. Eine gute entwickelte FMU führt hier die zu verwendenden Einstellungen auf. Falls in diesem Bereich keine Angaben gemacht wurden, sind vom Co-Simulator geeignete Default-Werte zu verwenden. Bei der stepSize werden häufig 0,001s (1ms) verwendet. Die stopTime wird auf die maximal mögliche Zeit („Inf“) gestellt, sodass die FMU möglichst lange läuft. Vor allem bei detaillierten FMUs ist es zu empfehlen, eine ideale stepSize vorzugeben. Gerade bei Differenzialgleichungen oder Einregel-Vorgängen können lange Zykluszeiten die Ergebnisse negativ beeinflussen.

```
<DefaultExperiment startTime="0" stopTime="3" stepSize="1e-2"/>
```

Abbildung 11: Beispielhafter Ausschnitt aus einer modelDescription.xml im Bereich Default Experiment.

5.4.2 Verwendung von verschiedenen Flags

Der FMI-Standard erlaubt es, in der modelDescription.xml bestimmte Flags zu setzen (vgl. Abbildung 12). Diese werden bereits bei der FMU-Erstellung festgelegt und sind als konstant zu betrachten. Über diese Einstellungen

können Informationen über Einschränkungen und Einstellungen für die Simulation an den Co-Simulator kommuniziert werden. Der Co-Simulator hat diese Flags zu prüfen und eine Hinweismeldung zu geben, sobald implementierte Funktionen nicht unterstützt werden. Der Co-Simulator kann die Flags nicht ändern. Solange eine Flag in der modelDescription.xml nicht aufgeführt wird, gilt diese als defaultmäßig „false“ [11]. Es ist dennoch eine gute Praxis, auch Flags mit der Einstellung „false“ in der XML aufzuführen.

```
canHandleVariableCommunicationStepSize="true"
canBeInstantiatedOnlyOncePerProcess="false"
canGetAndSetFMUState="true"
canSerializeFMUState="true"
```

Abbildung 12: Beispielhafter Ausschnitt aus einer modelDescription.xml im Definitions-Bereich der verschiedenen FMI-Flag

canHandleVariableCommunicationStepSize:
Wird diese Flag nicht gesetzt („false“), so erwartet die FMU immer eine bestimmte konstante Zykluszeit. Verschieden große, schwankende Zykluszeiten führen zu einem Fehler. Die FMU kann damit nicht umgehen. Aufgrund von unterschiedlichen Auslastungen in der Windows-Umgebung ist es sinnvoll, eine variable Zykluszeit zu erlauben. Im VIBN-Kontext kann unter Umständen auch mit einer fixen Stepsize umgegangen werden.

canBeInstantiatedOnlyOncePerProcess:
Über diese Flag wird dem Co-Simulator mitgeteilt, dass dieser die FMU pro Prozess nur einmal instanziieren darf. Die Automatisierungs-Komponente kann somit pro Prozess (Tool) nur einmal simuliert werden. Hintergrund für diese Flag sind verschiedene Arten, wie mit internen Variablen und Zuständen umgegangen wird (z.B. globale Variablen, Singletons). Werden gleiche Speicherbereiche von verschiedenen Instanzen genutzt, ist diese Flag zu setzen, da bei einer Mehrfachinstanziierung kein korrektes Modellverhalten möglich ist. Um mehrere Instanzen der FMU zu simulieren, dürfen die erstellten Modelle keine gemeinsamen Ressourcen nutzen. Im Kontext der VIBN laufen viele Co-Simulatoren auf einem Prozess und verteilen die FMUs in einzelne Threads. Aus diesem Grund führt diese Flag häufig zu Problemen. Sind mehrere gleiche Komponenten in einer Anlage verbaut, kommt es zu Komplikationen im digitalen Zwilling. Im Kontext der VIBN ist eine Mehrfachinstanziierung dadurch essenziell (vgl. Abschnitt 9.8). Diese Flag (*canBeInstantiatedOnlyOncePerProcess = true*) ist somit bei der FMU-Erstellung zu berücksichtigen. Die Fähigkeit einer Mehrfachinstanziierung ist häufig abhängig vom Export-Tool, sodass möglichst ein Erstellungs-Tool zu verwenden ist, welches diese Flag nicht setzt (*canBeInstantiatedOnlyOncePerProcess = false*).

canGetAndSetFMUState:
Ist es nötig, dass von einer FMU zu einem bestimmten Zeitpunkt der Zustand gelesen wird, ist diese Flag von Relevanz und muss auf „true“ gesetzt sein. Ebenfalls kann der Zustand vom Co-Simulator gesetzt sowie freigegeben werden. Der erfragte Zustand geht allerdings beim Simulationsstopp wieder verloren. Selbst wenn die FMU diese Funktion unterstützt, ist es im Co-Simulator nicht zwingend notwendig, die Funktionen aufzurufen.

canSerializeFMUState:
Sobald diese Flag auf „true“ gesetzt wird, muss auch *canGetAndSetFMUState=true* definiert werden. Durch die Flag canSerializeFMUState wird es dem Co-Simulator ermöglicht, Zustände, Inputs sowie Modellparameter aus der FMU zu erhalten und intern zu speichern. Auf diese Weise können die Werte beim nächsten Simulationsstart wieder gesetzt werden, sodass die Simulation zu einem gewissen Punkt wieder fortgesetzt wird. Selbst wenn die FMU diese Funktion unterstützt, ist es im Co-Simulator nicht zwingend notwendig, die Funktionen aufzurufen. Gerade im Bereich der azyklischen Bus-Kommunikation stellt dies eine bedeutende Funktionalität dar. Damit diese Funktionalität auch genutzt werden kann, muss sie im Co-Simulator entsprechend implementiert sein.

Viele VIBN-Tools zur Verhaltensmodellierung bieten die Möglichkeit eines Recorders bzw. Snapshots (Momentaufnahme eines Systems zu einem bestimmten Zeitpunkt). Dadurch kann ein bestimmtes Signalbild aufgenommen und wieder abgespielt werden. So können Fehlerzustände und Abläufe in der VIBN gezielt reproduziert werden. Die Flag *canSerializeFMUState* teilt mit, ob diese Funktionalität auch FMU-seitig unterstütz wird. In diesem Zusammenhang ist generell zu beachten, dass die FMU auch mit großen Wertsprüngen an den Inputs umgehen kann. Abstürze oder fehlerhafte Berechnungen aufgrund von Unstetigkeiten und großen Wertänderungen innerhalb eines Zyklus sind zu verhindern. Die FMU ist auch im Hinblick auf derartige Extremfälle zu validieren.

Die FMU-Flags sind auf Ersteller-Seite bei der Tool-Auswahl zu berücksichtigen. Im Leitfaden wird folgende Empfehlung gegeben:
canHandleVariableCommunicationStepSize = true
canBeInstantiatedOnlyOncePerProcess = false
canGetAndSetFMUState = true (erst zukünftig relevant)
canSerializeFMUState = true (erst zukünftig relevant)

5.4.3 Einträge unter Model Info

In diesem Bereich der modelDescription.xml sind allgemeine Informationen (Metadaten) zu hinterlegen (vgl. *Tabelle 3*). Darunter fallen beispielsweise der Autor, der Modell-Name, das Generierungs-Tool und das Datum sowie die Uhrzeit beim Bauen des Modells. Zusätzlich kann auch eine kurze Beschreibung über die Funktion und

den Umfang der FMU gegeben werden. Vom Co-Simulator werden nicht alle diese Informationen für die Simulation dringend benötigt und gelten deswegen für die FMU-Erstellung zum Teil als optional. Die Angaben unter Model Info können ausgelesen und über das VIBN-Tool (Co-Simulator) dem Anwender angezeigt werden (z.B. FMU-Blockschaltbild mit Metadaten im VIBN-Tool). Bei einer hohen Integrationsstufe von FMUs ist diese Visualisierung sinnvoll. Auch für die Ablage und die Integration (z.B. in Verwaltungsschale, System Structure and Parameterization (SSP), …) von FMUs sind diese Metadaten von Interesse. Unter diesen Attributen gibt es Angaben, welche dringend in der FMU zu hinterlegen sind. Dazu zählen der modelName sowie die fmiVersion und eine eindeutige ID.

Folgende Tabelle gibt einen Überblick der verschiedenen Attribute und unterteilt sie in optionale und verpflichtende Angaben:

Verpflichtende Attribute (fmiVersion 2.0 & 3.0)	
fmiVersion	FMU im 2.0 oder 3.0 Standard erstellt
modelName	FMU-Name analog zum Container-Name
GUID (fmiVersion 2.0) / instantiationToken (fmiVersion 3.0)	Eindeutige ID der FMU
Optionale Attribute (fmiVersion 2.0 & 3.0)	
description	Kurze Erklärung des Modells
author	Name des Autors
version	Modellversion
copyright	Copyright des Modells
license	Informationen über Lizenzen
generationTool	Name des Erstellungs-Tools
generationDateAndTime	Erstellungs-Datum und -Uhrzeit

Tabelle 1: Auflistung verschiedener Attribute aus der modelDescription.xml mit einer Einordnung in verpflichtende und optionale Angaben.

```
fmiVersion="3.0"
modelName="BouncingBall"
description="This model calculates the trajectory, over time, of a ball..."
generationTool="Reference FMUs (development build)"
instantiationToken="{8c4e810f-3df3-4a00-8276-176fa3c9f003}">
```

Abbildung 13: Beispielhafter Ausschnitt aus einer modelDescription.xml im Definitions-Bereich der verschiedenen Modell-Attribute (Model Info).

Diese Attribute geben im Allgemeinen Aufschluss darüber, wann die FMU erstellt wurde und woher sie stammt. Bei Problemen oder zur besseren Rückverfolgbarkeit können diese Informationen von extremer Bedeutung sein.

Wichtige Verwaltungs-Informationen wie die GUID (fmiVersion 2.0) bzw. der instantiationToken (fmiVersion 3.0) sowie die Version und der modelName müssen gepflegt werden, um eine korrekte Instanziierung beim Co-Simulator zu ermöglichen. Gerade für einen Modell-Update-Prozess in bestehenden Projekten sind diese Informationen von enormer Bedeutung. Bei unvollständigen oder schlecht gepflegten Attributen wird dieser Vorgang erschwert oder unter Umständen sogar unmöglich, was einen manuellen Update-Prozess oder die Neuverknüpfung von Signalen mit sich bringt. Auch die Angabe zur Lizenz ist als verpflichtend anzusehen, sobald ein Lizenzierungs-Mechanismus verwendet wird. Die Angabe des Generierungs-Tools hilft, Probleme bei der Co-Simulation frühzeitig einzuschränken und macht den Umgang mit FMUs transparenter.

Exkurs GUID/instantiationToken:
Die GUID (fmiVersion 2.0) bzw. der instantiationToken (fmiVersion 3.0) stellt eine eindeutige ID dar, welche bei der Modell-Erstellung generiert wird. Für verschiedene Modelle gilt es verschiedene IDs zu hinterlegen. Die GUID oder der instantiationToken dienen zur Prüfung, ob die modelDescription.xml und die kompilierte DLL jeweils zusammenpassen. Hierfür wird der Methode für die Instanziierung der FMU die ID aus der modelDescription.xml mitgegeben. Passen beide IDs nicht zusammen, schlägt die Instanziierung im Co-Simulator fehl [11]. Bei Änderungen in der modelDescription.xml ist, laut Standard, eine Anpassung der ID zwingend erforderlich [11]. Aufgrund der angepassten „generationDateAndTime“ findet bei jedem Build eine Anpassung in der XML statt. Somit wird in den meisten Tools bei jedem Build/Export der FMU eine neue ID generiert. Die GUID ist als String zu interpretieren, sodass auch Anführungszeichen oder geschweifte Klammern gültig sind. Dieser String muss global eindeutig sein und darf nicht mehrfach verwendet werden.

In der modelDescription.xml sind zwingend der modelName, die fmiVersion sowie die GUID bzw. der instantiationToken aufzuführen. Die restlichen Angaben gelten als Good Practices und sollten hinterlegt und gepflegt werden.

5.5 Umgang mit Bild/Icons im FMU-Container

Der FMI-Standard erlaubt es, in den gezippten Container optional auch ein Bild abzulegen. Hierfür wird in der obersten Ebene ein Bild mit der Bezeichnung „model.png“ abgelegt (FMI 2.0). Dieses kann später bei der Simulation angezogen und für Visualisierungszwecke verwendet werden. Im Idealfall ist ein Bild zu hinterlegen, welches einen Wiedererkennungswert der realen Komponente liefert. Denkbare Abbildungen sind hier das Logo des Komponenten-Herstellers, ein Bild der realen Komponente oder eine CAD-Abbildung. Möglich wäre auch, das Logo des FMI-Standards zu hinterlegen (vgl. Abbildung 14). Auf diese Weise würde dem Anwender im Falle einer hybriden Modellnutzung gezeigt werden, ob es ich um eine FMU oder ein natives Bibliothekselement handelt (vgl. Abschnitt 8.1). Von manchen FMU-Erstellungs-Tools werden automatisiert Bilder im Container hinterlegt. Zu

beachten ist, dass durch das Bild keine unbeabsichtigten Informationen über den Inhalt des Modelles (z.B. Screenshot vom Source-Code) nach außen geführt werden. Bei der Integration von FMUs in das VIBN-Tool kann das Bild dazu dienen, das Simulationsmodell der Gesamtanlage übersichtlicher zu gestalten. Dies ist vor allem bei einer hohen Integrationsstufe von FMUs erstrebenswert. Das Bild/Icon kann im jeweiligen Blockschaltbild dargestellt werden (vgl. Abbildung 14). Dabei wird die FMU als Blackbox mit den dazugehörigen Inputs, Outputs und den Parametern gezeigt.

Abbildung 14: Beispielhafte Darstellung eines Blockschaltbildes mit dem FMI-Logo als Icon.

Die Bildgröße sollte jedoch geringgehalten werden, damit das Handling der Dateien nicht erschwert und der Speicherbedarf nicht unnötig groß wird. Optional kann im Ordner „documentation“ eine .html-Datei hinterlegt werden, welche Informationen über die dazugehörigen Bildrechte enthält. Aktuell wird die Integration eines Bildes in vielen VIBN-Tools noch nicht unterstütz. Bezogen auf die unter Kapitel 10 genannten *weiterführenden Standards* besteht auch die Möglichkeit, das Icon in einer übergeordneten Schicht (z.B. in der Verwaltungsschale (AAS)) zu hinterlegen bzw. zu verlinken.

Das Hinterlegen eines Bildes im FMU-Container gilt aktuell als optional.
(zukünftig relevant)

5.6 Integration der Dokumentation

Zusätzlich bietet der Standard auch die Möglichkeit, im Container die Dokumentation (vgl. Abschnitt 7.3) des Verhaltensmodelles zu hinterlegen. Als Format für die Dokumentation ist .html oder .txt zu wählen. Somit ist auch ein entsprechender Export aus anderen Tools (z.B. Word) möglich. PDF wird laut Standard nicht unterstützt [11]. Anstelle der Daten-Ablage kann auch ein Link hinterlegt werden, welcher auf die Dokumentation verweist. Falls diese über das Internet bereitgestellt wird, ist zu beachten, dass das VIBN-Setup häufig in einem gekapselten Netzwerk isoliert ist und keinen Zugang zum Internet hat. Der Verweis auf das Dokument gewährleistet eine

möglichst schlanke Gestaltung der FMU, wird allerdings im Kontext der VIBN nicht empfohlen. Allerdings ist die Dateigröße der Dokumentation zu berücksichtigen, damit das Datenhandling nicht unnötig erschwert wird. Im VIBN-Tool (Co-Simulator) ist darauf zu achten, dass selbst bei einer umfangreichen Dokumentation die Performance (beim Simulationsstart aber auch während der Simulation) nicht negativ beeinflusst wird. In diesem Fall ist der Zeitpunkt für das Entpacken der FMU günstig zu wählen.

Insgesamt ist es gute Praxis, die Dokumentation über den FMU-Container bereitzustellen. Auf diese Weise kann der Co-Simulator (VIBN-Tool) auf die Dokumentation referenzieren und diese analog zur nativen „Hilfe-Funktion“ öffnen. Dies ermöglicht, gleichermaßen zur Hilfe bei nativen Verhaltensmodellen, die Dokumentation von FMUs zu verknüpfen und zu öffnen. Da die FMU selbst eine Blackbox darstellt, ist die Dokumentation ein essenzieller Bestandteil bei der VIBN. Gerade bei einer hohen Integrationsstufe von FMUs ist es für den Anwender von Bedeutung, direkt über das VIBN-Tool (Co-Simulator) auf die entsprechende Dokumentation zugreifen zu können. Diese Vorgehensweise stellt auch sicher, dass die Dokumentation zur vorliegenden FMU passt und mit der jeweiligen Versionierung übereinstimmt. Eine getrennte Ablage von Dokumentation und Modell kann zu Diskrepanzen führen. Aktuell ist es allerdings noch gängige Praxis, die Dokumentation dem Anwender getrennt von der FMU als PDF-Datei anzubieten.

Bezogen auf die unter Kapitel 10 genannten *weiterführenden Standards* besteht auch die Möglichkeit, die Dokumentation in einer übergeordneten Schicht (z.B. AAS) zu hinterlegen. Zusätzlich kann die Dokumentation auch auf der Internetseite des Komponenten-Herstellers zum Lesen angeboten werden. In der FMU sind in diesem Zusammenhang entsprechende Verweise, IDs und Links zu hinterlegen. Sollten vom FMU-Ersteller (Komponenten-Hersteller) verschiedene Bereitstellungsarten angeboten werden, ist auf Anwender-Seite darauf zu achten, die Dokumentation möglichst nicht redundant abzulegen.

Die Dokumentation ist mindestens in englischer Sprache zu erstellen. Gerade für den Einsatz im deutschsprachigen Raum sollte auch zusätzlich eine deutsche Fassung hinterlegt werden. Weitere Sprachen, welche im Kontext der VIBN häufig Verwendung finden, sind Spanisch, Französisch, Italienisch und Chinesisch.

Eine Dokumentation der FMU muss bereitgestellt werden. Aktuell kann die Dokumentation auch gesondert (z.B. als PDF) geliefert werden.
(zukünftig Integration in die FMU relevant)

5.7 Integration von externen Abhängigkeiten

Im FMI-Standard besteht die Möglichkeit, auf externe Bibliotheken oder Compiler (z.B. Python) zuzugreifen. Derartige Abhängigkeiten sind dem Anwender über eine Readme oder über die Dokumentation (vgl. Abschnitt 7.3) mitzuteilen, damit dieser nötige Installationen durchführen kann. Teilweise lassen sich FMUs nur im

Zusammenhang mit dem Generierungstool simulieren. Dies ist jedoch nicht das Ziel eines Standards. Im Kontext der VIBN müssen FMUs eigenständig laufen. Generell sind externe Abhängigkeiten somit zu vermeiden.

Externe Abhängigkeiten sind dringend zu vermeiden.

5.8 Nutzung von Logging in der FMU

Da die FMU eine gekapselte Blackbox bildet, ist es von enormer Bedeutung, Log-Meldungen nach außen zu führen. Dies kann zum einen über einen Log-Output geschehen, welcher verschiedene Zustände (z.B. Zustandsautomat), Fehler oder Diagnosen bereitstellt (häufig auch in der realen Komponente vorhanden), oder zum anderen über das Logging. Der FMI-Standard hat hierfür ein konkretes Vorgehen hinterlegt und verschiedene Log-Kategorien definiert. Gerade bei Fehlerfällen (die FMU geht beispielsweise in einen Fehler-Zustand/Error) ist der Anwender auf eine Log-Meldung angewiesen, um die Gründe für das Verhalten zu verstehen (Debugging). Ohne Logging wird dem Anwender keine Möglichkeit geboten, das Fehlverhalten der FMU näher zu analysieren und zu verstehen, da ihm kein Source-Code vorliegt. Gerade bei komplexen FMUs ist eine Logging Funktionalität sehr sinnvoll und hilfreich. In bestimmten Situationen (z.B. beim Einlesen oder Schreiben von externen Daten) oder beim Aufbau von Verbindungen (z.B. TCP/IP) ist es relevant, die internen Rückgaben von bestimmten Funktionen an den Anwender weiterzugeben. Auch in Bezug auf Lizenzierungen oder zeitlich begrenzte FMUs ist ein entsprechendes Logging essenziell. Bei Modellen mit einem zeitlichen Verhalten ist es sinnvoll, die simulierte Zeit in gewissen Situationen auszugeben. Auf diese Weise kann der Anwender überprüfen, ob die FMU im Co-Simulator korrekt aufgerufen wird. Es ist darauf zu achten, Meldungen nicht in jedem Simulationszyklus (doStep) zu loggen. Ein derartiges Vorgehen führt dazu, dass die Log-Datei vollläuft und die Analyse erschwert wird. Durch ein zu umfangreiches Logging können Performance-Probleme entstehen. Ein gutes und gezieltes Logging zeichnet eine qualitative FMU aus.

Das Logging hilft dem Komponenten-Hersteller bei der Optimierung und Weiterentwicklung seiner FMUs. So kann er vom Anwender Log-Files einfordern, welche bei Problemen der gezielten Analyse dienen. Allerdings ist darauf zu achten, dass die Log-Meldungen vom Anwender eingesehen werden können. In diesem Zusammenhang ist zu erwähnen, dass hier unter Umständen Know-how preisgegeben wird. Log-Meldungen sollten keine Informationen über die Implementierung in der realen Komponente preisgeben.

Auf der Co-Simulator-Seite müssen die Log-Meldungen auch an den Anwender weitergegeben werden und sollten nicht ins Leere führen. In den meisten Fällen werden die Log-Ausgaben an das Tool-Logging weitergegeben. Alternativ besteht die Möglichkeit, die FMU-Logs in einem separaten Log-File bereitzustellen. Im Idealfall kann der Anwender entscheiden, welche Log-Kategorien (fmiOK, fmiError, fmiFatal …) er nach außen führen will. Auf diese

Weise wird ein gezieltes Logging ermöglicht. Es muss im Co-Simulator die Option angeboten werden, das FMU-Logging gezielt zu aktivieren und zu deaktivieren.
Werden die vom Standard definierten Log-Kategorien sinnvoll eingesetzt und von beiden Seiten (Co-Simulator und Erstellungs-Tool) unterstützt, kann auf eigene interne Log-Mechanismen verzichtet werden. Hierunter fallen beispielsweise Parameter, über welche das Logging aktiviert und im Umfang definiert werden kann (z.B. Debug-, VIBN-Bit).

Gezieltes Logging wird empfohlen und bietet gerade beim Debugging einen enormen Mehrwert.

5.9 Nutzung von Terminals

Ab der FMI-Version 3.0 ist es möglich, so genannte Terminals zu definieren. Dabei werden Gruppen von Variablen gebildet und eine zusätzliche Ebene in die Input-/Output-Struktur eingefügt. Auf diese Weise können physikalische Verbindungen sowie Kommunikationsstrukturen zwischen FMUs leichter abgebildet werden. [11] Für VIBN-Tools ist es sinnvoll, diese neuen Umfänge zukünftig zu integrieren. Allerdings bedarf eine umfassende Nutzung von Terminals einer Standardisierung. Hierfür müssen die verwendeten Signale, die dazugehörigen Datentypen und auch die jeweiligen Einheiten klar definiert sein. Gerade im Hinblick auf die azyklische Kommunikation (z.B. IO-Link) können Terminals den Aufwand beim Verknüpfen der einzelnen FMUs deutlich verringern. Auch im Zusammenhang mit pneumatischen oder elektrischen Netzen erleichtern Terminals die Interaktion zwischen FMUs und ermöglichen einen höheren Detaillierungsgrad.

Die Kommunikation zwischen FMUs über Terminals gilt aktuell als optional. (zukünftig relevant)

6. Modellumfang

6.1 Detaillierungsgrad/Level of Detail (LoD)

Der Detaillierungsgrad eines Verhaltensmodells zählt sicherlich zu den am meisten diskutierten Punkten im Kontext von FMUs. So steht ein einfaches oberflächliches Modell einer hochkomplexen detaillierten Nachbildung der realen Komponente gegenüber (vgl. Abschnitt 6.1.4). Im Kontext der VIBN bedarf es einem sinnvollen Mittelweg, da in jedem Fall die simulierte Zykluszeit eingehalten werden muss. Performance-Probleme durch eine zu detaillierte Modellierung gilt es zu vermeiden. Die folgenden Abschnitte sollen den Komponenten-Hersteller unterstützen, einen geeigneten Detaillierungsgrad im Kontext der VIBN zu finden. Verhaltensmodelle im FMI-Standard können ebenfalls für die Konzeptphase, das Engineering sowie für den Betrieb der Automatisierungsanlage verwendet werden. Dieser Leitfaden beschränkt sich allerdings auf den Einsatz in der VIBN.

6.1.1 Logisches Verhalten

Gerade in Verbindung mit der VIBN bildet die Logik einen essenziellen Bestandteil des Verhaltensmodelles. Hier wird von der FMU ein exaktes logisches Verhalten gegenüber der Steuerung erwartet. Der Detaillierungsgrad muss so hoch sein, dass die Steuerung keinen Unterschied zur realen Komponente bemerkt. Nur so ist es möglich, das Steuerungsprogramm ohne Anpassungen an den digitalen Zwilling anzubinden. Hierfür muss das EA-Feld in Richtung SPS im vollen Umfang implementiert sein. Generell sind auch Parameter der realen Komponente analog zu definieren. Prozessdaten der Bus-Kommunikation stellen neben direkt verdrahteten EAs einen wesentlichen Bestandteil des Logik-Modelles dar. Bei verschiedenen Telegrammen (z.B. PROFIdrive) werden spezifische Ein-/Ausgangsworte zur Anbindung an die SPS benötigt, welche vom Verhaltensmodell zur Verfügung gestellt werden müssen. Diese gilt es, über Parameter oder unterschiedliche Verhaltensmodelle zu variieren. Gerade bei komplexen Komponenten reicht es teils aus, nur elementare Teile der Logik zu implementieren. Viele Sonderfunktionen werden unter Umständen nur in Ausnahmefällen verwendet.

Fehlertrigger sind entsprechend komponentenspezifisch abzubilden. Diagnose-Ausgänge, welche von der SPS eingelesen und ausgewertet werden, sind im Verhaltensmodell abzubilden. Im minimalen Fall muss hierfür ein Input auf den Diagnose-Ausgang durchgeschliffen werden.

Anzeige- und Eingabe-Elemente der realen Komponente sind für die VIBN nicht zwingend relevant und somit (in den meisten Fällen) als optional zu betrachten. Beispiele für diese Elemente bilden Fehlerlämpchen oder Lampen, welche Hinweise auf den aktuellen Betriebszustand geben (vgl. Abbildung 15). Unter Eingabeelementen werden z.B. Taster verstanden, welche zur händischen Ansteuerung oder Parametrierung dienen. Gerade im Kontext von Bedienerschulungen bietet die Modellierung diese Elemente einen deutlichen Mehrwert.

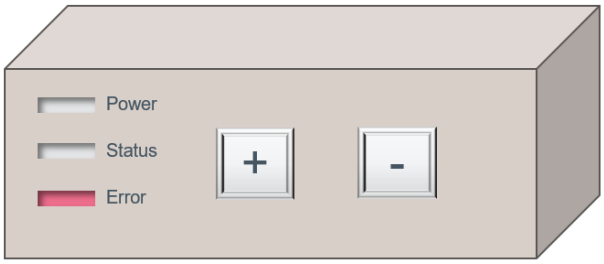

Abbildung 15: Abstrakte Darstellung von Eingabe-Elementen in Form von Anzeige-LEDs und Tastern („+“/„-“)

Bezogen auf die EAs (SPS-Signale) muss sich die FMU im Hinblick auf das logische Verhalten analog zur realen Komponente verhalten.

6.1.2 Zeitverhalten

Häufig sind Teile der Logik auch mit einem zeitlichen Verhalten angereichert. Zu nennen sind beispielsweise Lifebits (*periodische Taktsignale zur Funktionsüberwachung der Komponente*) oder zeitlich gesteuerte Impulse und Flanken. Auch die Verarbeitungszeit in der Komponente (oft bei azyklischer Kommunikation) stellt ein zeitliches Verhalten dar.

Das physikalische oder mechanische Verhalten ist in einem entsprechenden Detaillierungsgrad im Zusammenhang des Zeitverhaltens nötig, um Prozesse realitätsnah zu beschreiben. Auf diese Weise können Taktzeiten analysiert und optimiert werden. Durch ein einfaches zeitliches Verhalten (z.B. abstrahierte lineare Rampen bei Antrieben oder Zylindern) ist eine bessere Absicherung in der VIBN möglich. Im Idealfall werden Verfahrwege anstelle von linearen Annahmen über Beschleunigung und Verzögerung möglichst realitätsnah nachgestellt. Im einfachsten Fall ist dies über ein Weg-Zeit-Diagramm ohne dynamisches Verhalten zu realisieren. Durch ein derartiges Zeitverhalten können Bewegungen in der Visualisierung deutlich flüssiger dargestellt und Abläufe besser simuliert werden. In Abbildung 16 ist ein beispielhafter Verfahrweg anhand der Position und Geschwindigkeit bezogen auf die Zeit skizziert. Dabei werden verschiedene Detaillierungsgrade betrachtet. Im ersten Fall findet ein Sprung in der Position statt. Die Geschwindigkeit wird nicht abgebildet. Im mittleren Beispiel wird von einem Sprung in der Geschwindigkeit ausgegangen, wodurch die Positionsänderung linear angenähert wird. Im letzten Fall wird die Bewegung mit einer Beschleunigung und Verzögerung beschrieben. Der Detaillierungsgrad steigt zunehmend an. Bei Bewegungen im Kontext der VIBN ist die dritte Option zu bevorzugen, da Sprüngen zu Problemen in der Visualisierung führen können.

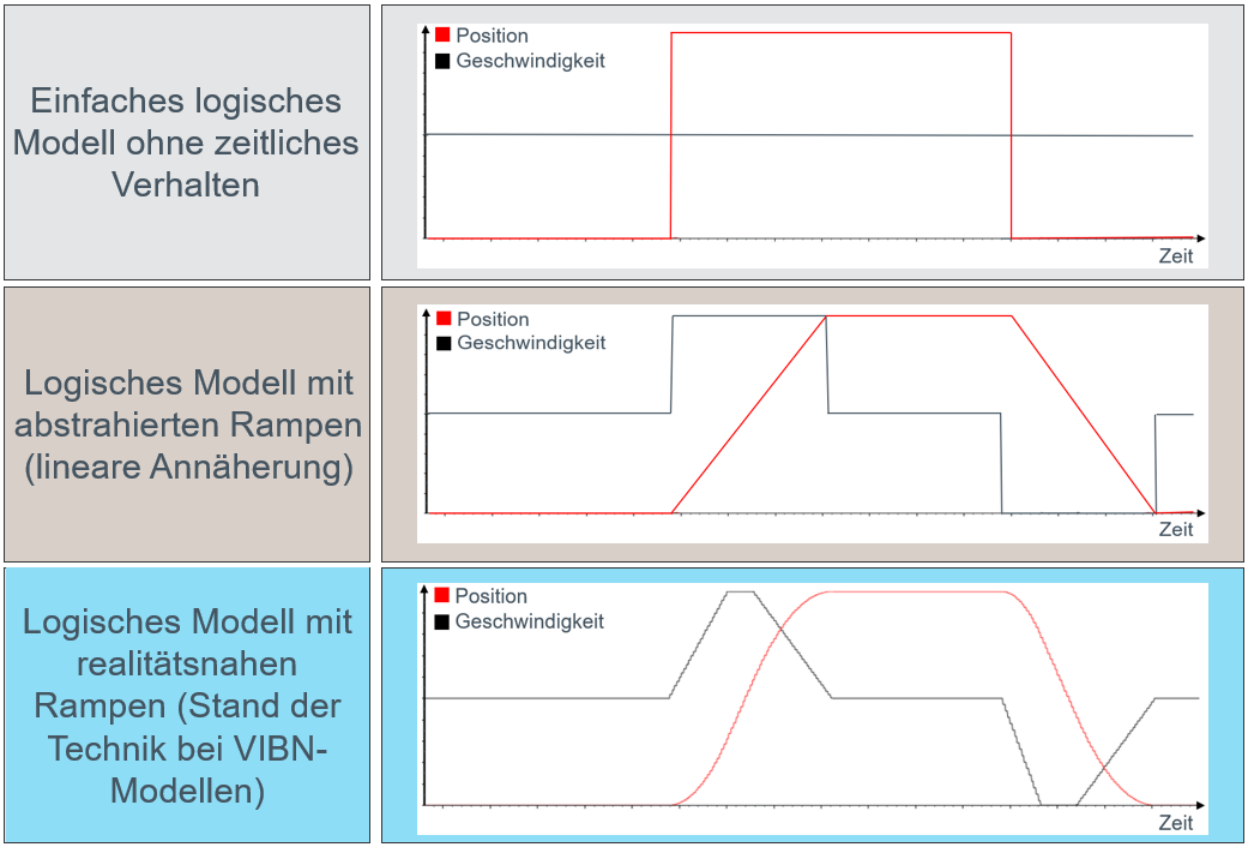

Abbildung 16: Beispielhaft skizziertes Zeit-Verhalten eines Antriebes mit Position und Geschwindigkeit in verschiedenen Detaillierungsgraden.

Zeitlich getaktete Signale und Prozesse (Bewegungen) sind abzubilden, sodass Überwachungszeiten und Abfolgen in der SPS validiert werden können.

6.1.3 Physikalisches Verhalten

Ein komplexes physikalisches Verhalten und das Modellieren von Reibung, Trägheit, Kräften, Momenten, thermischen sowie chemischen Verhalten, etc. wird in den meisten Anwendungsfällen nicht benötigt. Ebenfalls ist es nicht notwendig, ein detailliertes elektrisches Verhalten sowie dynamische mechanische Prozesse, wie Schwingungen und Vibrationen, zu simulieren. Derartige physikalische Aspekte werden meist bereits vorgelagert in der Konzeptphase, Dimensionierung und Auslegung (z.B. durch Sizing-Tools, Mehrkörpersimulation (MKS), ...) analysiert und betrachtet. Aus diesem Grund wird in der VIBN häufig ein logischer, idealisierter Materialfluss abgebildet und die Physik außen vorgelassen.

Bei einer realitätsnahen Simulation des Materialflusses mit Fördertechnik muss die Physik mit betrachtet werden. So sind physikalische Zusammenhänge in der Komponente abzubilden. Interne Trägheit, Reibung, etc. sind vom Komponenten-Hersteller zu modellieren. Es ist darauf zu achten, dass mit einer detaillierten Physik-Simulation im Verhaltensmodell die Komplexität des gesamten Simulationsmodelles steigt. So hat eine Kraft- und Momenten-Rückmeldung der Anlagenmechanik an das Verhaltensmodell zu erfolgen. Hierfür sind korrekte Reibungswerte und Gewichte sowie die Trägheit in der Visualisierung zu hinterlegen. Die Performance und die damit verbundene Zykluszeit im Visualisierungs-Tool stellt hier häufig einen limitierenden Faktor dar. Ebenfalls müssen die Schnittstellen zwischen Verhaltensmodell und Visualisierung performant genug sein, um den Einregelvorgang zwischen Antrieb und physikalischer Rückmeldung zu ermöglichen. Aktuell ist dies nicht immer gegeben, wodurch hochdetaillierte Komponenten-Modelle häufig nicht sinnvoll in den digitalen Zwilling integriert werden können. Die Ergebnisse aus der Simulation sind dadurch häufig nicht realitätsgetreu. Eine zeit- bzw. taktsynchrone Kopplung zwischen den VIBN-Tools (vgl. Abschnitt 9.11) kann in diesem Kontext unterstützen.

Die Simulation eines nicht idealisiert betrachteten Bewegungsvorgangs (vgl. vorheriger Absatz) ist im Kontext der VIBN in der Automobilindustrie selten nötig. In anderen Bereichen, wie z.B. dem Sondermaschinenbau spielen derartige Betrachtungsweisen inzwischen eine immer größer werdende Rolle.

Aktuell wird die Modellierung von physikalischen Zusammenhängen (Reibung, Kraft, Trägheit, …) meist nicht gefordert. Zukünftig werden die Anforderungen steigen.

6.1.4 Definition von verschiedenen Detaillierungsgraden

Eine konkrete Definition eines bestimmten Detaillierungsgrads ist nicht pauschal möglich, da, abhängig von der jeweiligen Komponente und vom Umfang der VIBN, verschiedene Betrachtungsweisen nötig sind. So ist der Detaillierungsgrad eines induktiven Näherungs-Sensors anders zu definieren als der eines Technologie-Systems, wie beispielsweise einer Schweißzange mit Subkomponenten. Durch die Definition von verschiedenen LoDs kann, je nach Bedarf, ein passender Modell-Umfang gewählt werden. Es ist zwischen „Basic“- und „Advanced“-Funktionalitäten zu unterscheiden. Im Rahmen des Leitfadens wird eine dreistufige Untergliederung vorgenommen (vgl. Abbildung 17):

- **S: „zeitlich abstrahiert, so viel, dass die SPS nicht meckert“**

  Beschreibt das Verhalten der EAs zueinander. Hier wird nur so viel abgebildet, wie für den Betrieb der Komponente benötigt wird, sodass die SPS keine Fehlermeldung wirft. Dieser Detaillierungsgrad bildet das Gegenstück zum SPS-Baustein und kann auf dessen Grundlage entwickelt werden. Auf diese Weise wird gegengleich die relevante Bus-Schnittstelle (z.B. PROFINET) abgebildet. In diesem

Detaillierungsgrad sind nur gängige Fehlerfälle über Fehler-Trigger zu realisieren. Damit eine VIBN ermöglicht wird, sind zeitliche Abfolgen im Signalfluss sowie einfache Bewegungen in Form von Rampen (falls vorhanden) zu implementieren. Dieser Detaillierungsgrad deckt die Mindestanforderungen und ist mit derzeitigen Bibliothekselementen in gängigen VIBN-Tools vergleichbar. Der LoD in der Basisversion („S“) sollte nicht über die in der Komponenten-Dokumentation bereitgestellten Informationen hinausgehen.

- **M: „hochdetailliertes Logikmodell mit zeitlich korrektem Verhalten“**

  Bildet das logische Verhalten analog zur realen Komponente ab. Alle Fehlerfälle (vgl. Abschnitt 6.6) und Parameter, welche in der Dokumentation der Komponente beschrieben sind, werden modelliert. Dieses Modell liefert eine höhere Detaillierung als bisherige Verhaltensmodelle (in VIBN-Tool-Bibliotheken).

- **L: „Verhalten analog zur realen Komponente mit Physik“**

  Dieser Detaillierungsgrad beschreibt eine „eins zu eins“ Abbildung der realen Komponente. Gewisse Teile der realen Firmware sind im Modell vorhanden. Wichtige physikalische Zusammenhänge werden im Modell berücksichtigt. Dies ermöglicht eine aussagekräftige Taktzeit-Analyse. Auch Untersuchungen von Luftdruck, Trägheit und sonstigen physikalischen Zusammenhängen sind möglich. Es ist zu beachten, dass derartig detaillierte FMUs (bezogen auf die Parameter-Anzahl) noch einfach in der Bedienung und Anbindung bleiben.

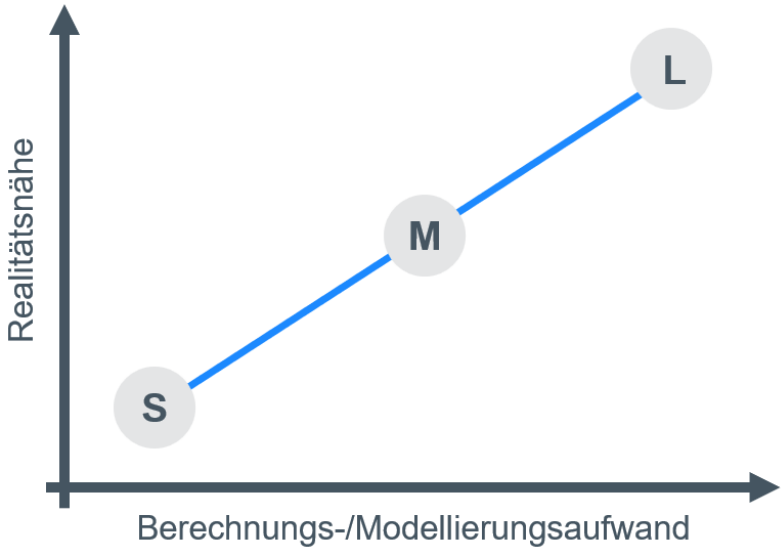

Abbildung 17: Einordnung der verschiedenen Detaillierungsgrade („S“/ „M“/ „L“) bzgl. Realitätsnähe des Modells und des Berechnungs- sowie Modellierungsaufwandes

Weniger umfangreiche Automatisierungskomponenten ohne mechanisches Verhalten, wie beispielsweise Sensoren oder Kommunikationsmodule, beschränken sich häufig auf die Stufen „S“ und „M“. Hier gilt es die logischen Grundfunktionen abzubilden. Komplexe Modellierungsumfänge für die Stufe „L“ sind hier möglicherweise die Simulation des Stromflusses über die jeweiligen Pins und Ports zur Abbildung von Überlasten. Komplexere Komponenten, wie beispielsweise elektrische Greif-Systeme oder Antriebs-Systeme, sind nur schwer in der Stufe „S“ abbildbar, da hier eine Vielzahl an Prozessdaten und Parameter zu integrieren sind. Möglicherweise existieren hier nur die Stufen „M“ und „L“. Der Umfang ist auch hier stark abhängig von der Komponente. Elektrische Komponenten (z.B. Antriebe) haben gerade im Bereich der Logik und der Firmware einen deutlich größeren Umfang vergleichen zu rein mechanischen Komponenten (z.B. Zylinder).

Generell können im Engineering-Prozess verschiedene Detaillierungsstufen für die jeweiligen Anwendungsszenarien verwendet werden. Möglicherweise sind SPS-Standardmodule (Bausteine) der Komponenten über die Stufe „M“ zu validieren. Hierbei werden alle möglichen Fehlerfälle durchgetestet. Die VIBN mit den bereits validierten Standardmodulen findet dann über die Stufe „S“ statt. Das Modell „S“ muss zwingend so detailliert sein, dass eine VIBN damit durchgeführt werden kann. Im Idealfall haben die verschiedenen Detaillierungs-Stufen gleiche Inputs und Outputs (Benennungen sowie Datentypen), sodass diese leicht austauschbar sind. Das Simulationsmodell kann somit bei Bedarf einfach mit einem detaillierten Modell angepasst werden.

Die verschiedenen Detaillierungsstufen können auch unterschiedlich bereitgestellt und lizenziert werden (vgl. Abschnitt 7.6). Insgesamt ist bei hochdetaillierten Modellen (vgl. Kapitel 11) zu bedenken, dass die FMU (analog zur realen Komponente) für Reverse-Engineering-Zwecke missbraucht werden könnte. Auch wenn der Code gekapselt vorliegt, können logische Zusammenhänge ausgewertet werden.

Für den jeweiligen Anwendungsfall können Komponenten in verschiedenen Detaillierungsgraden modelliert werden. Im Leitfaden wird zwischen drei Detaillierungsstufen unterschieden.

6.2 Anbindung an die Steuerung (Buskommunikation)

Generell ist zu berücksichtigen, dass nicht alle Automatisierungskomponenten über Feldbussysteme (z.B. PROFINET, EtherCAT, …) angeschlossen sind. Gerade nicht elektrische Komponenten (z.B. Pneumatik-Zylinder) kommunizieren teils nur indirekt (z.B. über Ventilinseln) mit der Steuerung, wodurch eine SPS-Anbindung in diesen Fällen entfällt. Im Kontext von elektrischen Antrieben oder sonstigen Busteilnehmer stellt dieser Abschnitt eine wichtige Basis, auch in Bezug auf den Detaillierungsgrad, dar.

In aktuellen VIBN-Projekten findet die Anbindung des Verhaltensmodells an die Steuerung durch das VIBN-Tool statt. In diesen Tools sind bereits Anbindungen an die realen sowie virtuellen Steuerungen implementiert. Der Bus selbst wird in vielen Fällen emuliert. Kurz- bis mittelfristig sind die FMUs über das jeweilige VIBN-Tool an die Steuerung anzubinden (vgl. Abbildung 18). Generell ist das Ziel der VIBN, die SPS-Logik abzusichern. Es wird davon ausgegangen, dass das Bus-Telegramm in der realen Komponente entsprechend korrekt funktioniert. Aus diesem Grund ist die realgetreue Abbildung von Bus-Protokollen aktuell nicht relevant.

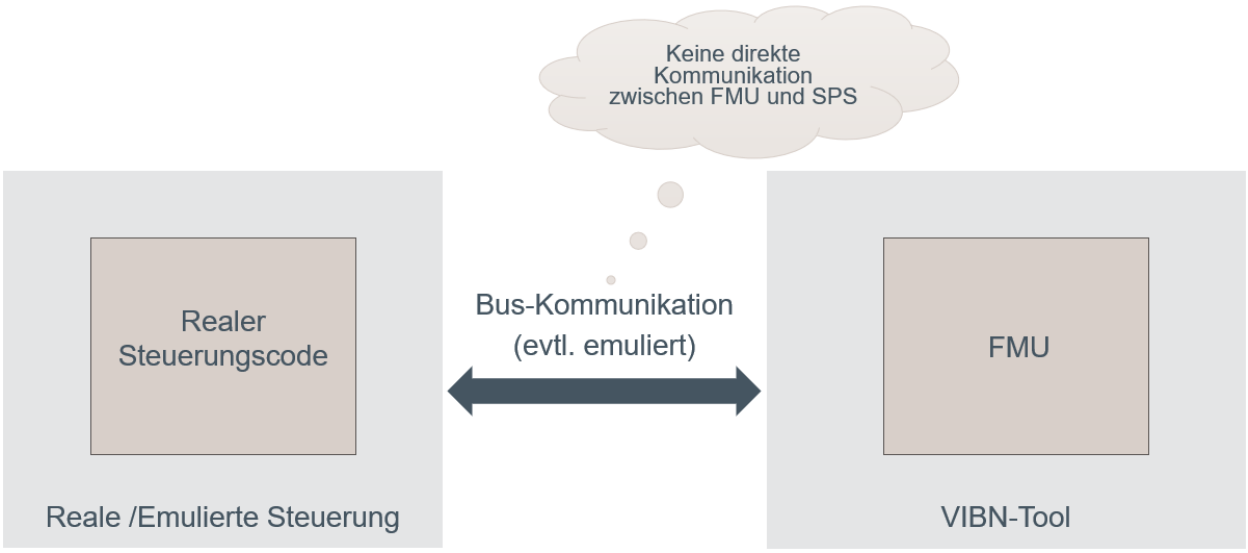

Abbildung 18: Anbindung einer FMU an den realen Steuerungscode (SPS) über ein VIBN-Tool (Co-Simulator)

Langfristig wäre eine Anbindung direkt an die virtuelle/reale Steuerung durch die FMU jedoch denkbar. Das Modell fungiert dann, analog zur realen Komponente, als Teilnehmer am entsprechenden Bus (z.B. PROFINET, EtherCAT, …). Neben der komplexen Implementierung sind aufgrund von Bus-Topologien evtl. auch Herausforderungen an die Hardware (Ethernet/Netzwerkkarten, Switches, …) zu betrachten.

Im Kontext der FMU-Erstellung ist es möglich, Bestandteile der realen Busanbindung für die Generierung und Parametrierung zu nutzen. So können beispielsweise standardisierte XML-Beschreibungen, wie Generic Station Description Markup Language (GSDML) oder IO Device Description (IODD), ausgelesen werden. Auf diese Weise wird eine teilautomatisierte FMU-Erstellung ermöglicht und der händische Implementierungsaufwand reduziert. Die daraus gewonnenen Informationen können auch für übergeordnete Beschreibungen und Standards (AutomationML (AML), Asset Administration Shell (AAS)) genutzt werden (vgl. Kapitel 10).

Die Anbindung an die jeweilige (reale/virtuelle) Steuerung wird aktuell vom VIBN-Tool übernommen und ist für die FMU-Erstellung nicht relevant.

6.3 Integration von azyklischer Kommunikation

Neben der im vorherigen Abschnitt betrachteten zyklischen Kommunikation nutzen einige Automatisierungskomponenten die azyklische Kommunikation. Auf diese Weise werden oft Parameter ausgetauscht und weniger zeitkritische Daten übertragen. Aktuell wird diese Funktionalität in den Verhaltensmodellen häufig nicht berücksichtigt. Auch wenn die reale Komponente azyklische Dienste unterstützt,

werden diese im SPS-Programm teils nicht genutzt. In Zukunft wird jedoch die Integration von azyklischer Kommunikation weiter ansteigen, was eine Modellierung in der FMU unumgänglich macht. Diese Entwicklung zeichnet sich bereits im Sondermaschinenbau ab.

Neben busspezifischen Protokollen existieren auch zusätzliche azyklische Anbindungsmöglichkeiten. Darunter fallen zum Beispiel OPC UA, TCP/IP oder IO-Link. Die Verwendung von azyklischer Kommunikation steigt mit den immer komplexer werdenden Automatisierungskomponenten weiter an. Aktuell wird die azyklische Kommunikation häufig über das VIBN-Tool abgebildet oder über ein externes Tool gebrückt. Diese Trennung zwischen Verhaltensmodell und azyklischer Kommunikation (VIBN-Tool) stellt eine Herausforderung an die Schnittstelle zum Verhaltensmodell (FMU) dar. So müssen die eingelesenen Parameter und Einstellungen an die FMU weitergegeben werden. Dies sind jedoch häufig mehrere hunderte Variablen. Gerade bei nicht standardisierten Protokollen und Kommunikationsmechanismen über TCP/IP oder OPC UA ist es sinnvoll, die Kommunikation mit in die FMU zu integrieren. Die azyklische Kommunikation auf Busstandard (z.B. PROFINET, EtherCAT) ist in der aktuellen Situation über das VIBN-Tool zu realisieren. Technisch ist eine Integration in die FMU zwar möglich, sollte aber aufgrund von Performance und vor allem aus Stabilitätsgründen derzeit nicht erfolgen. Falls die Kommunikation außerhalb der FMU stattfindet, ist das Modell durch gängige VIBN-Tools zu testen und zu validieren.

Kommunikations-Methoden wie TCP/IP oder OPC UA sind nur zu implementieren, solange auch die reale Komponente über diese Mechanismen verfügt (Nachbildung). Für interne Funktionen, wie das Speichern von Daten, die Parametrierung (sofern nicht analog zur realen Komponente), die Kommunikation zwischen FMUs, die Anbindung an das Netzwerk etc., ist der Aufbau von derartigen Kommunikations-Methoden aufgrund von Cyber-Security-Themen zu vermeiden.

Im aktuellen VIBN-Tool-Umfeld wird für die Integration von azyklischer Kommunikation folgende Empfehlung (vgl. Tabelle 2) gegeben. Es handelt sich dabei um eine grobe Richtlinie, welche im jeweiligen Kontext spezifisch zu bewerten ist:

Kommunikation	**Kommentar**	**Integration FMU**	**Integration VIBN-Tool**
azyklisches PROFINET (Siemens)	Emulation oder Anbindung an API der virtuellen Steuerung nötig	(x)	x
azyklisches EtherCAT (Beckhoff)	Emulation oder Anbindung an API der virtuellen Steuerung nötig	(x)	x
TCP/IP	meist sehr komponentenspezifisch	x	(x)
UDP	meist sehr komponentenspezifisch	x	(x)
OPC UA	meist sehr komponentenspezifisch	x	(x)
azyklisches IO-Link	da keine direkte Kommunikation mit der SPS, meist abstrakt gehalten	x	(x)

Tabelle 2: Übersicht der verschiedenen azyklischen Kommunikationswege und deren Eignung für die Integration in die FMU (x = Empfehlung; (x) = technisch möglich, aber nicht zu empfehlen)

Falls die azyklische Kommunikation im VIBN-Tool umgesetzt wird, ist eine zyklische Schnittstelle zur FMU zu implementieren, welche die eingelesenen Parameter zyklisch an die FMU weitergibt. Diese ist möglichst einfach und einheitlich zu gestalten. Analog zur zyklischen Kommunikation bieten auch hier standardisierte XML-Beschreibungen der realen Komponente die Möglichkeit, Informationen für die Modellierung sowie Parametrierung bereitzustellen. Beispielsweise kann im Falle von IO-Link hierfür die IODD-Datei verwendet werden.

Sofern azyklische Dienste im bereitgestellten Verhaltensmodell nicht vorhanden sind, jedoch für die VIBN benötigt werden, ist der FMU-Ersteller (Komponenten-Hersteller) zu kontaktieren. Auf Anwender-Seite ist es wichtig, derartige Bedürfnisse frühzeitig zu kommunizieren, damit die Relevanz von azyklischer Kommunikation auf Ersteller-Seite platziert wird.

Generell wird im Zusammenhang mit einer azyklischen Kommunikation das Thema Speicherung von FMU-internen Zuständen und Werten zunehmend wichtiger. Viele azyklische Parameter werden beim Starten der Steuerung geschrieben. Bei einem FMU-Start-Stopp gehen diese Werte verloren, wodurch auch Probleme bei der VIBN entstehen können. Ein Speichern von FMU-Zuständen kann über canGetAndSetFMUState sowie canDe/SerializeFMUState (vgl. Abschnitt 5.4.2) realisiert werden.

Azyklische Kommunikation spielt im Hinblick auf die Parametrierung eine immer größer werdende Rolle. Abhängig von der Kommunikationsart sind verschiedene Umsetzungen möglich. (zukünftig relevant)

6.4 Umfang der FMU (Einzelkomponente vs. Gesamtverbund)

In vielen Anwendungsfällen stellt sich die Frage, wo die Systemgrenzen einer Automatisierungs-Komponente im Kontext eines Verhaltensmodelles für die VIBN zu setzen sind. Als Beispiel ist hier ein Pneumatik-Zylinder zu nennen. Hierbei besteht die Möglichkeit, den Zylinder als Verbund aus Ventilen, Endanschlägen, Sensoren und dem eigentlichen Zylinder zu modellieren und bereitzustellen (vgl. Abbildung 19). Alternativ fällt die Aufgabe der Kombination von Einzelmodellen auf den Anwender ab. Hierdurch steigt die Flexibilität. Der Aufbau gestaltet sich analog zum realen Gesamtverbund. Ein weiteres Beispiel bildet die Positionier-Achse, welche aus Stromversorgung, Antrieb, Welle, Frequenzumrichter, Encoder und evtl. Sensoren zusammengesetzt ist.

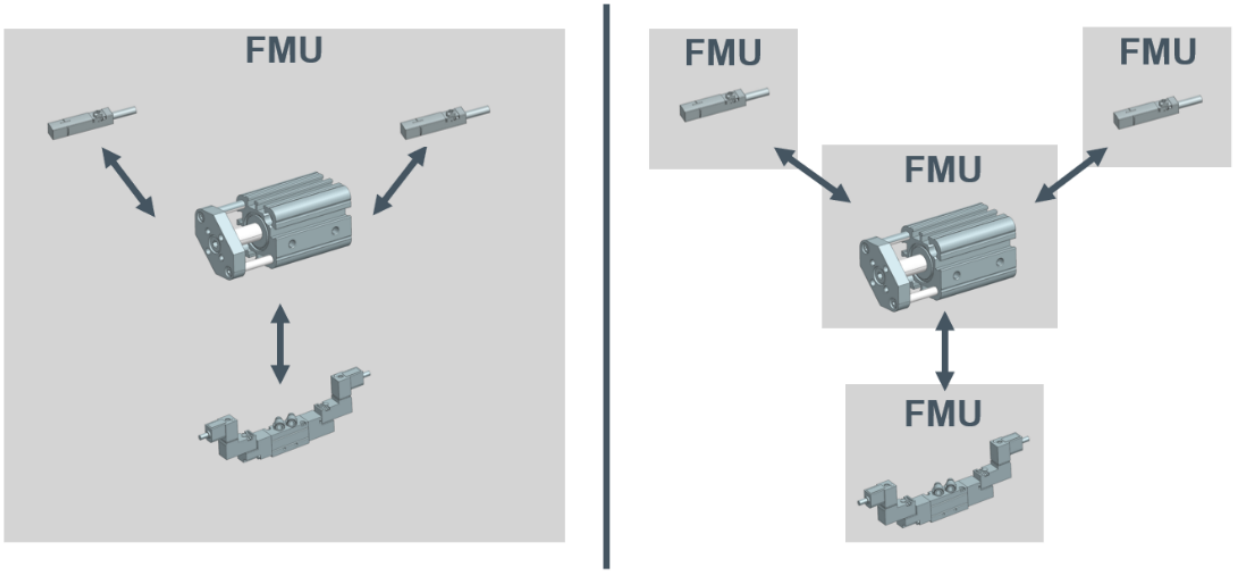

Abbildung 19: Abgrenzung zwischen Modul- und Einzelkomponenten-FMUs am Beispiel eines Pneumatik-Zylinders

Abhängig vom Umfang des Modelles müssen gegebenenfalls auch bestehende Assistenzen und Automatismen zur Modell-Erstellung angepasst werden. Die Zusammenfassung von mehreren Komponenten zu einer FMU erschwert eine Zuweisung der Signale über Betriebsmittelkennzeichen (BMKs), da gesamte Module zu verknüpfen sind. Das VIBN-Modell wird in vielen Fällen über die Stückliste (z.B. EPLAN) generiert. Hier erschweren die Modul-

FMUs den Prozess, da mehrere Artikel/Bestellnummern über eine Gesamt-FMU abgebildet werden. Somit ist der Weg über die Kombination von Einzelkomponenten durch den Anwender in den meisten Fällen der geeignetere.
Je feingranularer die Einzelkomponenten aufgelöst werden, desto schwieriger ist es, die Systemgrenzen zu definieren. Die Schnittstellen der FMU sind hierfür möglichst generalisiert und offen zu gestalten, damit Komponenten-FMUs von verschiedenen Herstellern miteinander verknüpft werden können. Die Abgrenzung der Komponente gestaltet sich analog zur Realität. In diesem Zusammenhang ist auf numerische Instabilitäten zu achten. Die FMU ist hier möglichst robust zu entwickeln. Kritische Situationen sind gezielt abzufangen. Generell sind für die VIBN lediglich die EAs zur Steuerung von Interesse.
Wenn eine gesamte Baugruppe von einem einzelnen Komponenten-Hersteller bereitgestellt wird, ist es sinnvoll, das Verhaltensmodell als Ganzes abzubilden. Das Verhalten des Gesamtmodells wird durch eine FMU oder SSP (durch mehrere FMUs vgl. Abschnitt 10.1) beschrieben. Zweiteres hätte den Vorteil, dass der Modellierungs-Umfang situativ kombinierbar ist. Für den Fall, dass eine Baugruppe oder eine Vorrichtung aus Komponenten verschiedener Hersteller zusammengesetzt wird (z.B. Motor und Umrichter von verschiedenen Herstellern), müssen die Einzelkomponenten miteinander kombiniert werden. Als Beispiel ist ein Greifer zu nennen, welcher an einer Ventilinsel angeschlossen ist.
Im Falle von gekapselten Applikationen mit einer eigenen Steuerung (z.B. Schweiß-, Klebe- oder Schraub-Applikationen) kann eine FMU für das gesamte System direkt vom Hersteller bereitgestellt werden. Dieser kennt sowohl das interne Verhalten als auch die genutzten Komponenten und hat relevante Dokumente sowie Spezifikationen bereitliegen. Aktuell wird hierfür meist vom Anwender ein Verhaltensmodell erstellt, welches die Applikation jedoch nur annähert und großen Aufwand sowie Unsicherheit über die Modellgüte mit sich bringt.

Der Modell-Umfang ist nicht pauschal festlegbar und abhängig von der jeweiligen Komponente.
Faustformel: Für jede Komponente mit einer eigenen BMK eine eigene FMU bereitstellen.

6.5 Individualität der FMU (spezifisches vs. abstraktes Modell)

Parameter erlauben es, FMUs für spezifische Anwendungsfälle anzupassen. Dadurch besteht die Möglichkeit, ganze Produktfamilien mittels eines Verhaltensmodells abzubilden. Als Beispiel ist eine Zylinder-FMU zu betrachten, welche über Parameter in der Länge variabel angepasst werden kann (vgl. Abbildung 20). Dieses Verhaltensmodell beschreibt folglich die Produktfamilie der Zylinder. Über diesen Weg muss der Komponenten-Hersteller weniger Verhaltensmodelle bereitstellen. Der Aufwand wird auf der Ersteller-Seite verringert. Allerdings steigt die Komplexität bei der Anbindung auf Anwender-Seite. Hier müssen viele Parameter gesetzt werden, was auch die Modellperformance teils negativ beeinflusst. Außerdem sind bei generalisierten FMUs viele Optionen, Einstellungen, Lastkurven, etc. zu hinterlegen, wodurch die Erstellung der Verhaltensmodelle unter Umständen

erschwert wird. Bei dieser Vorgehensweise ist auch eine klare Verbindung zu den entsprechenden realen Komponenten zu definieren. Für den Anwender muss ersichtlich sein, welche FMU mit welchem Parametersatz er für die entsprechende reale Komponente zu nutzen hat.

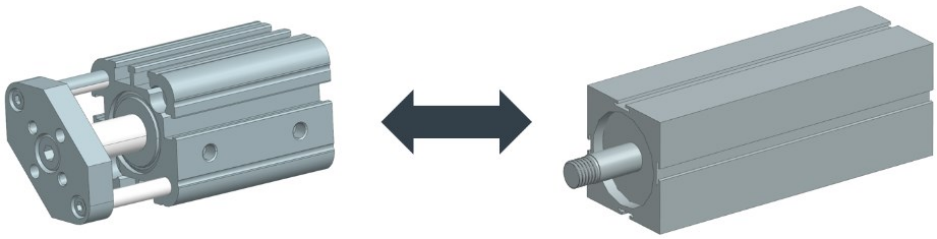

Abbildung 20: Vergleich zwischen einem artikelspezifischen Zylinder (links) und einem flexibel parametrierbaren Zylinder (rechts) analog zum CAD

Als Alternative kann beispielsweise für jede Komponente (Artikelnummer) ein eigenes Verhaltensmodell erstellt werden. Das Modell ist hierbei fest vorparametriert und nur für den spezifischen Anwendungsfall zu verwenden. Auf diese Weise werden potenzielle Fehler einer Falsch-Parametrierung vermieden. Auch die Zuordnung von Modellen zu den jeweiligen realen Komponenten auf Anwender-Seite wird vereinfacht. Neben dem artikelspezifischen CAD-Modell wird analog ein spezifisches Verhaltensmodell bereitgestellt. Der Aufwand auf Ersteller-Seite und in der Modellpflege (z.B. Versionierung, Optimierung, …) ist hier höher zu betrachten.
In beiden Fällen kann die Anbindung an das Simulationsmodell automatisiert erfolgen. So muss für einen bestimmten Artikel entweder auf eine spezifische FMU oder auf eine allgemeine FMU mitsamt einem zugehörigen Parametersatz referenziert werden. Die Parametrierung kann über einen Wrapper erfolgen, welcher unter Umständen auch für das Signal-Mapping verantwortlich ist. Generell ist diese Abgrenzung jedoch stark von der jeweiligen Komponente abhängig (z.B. induktiver Näherungs-Sensor vs. komplexen Antrieb) und kann nicht pauschal betrachtet werden.
Es besteht zudem die Möglichkeit, für jede real existierende Komponente (Seriennummer) eine bauteilspezifische FMU zu liefern. Der Verwaltungsaufwand ist hier jedoch enorm. Des Weiteren werden die FMU-Modelle meist vor der Bestellung/Lieferung der realen Komponente benötigt. Dieses Vorgehen hat sich somit als nicht praktikabel erwiesen.

Der Abstraktionsgrad ist nicht pauschal zu definieren und abhängig von der jeweiligen Komponente.
Faustformel: Solange die Parameter-Anzahl überschaubar bleibt, kann ein allgemeines Modell bereitgestellt werden.

6.6 Fehlertrigger im Anwendungsfall VIBN

Im Kontext einer VIBN ist es von großer Bedeutung, die SPS auf mögliche Fehlerfälle zu testen. Somit müssen in der FMU, neben dem idealen Verhalten, auch gewisse Fehl-Verhalten implementiert werden. Diese sind in der Realität häufig nicht oder nur mit großem Aufwand nachzustellen. Es ist zu unterscheiden zwischen Fehlern, welche das Verhalten der Komponente beeinflussen (sonstige Fehlerfälle), und Fehlermeldungen, welche über den Diagnose-Ausgang der Komponente (falls vorhanden) an die SPS weitergegeben werden (vgl. Abbildung 21). Falls die Automatisierungs-Komponente einen Diagnose-Ausgang für die SPS besitzt, ist dieser in jedem Fall in die FMU zu integrieren.

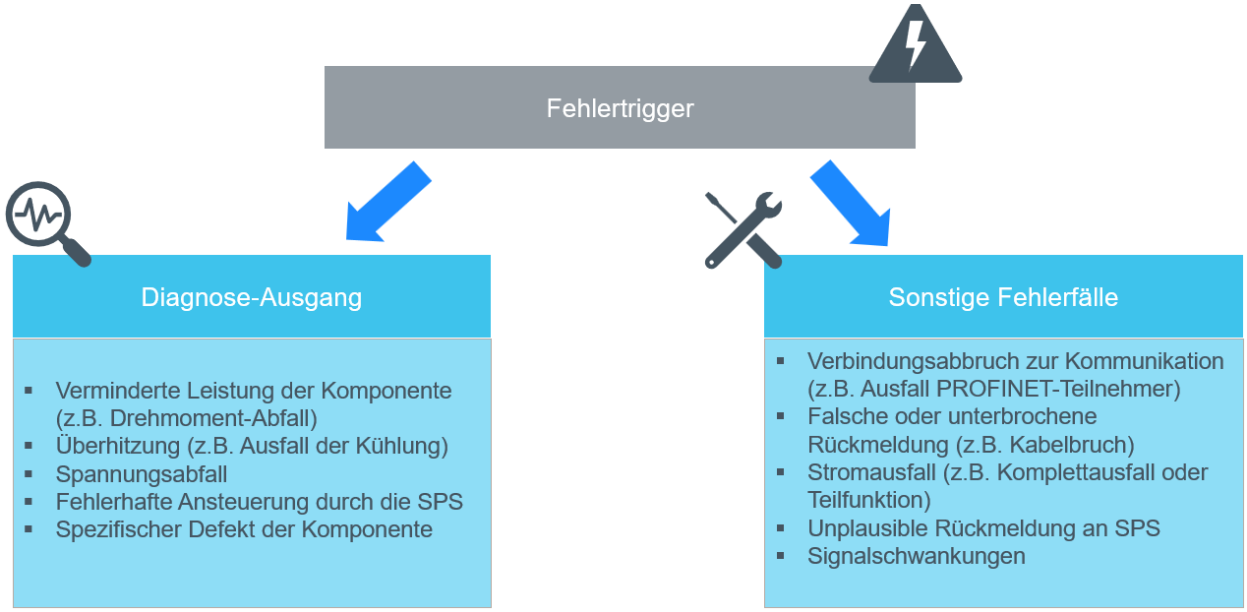

Abbildung 21: Unterteilung der Fehlertrigger in Meldungen durch den Diagnose-Ausgang und in sonstige Fehlerfälle mit dazugehörigen Beispielen

Als Beispiele für Fehlerfälle sind Kabelbrüche in der Safety (sonstige Fehlerfälle), die Überhitzung eines Motors (Diagnose-Ausgang) oder der Ausfall eines Kondensators (Diagnose-Ausgang) zu nennen. Ein weiterer Fehlerfall ist beispielsweise das Trennen der eingehenden Ist-Position von der Visualisierung (Encoder-Fehler). Dadurch erreicht der Antrieb nie die angestrebte Soll-Position und dreht immer weiter (sonstige Fehlerfälle). Zudem können beispielsweise Schleppfehler durch einen Fehlertrigger integriert werden.

Neben diesen Hardware-Fehlern existieren auch Software-Fehler. Verursacht werden diese Fehler z.B. durch unplausible Eingaben der Steuerung oder zeitliche Überwachungen (verspätete Rückmeldung). Beispiele sind Verständnis-Probleme beim der Programmumschaltung (z.B. Parameteränderung). In der FMU können beispielsweise Verarbeitungszeiten über einen Faktor oder Delay hochgestellt werden, sodass in der SPS ein

Überwachungsfehler ausgelöst wird. Für weitere komponentenspezifische Fehlerfälle sind Testtabellen aus der VIBN zu nutzen, um entsprechende Bedarfe abzuleiten.
Weitere Beispiele für Fehlertrigger sind der folgenden Auflistung zu entnehmen:

- Verbindungsabbruch der Kommunikation von/zur Steuerung
- falsche oder unterbrochene Sensor-Rückmeldungen
- verminderte Leistung der Komponente (z.B. geringes Drehmoment, geringe Kraft, geringer Volumenstrom, geringer Druck)
- Überhitzung bzw. Ausfall der Kühlung
- Spannungsabfall oder kompletter Strom-Ausfall
- fehlerhafte Safety-Signale (z.B. Signale fallen ab/passen nicht zusammen)
- fehlerhafte Bedienung der Anlage

Um derartige Fehlerfälle zu testen, bedarf es einer Integration im Verhaltensmodell. Die Ansteuerung gestaltet sich meist über so genannte „Fehler-Trigger“. Diese sind als Art „Pseudoeingänge“ zu verstehen, da sie in der realen Komponente nicht existieren. Aus diesem Grund ist es auch unerlässlich, die Fehlerfälle in die FMU-Dokumentation mit aufzunehmen.
Bei einer geringen Anzahl an Fehlerfällen sind die Trigger leicht über binäre Signale abzubilden. Damit bei einer Vielzahl an Fehlerfällen das Verhaltensmodell nicht unübersichtlich wird, kann auch ein Integer-Signal und ein Trigger-Bit verwendet werden, über welche der jeweilige Fehlerfall ausgelöst wird. Häufig genügt es, die Fehler auf oberster Ebene zu simulieren und nur die Rückmeldung an die SPS zu setzen. So können gewünschte Fehlerfälle vom Anwender definiert und direkt an die SPS weitergegeben werden. Die Fehlernummer ist über einen Input vom Anwender zu setzen.
Neben einzelnen Fehler können auch komplette Fehler-Szenarien und Testfälle abgebildet und beschrieben werden. Diese sind in der Dokumentation aufzuführen und evtl. über spezielle Trigger-Bits auszulösen. So können auch vollautomatisierte Tests durchgeführt werden. Es sind vor allem die Fehler zu überprüfen, welche in der Realität sehr selten auftreten. Im Hinblick auf automatisierte Fehler-Tests wird eine dazugehörige Quittierung nötig. Häufig ist es nur schwer möglich, nach einem Fehlerfall die Anlage wieder in Betrieb zu nehmen. In diesem Zusammenhang ist ein Quittierungsbit nötig, sodass der anliegende Fehler zurückgesetzt werden kann. Für den Fall, dass die reale Hardware über einen Reset/Quittierungs-Button verfügt, ist dieser auch im Verhaltensmodell zu implementieren.
Bei schwerwiegenden Fehlern muss unter Umständen sowohl die SPS als auch das Verhaltensmodell neu gestartet werden, um den Fehlerfall zu verlassen. In diesem Fall ist es möglich, über einen FMU-Reset im Co-

Simulator das Modellzurückzusetzen, ohne die Simulation neustarten zu müssen. Diese automatisierten Tests sind über eine externe Software (z.B. Expecto) zu verwalten.

Bei komplexen Komponenten existiert eine sehr große Anzahl an denkbaren Fehlerfällen. Falls nicht alle diese Zustände in der FMU vorgedacht und implementiert wurden, kann der Anwender diese teils über das VIBN-Tool realisieren. Hier werden häufig Mechanismen zur Verfügung gestellt, um manuelle oder gescriptete Fehler- und Störereignisse auszulösen. Auch eine Anbindung von Autotesttools wird über das VIBN-Tool ermöglicht.

Da die Kommunikation der Komponente am Bus aktuell vom VIBN-Tool emuliert wird, ist ein Ausfall des Bus-Teilnehmers ebenfalls über das VIBN-Tool zu simulieren. Ein Beispiel wäre der Ausfall eines PROFINET-Teilnehmers. In der FMU muss hierfür kein Trigger und Fehlerzustand vorgesehen werden. Im Zusammenhang mit dem Ausfall eines Bus-Teilnehmers steht auch die Anbindung an eine Stromversorgung. Damit auch ein Ausfall eines Netzwerks von Komponenten am gleichen Stromkreis simuliert werden kann, ist dem Verhaltensmodell über einen Eingang mitzuteilen, ob aktuell Strom am Gerät anliegt. Dies kann entweder über ein „Enable“-Bit geschehen oder über einen Real-Eingang, welcher die anliegende Spannung simuliert.

Neben Fehlertriggern für Komponenten sind diese auch für Prozesse zu hinterlegen. Da die Prozesse meist als idealisiert angenommen werden, bedarf es hinterlegter Fehlerszenarien. So müssen beispielsweise einzelne Sensoren ausgelöst werden, um in den gewollten Zustand zu gelangen. Des Weiteren muss die Möglichkeit geschaffen werden, „NiO“-Prozesse (z.B. beim Kleben oder Schweißen) anzutriggern. Häufig werden in der Prozesstechnik lange Zeiten in Anspruch genommen, bis die jeweiligen Zustände erreicht werden. Durch Fehlertrigger können lange „Wartezeiten“ in der VIBN gebrückt werden.

Fehlertrigger steigern den Mehrwert einer VIBN und ermöglichen Tests, die an der realen Anlage nur mit viel Aufwand nachzustellen sind. Fehlertrigger steigern die Qualität der FMU.

6.7 Steuerungsspezifische Anpassungen

Der Anschluss von komplexen Automatisierungskomponenten an die entsprechenden Steuerungen gestaltet sich häufig unterschiedlich. Neben den verschiedenen herstellerspezifischen Protokollen (z.B. PROFINET, EtherCAT, …) können die Komponenten auch über unterschiedliche Strukturen an die Steuerung angebunden werden. Hierfür sind an der FMU teils kleine Anpassungen nötigt. Beispiele hierfür sind der Byte-Flip oder die Aufteilung von Wörtern auf einzelne Bits und Bytes. Dies kann über geeignete Parameter direkt in der FMU erfolgen, sodass der Anbindungsaufwand für den Anwender minimal gehalten wird. Hierdurch erhöht sich jedoch die Anzahl an Schnittstellen-Signale und der Parametrierungs-Aufwand. Alternativ kann die Komponenten-FMU auch mit „Hilfs-FMUs“ verschalten werden, welche die Signale geeignet aufbereiten (siehe Abschnitt 10.1). Des Weiteren besteht die Möglichkeit, diese Anpassungen direkt im VIBN-Tool durchzuführen.

Generell sollte die FMU so allgemeingültig wie möglich gehalten werden. Spezifische Anpassungen für einzelne Anwender sollten vermieden und über einen alternativen Ansatz realisiert werden. Die Verhaltensmodelle bilden die realen Komponenten nach. Falls diese auf verschiedene Bus-Protokolle parametrierbar sind, ist diese Funktionalität (z.B. Byte-Flip) auch in der FMU über Parameter zu realisieren.
Folgende beispielhafte Empfehlungen können gegeben werden (vgl. *Tabelle 3*):

Anwendungsfall	Implementierung in FMU	Über Hilfs-FMU	Über Verhaltens-modellierer
Byte-Flip	x	(x)	(x)
Wort zu Byte	Wort in EAs	x	x
Wort zu Bit	Wort in EAs	x	x
Byte zu Wort	Wort in EAs	x	x
Bit zu Wort	Wort in EAs	x	x
Einheitenanpassung	x	(x)	(x)
Normierung	x	(x)	(x)

Tabelle 3: Übersicht von verschiedenen steuerungsspezifischen Anpassungen und deren Umsetzungsmöglichkeiten (x = Empfehlung, (x) = technisch realisierbar, aber wenn möglich, in FMU vorzusehen).

Die FMU ist möglichst neutral zu erstellen. Steuerungsspezifische Eigenheiten sind über Hilfs-FMUs oder durch das VIBN-Tool anzugleichen. Die Integration in die FMU ist nur sinnvoll, wenn die reale Komponente auch diesbezüglich parametrierbar ist.

6.8 Rolle der Visualisierung (CAD)

In den meisten Fällen werden im Kontext der VIBN die Verhaltensmodelle (FMUs) an eine Visualisierung (z.B. YAMS, SIMLINE, NX MCD) gekoppelt. Hier wird unter anderem der Materialfluss abgebildet oder eine Kollisionsprüfung durchgeführt. Die Visualisierung bietet die Möglichkeit, zusätzliche Aspekte im Rahmen einer VIBN abzusichern. Vor allem Prozesse und Materialfluss können mit reinen Verhaltensmodellen nur schwer abgebildet werden. Auch die Absicherung von Safety wird über die Visualisierung erleichtert. So können beispielsweise Lichtgitter oder einzelne Strahlen modelliert und an das jeweilige Verhaltensmodell weitergegeben werden. Zusätzlich bietet die Visualisierung dem Anwender ein visuelles Feedback. Der digitale Zwilling ist anschaulich und leichter zu interpretieren.

Nicht jede Komponente benötigt eine Schnittstelle zur Visualisierung. Hierunter fallen beispielsweise Kommunikationsmodule oder Temperatursensoren. Für viele Bauteile ist eine Anbindung an eine 3D-Simulation allerdings von Bedeutung. Zu nennen sind Sensoren, Antriebe, Zylinder oder Roboter.

Die Schnittstellen-Signale an die Visualisierung sind nicht standardisiert festgelegt. So unterscheiden sich beispielsweise die genutzten Einheiten (z.B. Millimeter und Meter). Zudem werden unter Umständen keine direkten Prozesswerte, sondern Prozentwerte an die Visualisierung übergeben. Als Beispiel ist ein Pneumatik-Zylinder zu nennen, welcher an die Visualisierung einen Wert zwischen 0 und 100 % übergibt. Aus diesem Prozentwert wird in der Visualisierung der resultierende Verfahrweg abgeleitet.

Im Falle eines Zylinders können die Sensoren für die Endanschläge („aus-“/„eingefahren“) sowohl direkt im Verhaltensmodell als auch in der Visualisierung abgebildet werden (vgl. Abbildung 22). Vor allem bei Roboter-Kopplungen oder mechanischen Zusammenhängen ist es sinnvoll, die Sensorrückmeldung über die Visualisierung zu modellieren. Bei Spannzylindern oder nicht konstruktiv abgebildeten Sensoren reicht häufig eine Anbindung über das Verhaltensmodell aus. Im Fall „1“ (Abbildung 22) bekommt die Steuerung eine direkte Rückmeldung vom Verhaltensmodell. Aus diesem Grund kann der Zylinder auch ohne eine Visualisierung betrieben werden. In der Version „2“ gibt das Verhaltensmodell eine Position an die Visualisierung, welche über Sensoren die Lage des Zylinders erkennt und eine Rückmeldung an die SPS gibt. In Möglichkeit „3“ wird der Zustand aus der Visualisierung an das Verhaltensmodell rückgeführt. Dieses wertet das Ergebnis aus der Visualisierung aus und meldet es an die SPS weiter.

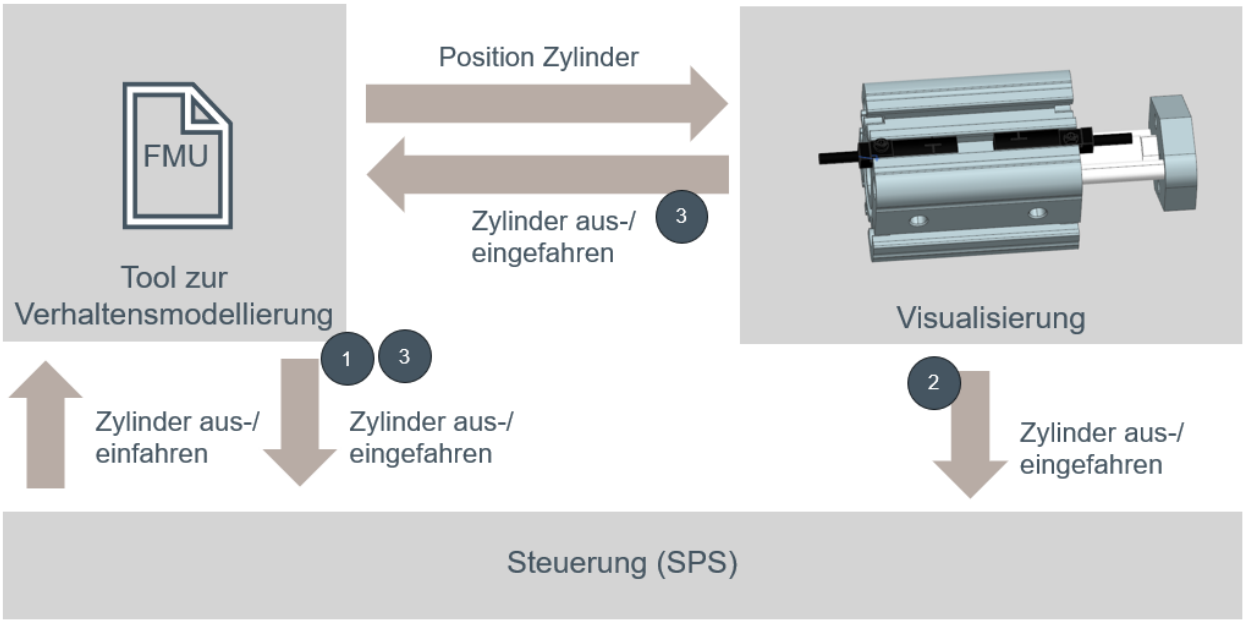

Abbildung 22: Verschiedene Möglichkeiten der Rückmeldung an die Steuerung am Beispiel eines Pneumatik-Zylinders mit Endanschlägen (1=ohne Visualisierung; 2=direkt aus Visualisierung; 3=über Visualisierung und Verhaltensmodell)

Generell ist es (abhängig vom Einsatzgebiet) zu empfehlen, dass die FMU möglichst eigenständig betrieben werden kann. Wesentliche Funktionen der einzelnen Komponenten sind direkt über die FMU zu simulieren. Dadurch steigen die Stabilität und Flexibilität des digitalen Zwillings. Die Anzahl an Schnittstellen-Signale sinkt. Zudem kann die VIBN (z.B. EA-Test, einfache Logiken) in einem frühen Stadium ohne Visualisierung durchgeführt werden. Später dient die Visualisierung als Unterstützung. Bei mechanischen Komponenten kann im Idealfall eine Anbindung an die Visualisierung (z.B. Antriebe mit Rückmeldung aus dem CAD) über einen Parameter an- und abgewählt werden. So besteht die Möglichkeit, je nach Simulationskette und Testfall, die entsprechenden Funktionen zu aktivieren bzw. zu deaktivieren.

Gerade in Verbindung mit hochkomplexen physikalischen Verhaltensmodellen (vgl. Abschnitt 6.1.3) spielt die Visualisierung eine große Rolle. Werden beispielsweise Kräfte und Momente an die FMU rückgeführt, bedarf es einer entsprechenden Physik-Simulation in der Visualisierung. Hier gilt es entsprechend Gewichte, Reibwerte und Trägheit zu hinterlegen. Voraussetzung hierfür ist, dass das Visualisierungstool einen physikbasierten Solver bereitstellt.

Viele Verhaltensmodelle interagieren mit einem CAD-Modell (Visualisierung). Hierfür sind entsprechende Schnittstellen in der FMU bereitzustellen.

6.9 Ausblick: Modellierung von Prozessen, Gebäuden und Produkten

Im digitalen Zwilling sind neben Automatisierungskomponenten häufig auch Prozesse abzubilden. Die Erstellung von Prozess-FMUs ist dabei nicht an den Komponenten-Hersteller adressiert. In diesem Kontext liegt das Prozess-Wissen beim Anlagenbauer bzw. beim OEM. Häufig werden auch Module mit Teilprozessen bereitgestellt (z.B. Klebeapplikationen, Schweißzangen, etc.). In diesem Fall wird eine FMU-Erstellung durch den Modul-Hersteller empfohlen. In vielen Fällen reichen einfache Logikmodelle aus. Im Idealfall sind diese möglichst automatisiert zu generieren, sodass der Erstellungsaufwand minimiert wird. Im Falle von Prozess-FMUs dienen Inputs und Parameter dazu, eine Bedienerschulung zu ermöglichen. Durch Anpassungen während der Simulation können Parameter eingestellt und optimiert werden. Fehlerfälle können leicht simuliert werden (vgl. Abschnitt 6.6).

Neben Prozessen können auch Modelle von Gebäuden und der Infrastruktur über FMUs abgebildet werden. Darunter fallen beispielsweise die Druckluft- oder elektrische Versorgung in einer Halle sowie externe Einflüsse wie Temperatur oder Feuchtigkeit.

Vor allem bei komplexen Produkten müssen diese für eine aussagekräftige VIBN zunehmend modelliert werden. In der Automobilindustrie kommunizieren beispielsweise bei End-of-Line-Prüfständen die Fahrzeuge mit ihren Prüfständen. Häufig müssen auch Seriennummern ausgelesen oder sonstige Parameter im Produkt gespeichert werden. Diese Anwendungsfälle können ebenfalls toolunabhängig über FMUs beschrieben werden.

Analog zu den Automatisierungskomponenten können auch Prozesse, Gebäude, Produkte oder Teilsysteme (Module) über den FMI-Standard modelliert und angebunden werden.

7. FMU-Erstellung und Übergabe an den Anwender

Aus Sicht des Anlagenbauers und -betreibers (OEM) werden im Zielbild vom Komponenten-Hersteller für alle/neue Komponenten FMUs über eine Plattform bereitgestellt. Dies hat den Vorteil, dass der Anlagenbauer die Komponentenmodelle bei Bedarf herunterladen und in das Simulationsmodell integrieren kann. Eine FMU-Erstellung auf Anfrage, wie es aktuell häufig gehandhabt wird, hat den Nachteil, dass die Bereitstellung meist lange dauert. Das Modell kann im entsprechenden VIBN-Projekt nicht mehr genutzt werden. Dadurch muss der Anlagenbauer selbst Verhaltens-Modelle (FMUs) anfertigen. Im Zielbild stellt bei neuen Komponenten der Komponenten-Hersteller das entsprechende Verhaltensmodell als FMU während der Freigabe dem Anlagenbauer/OEM zur Verfügung.

7.1 Datendurchgängigkeit (Nutzung von Engineering-Daten)

Im Idealfall werden Verhaltensmodelle mittel- und langfristig direkt aus der Komponenten-Firmware exportiert. So können auch Anpassungen in der realen Komponente direkt auf die Modelle übertragen werden. Zusätzlich werden die Kosten für die Modell-Erstellung niedrig gehalten. Eine direkte Übernahme und Integration der Original-Firmware bringt hohe Komplexität in das Modell und hat gegebenenfalls auch Auswirkungen auf die Performance im Co-Simulator. Häufig ist eine direkte Übernahme der Software nicht oder nur schwer möglich, da verschiedene Abhängigkeiten und Einschränkungen aus der realen Firmware-Entwicklung existieren. Das mechanische Verhalten (z.B. bei Zylindern) ist nicht Teil der Firmware und lässt sich somit auch nicht ausleiten. Gleichermaßen sind in den FMUs Fehlertrigger zu implementieren (vgl. Abschnitt 6.6), welche ebenfalls nicht Teil der Firmware sind.

Da die Verhaltensmodelle derzeit meist ohne das Wissen der Komponenten-Hersteller generiert werden, beschränkt sich der Umfang meist auf das Handbuch-Wissen. In der VIBN werden aktuell vereinfachte Modelle genutzt, welche nur Teilbereiche der Firmware-Logik beschreiben. Bei vielen Komponenten-Hersteller wird derzeit die Logik der FMU-Modelle parallel zur Firmware entwickelt. Damit der Aufwand zur Implementierung dennoch geringgehalten wird, legen sich die Komponenten-Hersteller in ihrem FMU-Export-Tool Bibliotheken an. Bei der FMU-Erstellung werden schließlich einzelne Bibliotheks-Elemente miteinander verschalten.

Eine vollumfängliche Ausleitung von FMUs aus der Firmware gestaltet sich schwierig. Fehlertrigger und mechanische Zusammenhänge müssen zusätzlich zur Logik erweitert werden. Eine automatisierte (Teil-) Generierung von FMUs ist mittelfristig anzustreben.

7.2 Tools zur Erstellung von FMUs (FMU-Export)

Der FMI-Standard erfährt immer mehr Aufmerksamkeit. Aus diesem Grund unterstützten inzwischen fast 250 Tools den Umgang mit FMUs. Eine Auflistung ist unter folgendem Link zu finden: https://fmi-standard.org/tools/ [35].

Prinzipiell gibt es im Kontext der FMU-Entwicklung für die VIBN keine speziellen Tools, über welche die Verhaltensmodelle zu erstellen sind (vgl. Abbildung 23). Ein geeignetes Tool muss immer unter Berücksichtigung des Unternehmens-Kontextes gefunden werden. Eine pauschale Aussage zu Eignung spezieller Erstellungs-Tools in Form eins Rankings ist nicht sinnvoll. Wichtig ist allerdings, dass die exportierten FMUs den in diesem Leitfaden definierten Anforderungen genügen. Dies gilt es im Einzelfall zu überprüfen.

Im Allgemeinen lassen sich die Tools und Umgebungen zur Erstellung von FMUs in zwei Kategorien aufteilen. Zum einen bieten verschiedene Simulations- und Modellierungs-Tools einen FMU-Export an (z.B. AMESim, MATLAB/Simulink, OpenModelica). Die zweite Möglichkeit stellt eine FMU-Erstellung direkt auf Basis des FMI-Standards dar. Hier sind verschiedenste FMU-Vorlagen, so genannte Templates, zu nennen. Die FMUs werden über Programmiersprachen wie C oder C++ entwickelt. Generell ist es die Aufgabe des Komponenten-Herstellers, das für Ihn passende Tool zu identifizieren. Da die Lebenszeit der FMU an die reale Komponente gekoppelt ist, gilt es zu beachten, dass das gewählte Tool auch für den mittel- oder langfristigen Betrieb geeignet ist. Andernfalls sind keine Anpassungen an den FMU-Modellen möglich. Die aus dem VIBN-Kontext gestellten Anforderungen an die FMUs sind bei der Tool-Auswahl zu berücksichtigen. Andernfalls kommt es zu Problemen bei der Integration in die VIBN-Toollandschaft. Ein Beispiel hierfür wäre die Mehrfachinstanziierung von FMU-Modellen über die Flag „canBeInstantiatedOnlyOncePerProcess“ (vgl. Abschnitt 5.4.2).

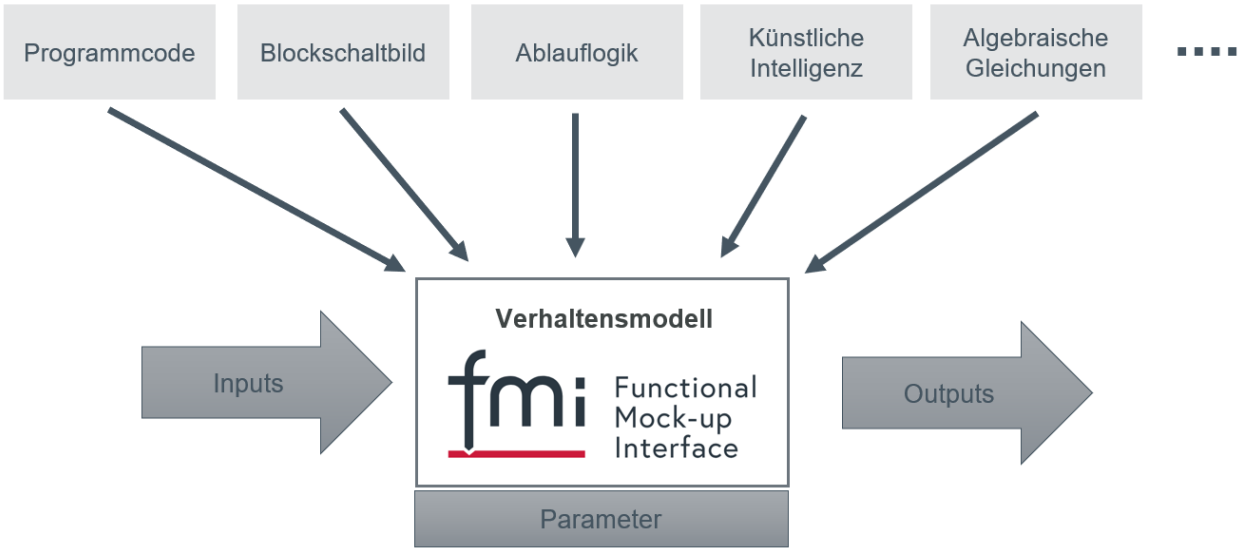

Abbildung 23: Überblick der verschiedene Modellierungsmöglichkeiten von FMUs

Simulations- und Modellierungs-Tools (1):

- Werden aktuell häufig verwendet, um FMUs für die VIBN in einer weniger komplexen Detaillierungsstufe bereitzustellen (parallel zur Firmware entwickelt).
- Ermöglichen eine schnelle FMU-Erstellung, da der Exporter bereits implementiert ist und die Modellierung häufig recht intuitiv ist. Aus diesem Grund werden die Tools häufig als kurz- bis mittelfristige Lösung genutzt. Eine tiefe Einarbeitung in den Standard und die Validierung der exportieren FMUs ist nicht nötig. Das Simulations- und Modellierungs-Tool verantwortet eine konforme und stabile Ausleitung von FMUs. Anpassungen durch den Anwender sind nicht oder nur schwer durchführbar.
- Gerade im Kontext von Physik-Simulationen sind hier häufig umfangreiche Bibliotheken hinterlegt. Über die Einstellungen können verschiedene Solver-Verfahren gewählt und die Berechnung der Differentialgleichungen in der FMU beeinflusst werden.
- Unterstützen teils den Import sowie Export in verschiedenen Betriebssystemen (Windows und Linux).
- Bieten meist auch einen Co-Simulator an, mit welchem die FMUs validiert werden können.
- Setzen bestimmte Flags (vgl. Abschnitt 5.4.2) automatisch.
- Der Anwender ist auf den Umfang der jeweiligen nativen Bibliotheken begrenzt.

FMU-Templates (2):

- Basieren meist auf einer Hochsprache (C, C++) und ähneln damit der Firmware-Entwicklung. Eine Integration von Firmware in das Verhaltensmodell ist dadurch denkbar.
- Können in eine Entwicklungsumgebung (z.B. Visual Studio) eingebunden werden, welche der Komponenten-Hersteller unter Umständen im Haus hat. (keine Neuanschaffung)
- Können auf den jeweiligen Einsatzzweck erweitert werden. Die Integration von zusätzlichen FMI-Funktionalitäten ist bei Bedarf möglich.
- Durch die Nutzung von C/C++ stehen umfangreiche Bibliotheken/Methoden zur Verfügung. So können beispielsweise komplexe Funktionen, wie eine Datenspeicherung, ein API-Aufruf oder eine TCP/IP-Kommunikation, abgebildet werden.
- Versionierungs- und Verwaltungs-Tools aus der Software-Entwicklung (wie z.B. Git, SVN) können verwendet werden. So entstehen Analogien und Synergieeffekte aus der Firmware-Entwicklung. Beispiele bilden Konventionen in der Variablenbezeichnung, die Code-Formatierung, die Nutzung von bekannten Bibliotheken oder das Vorgehen beim Kommentieren des Codes.
- Ein gezieltes Logging ist möglich. Das Debugging der FMU wird dadurch erleichtert.
- Die Stabilität der FMU ist in den Händen des Entwicklers. Abstürze und Falscheingaben können gezielt abgefangen werden.
- Da die modelDescription.xml häufig händisch oder über Hilfstools generiert wird, ist eine Gegenprüfung über den FMU-Checker (vgl. Abschnitt 7.4) von großer Bedeutung.
- Auf eine qualitative Software-Entwicklung und Implementierung ist zu achten, da ein unsauberer Umgang mit Ressourcen und Methoden zum Absturz der FMUs führen kann. Die FMUs sind sorgfältig zu validieren.

VIBN-Tools (3):

- Aktuell wird von keinem VIBN-Tool heraus der Export von FMUs angeboten. Nur der FMU-Import und die Simulation werden unterstützt. Es ist aber generell nicht ausgeschlossen, dass VIBN-Tools einen Export von nativ erstellten Verhaltensmodellen oder auch Bibliotheks-Elementen ermöglichen.
- Nicht jedes Tool ist für den FMU-Export geeignet. Es bestehen Abhängigkeiten zur Struktur des Tools, zur Lizenzierung, etc. Der Aufwand bei der Implementierung und Pflege eines FMU-Exportes ist hoch.

Insgesamt können Tools aus verschiedensten Bereichen und Branchen für die FMU-Erstellung genutzt werden. Aufgrund des FMI-Standards ist keine VIBN-spezifische Software nötig, um Verhaltensmodelle für die Anwendung im digitalen Zwilling zu erstellen. Generell ist darauf zu achten, dass einige Tools den Export von FMUs zwar

erlauben, jedoch keine reine C-Code-Binärdatei ausleiten. Hierbei wird ein Wrapper bzw. eine Hülle um das eigentliche Modell gelegt (vgl. Python-FMU). Diese FMUs benötigen häufig eine installierte Runtime im Hintergrund, da sie nicht auf einer kompilierten C-Datei basieren.

Die Herausforderung für den Tool-Hersteller besteht darin, ein geeignetes Tool zu finden. Zum einen müssen die Anforderungen für FMUs im Kontext der VIBN erfüllt werden. Anderseits muss das Tool auch in den Unternehmenskontext passen und möglichst viele Synergien aufweisen. Im besten Fall können bestehende Modelle oder die Firmware der realen Komponente in den Erstellungsprozess eingebunden werden. Die Tool-Auswahl ist ein komplexer Prozess, welcher Vorab-Analysen und Tests voraussetzt. Die Auswahl ist unternehmensspezifisch zu treffen. Unter Umständen kann auch eine Kombination von mehreren Erstellungs-Tools für verschiedene Komponenten (Produktfamilien) in Frage kommen.

Dieser Leitfaden unterscheidet bei den FMU-Erstellungs-Tools zwischen: Simulations- und Modellierungs-Tools (1), FMU-Templates (2) und VIBN-Tools (3). Abhängig von unternehmens-spezifischen Gegebenheiten und dem vorhandenen Know-how ist ein passendes Tool zu finden.

7.3 Dokumentation einer FMU

Ein wichtiger Bestandteil eines Verhaltensmodelles ist die Dokumentation (vgl. Abschnitt 5.6). Hier sind allgemeine Informationen, wie der verwendete Standard (z.B. FMI 2.0/3.0), der Autor, das Erstellungs-Datum, der Status, Versionierungen oder das genutzte Generierungs-Tool, aufzuführen. Da es sich bei einer FMU um eine Blackbox handelt, bietet dieses Dokument (ausgenommen der modelDescription.xml) die einzige Möglichkeit, dem Anwender Informationen über das Modell bereitzustellen. So sind unter anderem die Inputs, Outputs und Parameter als Schnittstellen der FMU mitsamt deren Signalbenennungen, Datentypen, Default-Werten, Maximal-/Minimal-Werten, Einheiten und Kommentaren zu nennen. In der Signal-Beschreibung ist eindeutig zu hinterlegen, welche Funktion dem jeweiligen Signal zugeordnet und wie es zu verwenden ist. Die Ein- und Ausgänge können in Form eines Blockschaltbildes (vgl. Abbildung 24) dargestellt werden.

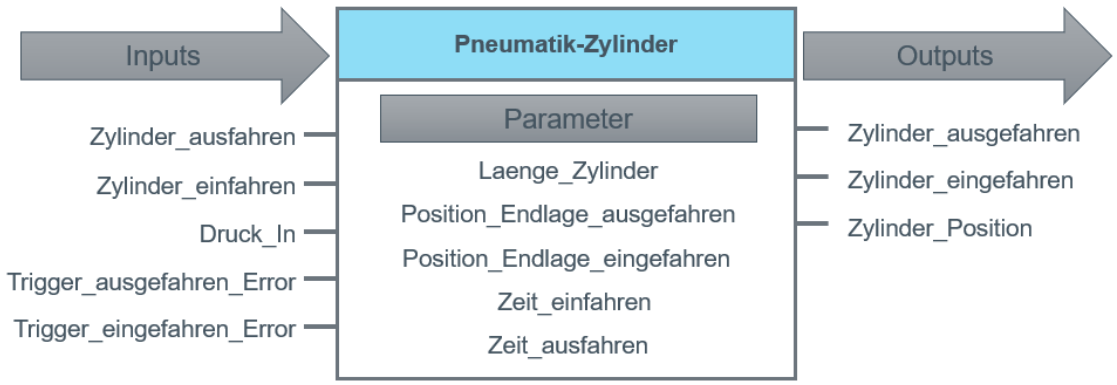

Abbildung 24: Prinzipieller Aufbau eines Blockschaltbildes am Beispiel eines Pneumatik-Zylinders

Die Dokumentation muss Informationen darüber enthalten, welche Inputs, Outputs und Parameter zwingend angesprochen werden müssen (Grundfunktionieren) und welche als optional gelten (Zusatzfunktionen). Des Weiteren ist zu beschreiben, wie die Daten in der Blackbox verarbeitet werden. Hierunter fallen beispielsweise die Auswirkungen von gesetzten Parametern auf das Verhalten. Falls Fehler-Trigger und Fehlerfälle in der FMU abgebildet werden, sind diese dringend in der Dokumentation zu beschreiben. Der Anwender muss hier detaillierte Informationen erhalten, was im Hintergrund bei der Berechnung abläuft. Fehlercodes und entsprechende Log-Meldungen sind zu ebenfalls zu beschreiben. Auf diese Weise wird ein grobes Verständnis für das Verhalten geschaffen, welches bei Fehlverhalten und Problemen Debugg-Versuche auf Anwenderseite ermöglicht.

Die Default-Werte für Parameter und Inputs sind so zu wählen, dass der Anwender das Modell schnell testen kann. Grundfunktionen sollen ohne große Parameter-Einstellungen beim Simulationsstart ermöglicht werden. Ebenso ist in der Dokumentation ein kleines Tutorial aufzuführen, wie die FMU in Betrieb genommen werden kann (z.B. Startreihenfolge bei Zustandsautomaten, Einstellung von Parametern, Inputs wie „Enable“) und welche Ergebnisse zu erwarten sind (Testfunktionen). So kann das Verhaltensmodell im jeweiligen Co-Simulator vor der VIBN getestet werden.

Ein weiterer wichtiger Bestandteil einer Dokumentation sind Traces und Diagramme (vgl. Abbildung 25 und

Zeit [s]	**Ziel-Position [mm]**	**Enable**	**Ziel-Geschwindigkeit [mm/s]**	**Ziel-Beschleunigung [mm/s2]**
10	100	True	10	10
20	120	True	10	10
30	0	True	10	10
40	-100	True	10	10
50	0	True	10	10

Tabelle 4). Gerade wenn ein zeitliches Verhalten (Rampen, Signalfolgen, etc.) abgebildet wird, ist es wichtig, den modellierten Verlauf zu dokumentieren. Auf Grundlage dieser Referenz kann der Anwender Probleme im Co-Simulator (evtl. Performance, Auslastung, …) erkennen. In der VIBN ist dies von Relevanz, da häufig mit Überwachungszeiten gearbeitet wird, sodass Fehlermeldungen und Störungen im SPS-Programm die VIBN beeinträchtigen können. In diesem Zusammenhang sollte in der Dokumentation auch das Vorgehen bei der Validierung beschrieben werden. (vgl. Abschnitt 7.4) Aussagen zur Modell-Performance (vgl. Abschnitt 9.11) und zu Konformität (nach FMI 2.0/3.0) sind zu treffen.

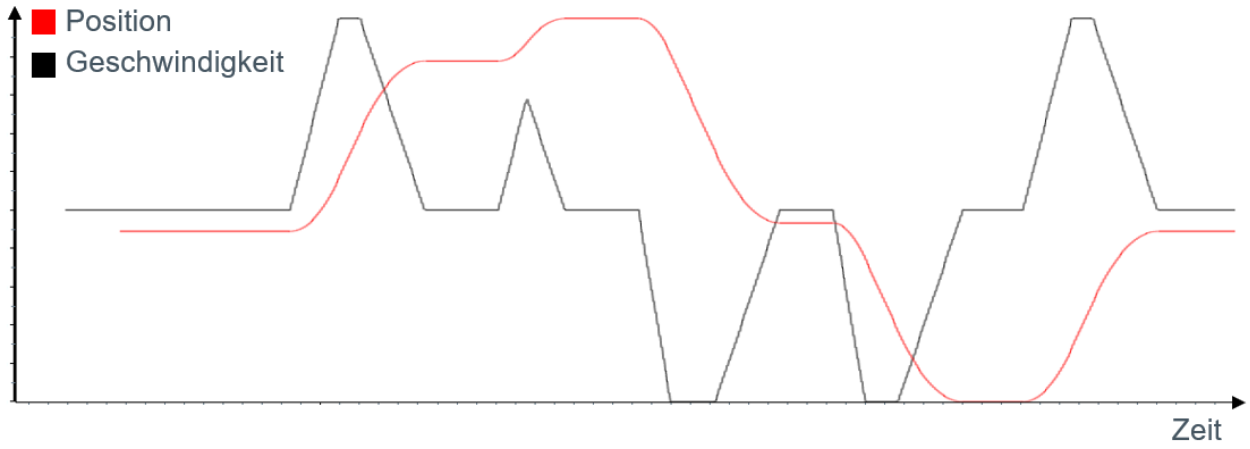

Abbildung 25: Beispielhafte Scope-Aufzeichnung eines Positionier-Antriebes

Zeit [s]	Ziel-Position [mm]	Enable	Ziel-Geschwindigkeit [mm/s]	Ziel-Beschleunigung [mm/s²]
10	100	True	10	10
20	120	True	10	10
30	0	True	10	10
40	-100	True	10	10
50	0	True	10	10

Tabelle 4: Zur Scope-Aufzeichnung zugehörige, beispielhafte Auflistung der angelegten Inputs und Parameter eines Positionier-Antriebes

Zudem ist in der Dokumentation zu erwähnen, welche Funktionen, Parameter sowie sonstige Umfänge der realen Komponente im Modell implementiert wurden und welche nicht. So kennt der Anwender die Grenzen des Simulationsmodelles und kann den Testumfang der VIBN anpassen. Der Detaillierungsgrad wird dadurch ersichtlich. Dies ermöglicht eine klare Abgrenzung zur realen Komponente. Der Anwender muss wissen, was er vom Modell erwarten kann und was nicht. Beispielsweise wird eine Schraube vom Baumarkt auch nicht in einer

Rakete verbaut, die bis zum Mars fliegt. Bei Bedarf sind einzelne Tests auf die reale Inbetriebnahme (IBN) auszulagern.

Da der FMI-Standard keine ausreichende Möglichkeit bietet, das Bedienfeld (die GUI zur Bedienung der FMU) standardisiert zu beschreiben und in die Simulationsumgebung zu integrieren, bedarf es im Idealfall eines Beispiels in der Dokumentation. Hier ist es dem Komponenten-Hersteller überlassen, diese abstrahiert zu skizzieren (vgl. Abbildung 26) oder einen Screenshot aus einem beliebigen VIBN-Tool anzuhängen. Über die GUI können beispielsweise Fehlertrigger (z.B. Kurzschlüsse, Kabelbrüche) über Taster ausgelöst oder verschiedene Parameter über eine textuelle Eingabe gesetzt werden. Das in der Dokumentation aufgeführte Konzept kann vom Anwender im jeweiligen Ziel-Tool nachgestellt werden und dient als Beispielvisualisierung. Neben dieser Beschreibung in der Dokumentation können auch Bilder (Vektor-Grafiken) in der FMU mitgeliefert und für die GUI im Ziel-Tool genutzt werden. So sind beispielsweise Abbildungen von Signallampen, Türschalter, etc. im Ressourcenordner zu hinterlegen. Dadurch entfällt die Suche von geeigneten CAD-Daten oder Abbildungen der realen Komponente auf Anwender-Seite. Alternativ zum Ressourcen-Ordner kann auch ein weiterführender Standard wie die Verwaltungsschale (vgl. Kapitel 10) hierfür genutzt werden. Des Weiteren besteht die Möglichkeit, über Links in der Dokumentation auf entsprechende Abbildungen zu verweisen.

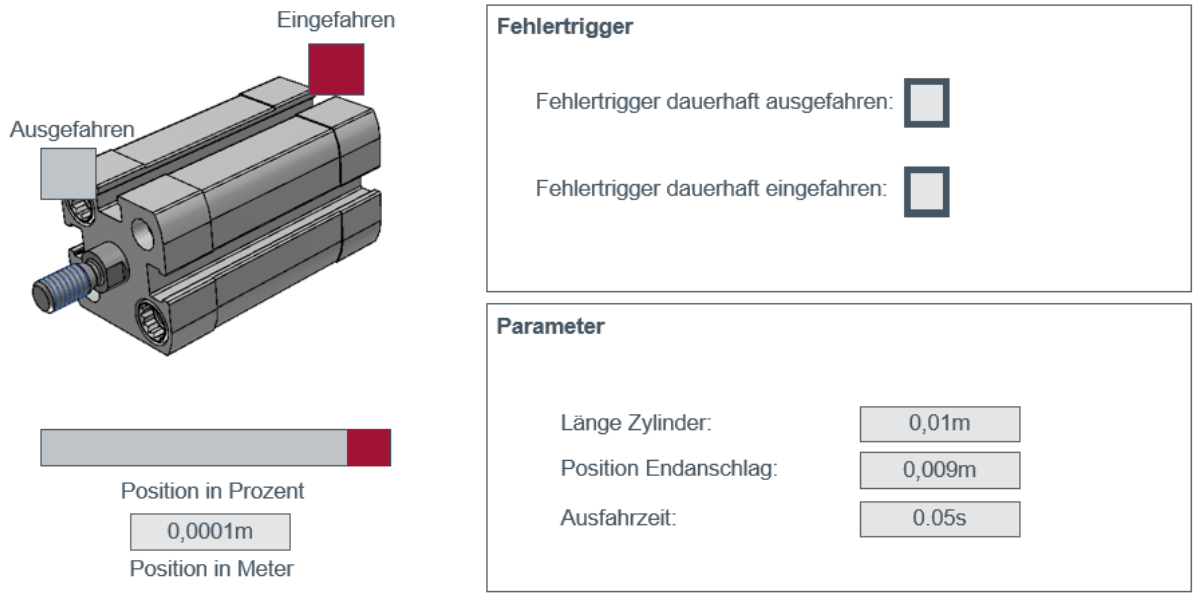

Abbildung 26: Beispielhafte Darstellung eines Bedienfeldes von einem Pneumatik-Zylinder mitsamt Fehlertriggern, Endlagensensoren und Parametern

Performance-Tests sind dringend durchzuführen und in der Dokumentation zu protokollieren. Wichtig ist, die verwendetet Hardware (Rechner), Software (Betriebssystem) und die Anzahl an integrierten FMU-Instanzen sowie die erreichte Zykluszeit zu nennen. In diesem Zusammenhang sind auch Mindestvoraussetzungen bzw. Zielanforderungen an die Hardware und Software zu nennen. Falls zusätzliche Hardware (z.B. Netzwerkkarten für TCP/IP-Kommunikation) benötigt wird, ist dies ebenfalls in der Dokumentation hervorzuheben. Auf diese Weise werden Angaben zum benötigten Rechenbedarf gemacht.

Sofern die FMU lizenziert ist (vgl. Abschnitt 7.6), gilt es dies in jedem Fall in der Dokumentation zu erwähnen. Details in Bezug zur Anwendung und Restriktionen sind hier aufzuführen.

Falls in der FMU Bilder oder sonstige Dokumente hinterlegt werden, sind diese im Rahmen der Dokumentation zu nennen und zu beschreiben.

Um zu garantieren, dass die FMU von einer vertrauenswürdigen Quelle stammt, wird unter Umständen eine Signatur angehängt (vgl. Abschnitt 9.9). Diese ist im Rahmen der Dokumentation zu nennen und zu beschreiben.

Externe Abhängigkeiten (vgl. Abschnitt 5.7) und Systemanforderungen sind ebenfalls in der Dokumentation aufzuführen. Hierunter fallen relevante Einstellungen im Co-Simulator (z.B. vorgegebene maximale oder minimale Stepsize). Auch Anpassungen oder Einstellungen für die Anbindung an die Visualisierung sind in der Dokumentation zu erwähnen.

Generell ist es von enormer Bedeutung, einen Ansprechpartner oder eine Support-Adresse zu hinterlegen (vgl. Abschnitt 7.9). Da es sich bei der FMU um eine Blackbox handelt, muss der Anwender wissen, an wen er sich bei Problemen wenden kann. Nur so kann eine Brücke zwischen dem Anwender und dem Entwickler geschaffen werden. In der VIBN sind Probleme teils zeitkritisch. Zudem kann vom Anwender Feedback gegeben und die FMU gezielt weiterentwickelt werden.

Ein weiterer Bestandteil des FMU-Handbuches ist die Dokumentation von Änderungen in bisherigen FMU-Versionen. So wird deutlich, welche Anpassungen gegenüber Vorgängerversionen stattgefunden haben und ob ein Update des Modelles für den Anwender relevant ist. Zum Beispiel sind Änderungen von Signalnamen hier aufzuführen, damit der Update-Prozess koordiniert werden kann. In der Dokumentation kann dies über eine Änderungshistorie beschrieben werden. Es ist auch wichtig, in der Versionierung Bezug zur realen Komponente zu nehmen. Es gilt zu beschreiben, welche Firmware-Versionen und Artikelnummern durch die jeweilige FMU abgebildet werden.

Es besteht die Möglichkeit, auf die Dokumentation der realen Komponente zu verweisen. Einzelne Abschnitte (z.B. Parameter, Funktionen, die Bedeutung von einzelnen Bit-Mustern) können evtl. direkt übernommen werden. Falls Abkürzungen verwendet werden, sind diese auch in einem Abkürzungsverzeichnis aufzuführen und zu erläutern. Die Dokumentation muss zwingend zur bereitgestellten FMU passen.

Die Dokumentation ist von enormer Bedeutung, da die FMU selbst eine gekapselte Blackbox darstellt. Eine qualitative FMU enthält Informationen über Inputs/Outputs/Parameter, Metadaten, Trace-Aufzeichnungen, beispielhafte Bedienfelder und den abgebildeten Detaillierungsgrad sowie Auskunft über logische Verhaltensmuster.

7.4 Validierung des Modelles

Nachdem die FMU erstellt wurde, folgen intensive Tests und eine gezielte Validierung. Aufgrund der Kapselung in einer Blackbox wird das Debugging bei Problemen erschwert. Umso wichtiger ist es, dass der Reifegrad bei der Auslieferung entsprechend hoch ist. Generell gilt, je automatisierter die Komponente erstellt wurde, desto geringer fällt der Aufwand für die Validierung aus (*ausgeschlossen einer Generierung durch KI*). Wie überall, führen manuelle Arbeiten zu Fehlern. Werden Simulations- und Modellierungstools zur Erstellung der FMU genutzt (vgl. Abschnitt 7.2), sind erste Tests meist bereits in dieser Entwicklungsumgebung möglich. Im Falle von z.B. MATLAB/Simulink oder OpenModelica können FMUs simuliert und über eine „Scope“-Aufzeichnung gegengetestet werden. Ein finaler Test ist im direkten VIBN-Umfeld durchzuführen. Dadurch wird auch ein korrektes Verhalten in einer „echtzeitfähigen“ Umgebung sichergestellt. Am Ende der Validierung muss sich die FMU, vor allem die Signale von und zur SPS, analog zur realen Komponente verhalten.

Zur Validierung des logischen Verhaltens kann auch durch den OEM ein Standardmodul (SPS-Baustein) zur Verfügung gestellt werden, anhand dem die FMU gegengeprüft wird (vgl. Abbildung 27). Teils existiert beim Komponenten-Hersteller für komplexe Bauteile bereits ein eigener SPS-Baustein. Ein einfaches Test-Projekt kann genutzt werden, um das Verhaltensmodell in einem „Unit-Test“ (zumindest für einen spezifischen Anwendungsfall bei einem Anwender) zu validieren. Verhält sich die FMU in diesem Test-Projekt korrekt und ohne Fehlermeldungen, kann davon ausgegangen werden, dass in der VIBN (für diesen Anwendungsfall) auch keine Probleme auftreten. Ebenso können auch anhand der Dokumentation der realen Komponente Testfälle abgeleitet und einzelne Aspekte validiert werden. Die Validierung ist an den implementierten Umfang und auf den speziellen Use-Case der FMU anzupassen.

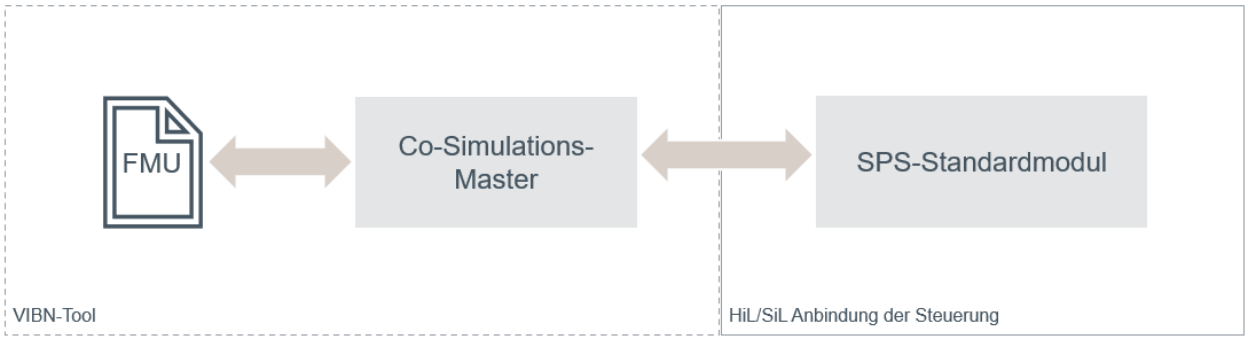

Abbildung 27: Prinzipieller Aufbau des Setups zur Validierung einer FMU anhand eines SPS-Standardmodules

Ein detailliertes physikalisches Verhalten wird derzeit im Kontext der VIBN in der Automobilindustrie selten implementiert. Zeitliche Abfolgen, wie z.B. Rampen, können anhand des idealen Verhaltens (aus dem Handbuch) validiert werden. Aufgrund einer unvollständigen Abbildung der Physik (z.B. Reibung, Temperatur, Alterung, …) in der Visualisierung, ist eine Validierung gegenüber der realen Komponente hier nur bedingt zielführend. In diesem Zusammenhang wird es aktuell immer Abweichungen geben. Bei komplexen physikalischen Modellen ist die Funktionalität der FMU auf die in den Abschnitten 5.4.2 und 6.4 erwähnten Sprünge in den Eingängen zu überprüfen. Auch bei großen Unterschieden in den Eingängen zwischen zwei Simulations-Zyklen muss die FMU stabil laufen und darf nicht abstürzen.

Generell ist es essenziell, den Standard als solches ohne jegliche Abweichungen einzuhalten und zu implementieren. So sind z.B. die von der FMI-Organisation bereitgestellten FMI-Header-Files in das Export-Tool und in den Co-Simulator zu integrieren. Anpassungen dürfen hier nicht vorgenommen werden. Die modelDescription.xml muss in jedem Fall zu der bereitgestellten Binärdatei passen. Bei Änderungen müssen beide Dateien synchronisiert werden.

Die FMI-Association bietet auch einen so genannten „FMU-Checker“ (https://fmu-check.herokuapp.com/) an, mit welchem die modelDescription.xml der FMU auf Konformität geprüft werden kann. Weitere Möglichkeiten zur Validierung und Überprüfung stehen über folgende Links zur Verfügung:

- https://fmi-standard.org/validation/
- https://github.com/modelica/reference-fmus
- https://github.com/INTO-CPS-Association/FMI-VDM-Model
- https://github.com/modelica-tools/FMUComplianceChecker/releases/tag/2.0.4

Über den „FMU-Checker“ werden Fehler in der Umsetzung der XML deutlich. Dieser Syntax-Test bildet einen wesentlichen Bestandteil der Validierung. Wenn die FMU im Checker einen Fehler wirft, wurde die modelDescription.xml nicht korrekt erstellt. Die Überprüfung der XML auf Konformität sollte im besten Fall sowohl beim Komponenten-Hersteller als auch im Co-Simulator (vor der Simulation) durchgeführt werden. In manchen Erstellungs-Tools findet die Überprüfung durch den „FMU-Checker“ bereits beim Bauen der FMU automatisch statt. Generell sollte der Komponenten-Hersteller den Anspruch haben, keine fehlerhaften FMUs für den produktiven Betrieb in den Umlauf zu bringen.
Im Co-Simulator besteht die Möglichkeit, beim Import der FMU eine kurzen Infotext auszugeben. Hier können beispielsweise die aus der modelDescription.xml gelesenen Informationen aufgezeigt werden. Nicht vollständige Angaben, Annahmen durch den Co-Simulator (z.B. DefaultExperiment) oder Fehler in der XML können geloggt und angezeigt werden. Wenn die FMU nicht kompatibel mit dem Co-Simulator ist, bedarf es einer eindeutigen Warn-Meldung. Ziel ist es, dass der Standard in allen Anwendungsfällen eingehalten und eine reibungslose Simulation im Co-Simulator ermöglicht wird.

Eine systematische und vollumfängliche Validierung der FMUs wird vorausgesetzt. Fehlverhalten und Probleme im Modell können die VIBN beeinträchtigen und zu Verzögerungen führen.

7.5 Versionierung, Handling und Updates von FMUs

Die exportierte FMU bildet nur eine Momentaufnahme des Source-Codes zu einem bestimmten Zeitpunkt. Um einen Überblick zwischen verschiedenen Modell-Versionen zu schaffen, bietet der FMI-Standard auch die Möglichkeit einer Versionierung. Die entsprechende Versions-Nummer wird hierfür in der modelDescription.xml hinterlegt. Redundante Informationen sind zu vermeiden, sodass die Version möglichst nicht zusätzlich in der FMU-Benennung zu platzieren ist. Eine geeignete und gepflegte Versionierung ermöglicht es, Verhaltensmodelle in VIBN-Projekten gezielt zu aktualisieren. Es empfiehlt sich, bei der FMU-Erstellung, analog zur Software-Entwicklung, Datenmanagement-Tools (wie z.B. Git, SVN) zu verwenden. Ein Zusammenhang zwischen der FMU-Versionsnummer und dem Firmware-Stand der realen Komponente muss hergestellt werden. Dies kann beispielsweise über die Dokumentation geschehen (vgl. Abschnitt 7.3). Ändert sich die Funktion der realen Komponente, muss auch die Versionsnummer der FMU hochgezählt werden. Solange die FMU nicht direkt aus der Firmware ausgeleitet wird, macht es wenig Sinn, die Versionierung der Firmware-Entwicklung für die FMU-Erstellung direkt zu übernehmen.

Analog zur Software-Entwicklung kann mit dem Prinzip von „Major“, „Minor“ und „Patch“ (vgl. Abbildung 28) gearbeitet werden (vgl. Semantic Versioning 2.0.0) [13]. Die erste validierte Version startet mit der Versionsnummer „1.0.0“. Dabei wird die Major-Version hochgezählt, sobald gravierende Änderungen

vorgenommen werden, welche keine Kompatibilitäten zu vorherigen Modellen zulassen. Beispielsweise wurden gravierende Anpassungen im Aufbau der FMU vorgenommen. Die FMU ist somit nicht mehr eins zu eins austauschbar, da es Änderungen an den Schnittstellen (Inputs, Outputs, Parameter) gegeben hat. Ein Update des Verhaltensmodells sollte nicht im laufenden VIBN-Projekt erfolgen, sofern die Änderungen nicht dringend erforderlich sind. Die vorherige Major-Version wird nicht mehr weitergepflegt. Werden neue oder erweiternde Funktionen integriert, so ist die Minor-Version hochzuziehen. Die neue FMU-Version ist eins zu eins mit der Vorgänger-Version kompatibel. Ein Umstieg ist nicht trivial und sollte, nur getätigt werden, sofern es das Projekt fordert. Ein Bugfix (Fehlerbehebung) oder eine Anpassung sowie Optimierung in der Logik führt zur Änderung der Patch-Version. Die Patch-Version kann möglicherweise auch der Revisionsnummer der FMU entsprechen. Ein Update der FMU bei einer Patch-Änderung im laufenden Projekt wird empfohlen.

Abbildung 28: Zusammensetzung einer Versions-Nummer nach dem Standard Semantic Versioning

Bei Bedarf kann die in *Abbildung 28* aufgeführte Zusammensetzung während des Entwicklungsprozesses erweitert werden. So besteht die Möglichkeit, über ein „dev“ dem Anwender mitzuteilen, dass es sich um einen frühen Entwicklungsstand mit exemplarischen Teilfunktionen handelt. „beta“ signalisiert eine abgeschlossene Entwicklung. Allerdings steht eine Validierung anhand eines SPS-Projektes und der Einsatz in einem Pilotprojekt noch aus.

Eine wichtige Fragestellung besteht darin, wie der Anwender mitbekommt, dass eine neue Version des Verhaltensmodelles existiert. Dies kann direkt vom Komponenten-Hersteller gepusht werden, sobald dieser eine Verbesserung im Modell durchgeführt hat. Alternativ muss der Anwender, z.B. bei Problemen, aktiv nach einer neuen Version des Verhaltensmodells suchen.

Eine neue Modellversion fordert ein Update und möglicherweise die Integration in ein bestehendes VIBN-Projekt. Logische Änderungen in der FMU (z.B. bei Bugfixes) können leicht nachgepflegt werden, da nur die Binärdatei neu geladen werden muss. Signaländerungen (z.B. Benennung, neue Signale) fordern meist manuelle Anpassungen durch den Anwender. Derartige Änderungen gilt es möglichst zu vermeiden. Generell ist beim FMU-Update höchste Vorsicht geboten. Ein Austauschen von FMUs oder Teilen davon (z.B. DLL, modelDescription.xml) kann

zu schwerwiegenden Fehlern führen. Um den Update-Prozess für den Anwender möglichst angenehm zu gestalten, kann der Co-Simulator bzw. die Assistenz die modelDescription.xml beider FMU-Versionen miteinander vergleichen. Auf diese Weise wird deutlich, ob sich Signale, Datentypen oder Signalnamen geändert haben. Es ist zu beachten, dass alle Instanzen einer FMU geupdatet werden. Im Mischbetrieb zweier verschiedener Versionen können Fehler auftreten. Ein automatisierter Update-Prozess durch den Co-Simulator ist anzustreben. Bevor die Modelle geupdatet werden (z.B. automatisiert über einen globalen Importer), ist ein Hinweis an den Anwender zu geben und auf dessen Bestätigung zu warten. Bei einem händischen Neuimport durch den Anwender kann es passieren, dass alle verknüpften Signale und Parameter neu definiert werden müssen.
Der Co-Simulator kann die Version der FMU aus der modelDescription.xml auslesen und in seinen Metadaten hinterlegen. Zusätzlich besteht für den Co-Simulator die Möglichkeit, die GUID (vgl. Abschnitt 5.4.3) ebenfalls zu prüfen. Diese ändert sich meist bei jedem Build der FMU und signalisiert somit Änderungen im Modell, auch wenn die Version nicht hochgezählt wurde. Anpassungen in den Schnittstellen-Signalen müssen dringend erkannt werden, da andernfalls Abstürze der FMU einhergehen können.

Eine gepflegte, strukturierte Versionierung unterstützt sowohl den Ersteller als auch den Anwender der FMU. Die Fehlersuche, der Modell-Update-Prozess sowie die kontinuierliche Modell-Optimierung werden dadurch erheblich vereinfacht.

7.6 Lizenzierung von FMUs

Bei der Erstellung einer FMU müssen interne Lizenzen, Lizenzen von Templates und Bibliotheken sowie die Bild- und Verwendungsrechte abgelegt werden. Generell ist es zu empfehlen, bei jeglicher Art von Software eine Lizenz zu hinterlegen. Diese bietet rechtlichen Schutz und garantiert die Kontrolle über die Nutzung der Software. Selbst wenn die FMU zur freien Nutzung erstellt wird, sollte dies in Form einer entsprechenden Lizenz dokumentiert werden. Bei der FMU-Erstellung werden teils Third-Party Bibliotheken oder Open-Source-Software verwendet. Die dazugehörigen Lizenzen sind entsprechend in der FMU bereitzustellen. Hierfür ist in der FMU-Struktur im Ordner „documentation“ ein Unterordner „licenses“ anzulegen, welcher als Ablageort für derartige Lizenzen dient. Der FMI-Standard selbst (z.B. C-Header, XML-Schema-Dateien) wurde offen unter der CC-BY-SA 4.0 Lizenz veröffentlicht und steht über eine 2-Clause BSD zur Verfügung [11].

Der FMI-Standard bietet die Möglichkeit, Verhaltensmodelle zu lizenzieren. Es ist jedoch zu beachten, dass im Anlagenbau eine Vielzahl an Komponenten-Hersteller Einsatz finden, sodass bei unterschiedlichen Lizenzierungsmechanismen schnell eine unübersichtliche Gesamtsituation entsteht. Die Lizenzierungsmethoden sind verschieden. Sie reichen von Dongles über Lizenzschlüssel bis hin zu Servern, Clouds sowie Verträgen (vgl. Abbildung 29). Diese verschiedenen Lizenzmechanismen müssen aufgesetzt, integriert und unter Umständen

umgezogen werden. Zudem sind viele VIBN-Setups vom Firmennetzwerk gekapselt und bilden ein isoliertes System. Auch der Aufwand für die Pflege und die Verlängerung der Lizenzen ist enorm. Der Anlagenbetreiber (OEM) muss die Modelle (digitaler Zwilling) von verschiedenen Anlagenbauern betreiben, sodass dieser eine noch größere Vielfalt abdecken muss. Zudem müssen auch alte Modelle nach mehreren Jahren weiterhin lauffähig sein. Generell ist die individuelle Lizenzierung von FMUs im Kontext der VIBN nur schwer umsetzbar und wird vom Anwender häufig nicht geduldet. Lizenzprobleme während der VIBN würden den gesamten Prozess zum Stoppen bringen. Somit muss im Falle einer Lizenzierung eine stabile, zuverlässige sowie einfache Lösung gefunden werden. Eine häufige Lizenzprüfung kann zu Performance-Problemen führen. Ebenfalls ist darauf zu achten, dass die FMU auch offline betrieben werden muss (ohne Verbindung an das Firmen-Netzwerk oder Internet). Eine Art Lizenzierung könnte auch ein auslaufendes Zertifikat (vgl. Abschnitt 9.8) bilden.

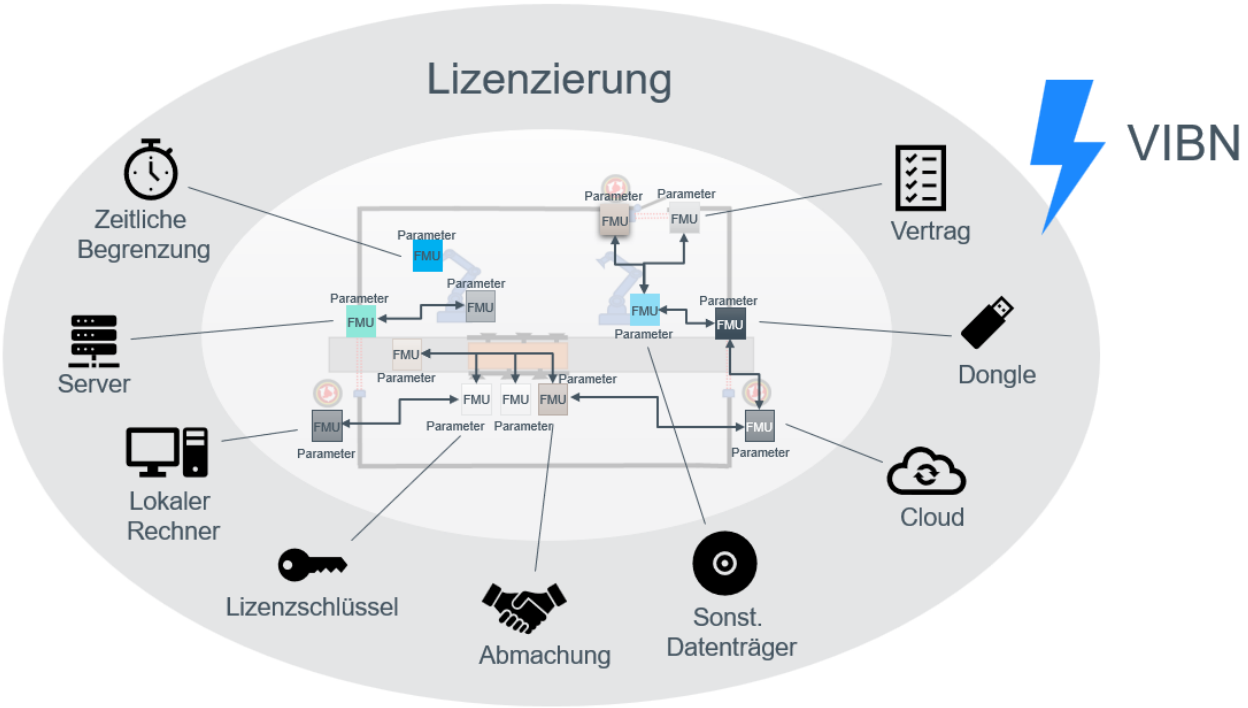

Abbildung 29: Komplexität durch die vielfältigen Möglichkeiten einer Lizenzierung von FMUs im Kontext der VIBN

Die Auslieferung von zeitlich begrenzten Modellen wird als sehr kritisch angesehen. In vielen Fällen führt eine derartige „Lizenzierung“ dazu, dass das bereitgestellte Modell nicht für den Produktivbetrieb (VIBN) genutzt wird. Viel zu groß ist die Gefahr, dass während der VIBN die FMU abläuft und die SPS- und Roboter-Programmierer das Simulationsmodell nicht nutzen können. Die VIBN dauert in manchen Fällen mehrere Monate bis hin zu Jahren. Eine zeitliche Begrenzung von FMUs ist nur für Beta-Versionen zu empfehlen.

Zwischen Komponenten-Hersteller und Anwender muss klar geregelt sein, für welche Zwecke die FMU genutzt und inwieweit die FMU an Kunden oder Lieferanten weitergegeben werden darf. Für den Fall, dass der digitale Zwilling an den Anlagenbetreiber (OEM) weitergegeben wird, müssen auch Lizenzen und Rechte firmenübergreifend übertragen werden. Sollten in diesem Zusammenhang Einschränkungen vorliegen, so ist dies bei der Übergabe oder in der entsprechenden Lizenzierungs-Methode eindeutig zu definieren. Es ist zudem abzuklären, in welchem Umfang Wartungs- und Optimierungsmaßnahmen zu erwarten sind.

Aktuell werden Verhaltensmodelle über die Bibliotheken im jeweiligen VIBN-Tool lizenziert. Austausch-Plattformen für Komponenten-Modelle (wie z.B. CADENAS, TwinStore, …) werden ebenfalls über spezifische Zugänge verwaltet. Im Falle von FMUs können auch die Co-Simulatoren auf Seiten der VIBN-Tools lizenziert werden.

Dieser Abschnitt soll sensibilisieren, welche Herausforderungen mit einer Lizenzierung einhergehen. Gerade für die Einführung des FMI-Standards ist es wichtig, möglichst keine zusätzlichen Hürden aufgrund von aufwendigen Lizenz-Mechanismen in den Weg zu stellen.

Der FMI-Standard bietet die Möglichkeit einer Lizenzierung. Generell sind aufwendige Lizenz-Mechanismen zu vermeiden, da diese die VIBN bezogen auf die Performance und den Verwaltungsaufwand negativ beeinflussen können.

7.7 Rückverfolgbarkeit der Modellnutzung

Im Hinblick auf Exportbeschränkungen oder zur Plagiatsabsicherung wäre es für den Modell-Ersteller wünschenswert, zu wissen, wo seine FMU Anwendung findet. Eine Möglichkeit bietet die Individualisierung der FMU selbst. So könnte für jede reale Komponente auf Basis der Seriennummer ein eigenes Modell erstellt werden, welches nur über die Eingabe der entsprechenden Seriennummer funktioniert. Dieses Verfahren entspricht einer Lizenzierung über Lizenzschlüssel. Da der Anwender in einer frühen Phase jedoch meist keine reale Komponente besitzt, stellt sich dieses Vorgehen als schwierig dar. Häufig steht nicht einmal die genaue Artikelnummer fest.

Alternativ kann dem FMU-Ersteller über das Internet mitgeteilt werden, wo und wie viele seiner FMUs im Einsatz sind. Dies steht allerdings im Widerspruch zur Anforderung, dass die FMU keine Anbindung an das Netzwerk haben darf, solange dies in der realen Komponente nicht vorgesehen ist (vgl. Abschnitt 6.3). Das VIBN-Setup ist häufig getrennt vom Internet aufgebaut (vgl. Abschnitt 9.9).

Im Allgemeinen hilft eine passende Lizenzierung dabei, die Rückverfolgbarkeit und Modellnutzung zu gewährleisten. Häufig kann jedoch nur eingesehen werden, ob der Anwender das Modell nutzt. Die Information über die Anzahl der betriebenen Instanzen kann bei vielen Lizenz-Mechanismen nicht bereitgestellt werden.

Generell erfordern derartige Vorgehensweisen zur Rückverfolgbarkeit einen erhöhten Aufwand beim Komponenten-Hersteller, aber auch beim Betreiber. Der Aufwand muss hier so gering wie möglich gehalten werden.

CAD-Dateien (z.B. .step) ermöglichen aktuell keine Rückverfolgbarkeit. Da die FMU eine gekapselte Blackbox bildet, ist eine Verbreitung von Daten nicht kritisch. Vielmehr sollte in diesem Zusammenhang eine Art Lizenzvertrag zwischen Komponenten-Hersteller und Anwender geschlossen werden. Hierbei ist klar festzulegen, welche Beschränkungen für die Weitergabe an Dritte oder die Veröffentlichung des Modelles gelten (Stichwort: Exportbeschränkungen). Zudem ist zu erwähnen, dass die FMU nicht für das Reverse Engineering genutzt werden darf. Über Verträge, AGBs, NDAs und FMU-Vorschriften kann bestimmt werden, in welchen Bereichen eines Unternehmens (Entwicklung, Produktion, …) die FMU genutzt werden darf.

Die Rückverfolgbarkeit der Modellnutzung ist technisch schwer umsetzbar. Aus diesem Grund sind Vereinbarungen und Verträge bei der Modell-Übergabe abzuschließen.

7.8 Auslieferung der Verhaltensmodelle

Die Bereitstellung der FMU kann auf verschiedene Arten erfolgen. Hierbei geht es darum, wie der Komponenten-Hersteller dem Anwender (Anlagenbauer oder Anlagenbetreiber) die Modelle und dazugehörigen Daten zur Verfügung stellt. Generell wird nicht erwartet, dass der Source-Code mit ausgeliefert wird, sofern die FMU-Entwicklung nicht gezielt durch den Anwender beauftragt wurde. Es ist davon auszugehen, dass die FMU bei Auslieferung bereits validiert wurde. Da sich das Thema FMU-Bereitstellung noch in der Anfangsphase befindet, ist die Marktsituation noch unklar. Eine Entscheidung für ein Format oder eine generalisierte Plattform ist noch nicht getroffen. Der Komponenten-Hersteller muss neben den Anforderungen für die Automobilindustrie auch noch andere Branchen und Bereiche bedienen. Eine Möglichkeit bietet die Verwaltungsschale (vgl. Abschnitt 10.2). Hierbei kann die FMU als AAS-Typ2 über einen Server bereitgestellt werden.

Für den Anwender (Anlagenbauer oder Anlagenbetreiber) wäre ein komponentenbezogener Austausch analog zu den CAD-Daten (evtl. über vergleichbare Plattformen, wie CADENAS) wünschenswert, da hierdurch die Daten gebündelt bereitgestellt und verwaltet werden können. Falls es sich um externe Datenbanken handelt, ist es wichtig, dass diese über eine Schnittstelle (API) angebunden werden können. Bei vielen OEMs existieren eigene Plattformen, über welche der Komponenten-Hersteller bereits CAD-Daten und Handbücher bereitstellt. Hierüber kann auch die Auslieferung der FMUs verwaltet und dokumentiert werden. Der Austausch über eine Plattform sorgt für Verfügbarkeit und bietet Sicherheit gegen Cyber-Gefahren (vgl. Abschnitt 9.9), da die Daten über einen spezifischen und abgesicherten Kanal bereitgestellt werden. Alternativ können die FMUs auch über die Homepage des Komponenten-Herstellers zum Download angeboten werden.

Die Auswahl für eine generalisierte Austausch-Plattform wurde bislang nicht getroffen. Eine Auslieferung von FMUs über die Verwaltungsschale (AAS) ist technisch realisierbar und wird zukünftig angestrebt.

7.9 Haftung für bereitgestellte Modelle

Hierbei ist zu unterscheiden zwischen Schaden durch Schadsoftware (Viren, etc.) und durch Fehlverhalten im Modell. Ersteres wird über Mechanismen, wie die Zertifizierung (vgl. Abschnitt 9.9), abgefangen. Fehlverhalten im Modell kann zu Verzögerungen in der VIBN (Kosten durch verspätete Liefertermine) oder beispielsweise zu Kollisionen an der realen Maschine führen. Auch Personen-schäden während der IBN können daraus resultieren. In diesem Kontext stellt sich die Frage, wer hierbei die Haftung übernimmt.

Im Idealfall verhält sich das validierte Modell analog zur realen Komponente, sodass kein Fehlverhalten auftritt. Generell gilt zu unterscheiden, ob der Komponenten-Hersteller die FMU zur freien Verfügung anbietet oder im Auftrag des Anwenders erstellt. Zusätzlich spielt es auch eine Rolle, ob das Fehlverhalten durch die Implementierung oder durch eine falsche Parametrierung entsteht.

Generell ist eine gute Dokumentation der FMU von Bedeutung (vgl. Abschnitt 7.3). Hierdurch können Fehlersuchen deutlich beschleunigt und Anwenderfehler aufgedeckt werden. Es ist zu empfehlen, dass der Anwender (Anlagenbauer/OEM) die FMU nach Erhalt anhand eines Testprojekts oder über einen EA-Test vor der Nutzung in der VIBN prüft und entsprechend für seinen Anwendungsfall validiert.

Zudem ist es unabdingbar, dass bei der Auslieferung der FMU ein Ansprechpartner seitens Komponenten-Hersteller hinterlegt wird. Dies kann über die Dokumentation (vgl. Abschnitt 7.3) oder die modelDescription.xml erfolgen. Der Ansprechpartner muss bei Problemen erreichbar sein und eine schnelle Abhilfe schaffen. Da die FMU eine Blackbox darstellt, kann der Anwender keine Anpassung am Verhaltensmodell durchführen. In erster Linie verantwortet der Modell-Ersteller (Komponenten-Hersteller, Plattformbetreiber, Anlagenbauer oder Dienstleister) das Verhalten der FMU. Aktuell stellt die Bereitstellung von FMUs ein Angebot vom Komponenten-Hersteller dar. Die Modelle werden nach bestem Wissen und Gewissen modelliert sowie validiert und bieten keinen Anspruch auf Vollständigkeit und fehlerfreie Funktionalität. Zudem ist die Erstellung von Modellen im FMI-Standard für viele Unternehmen neu. Erste Erfahrungen müssen gesammelt werden.

Der FMU-Ersteller kann sich bei der Bereitstellung des Verhaltensmodelles zusätzlich über standardisierte Formulierungen (in der Dokumentation der FMU oder in der modelDescription.xml) und Verträgen bezüglich der Haftung absichern. Viele Firmen haben bereits standardisierte AGBs für Software-Haftung. Diese können ebenfalls für FMUs verwendet und gegebenenfalls angepasst werden. Auch beim Austausch von CAD-Daten existieren derartige Haftungsausschlüsse. Das Thema Haftung ist nicht generalisierbar und somit fallspezifisch zu betrachten.

Fazit:

Generell bietet die VIBN eine Möglichkeit, Fehler und Gefahren vorzeitig zu entdecken. Das Risiko, ohne VIBN in die IBN zu gehen, ist aktuell deutlich höher, sodass keine Haftungsansprüche gestellt werden. Aktuell liegt in den meisten Fällen als Datengrundlage für die Modell-Erstellung nur die Dokumentation der realen Komponente und der SPS-Code (Baustein) vor. Dadurch werden keine Verhaltensmodelle erwartet, welche zu 100% der realen Komponente entsprechen. Die nachgelagerte IBN stellt eine zusätzliche Validierung der Anlage dar, welche mögliche Unzulänglichkeiten im Modell aufdeckt und eine weitere Absicherung darstellt. Durch die Nutzung von FMUs und die Einbindung des Komponenten-Herstellers steigt die Modellqualität, sodass generell eine Verbesserung der aktuellen Situation erreicht wird. Das Thema Haftung für bereitgestellte Modelle wird derzeit als unkritisch gesehen. Für bereitgestellt CAD-Modelle gilt dies gleichermaßen. Derzeit werden Verhaltens-Bibliotheken vom OEM freigegeben und für die VIBN genutzt. In diesem Prozess ergibt sich kein Unterschied, ob FMUs oder native Bibliothekselemente freigegeben werden. Für die Freigabe der Verhaltensmodelle werden diese vom OEM geprüft und validiert. Durch diesen Prozess wird die Verantwortung klar definiert.

Das Thema ist analog zur jetzigen Situation zu betrachten. Vor der Anwendung müssen die Modelle durch den OEM freigegeben werden. Eine Klausel zum Ausschluss von Haftung sichert den Komponenten-Hersteller zusätzlich ab.

7.10 Nutzen von FMUs beim Komponenten-Hersteller

Haupt-Anwender von FMUs sind Anlagenbauer bzw. OEMs. Der größte Mehrwert von standardisierten Verhaltensmodellen wird jedoch dann erreicht, wenn der Komponenten-Hersteller selbst von seinen erstellten FMUs profitiert. So ist es denkbar, dass dieser in Zukunft anstelle der realen Komponente (Hardware in the Loop (HiL)) die virtuelle Repräsentation (Software in the Loop (SiL)) in den Entwicklungs-Prozess einbindet. Die FMU steht bereits vor der realen Komponente zur Verfügung. Auf diese Weise kann der Engineering-Prozess beim Komponenten-Hersteller verkürzt werden, da bereits erste Absicherungen in einer sehr frühen Phase möglich sind. Auf diese Weise unterstützen FMUs auch bei der internen Entwicklung und bei Systemtests. Konkrete Beispiele hierfür sind Integrationstests in bestimmte Steuerungssysteme (z.B. Technologieobjekte im TIA Portal), die Entwicklung von Standard-Modulen (SPS-Bausteinen) oder die Anbindung an Parametrierungs-Tools. Der Komponenten-Hersteller kann zudem detailliertere Modelle für eigene Anwendungen erstellen, welche genau auf den jeweiligen Anwendungsfall zugeschnitten sind. Zudem bietet der FMI-Standard die Möglichkeit, verschiedene Tools miteinander zu koppeln und dadurch die Datendurchgängigkeit im Entwicklungsprozess der Komponente zu verbessern.

Häufig besitzen Komponenten-Hersteller auch einen eigenen Sondermaschinenbau oder entwickeln Test-/Prüfstände für ihre Komponenten. In diesem Zusammenhang können die Verhaltensmodelle (FMUs) auch für interne VIBN-Aktivitäten Verwendung finden.

Um den Mehrwert von Verhaltensmodellen zu maximieren, ist auch eine Nutzung der FMUs beim Komponenten-Hersteller (z.B. im Entwicklungs-Prozess) anzustreben.

8. Integration in die VIBN-Tool-Landschaft

Das VIBN-Setup beim Anlagenbauer/OEM setzt sich aus mehreren Software-Tools zusammen, welche verschiedene Aufgabenbereiche übernehmen. Es wird von einer so gennannten Tool-Landschaft gesprochen. Diese besteht aus Emulations-Tools der Steuerungen (z.B. PLCSIM Advanced, TwinCAT, …), einer Emulation des Roboter-Controllers (z.B. KUKA OfficeLite, ABB RobotStudio, Fanuc ROBOGUIDE), einem Tool zur Verhaltensmodellierung (z.B. SIMIT, WinMOD, ViPer) und einer Visualisierung (z.B. MCD, Process Simulate, SIMLINE, YAMS). In manchen Fällen sind die Visualisierung und Verhaltensmodellierung in einem Tool kombiniert.

8.1 Integration in das Tool zur Verhaltensmodellierung

Die FMU als neutrales Format kann prinzipiell durch einen Co-Simulator an jedes der unter Abschnitt 8 aufgeführten Tools angebunden werden. Da es sich bei den FMUs hauptsächlich um Verhaltensmodelle handelt, hat sich die Integration in das Tool zur Verhaltensmodellierung (z.B. SIMIT, ViPer) als sinnvoll erwiesen (vgl. Abbildung 30). Auf diese Weise wird auch die Möglichkeit geboten, existierende Assistenzen und Automatismen zur Modellgenerierung leicht an den Einsatz mit FMUs anzupassen. Dies geschieht über einen so genannten „Wrapper“, welcher die FMU an ein bestimmtes Schema anpasst und bei der Integration der FMU in die native Tool-Bibliothek unterstützt. Die FMU bildet einen Funktionsbaustein als Blockschaltbild mit Inputs, Outputs und Parametern. Bei geschickter Integration von FMUs merkt der Anwender keinen Unterschied zu nativen Bibliothekselementen. Dadurch wird die Integration von FMUs vereinfacht. Die FMU-Runtime ist meistens direkt mit dem VIBN-Tool gekoppelt, sodass die FMUs mit der Verhaltenssimulation gestartet und gestoppt werden. Häufig bieten diese Tools die Möglichkeit eines Recordings. Eingänge, Ausgänge und interne Zustände werden gespeichert, wodurch bestimmte Fehlerfälle nachgestellt und analysiert werden können. In diesem Kontext müssen FMUs diese Funktionalität auch unterstützen (vgl. Abschnitt 5.4.2 canGetAndSetFMUState und canDe/SerializeFMUState).

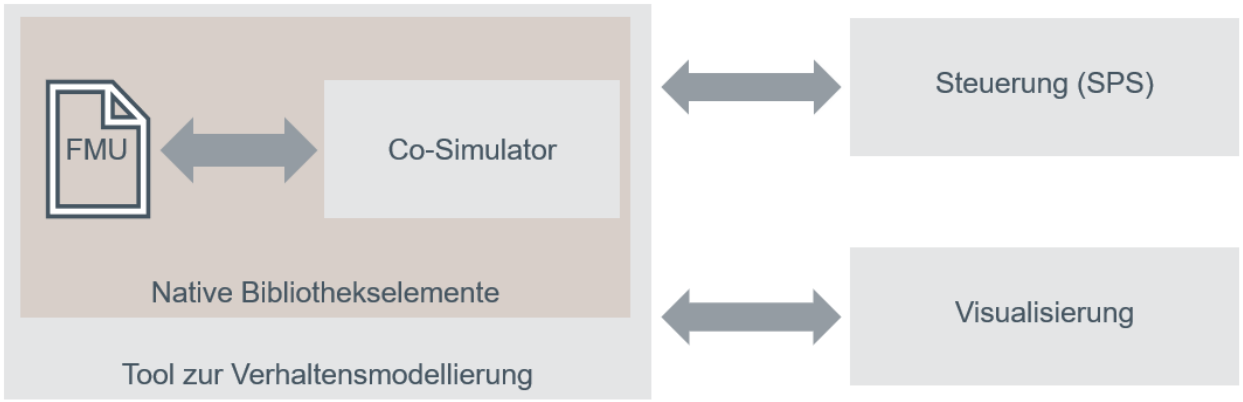

Abbildung 30: Prinzipielle Anbindung von FMUs über das Tool zur Verhaltensmodellierung

Diese Art der Anbindung erlaubt einen hybriden Betrieb mit nativen Verhaltensmodellen und FMUs. Bestehende Bibliotheken können weiterhin genutzt werden. Wird vom Komponenten-Hersteller eine FMU angeboten, ist diese gezielt zu integrieren. So werden die Bibliotheken Stück für Stück mit FMUs angereichert (vgl. Abbildung 31). Für den Fall, dass die bereitgestellten FMUs nicht den Anforderungen des Anwenders genügen, können diese im jeweiligen VIBN-Tool in einem gewissen Rahmen optimiert werden. Beispielhaft sind hier das Aufsplitten eines Wortes auf einzelne Bits oder die zusätzliche Normierung von Signalwerten zu nennen. Diese Form von hybriden Bibliotheken wird kurz- bis mittelfristig in vielen Anwendungsfällen in der Automobilindustrie vorliegen.

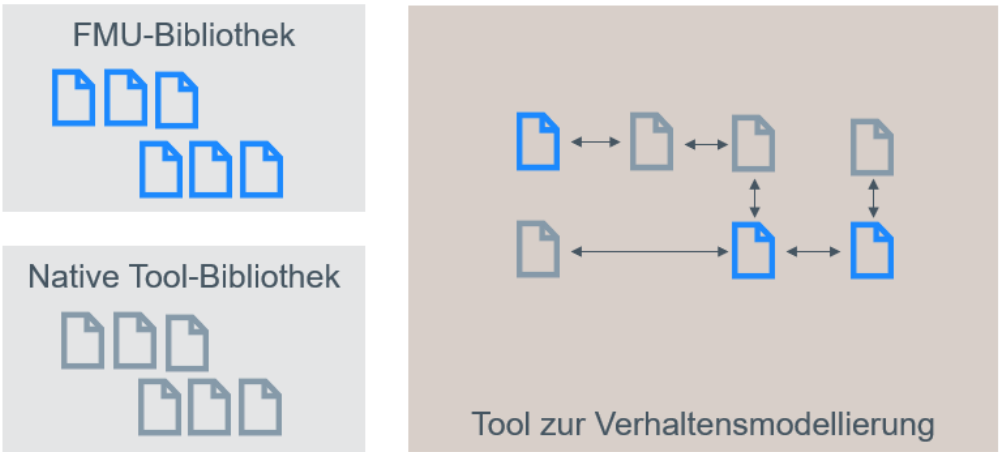

Abbildung 31: Vereinfachte Darstellung eines hybriden Betriebes von FMUs und nativen Bibliothekselementen im Tool zur Verhaltensmodellierung.

8.2 Integration durch einen externen Co-Simulations-Master

Neben der Integration in das Tool zur Verhaltensmodellierung besteht auch die Möglichkeit, die FMUs auszulagern und in einem externen Co-Simulations-Master (z.B. WinMOD, iSILOG PLCConnect) zu betreiben (vgl. Abbildung 32). Die Anbindung an das jeweilige VIBN-Tool findet über eine Schnittstelle (z.B. Shared Memory (SHM), API, TCP/IP) statt. Gerade bei Performance-Problemen im VIBN-Tool kann diese Vorgehensweise helfen, um komplexe Modelle auf einen externen Prozess auszulagern. Auf diese Weise können die Rechenaufwände beliebig skaliert werden, da auch eine Verteilung auf mehrere Systeme ermöglicht wird. Zusätzlich verbessert dieses Vorgehen die Systemstabilität.

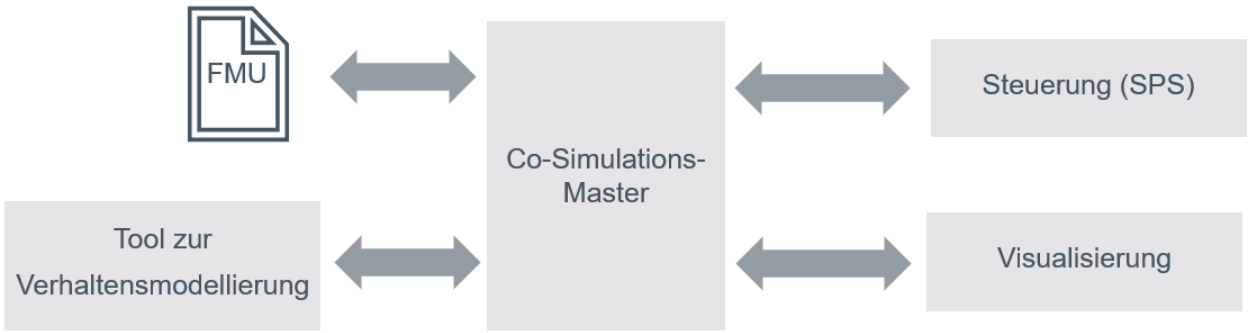

Abbildung 32: Prinzipielle Anbindung von FMUs über einen externen Co-Simulations-Master

Auch in diesem Fall ist ein hybrider Betrieb von FMUs und nativen Bibliothekselementen möglich. FMUs können über Dummy-Bausteine und Makros in das Tool zur Verhaltensmodellierung integriert und an die jeweiligen Steuerungen weitergeschliffen werden. Vorhandene Bausteine können weiterhin nativ im jeweiligen VIBN-Tool simuliert werden.

8.3 Integration in die Visualisierung

Prinzipiell ist es möglich, FMUs auch in das Visualisierungstool (meist 3D-CAD-Umgbung) zu integrieren (vgl. Abbildung 33). Performance-technisch sind diese Tools jedoch häufig stark ausgelastet (v.a. bei physikbasierter Simulation), sodass die Integration von FMUs hier teils Probleme verursacht. Die Visualisierungstools bieten meist nur rudimentäre Möglichkeiten zur Verhaltensmodellierung und sind somit klar von den im Abschnitt 8.1 genannten Tools abzugrenzen.

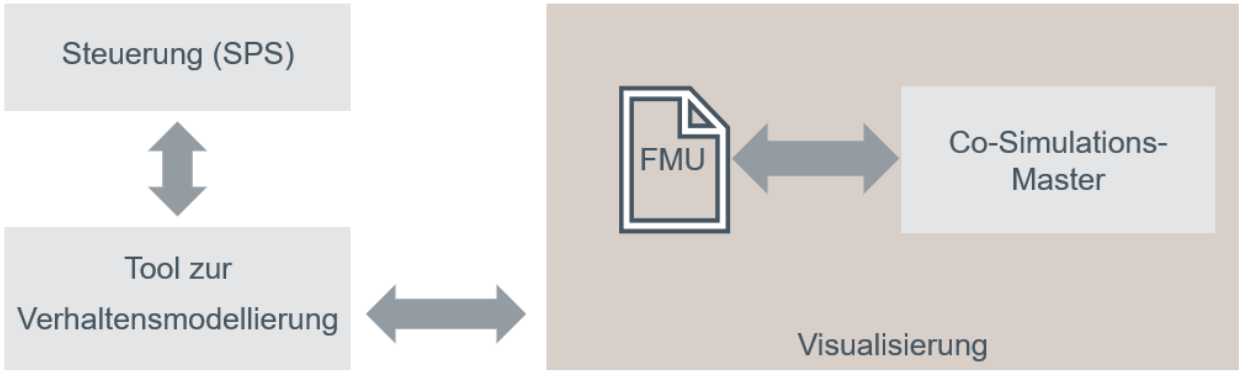

Abbildung 33: Prinzipielle Anbindung von FMUs über die Visualisierung

Des Weiteren existieren VIBN-Tools, welche Visualisierung und Verhaltensmodellierung vereinen. Diese können FMUs direkt einbinden und analog zu ihren nativen Verhaltensmodellen anbinden.

8.4 Zusammenfassung

Da die Verhaltensmodelle als FMUs an mehreren Stellen in das Simulationssetup integriert werden können, gilt es im jeweiligen Anwendungsfall eine geeignete, optimale Lösung zu finden. Eine pauschale Festlegung, wie die FMUs verteilt werden sollen, ist nicht sinnvoll. In allen Fällen muss ein hybrider Betrieb von FMUs und nativen Bibliothekselementen ermöglicht werden. Ein 100%iger Einsatz von FMUs ist kurz- und mittelfristig nur schwer realisierbar.

Aktuell bietet eine Vielzahl an VIBN-Tools bereits eine FMU-Schnittstelle an. Die Integration von FMUs erfolgt jedoch teils noch händisch ohne eine automatisierte Instanziierung. Jedoch sind erste Konzepte bereits validiert, sodass bei gestiegener Nutzung von FMUs weitere Optimierungen vorgenommen werden. In der Automobilindustrie ist der Anteil von FMUs aktuell <5%. Im Sondermaschinenbau wurden bereits Projekte mit >95% FMUs erfolgreich durchgeführt.

Es existieren verschiedene Möglichkeiten, wie FMUs in die VIBN-Toollandschaft integriert werden können. Es ist im jeweiligen Anwendungsfall eine passende Lösung zu finden.

9. Anbindung an das Simulationsmodell

Um eine problemlose Integration von FMUs in das VIBN-Tool zu ermöglichen, gilt es verschiedene Aspekte zu betrachten. Diese müssen sowohl auf Ersteller- als auch auf Anwender-Seite berücksichtigt werden. Kapitel 9 führt hier verschiedene Themenbereiche auf und unterstützt bei der Anbindung von FMUs an den digitalen Zwilling.

9.1 Umsetzung des Standards (Implementierungstiefe)

Um einen reibungslosen Einsatz des FMI-Standards im Umfeld der VIBN zu ermöglichen, ist es unumgänglich, dass die genutzten VIBN-Tools den FMU-Co-Simulator in einer ausreichenden Implementierungstiefe umgesetzt haben. Der FMI-Standard ist umfangreich. Nicht alle Funktionalitäten und Optionen sind im Kontext der VIBN von Relevanz. Es ist auf Kapitel 5 zu verweisen, welches die erforderlichen Umfänge des Standards aufführt. Hier werden verschiedene Flags beschrieben und relevante Funktionen genannt. Prinzipiell ist der Co-Simulator nicht anhand spezifischer Beispiele zu implementieren. Vielmehr bedarf es einer möglichst umfänglichen Integration des FMI-Standards. Für den Fall, dass bestimmte Funktionalitäten und Methoden nicht umgesetzt werden, gilt es die entsprechenden Flags auszuwerten und dem Anwender eine klare Rückmeldung zu geben. An dieser Stelle ist zu erwähnen, dass sich der FMU-Co-Simulator möglicherweise in einer anderen Systemumgebung ungleich verhält. Im Windows-Kontext sind Tests auf Windows Desktop (Windows 10/11) sowie auf Windows Server (sofern unterstützt) nötig.

Neben FMUs aus verschiedenen Erstellungs-Tools müssen weitere Anwendungsfälle analysiert werden. Zu Testen sind beispielsweise Modelle mit einer großen Anzahl an Ein- und Ausgängen (>500). Hier wird ersichtlich, ob das VIBN-Tool (Co-Simulator) auch visuell damit umgehen kann. Viele interne Status-Variablen können ebenfalls zu Problemen im Co-Simulator führen. Zusätzlich besteht die Möglichkeit, dass umfangreiche FMUs die Speicherkapazität des Solvers übersteigen und diesen zum Absturz bringen. Es gilt im Co-Simulator zu untersuchen, wie viele Instanzen vom gleichen Modell geladen werden können (Mehrfachinstanziierung). Ebenso bedarf es einer Analyse, ob der Co-Simulator mit verschiedenen Versionen des Standards (hybrider Betrieb) und mit FMUs aus verschiedenen Erstellungs-Tools umgehen kann. In diesem Kontext sind auch Skalierungs-Tests mit mehreren hunderten FMUs durchzuführen.

Die von verschiedenen Tools generierten FMUs müssen problemlos simuliert werden. So ist es beispielsweise nicht ausreichend, seinen Co-Simulator ausschließlich anhand der Standard-Test-FMUs (z.B. BouncingBall) zu validieren. Es ist zu empfehlen, dass die Hersteller von Templates und Export-Tools Beispiel-FMUs zur Verfügung stellen, anhand derer der Co-Simulator getestet werden kann. Auf diese Weise werden Fehlinterpretationen des Standards und Unzulänglichkeiten früh erkannt und der Co-Simulator gezielt optimiert.

In der aktuellen Situation müssen FMUs vor ihrem Einsatz immer auf Funktionalität im entsprechenden Co-Simulator geprüft werden. Zukünftig soll dieser Vorabtest beim Anwender möglichst entfallen. Sobald die FMU vom Komponenten-Hersteller freigegeben wurde, muss diese später auch mit den gängigen VIBN-Tools kompatibel sein. Die Herausforderung liegt hier vor allem auf Seiten des Co-Simulators, welcher verschiedene Funktionen, Datentypen oder Flags vollständig implementieren muss.
Generell sollte auf Ersteller-Seite auch berücksichtig werden, dass nicht alle Co-Simulatoren die volle Funktionalität umgesetzt haben. Die FMU ist so zu entwickeln, dass erweiterte Funktionalitäten nicht zwingend für die Simulation der FMU nötig und als optional anzusehen sind. Die Advanced-Eigenschaften können genutzt werden, falls der Co-Simulator diese unterstützt. Die Core-Funktionalität muss sichergestellt sein. Als Beispiel ist die Flag „canGetAndSetFMUState“ zu nennen (vgl. Abschnitt 5.4.2). Hier muss die FMU auch lauffähig sein, wenn der Co-Simulator diese Funktion nicht ermöglicht. Ein weiteres Beispiel sind nicht unterstützte Datentypen, welche auf die im VIBN-Tool vorhandenen gemappt werden. Im Co-Simulator ist dem Anwender eine klare Meldung bereitzustellen, welche Funktionalitäten nicht genutzt werden können und wo Abstriche in der Simulation zu erwarten sind.

Der Co-Simulator muss alle für die VIBN relevanten Bereiche des FMI-Standards implementieren. Eine reine Validierung anhand der Standard-Beispiel-FMUs ist nicht ausreichend.

9.2 Benennungskonzept von Ein-/Ausgängen und Parametern

Generell sind die Signalnamen in englischer Sprache zu benennen. Sonderzeichen (z.B. @, “, &, |, %) führen in vielen Tools zu Problemen und sind deshalb zu vermeiden. Auch wenn Umlaute inzwischen von vielen VIBN-Tools unterstützt werden, ist es sinnvoll, darauf in der Signalbenennung zu verzichten. Generell sind die Signale so zu benennen, dass sie für den Anwender verständlich sind. Ein kurzer aussagekräftiger Name ist anzustreben. Die Signalbenennung der FMU ist dem Komponenten-Hersteller überlassen. Es ist zu empfehlen, sich bei der Benennung möglichst an die reale Komponente zu halten und die Bezeichnungen zu übernehmen. Auch wenn es der FMI-Standard erlaubt, ist darauf zu achten, dass nicht der gleiche Signalname für einen Ein-, Ausgang oder Parameter verwendet wird. Ein einheitliches Benennungs-Schema wie z.B. Camel Case, Pascal Case oder Snake Case ist einzuhalten. Dadurch sind die Signalnamen besser lesbar.
Aktuell wird in der Automobilindustrie von den OEMs ein jeweils eigener Standard für die Signalbenennung verwendet, welcher das Mapping der verschiedenen Signale vereinfacht. Generell ist es bei standardisierten Verhaltensmodellen nicht sinnvoll, den Benennungsstandard von einzelnen Firmen (OEMs) zu nutzen. Ein globaler Standard ist derzeit nicht vorhanden. Häufig können Komponenten an verschiedene Bussysteme angebunden werden. Es ist nicht sinnvoll, die Benennung an eine bestimmte Kommunikation (z.B. PROFINET) zu koppeln. Auch

wenn die Integration auf Anwender-Seite dadurch erschwert wird, ist die Benennung möglichst generalisiert durchzuführen. Der Aufwand des Mappings fällt auf den Anlagenbauer (OEM) ab. Die Zuweisung auf die standardisierte Signalbenennung (OEM) findet möglicherweise im jeweiligen VIBN-Tool oder automatisiert in Form einer Assistenz über einen Wrapper statt.

Bei einer geeigneten Signalbenennung wird der Aufwand für diesen Wrapper reduziert. Ein einheitliches Konzept vereinfacht die Interpretierbarkeit von verschiedenen Signalen. So kann beispielsweise der Signalname darauf hindeuten, ob es sich um ein SPS-Signal handelt oder ob die Kommunikation an die Visualisierung gerichtet ist. Es ist nicht relevant, Informationen über die Datentypen in der Signalbenennung zu hinterlegen, da diese bereits in der modelDescription.xml genannt sind. Hierdurch entstehen nur Redundanzen, Fehlerquellen sowie unnötig komplexe Signalnamen. Auch ein Bezug auf die Signalrichtung ist irrelevant (z.B. „von“, „an“), da über die modelDescription.xml eindeutig beschrieben wird, ob es sich um einen In- oder Output handelt. In diesem Zusammenhang werden rein redundante Informationen hinterlegt.

Bezogen auf eine Anreicherung von Informationen im Signalnamen existiert bislang kein einheitlicher Standard. Im Rahmen des DIAMOND-Forschungsprojektes wurde folgende Konvention als Orientierungshilfe getroffen (vgl. Abbildung 34):

- **PLC_**: Mit diesem Präfix werden Signale benannt, welche von bzw. an die Steuerung gehen. Diese Signale existieren auch in der realen Komponente und bilden die Schnittstelle zur SPS. Die Datentypen sind hier analog zur realen Komponente zu wählen.
- **VISU_**: Dieses Präfix signalisiert, dass das Signal von oder an die Visualisierung geht. Die Signale existieren in der realen Komponente nicht. (z.B. Trigger-Signal eines Sensors von der Visualisierung, Positionswert von einem Antrieb an die Visualisierung) Die Visualisierung kann auch einer 2D-Simulation (z.B. Etiketten-Simulation) entsprechen.
- **TRIG_**: Auf diese Weise wird signalisiert, dass es sich bei dem Signal um ein zusätzliches Signal des Komponenten-Modells handelt, welches in der Realität nicht existiert. Die Signale werden meist über ein Bedienfeld im VIBN-Tool angesprochen. Darunter fallen beispielsweise Fehlertrigger oder sonstige Signale zur Werker-Interaktion (z.B. Bedienknöpfe, Türzuhaltungen, Berechtigungen).
- **PARA_:** Hierüber werden alle Parameter zusammengefasst, welche dazu verwendet werden, das Verhaltensmodell auf eine bestimmte Komponente anzupassen oder zu Parametrieren. Zu nennen sind beispielsweise die Zylinderlänge, Betriebsart, Zeiten, Positionen, Geschwindigkeiten oder Beschleunigungen.
- **PROC_**: Zur Beschreibung von Prozess-Informationen kann diese Abkürzung Verwendung finden. Mögliche Aspekte stellen die Position, der interne Zustand, die Kraft, Drehzahl oder die anliegende

Druckluft dar. Ebenso schließt diese Bezeichnung Signale mit ein, welche zum Informations-Austausch mit weiteren Verhaltensmodellen dienen. (z.B. Luft, Strom, Fluid).

- **ROBOT_**: Über dieses Präfix kann ein Signal gekennzeichnet werden, welches an die Roboter-Steuerung angebunden wird. Beispiele hierfür bilden Schweißzangen oder Robotergreifer.

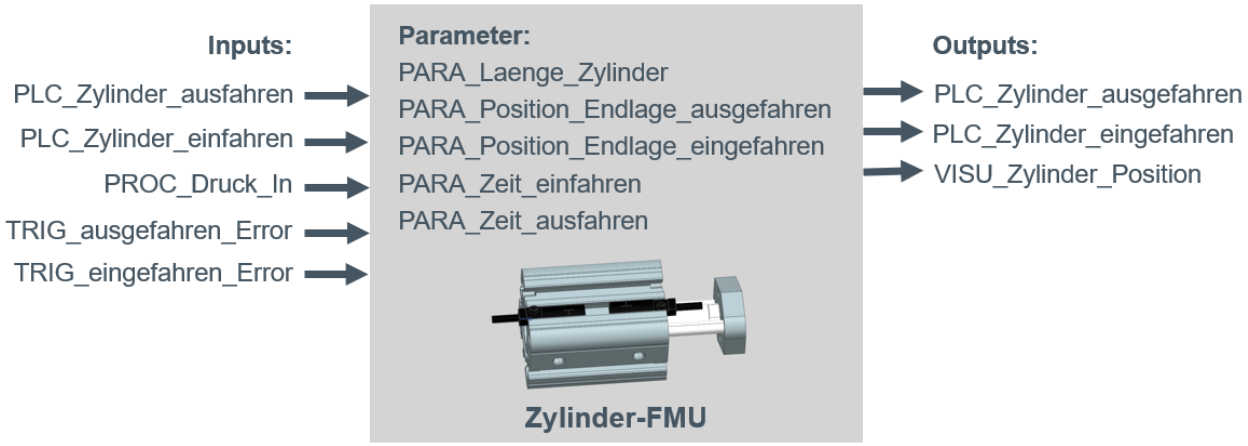

Abbildung 34: Bezeichnung mit Präfix zur Vereinfachung der Anbindung von FMUs am Beispiel eines Pneumatik-Zylinders

Für den Fall, dass der Anwender Schwierigkeiten mit den Signalnamen hat und Anpassungen für seinen Wrapper nötig sind, kann er über den SSP-Standard (vgl. Abschnitt 10.1) eine Schale um die FMU legen. Über SSP-Variablen können auf Anwender-Seite die Signalnamen frei umbenannt werden.
Die Komponenten-Bezeichnungen vom Anlagenbauer (z.B. BMKs) sind nicht Bestandteil der Signal- oder FMU-Namen und werden bei der Instanziierung der FMUs vergeben. Die Vergabe des FMU-Namens ist dem Ersteller frei überlassen. Allerdings ergibt es Sinn, den Firmen-Namen des Komponenten-Herstellers in die Benennung zu integrieren. Des Weiteren ist es zu empfehlen, die Artikelnummer der spezifischen Komponente aufzuführen. Der FMU-Name wird in diesem Abschnitt aufgeführt, da dieser, je nach Art der Instanziierung, Teil des Signalnamens sein kann. Falls mehrere Instanzen einer FMU im Co-Simulator simuliert werden, ist es gängige Praxis, die FMU-Signalnamen mit den jeweiligen Instanz-Namen anzureichern. Dadurch werden die Signale eindeutig. Die Benennung der FMUs sollte sich bei der Instanziierung an den BMK-Bezeichnungen aus dem EPLAN orientieren. Dadurch können Signale später automatisiert miteinander verbunden werden.

Die Benennung der Signale ist dem FMU-Ersteller überlassen. Präfixe können dem Anwender bei der Anbindung an das Simulationsmodell helfen.

9.3 Default-Werte

Die Default-Werte sind aus der modelDescription.xml zu entnehmen. Vor allem bei Parametern und ggf. Inputs ist dies von besonderer Bedeutung. Zudem ist es wichtig, dass der Anwender diese Werte in der Simulation überschreiben kann. Dadurch wird eine Parametrierung über den Co-Simulator/das VIBN-Tool ermöglicht. Die Default-Werte sind so zu wählen, dass die FMU nach Simulationsstart möglichst lauffähig ist (vgl. Abschnitt 7.3).

Die Default-Werte sind aus der modelDescription.xml in den Co-Simulator zu übernehmen.

9.4 Anforderung an die Schnittstellen (Datentypen)

Abhängig von der Version des Standards (FMI 2.0/3.0) sind verschiedene Datentypen definiert (vgl.

Datentyp	modelDescription.xml	FMI 2.0	FMI 3.0
Float	single		✓
Double	double	✓	✓
Int8	byte		✓
UInt8	unsignedByte		✓
Int16	short		✓
UInt16	unsignedShort		✓
Int32	int	✓	✓
UInt32	unsignedInt		✓
Int64	long		✓
UInt64	unsignedLong		✓
Bool	boolean	✓	✓
Char	string	✓	✓

Tabelle 5). Der FMI-Standard 2.0 bildet ausschließlich gebräuchliche Typen, wie Boolean, Integer, Real oder String ab. Sobald verschiedene Integer-Formate (8/16/32-Bit, signed oder unsigned) benötigt werden, ist die neuere Version 3.0 zu verwenden. Des Weiteren werden erst mit der FMI-Version 3.0 Arrays unterstützt. In vielen Fällen helfen Arrays dabei, die FMU übersichtlicher zu gestalten und die Anbindung an das Simulationsmodell und die Steuerung zu vereinfachen. Gerade bei Komponenten wie Kameras, Drucker oder RFID-Leser müssen häufig lange Arrays an die Steuerung weitergeben werden [27]. Im Bereich der Steuerung (SPS und Roboter-Controller) ist es

wichtig, sich an die im SPS-Programm hinterlegten Datentypen zu halten. Diese Datentypen sind analog zur realen Komponente zu wählen. Andernfalls gehen Probleme und möglicherweise Fehlerquellen einher.

Datentyp	modelDescription.xml	FMI 2.0	FMI 3.0
Float	single		✓
Double	double	✓	✓
Int8	byte		✓
UInt8	unsignedByte		✓
Int16	short		✓
UInt16	unsignedShort		✓
Int32	int	✓	✓
UInt32	unsignedInt		✓
Int64	long		✓
UInt64	unsignedLong		✓
Bool	boolean	✓	✓
Char	string	✓	✓

Tabelle 5: Übersicht der unterstützten Datentypen im FMI-Standard 2.0 und 3.0 [11]

In vielen Fällen setzen sich die Signale einer FMU aus verschiedenen Datentypen zusammen. Je nach angebundener Steuerung kann für Integer-Werte auch ein Byte-Flip von Nöten sein, damit der jeweilige Wert richtig interpretiert wird. Der Komponenten-Hersteller hat für jedes Signal einen geeigneten Datentyp zu wählen. Im Hinblick auf die steigende Komplexität in der Kommunikation werden in Zukunft Arrays zunehmend an Bedeutung gewinnen. Eigens definierte Strukturen und Datentypen sowie dynamische Strings, wie sie häufig in der IT-Anwendung finden, werden im Kontext der VIBN nicht benötigt. Aktuell besteht die Herausforderung darin, dass VIBN-Tools teils nur eine begrenzte Auswahl an Datentypen unterstützen und die FMI 3.0 Datentypen mappen müssen. Bei einem Mapping auf einen anderen Datentyp kommt es häufig zu Fehlern und Informationsverlusten. Die Schnittstellen zur SPS-Software sowie zu weiteren VIBN-Tools fordern ebenfalls vermehrt zusätzliche Datentypen, sodass eine Erweiterung, unabhängig vom FMI-Standard, in den verschiedenen Tools und Steuerungen zu erfolgen hat. Teils ist dies bereits geschehen. In der Anbindung zu Visualisierungs-Tools ist es zu empfehlen, möglichst gängige Datentypen zu verwenden. Eine Standardisierung in diesem Bereich ist nicht vorhanden. Die Datentypen aus der Standard-Version 2.0 werden in den gängigen VIBN-Tools (mit der Ausnahme von Strings) vollumfänglich unterstützt.

Der Datentyp String (auf eine bestimmte Länge begrenzt) verursacht aktuell in der VIBN-Toollandschaft teils Probleme, da nicht alle Tools diesen unterstützen. Zudem werden Strings in der Kommunikation zur SPS selten verwendet. Aus diesem Grund wird sich häufig der ASCII-Codierung bedient, welche es erlaubt, Zeichen als Integer/Bytes auszutauschen. Strings sind in der FMU nur zu verwenden, wenn dies in Bezug auf das SPS-Programm von Nöten ist und analog in der realen Komponente vorliegt. In Zukunft werden immer mehr VIBN-Tools den Datentyp String unterstützen. Für den Anwender sind Strings leichter und schneller zu interpretieren. Komplexe Daten können einfach ausgetauscht werden. Gerade für Parameter (z.B. IP-Adressen, Seriennummern, Pfade, Konfigurationen) ist der Datentyp String von Bedeutung.

In Bezug auf die Performance des Gesamt-Systems (digitaler Zwilling) ist es erstrebenswert, die Anzahl an Signalen (Inputs, Outputs) möglichst gering zu halten. Die Parameter können vor Simulationsstart gesetzt werden und beeinflussen somit die Kommunikation während der Simulation nicht zwingend. Generell sind die Schnittstellen auf große Datenmengen (>1000 Signale) ausgelegt. Es ist somit ein geeignetes Mittelmaß zu finden und eine sinnvolle Anzahl an Signalen anzustreben: wichtige Signale müssen nach außen geführt werden, optionale Information (z.B. interner Zustand) können beispielsweise auch über das FMI-Logging nach außen geführt werden. Wörter und Integer-Werte sind als solche an die Schnittstelle zu übergeben und nicht in einzelne Bits aufzusplitten. So bleibt das Modell auch übersichtlicher. Es ist von Vorteil, große Daten-Pakete und Strukturen über Arrays und Terminals (vgl. Abschnitt 5.9) abzubilden. Generell gilt der Leitspruch „so wenig wie möglich, so viel wie nötig“. Der SSP-Standard (vgl. Abschnitt 10.1) kann genutzt werden, um Signale intern zu verschalten und die Schnittstellen zu entlasten. Bezüglich dieses Themas ist auf Abschnitt 9.10 zu verweisen, welcher den Aspekt Performance noch näher aufgreift.

Der Co-Simulator hat alle im FMI 3.0 Standard definierten Datentypen zu unterstützen. In Zukunft wird die Integration von Arrays von zunehmender Relevanz.

9.5 Einheiten

Beim standardisierten Austausch von Daten über Systemgrenzen hinweg, spielen Einheiten eine große Rolle. Der übermittelte Wert in Form einer Zahl muss richtig interpretiert werden [30]. Somit empfiehlt es sich, zur Verknüpfung der einzelnen Komponenten und zur Anbindung an weitere Systeme, wie Steuerungen oder die Visualisierung, an Standards zu halten. Durch die Nutzung eines Einheiten-Standards wird die Anbindung vereinfacht und das Risiko für Fehler reduziert. In diesem Kontext hat sich der weit verbreitete SI-Standard als sinnvoll erwiesen. Sollten im Ausnahmefall Modelle von diesem Standard abweichen (z.B., weil die reale Komponente oder der SPS-Baustein andere Einheiten verwendet), ist dies in der Dokumentation und in den

Signal-Kommentaren besonders hervorzuheben. Im Zweifelsfall dient die reale Komponente zur Orientierung. Auch reale Parameter-Einheiten sind analog in die FMU zu übernehmen.

In diesem Zusammenhang ist zu erwähnen, dass verschiedene VIBN-Tools unterschiedliche Grundeinheiten verwenden. So sind nicht immer defaultmäßig SI-Einheiten hinterlegt. In den meisten Tools können die Einheiten aber in den Einstellungen definiert werden. Eine weitere Möglichkeit ist, über einen Parameter in der FMU zwischen verschiedenen Einheiten umzuschalten. Diese Funktionalität vereinfacht die Anbindung der FMU. Als mögliches Beispiel ist hier der Druck zu nennen. Häufig wird dieser in der Einheit „bar“ ausgedrückt. Laut SI-Standard wird jedoch die Einheit „Pascal“ gesetzt. Weitere nicht SI-Einheiten, welche häufig genutzt werden, sind „mm/s“ oder „l/min“.

Der unter Abschnitt 10.3 erwähnte AML-Standard beschäftigt sich ebenfalls mit einer standardisierten Definition von Einheiten. Da der SI-Standard gerade die Bereiche Transport, Verpackung, Verkauf sowie Informationstechnologie nicht vollständig abdeckt, verweist AML auf UNECE Recommendation N°20 “Codes for Units of Measure Used in International Trade” [30]. Dies stellt eine Art Erweiterung zum SI-Standard dar und ist zu verwenden, sobald zusätzliche Bedarfe in der Definition von Einheiten existieren [28]. Des Weiteren kann auf den ECLASS-Standard zurückgegriffen werden, falls die Recommendation N°20 keine zufriedenstellende Lösung bietet. Hier werden verschiedenste Eigenschaften über Attribute definiert und mittels vorgegebener Einheiten beschrieben. Dabei bedient sich ECLASS an Standard-Einheiten nach DIN und ECE [9].

Der FMI-Standard bietet die Möglichkeit, Einheiten in der modelDescription.xml zu hinterlegen. Dabei werden die Einheiten aus den Grund-SI-Einheiten zusammengesetzt. Zudem kann auch eine Umrechnung angegeben werden. Dem Co-Simulator wird auf diese Weise ermöglicht, einen Einheiten-Check bei der Signalverknüpfung und gegebenenfalls eine Konvertierung durchzuführen. Im Kontext der VIBN sind derartige Informationen und Funktionalitäten aktuell selten implementiert. Mittel- und langfristig bietet diese Option aber ein großes Potential zur Fehlerreduktion und Modelloptimierung [11].

Bei der Modellierung von FMUs und in den Schnittstellen sind gängige SI-Einheiten zu verwenden.

9.6 Bedienfeld für die Ansteuerung im VIBN-Tool

Wie in Abschnitt 7.3 bereits erwähnt, unterstützt der FMI-Standard auch in der Version 3.0 keine Visualisierung für eine GUI. Das Bedienfeld, wie es im VIBN-Tool aktuell modelliert wird, kann nicht standardisiert beschrieben werden. Dadurch wird es weiterhin im Tool zur Verhaltensmodellierung abgebildet. Im einfachsten Fall beschreibt das Bedienfeld das Blockschaltbild der Blackbox mit den Inputs, Outputs und den Parametern. Diese Form wird aktuell in vielen VIBN-Tools beim Import einer FMU zur automatisiert generierten Visualisierung genutzt. FMU-Parameter werden entweder als Eingänge abgebildet oder mit globalen Variablen verknüpft. Zur Handhabung in

der VIBN gestaltet sich dieses automatisiert abgeleitete Bedienkonzept als nicht praktikabel. Es bedarf einer Anbindung von Tastern und Eingabefeldern für beispielsweise Fehler-Trigger. Zusätzlich sind Anzeigen zu erstellen, welche es dem Anwender ermöglichen, wichtige Werte auf einen Blick zu erkennen. Es ist die Aufgabe von Assistenzen und Automatismen, die jeweiligen FMUs mit den im VIBN-Tool modellierten Bedienfeldern zu verknüpfen.

Das Bedienfeld ist nicht Bestandteil der FMU und im VIBN-Tool zu realisieren. Ein beispielhafter Aufbau ist in der Dokumentation der FMU bereitzustellen.

9.7 Parametrierung des Verhaltensmodelles

Mit einem höheren Detaillierungsgrad der Verhaltensmodelle steigt die Anzahl an Parametern und der damit verbundene Aufwand bei der Parametrierung. Während beispielsweise die Verfahrwege eines Pneumatik-Zylinders bislang teils nur in der Visualisierung hinterlegt wurden, ist es bei detaillierteren FMUs auch nötig, diese im Verhaltensmodell zu definieren. Die Parameter einer FMU teilen sich grob in zwei Bereiche auf. Zum einen existieren Parameter, die ein generalisiertes Verhaltensmodell auf einen bestimmten Einsatz oder eine spezifische Komponente anpassen. Je mehr eine FMU abdeckt (ein Modell für eine ganze Produktfamilie), desto größer ist auch die Anzahl an Parametern. Der zweite Parameterbereich orientiert sich an der realen Komponente und dient dazu, Einstellungen für den jeweiligen Anwendungsfall zu treffen. Die Parametrierung gestaltet sich für jeden dieser Bereiche unterschiedlich.

Im idealen Fall bietet eine FMU die Möglichkeit, Parameter, welche analog zur realen Komponente vorliegen, auch auf dieselbe Weise einzustellen. Die Umsetzung kann beispielsweise über eine Web-UI oder eine TCP/IP-Anbindung an ein Parametrierungstool erfolgen, welches die FMU, analog zur realen Komponente, anspricht. Die Parameter zur Einstellung des generalisierten Verhaltensmodelles sind möglichst automatisiert zu definieren. Hier können Informationen von Datenbanken, internen Tools, VIBN-Tools, vom Schaltplan oder aus den CAD-Daten genutzt werden. Zu nennen ist hier beispielsweise der Betriebsmodus eines Antriebes oder die Längen eines Pneumatik-Zylinders. Diese Parameter werden über das VIBN-Tool (Co-Simulator) gesetzt und können über Makros analog zu bisherigen Verhaltensmodellen festgelegt werden. Bei einer geeigneten Integration des Co-Simulators in das VIBN-Tool kann die Parametrierung über die existierenden Assistenzen automatisch erfolgen.

Generell sind im Kontext der VIBN möglichst Parameter vom Typ „tuneable“ zu definieren. Diese können auch während der Laufzeit angepasst werden. Dadurch besteht die Möglichkeit, Fehlerszenarien zu simulieren, da die FMU bei einer Änderung unplausible Werte liefert. „fixed“ Parameter können zu Problemen führen, da die Werte teils vor Simulationsstart der FMUs noch nicht zur Verfügung stehen. Unter Umständen sind hierfür Anstart-Reihenfolgen zu definieren.

Im Zusammenhang mit der Parametrierung ist auf den SSP-Standard zu verweisen, welcher im Abschnitt 10.1 näher erläutert wird. Hier können Parameter (auch „fixed“ Parameter) gesetzt werden, welche bei der Instanziierung die Default-Werte der FMU überschreiben.

Generell sind Parameter als „tuneable“ zu definieren und über den Co-Simulator zu setzen. Der SSP-Standard kann hier bei der Instanziierung zusätzlich unterstützen.

9.8 Instanziierung

Durch die Instanziierung wird aus dem Verhaltensmodell einer Komponente eine spezifisch definierte Instanz (Seriennummer/BMK). Die Bezeichnung der jeweiligen Instanzen muss eindeutig sein (vgl. Bezeichnung im Schaltplan). Über geeignete Assistenzen kann die Instanziierung der FMUs auf Grundlage von hinterlegten Verhaltensmodellen automatisiert erfolgen. In diesem Zusammenhang ist im jeweiligen VIBN-Tool oder Co-Simulator eine entsprechende API, Schnittstelle oder ein Import von Listen (.txt, .csv, …) zu implementieren. Bei geeigneter Integration des Co-Simulators in das VIBN-Tool kann die Instanziierung analog zu bisherigen Vorgehensweisen erfolgen. Unter Umständen ist eine Mapping-Tabelle relevant, um auf die jeweiligen FMUs zu referenzieren.

Eine automatisierte Instanziierung mithilfe des Co-Simulators/VIBN-Tools ist anzustreben.

9.9 Sicherheit (Cyber Security)

Generell gehen aktuell im Bereich der VIBN aufgrund von Netzwerkaktivitäten, gelockerten Windows- sowie Firewall-Regeln und der Vielzahl von Tools und Modellen Sicherheits-Risiken einher. Da die FMU eine gekapselte, kompilierte Datei darstellt, sind mit der Co-Simulation und der damit verbundenen Ausführung einer Binärdatei gewisse Risiken verbunden. Diese gilt es zu minimieren. Es ist nicht ersichtlich, was das Verhaltensmodell (FMU) im Hintergrund auf der Anwender-Seite durchführt. So kann im schlimmsten Fall Malware installiert oder ausgeführt werden. Aktuell werden die meisten FMUs direkt vom Ersteller bereitgestellt, sodass es sich hierbei um eine vertrauenswürdige Quelle handelt. Bei einer umfangreichen Nutzung von FMUs ist es für den Anwender ein deutlicher Mehraufwand, alle FMUs direkt vom Komponenten-Hersteller über eine sichere, analoge Bereitstellungs-Art zu beziehen. Sobald FMUs über das Internet und verschiedenen Plattformen ausgetauscht werden, geht der Herkunftsnachweis verloren. FMUs von einer unbekannten Quelle können sich unter einem falschen Namen ausgeben.

Aus diesem Grund sind Signaturen/Zertifikate von großer Bedeutung. Dadurch kann sichergestellt werden, dass die FMU direkt vom Komponenten-Hersteller (bzw. Dienstleister) stammt und nicht durch Dritte verändert wurde. Die Verantwortlichkeiten sowie die Herkunft der Modelle können durch eine Signatur eindeutig definiert werden. Es wird sichergestellt, dass die FMU aus einer vertrauenswürdigen Quelle stammt und im Nachgang nicht abgeändert wurde.

Mit der Ausführung von FMUs sind Risiken verbunden. Die FMU muss von einer vertrauenswürdigen Quelle stammen. (Signierung oder sicherer Austauschkanal)

9.9.1 Grundsätzliches Prinzip einer Signatur

Die kryptografische Signatur gestaltet sich über private und öffentliche (Public) Schlüssel (Keys) (vgl. Abbildung 35). Über die zu signierenden Dateien wird ein HASH-Code (z.B. SHA512) erzeugt, welcher mittels des privaten Schlüssels in die Signatur der FMU gewandelt wird (z.B. RSA-Signatur, Padding Mode, PKCS). Diese wird ungültig, sobald Anpassungen in der FMU vorgenommen werden (z.B. Anpassung/Austausch der DLL oder modelDescription.xml). Über den öffentlichen Schlüssel (Public Key) kann die Signatur der FMU auf Echtheit geprüft werden. Dieser Schlüssel muss dem Anwender zugänglich gemacht werden. Die Herausforderung besteht in der Bereitstellung dieser öffentlichen Schlüssel [26]. Neben einem direkten Austausch bilden beispielsweise Public Key Infrastructure (PKI) Zertifikate eine weitere Möglichkeit. Über diesen Weg können neben dem Schlüssel zusätzlich Meta-Informationen übermittelt werden. [16] Im letzten Schritt wird aus der Signatur und dem öffentlichen Schlüssel (Public Key) ein Hash generiert, der mit dem Hash der FMU verglichen wird. Auf diese Weise kann geprüft werden, ob es ich um die originale FMU handelt oder ob diese im Nachgang abgeändert wurde. Es besteht die Möglichkeit, für jede Datei (FMU) mehrere Signaturen zu erstellen.

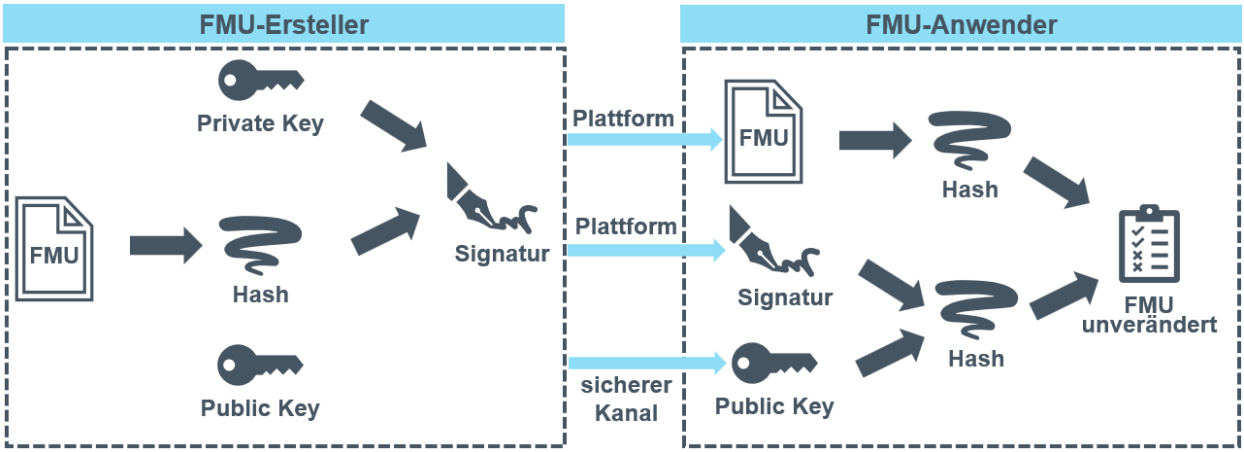

Abbildung 35: Prinzipielles Vorgehen beim Signieren einer FMU über Private und Public Keys

Im Kontext von FMUs können gängige Vorgehensweisen zur Signierung genutzt werden.

9.9.2 Nutzung von Signaturen im Kontext der VIBN

Die Signierung dient zur Authentifizierung des FMU-Lieferanten. Zusätzlich kann eine Signatur vom OEM vergeben werden, um die FMUs für eine weitere Verwendung (z.B. beim Anlagenbauer) freizugeben. Auf diese Weise können gesamte FMU-Bibliotheken sicher ausgetauscht werden. Durch die Signierung wird die FMU auf valide gesetzt und eine klare Verantwortlichkeit über den Inhalt definiert.

Der Co-Simulator (VIBN-Tool) kann in diesem Zusammenhang einen Hinweis beim Ausführen bzw. Laden der FMU geben und den Anwender auf das Risiko hinweisen, sobald kein passendes Zertifikat (Signatur) hinterlegt ist oder die Signatur gebrochen wurde. Durch diese Warnung informiert der Tool-Hersteller den Anwender über die potenzielle Gefahr. Dieser muss entscheiden, ob er das Risiko eingehen will. Die Verantwortung liegt dadurch nicht beim Tool, sondern beim Anwender. Ebenfalls kann ein Hinweis in den Nutzungsbedingungen des Tools auf die Risiken beim Ausführen von FMUs von Drittanbietern ohne Signatur hinweisen. Erlaubte Signaturen (von den jeweiligen Firmen) können über so genannte „White/Trust-Lists“ hinterlegt werden. Falls die FMU mehrere Signaturen beinhaltet, reicht es für die Prüfung aus, wenn eine davon in der „Trust-List“ hinterlegt wurde. Es ist zu empfehlen, dass diese Root-Zertifikate von der IT bei der Installation definiert werden, sodass sich der Tool-Anwender mit dieser Thematik nicht auseinandersetzen muss. Diese Root-Zertifikate („öffentliche Schlüssel“) müssen über eine sichere Plattform zwischen Signatur-Ersteller (Komponenten-Hersteller) und Anwender ausgetauscht werden. Sobald zusammen mit der FMU auch ein Root-Zertifikat angeboten wird, ist Vorsicht geboten. Es könnte sich um ein gefälschtes Zertifikat handeln. Globale Plattform-Anbieter können unter Umständen die Zertifikate der Komponenten-Hersteller sicher verteilen.

Die Signatur ist bei der FMU-Erstellung zu hinterlegen und bei der FMU-Anwendung zu überprüfen.

9.9.3 Signierung in der FMU selbst

Bis in die Version FMI 3.0 wird das Thema Zertifizierung und Signatur nicht im Standard aufgegriffen. Aus diesem Grund ist in diesem Zusammenhang auf die im Abschnitt 9.9.6 aufgeführte Best Practice aus dem DIAMOND-Projekt zu verweisen [28].

Generell ist zu beachten, dass der gesamte Bereich einer FMU (inkl. der modelDescription.xml, dem ressourcen-Ordner, …) in die Signatur miteingeschlossen wird. Dadurch wird der gesamte Inhalt, außer dem Signierungs-Schlüssel selbst, integriert. Dieser Prozess muss vor dem Zippen des Containers geschehen. Die Signierung sowie die Überprüfung des Zertifikates können entweder selbst implementiert werden. Andernfalls existieren Tools (evtl. auch Open-Source), welche bei diesem Prozess unterstützen und in gewisser Weise die Verantwortung übernehmen. Die Signatur ist entweder direkt beim Bauen der FMU zu integrieren (vor allem bei FMU-Templates). Andernfalls besteht die Möglichkeit, die FMU im Nachgang zu signieren. In diesem Kontext können einfache Signierungs-Tools entwickelt oder verwendet werden.

Prinzipiell kann die FMU mit einem zusätzlichen Ordner z.B. „certificate“ (vgl. Abbildung 36) angereichert werden, welcher Signaturen und Dokumente mit Metadaten (Informationen über die Zertifikate/Signatur) enthält [16]. Laut FMI-Standard sind zusätzliche Ordner im Unterordner „extra“ anzulegen [11]. Alternativ besteht die Möglichkeit, beispielsweise einen Unterordner „signatures“ im Ordner „documentation“ anzulegen. Des Weiteren können gängige Zip-Signaturen mit dem Unterordner „META-INF“ (vgl. Abschnitt 9.9.4) genutzt werden. Hierbei wird allerdings ein neuer Ordner auf oberster Ebene (z.B. „META-INF“) erstellt, welcher zum Zeitpunkt der Veröffentlichung dieses Leitfadens nicht im FMI-Standard inbegriffen ist.

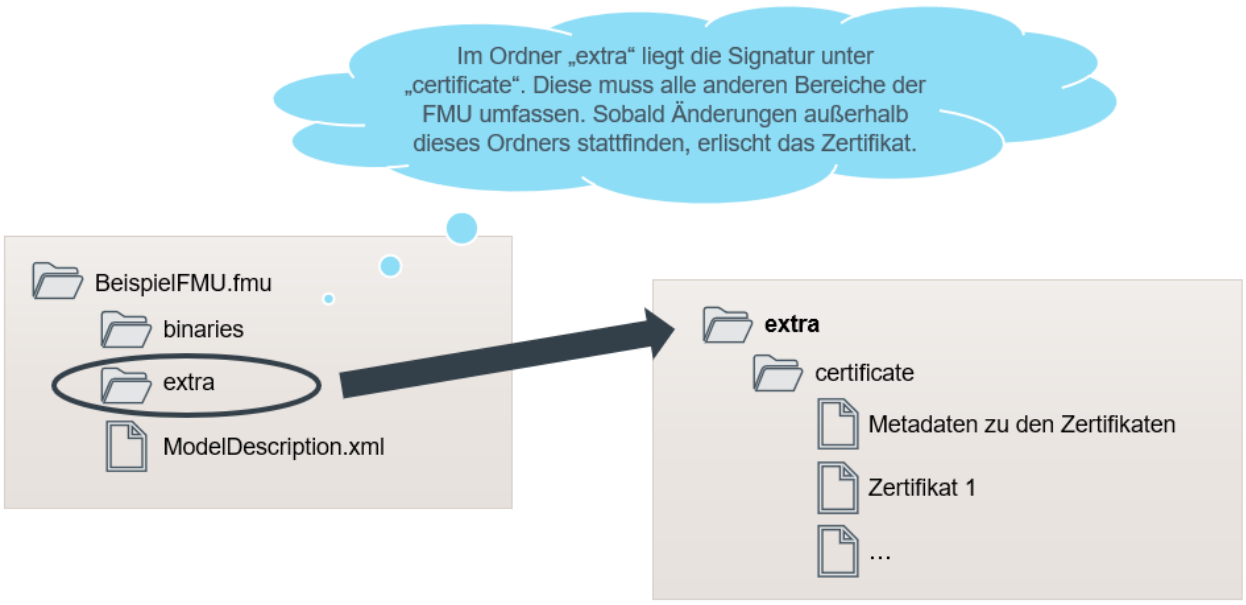

Abbildung 36: Mögliche Umsetzung der Integration von Zertifikaten in die FMU

Aktuell ist im FMI-Standard kein konkretes Vorgehen zur Signierung hinterlegt. Dennoch kann die Signatur im FMU-Container hinterlegt werden.

9.9.4 Signierung über den Zip-Container der FMU

Eine weitere Möglichkeit besteht darin, die Signatur direkt über den Zip-Container zu integrieren. Hierdurch wird der Inhalt der FMU nicht geändert, wodurch die Signatur den gesamten Container umfasst. Dies ist bei einer Integration über den „certificate“- [16] oder „signatures“-Ordner (vgl. Abschnitt 9.9.3) nicht gegeben.

Da es sich bei einer FMU um einen Zip-Container handelt, können auch gängige Vorgehensweisen zum Signieren einer Zip-Datei eingesetzt werden.

9.9.5 Signierung in einer übergeordneten Schale

Unter *Weiterführende Standards* (vgl. Kapitel 10) werden verschiedene Ansätze aufgeführt, wie FMUs in ein übergeordnetes System eingebettet werden können. Dabei besteht auch die Möglichkeit, die Signierung außerhalb der FMU zu platzieren. Das Vorgehen gestaltet bei allen drei Lösungen (AML, AAS, SSP) analog [16]. Generell sind auch in den übergeordneten Standards keine detaillierteren Aussagen vorhanden, was zu signieren ist und wie die Signatur auszusehen hat. Den Prozess zeigt folgende Abbildung 37. Dabei muss vom FMU-Ersteller die FMU über einen weiterführenden Standard bereitgestellt werden. Eine externe Signatur ist nur valide, wenn

die FMU in die Struktur integriert wurde und in einer der standardisierten Schalen abliegt. Bei Verwendung dieses Vorgehens, kann die FMU nur über den jeweiligen übergeordneten Standard ausgetauscht werden. Die alleinstehende FMU als solche beinhaltet keine Signatur.

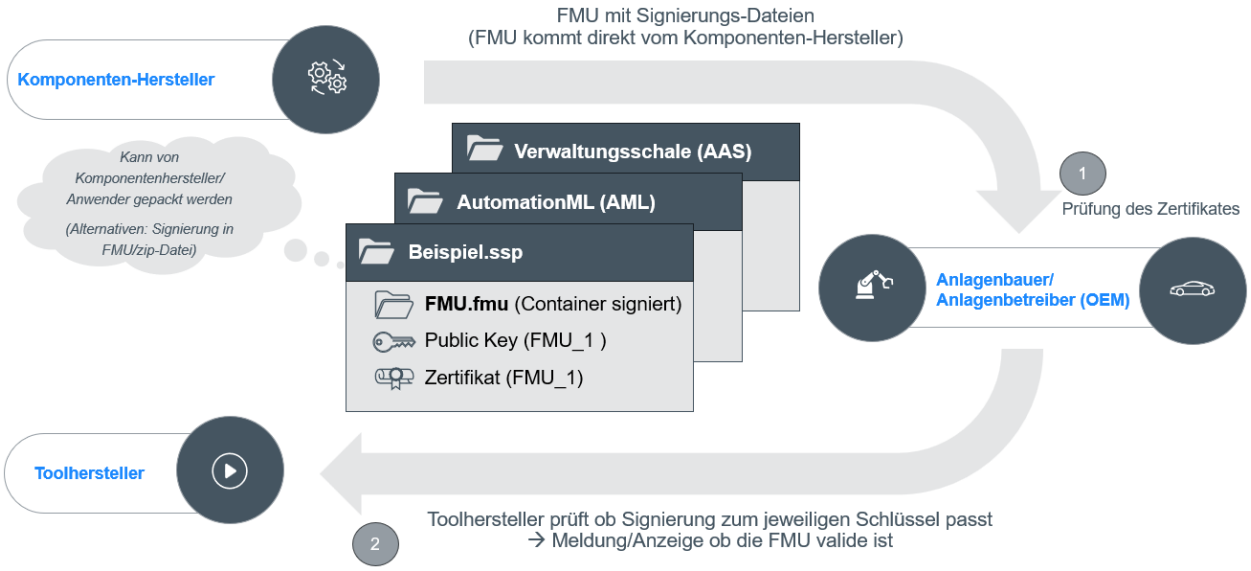

Abbildung 37: Prozess der Bereitstellung von FMUs zwischen den einzelnen Rollen bei einer Signierung mittels übergeordneter standardisierter Standards (z.B. SSP, AML und AAS)

Ein konkretes Beispiel zur Einbindung von Signaturen in Bezug auf FMUs über den AML-Standard wurde im Beitrag „Secure Exchange of Black-Box Simulation Models using Functional Mockup Interface in the Industrial Context“ [26] bereits untersucht und validiert.

Alternativ können FMUs mittels eines übergeordneten Standards (z.B. SSP, AML, AAS) signiert werden. Ein sicherer Austausch von alleinstehenden FMUs ist über diesen Weg nicht möglich.

9.9.6 Möglichkeiten für eine einheitliche Verwendung in der VIBN

Im Rahmen des Forschungsprojekts DIAMOND wurden verschiedene Möglichkeiten untersucht, wie eine einheitliche Nutzung von Signaturen realisiert werden kann. Hierfür wurde unter [27] eine Best-Practice formuliert. Diese kann nun mittels eines „Layered Standards“ in die Modelica Association oder Vereine (wie z.B. ProSTEP iViP) eingesteuert werden. Des Weiteren besteht die Option, für die Signierung sowie die Überprüfung der Signatur ein entkoppeltes Tool zu nutzen. Dadurch wird die Verantwortung ausgelagert und die Anwendung

vereinfacht. Für eine flächendeckende Nutzung von FMUs in der VIBN ist eine strikte Verwendung von Signaturen unerlässlich [16].

Um ein einheitliches Vorgehen zu ermöglichen, sind Best-Practices zu entwickeln (vgl. [27]) und im FMI-Standard oder in Vereinen zu platzieren.

9.9.7 Schutz vor Schadsoftware und ungewollten Aktivitäten

Die Signatur verhindert im Allgemeinen nicht, dass sich Schadsoftware (z.B. Viren) in der FMU befindet. Gerade bei der Verwendung von Third Party DLLs muss der FMU-Ersteller sichergehen, dass diese auch frei von Schadsoftware sind. Signaturen geben lediglich einen klaren Herkunftsnachweis und definieren die Verantwortung. Andere Möglichkeiten, wie die Sicherheit im Umgang mit FMUs gesteigert werden kann, sind folgend aufgeführt:

Eine Vorab-Analyse von DLLs ist allgemein schwierig. Der sicherste Weg wäre die Integration eines Schadsoftware-Checkers direkt in den Co-Simulator, welcher vor dem Laden der DLL eine Prüfung durchführt. Ein FMU-Checker, ähnlich wie er zur Überprüfung der modelDescription.xml genutzt wird (in Form eines Viren- und Schadsoftware-Prüfers) existiert nicht und ist nur mit großem Aufwand realisierbar. Unter Umständen können gängige Virenscanner Schadsoftware beim Ausführen erkennen. Allerdings ist in diesem Kontext keine Garantie gewährleistet.

- Generell besteht die Möglichkeit, bestimmte Aktivitäten der FMU zu überprüfen, bevor diese in Einsatz kommt. So muss sichergestellt werden, dass die FMU keine Verbindung zu einem Server außerhalb des VIBN-Netzwerkes aufbaut. Aktuell werden auch bestimmte reale Komponenten (IT-Endgeräte, Server-Units, …) vor ihrem Einsatz von der IT analysiert und freigegeben (Sandbox-Tests). Ähnliches könnte auch in Zukunft für FMUs genutzt werden. Es ist zudem wichtig, dass die FMU-Applikationen ohne Admin-Rechte ausgeführt werden. Auf diese Weise können wichtige Softwaremechanismen des Betriebssystems (Windows) aktiv bleiben.
- Das VIBN-Setup wird aktuell häufig vom Firmennetzwerk und Internet getrennt und als gekapseltes System (Insel-Netz) betrachtet. Grund hierfür ist auch die Nutzung von Produktdaten (z.B. CAD-Daten), welche es besonders zu schützen gilt. Diese Trennung vom Firmennetz gilt es in Zukunft möglichst zu vermeiden, da Einschränkungen (z.B. beim Datenaustausch, Lizenzservern) mit sich gehen. Generell ist das Thema Schadsoftware auch hier über Verantwortlichkeiten und Nachverfolgung in Form von Signaturen handhabbar.
- Der Freigabe-Prozess von FMUs sollte sich im besten Fall an dem Lieferanten-Daten-Freigabe-Prozess orientierten. Eine Freigabe über die IT (analog zur Software) wird im Falle der VIBN zu lange dauern.

Die Signierung schafft Rückverfolgbarkeit, schließt aber Schadsoftware nicht aus.

9.10 Stabilität von FMUs

Bei der Simulation über FMUs werden viele verschiedene, gekapselte Binärdateien zyklisch ausgeführt. Dennoch muss während der VIBN die Stabilität der Software (Co-Simulator) gewahrt werden. Abstürze von Programmen und im schlimmsten Fall von Rechnern sind zu vermeiden. Ein Neustart bringt häufig lange Wartezeiten und Verzögerungen mit sich. So müssen auch FMUs stabil betrieben und möglicherweise auftretende Fehler abgefangen werden. Hierunter fallen beispielsweise Speicherplatz-verletzungen oder der Zugriff auf nicht vorhandene Ressourcen. Unter Umständen ist es sinnvoll, die FMU-Simulation auf einen eigenen Prozess zu legen, damit eventuell auftretende Probleme nicht zum Absturz des Hauptprozesses (Tool) führen. Im besten Fall sind die FMUs schon beim Laden zu überprüfen und zu validieren. Erste Fehler können hier bereits abgefangen werden. Um die Analyse bei Problemen zu vereinfachen, sind Fehler und Warnungen abzufangen und an den Anwender über die Ausgabe (u.a. FMU-Logs vgl. Abschnitt 5.8) weiterzugeben.

Häufig wird der Simulationsstart im VIBN-Tool mit dem Start der FMU-Co-Simulation gekoppelt. Die gleiche Vorgehensweise liegt auch beim Stoppen der Simulation vor. Im Hinblick auf die Stabilität und Laufzeiteigenschaften, ist es zu empfehlen, diese zwei Vorgänge voneinander zu entkoppeln.

Für den Tool-Hersteller ist es schwierig, ein Fehlverhalten zuverlässig abzufangen. Aus diesem Grund muss der FMU-Ersteller das Hauptaugenmerk auf eine qualitative und stabile Implementierung legen. Die Erstellungs-Tools zum Export von FMUs gilt es ausreichend zu validieren, um ein Fehlverhalten auszuschließen. Es ist unerlässlich, die FMU beim Ersteller vollumfänglich zu testen und auf möglicherweise auftretende Fehler zu prüfen.

In der VIBN ist ein stabiles Anlagenmodell von enormer Bedeutung. Bei der FMU-Entwicklung ist auf Robustheit zu achten. Der Co-Simulator hat Fehlerfälle möglichst abzufangen und zu loggen.

9.11 Modell-Performance und Berechnungsaufwand

Im Rahmen einer VIBN ist es wichtig, dass das Verhaltensmodell in einer bestimmten Zykluszeit berechnet werden kann. Diese ist abhängig von der jeweiligen Hardware, vom Betriebssystem, aber auch von der Auslastung sowie der Anzahl an simulierten FMUs. Generell beeinflussen auch die Implementierung des FMI-Standards im Co-Simulator, das verwendete Rechner-Setup, die eingesetzte Hardware sowie parallellaufende Software die Performance der FMUs.

Verglichen mit den in der VIBN angebundenen Steuerungen (SPS) muss die Zykluszeit von FMUs generell gleich oder kürzer sein. Läuft das Verhaltensmodell langsamer als die Steuerung, kann dies zu Problemen führen. Die FMUs sind dann für diesen Anwendungsfall nicht ausreichend „echtzeitfähig“. Ausnahmen bilden hier

Verhaltensmodelle, welche nicht die Komponenten, sondern Prozesse nachbilden. Hier sind auch Zykluszeiten bis 500ms denkbar. In der Automobilindustrie laufen Steuerungen in komplexen Anlagen teils mit einer Zykluszeit von 20 – 200ms. Im Normalfall liegen diese deutlich darunter (ca. 1 – 2ms). Komponenten-FMUs haben zwingend im Bereich dieser Zykluszeiten zu liegen, da andernfalls eine reibungslose VIBN nicht gewährleistet werden kann. Im Windows-Umfeld ist eine Zykluszeit in einem gängigen VIBN-Co-Simulator von <10ms anzustreben. Eine verspätete Rückmeldung (Feedback) aus der FMU führt dazu, dass Überwachungszeiten in der SPS nicht mehr eingehalten werden können, wodurch Fehlermeldungen aus dem realen SPS-Code die VIBN einschränken.
Die Modellperformance darf nicht unter einem hohen Detaillierungsgrad leiden. Bei einer langsamen Zykluszeit (>50ms) gilt es den LoD gezielt zu reduzieren und die FMU diesbezüglich zu optimieren. Die FMU muss eine vorgegebene Zykluszeit (<10ms) erreichen. Bei einem nicht zu detaillierten Modell ist diese Zielvorgabe mit verschiedenen Erstellungs-Tools meist gut zu realisieren. Häufig bildet nicht die FMU, sondern der Co-Simulator den begrenzenden Faktor, da dieser meist nur eine minimale Zykluszeit von 1ms erlaubt. Viele FMUs können deutlich schneller simuliert zu werden. Im Allgemeinen sind FMUs und Co-Simulatoren sehr performant. So ist es möglich, mehrere hundert FMUs mit einer Zykluszeit <10ms zu simulieren. Bei der FMU-Erstellung sollten Performance-Test in dieser Größenordnung (je nach Komponente) durchgeführt werden.
Grundsätzlich sollte nach dem Prinzip „so schnell wie möglich“ gearbeitet werden. Gerade wenn verschiedene FMUs miteinander über den Co-Simulator oder mit anderen Tools synchronisiert werden, sind Zykluszeiten von 1 – 2ms von Bedeutung. Häufig werden alle instanziierten FMUs miteinander synchronisiert und laufen in einem Takt. Somit bremsen langsame FMUs unter Umständen die gesamte Simulation aus. Bei Systemen mit verschiedenen Zeitscheiben können schnellere FMUs auch langsamer betrieben werden. Bei der betrachteten Zykluszeit handelt es sich um die Zeit zwischen zwei Kommunikationsschritten, an denen die Inputs, Outputs und Parameter mit der restlichen Simulation ausgetauscht werden. FMU-intern kann allerdings mit feineren oder auch gröberen Schritten gearbeitet werden. Passen die internen-Zykluszeiten nicht mit den Co-Simulations-Zeiten zusammen, kann auf Interpolation zurückgegriffen werden. In der Standard-Version 3.0 werden auch so genannte Timer und Clocks angeboten, welche zur Synchronisierung genutzt werden können. Das Handling ist in diesem Zusammenhang komplex und wird aktuell nicht produktiv eingesetzt.
In diesem Kontext wird vom Tool-Hersteller (des Co-Simulators) erwartet, dass dieser die jeweiligen Zykluszeiten der einzelnen FMUs anzeigt oder loggt. Dadurch können Performance-Probleme und hohe Rechenzeiten leichter identifiziert werden. Für den Fall, dass das VIBN-Tool den Co-Simulator integriert, ist eine asynchrone Anbindung zu empfehlen. Dadurch können FMUs mit dynamischen sowie fixen Schrittweiten (fixed/dynamic step size) angebunden werden. Bei einer synchronen und gekoppelten Simulation können „langsame“ FMUs das komplette VIBN-Tool ausbremsen. Hierbei wird häufig von Performance-Problemen in Verbindung mit FMUs gesprochen. Es

ist zu empfehlen, den Co-Simulator auf einen eigenen Prozess auszulagern, damit die Integration von FMUs keinen negativen Einfluss auf das Tool selbst hat.

Durch die Co-Simulation von Verhaltensmodellen (FMUs) lassen sich die entsprechenden Rechenaufwände beliebig auslagern. So können kritische FMUs auf einen neuen Prozess, einen anderen CPU-Kern oder sogar auf weitere Rechner sowie Server verteilt werden. Es ist zu beachten, dass die Anzahl von simulierten FMUs Einfluss auf die Performance hat. Jede FMU benötigt Ressourcen, auch wenn die Komplexität der FMU selbst geringgehalten wird. Die FMUs müssen auf dem Zielsystem als Thread oder Task verteilt werden. Dies spricht generell für eine Zusammenfassung einfacher Komponenten zu einem System/Modul (vgl. Abschnitt 6.2). Aufgrund der Auslagerung über Co-Simulation können mehrere tausende FMUs mit einem gängigen Detaillierungsgrad stabil simuliert werden. Mit der steigenden Anzahl an FMUs erhöht sich meistens die Zeit zum Starten der Simulation. Das Laden, Entpacken und Instanziieren der FMUs dauert (abhängig von der Implementierung) unter Umständen mehrere Sekunden.

Das Auslagern von Rechenaufwänden ist immer stark vom konkreten Anwendungsfall abhängig. Latenzzeiten und Unregelmäßigkeiten bei der Kommunikation zu anderen Rechnern, Servern, etc. erfordern häufig eine Simulation direkt auf dem Zielsystem. Hierin besteht das größte Hemmnis für eine Simulation von FMUs in der Cloud. Die Schnittstellen sind hier meist zu langsam. Somit können vor allem rechenaufwendige FMUs, welche weniger zeitkritische Signale austauschen, (evtl. künstliche Intelligenz (KI) Anwendungen mit neuronalen Netzen) problemlos auf weitere Systeme ausgelagert werden.

Mit der Verteilung der Verhaltensmodelle geht ein zyklischer Austausch der Schnittstellen-Signale (Inputs, Outputs, Parameter) einher. Es entsteht eine Kommunikation zwischen verschiedenen Prozessen und Systemen. In diesem Zusammenhang ist es zu empfehlen, die Anzahl an Schnittstellen-Signalen auf das Wesentliche zu begrenzen. Bei einer Skalierung von FMUs summieren sich die Signale schnell auf und führen zu großen Datenmengen, welche zyklisch im Millisekunden-Takt ausgetauscht werden müssen. Die Geschwindigkeit leidet darunter. Gerade bei systemübergreifender Verteilung kann der performante SHM nicht genutzt werden. In diesem Zusammenhang ist auf Abschnitt 9.4 zu verweisen.

Die FMUs sind möglichst performant zu entwickeln, sodass Zykluszeiten von <10ms realisiert werden können. Der Co-Simulator (VIBN-Tool) muss die Rechenaufwände gezielt verteilen.

10. Weiterführende Standards

Keiner der im Folgenden aufgeführten Standards steht mit dem FMI-Standard in Konkurrenz oder schließt die Verwendung von FMUs im Kontext der Verhaltensmodellierung aus. Alle diese Standards bieten die Möglichkeit, FMUs zu integrieren und mit zusätzlichen Metadaten anzureichern. Der Co-Simulator kann eine Anbindung der genannten Standards realisieren, sodass FMUs automatisch extrahiert und simuliert werden. Die Standards dienen einem einheitlichen Austausch von Modellen. Sie können vom Komponenten-Hersteller somit auch zum Ausliefern der FMUs an den Anwender verwendet werden. Die Standards sind untereinander integrier- bzw. referenzierbar.

Der FMI-Standard ist mit den Standards SSP, AAS und AML kompatibel und steht in keiner Konkurrenz.

10.1 System Structure and Parameterization (SSP)

Der Standard *System Structure and Parameterization* (SSP) stammt analog zum FMI-Standard von der Modelica Association [1]. Im Kern dient er dazu, mehrere FMUs miteinander zu verknüpfen und verschiedene Parametersätze zu definieren. Im Kontext der Verhaltensmodellierung für die VIBN besteht die Möglichkeit, mehrere FMUs zu einem Gesamtverbund zusammenzufassen und zu parametrieren (vgl. Abbildung 38). Diese Einheit kann wiederum mittels eines übergeordneten Standards, wie AML, in die Struktur der Gesamtanlage eingebunden werden.

Der Komponenten-Hersteller kann den SSP-Standard nutzen, um einzelne Komponenten zu einem Modul oder einer Unit zu verschalten. Somit ermöglicht es der Standard, die Abgrenzung von Einzelkomponenten und Systemen flexibel zu gestalten (vgl. Abschnitt 6.4). So kann beispielsweise ein Gesamtsystem aus mehreren Komponenten (z.B. Portal, pneumatische Einheit) als SSP mitsamt allen beteiligen Einzel-FMUs als Gesamtlösung bereitgestellt werden. Des Weiteren dient SSP dazu, FMUs zu parametrieren. Auf diese Weise können generalisierte FMUs (vgl. Abschnitt 6.5) für den spezifischen Anwendungsfall konfiguriert werden. Generell ist der Standard sehr umfangreich und wird aktuell nur in geringem Umfang in der VIBN genutzt. SSP bietet beispielsweise zusätzlich die Möglichkeit, bei komplexen Anwendungsszenarien verschiedene Parametersätze zu definieren und eine Parameter-Optimierung durchzuführen.

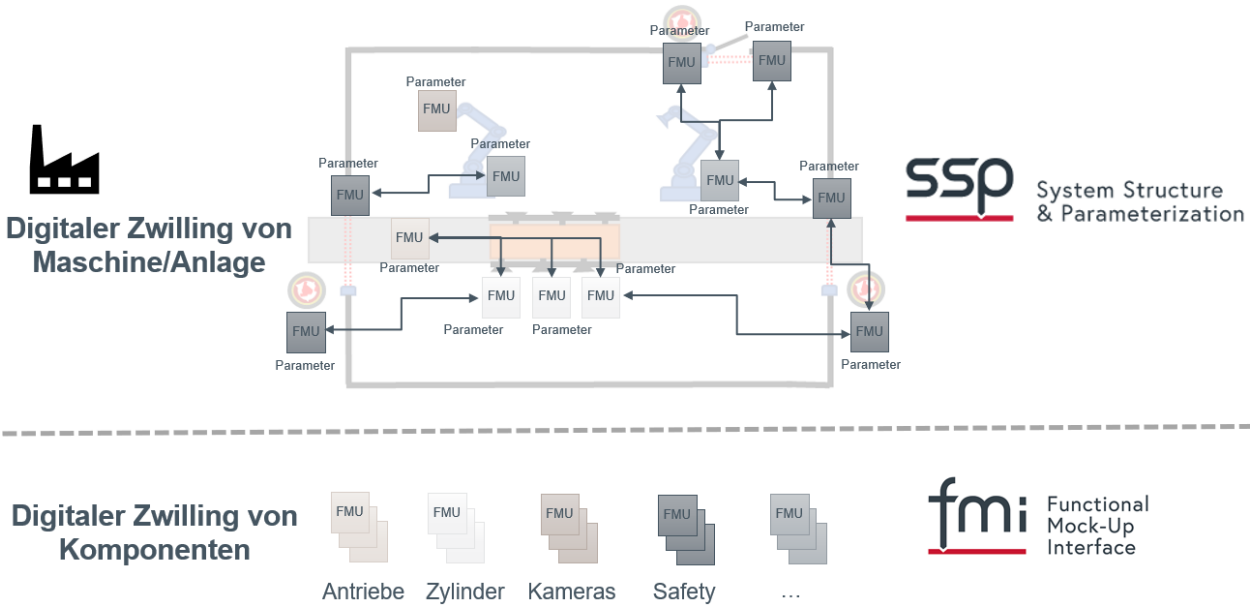

Abbildung 38: Schematischer Aufbau bei der Zusammenführung von FMUs zu einer SSP

Im Kontext der VIBN kann der SSP-Standard für einen so genannten „Wrapper“ verwendet werden, um die Anbindung von bisherigen Automatismen und Assistenzen weiterhin zu ermöglichen. Dabei werden die Signalnamen der FMUs auf den jeweiligen Firmenstandard umbenannt. Zudem besteht die Möglichkeit, die Instanz-Namen zu vergeben sowie eine spezifische Parametrierung zu setzen. Der SSP-Standard erlaubt es, Signale intern zu verschalten und so den Umfang der Schnittstellen-Signale zu reduzieren. Die Performance wird dadurch erhöht (vgl. Abschnitt 9.4). Derartige Signalverknüpfungen können über Assistenzen automatisiert angelegt werden.

Mit Verweis auf Abschnitt 9.9 kann der SSP-Standard auch genutzt werden, um die Signaturen der einzelnen FMUs zu hinterlegen (vgl. Abbildung 39). Im SSP-Standard wurde eine Vorgehensweise bereits definiert und in der Version 2.0 veröffentlicht [12].

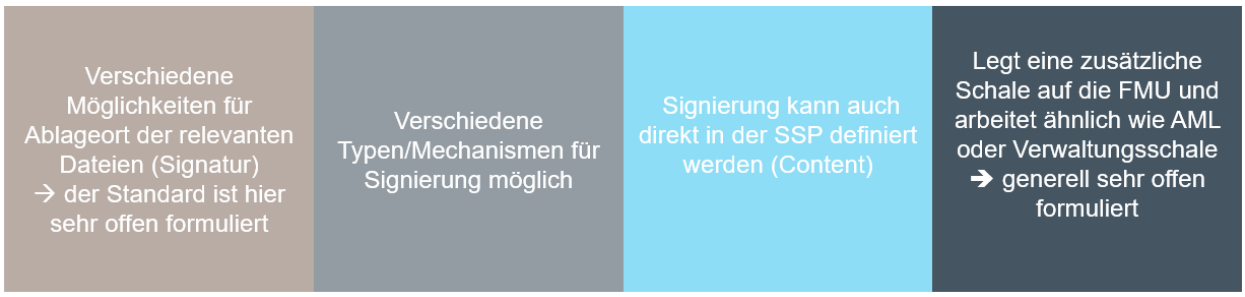

Abbildung 39: Integration von Signaturen über den SSP-Standard in der Version 2.0

Verglichen mit dem FMI-Standard ist SSP bislang in der VIBN-Toollandschaft wenig integriert. Nur wenige Anwendungen unterstützen die Erstellung sowie Simulation von SSPs. Generell ist der SSP-Standard bislang nur in 15 Tools implementiert [12]. Dennoch wurden im Kontext der VIBN bereits Produktiv-Projekte erfolgreich mit Hilfe von SSPs durchgeführt. Hierbei wurden mehrere hunderte FMUs integriert und miteinander verschaltet. Der SSP-Standard wird künftig an Bedeutung gewinnen, da er eine Schale über die FMUs legt und für die Integration in weitere Strukturen (wie AAS und AML, vgl. nachfolgende Abschnitte) geeignet ist.

Der SSP-Standard kann dazu verwendet werden, einzelne FMUs miteinander zu verschalten und zu parametrieren. SSP beinhaltet ein Vorgehen zur Signierung.

10.2 Verwaltungsschale (AAS)

Die Verwaltungsschale bietet eine Technologie, um Daten und Engineering-Inhalte zu transportieren und zu übermitteln. Formuliert und weiterentwickelt wird der Standard durch den Industrial Digital Twin Association e.V. (IDTA) [36]. Bei vielen Komponenten-Herstellern bietet AAS ein führendes Format für digitale Informationen. Im Kontext von DIAMOND wird die Verwaltungsschale dafür verwendet, Daten und Informationen zwischen Komponenten-Herstellern und Anlagenbauern über eine standardisierte Struktur (Container) auszutauschen. Verwaltungsschale hat den Vorteil, dass die Daten gebündelt abgelegt und unabhängig vom Ziel-Tool geladen werden können. Der Standard beinhaltet beispielsweise die Integration von CAD-Daten, Dokumentationen sowie Verhaltensmodellen (FMUs). Die FMUs werden dabei in das Submodell Simulation (SM_Simulation, vgl. Abbildung 40) integriert [3]. Einzelne Komponenten-Hersteller wollen in Zukunft FMUs nur innerhalb einer Verwaltungsschale und nicht als Einzeldatei bereitstellen.

Im Kontext der Verwaltungsschale sind zwei verschiedene Typen zu unterscheiden:

- **Typ 1:** Austausch über die aasx-Datei, welche einen gezippten Container darstellt.
- **Typ 2:** Austausch über einen Server, welcher auf Abruf Daten und Informationen bereitstellt.

Die Integration und Veröffentlichung einer FMU über die Verwaltungsschale Typ 2 wurde bereits prototypisch untersucht und stellt eine Möglichkeit zur Bereitstellung von FMUs dar. Die Verhaltensmodelle werden im Rahmen einer AAS über einen Server angeboten. Denkbare Umsetzungen sind die Integration von AAS in den Shop oder die Website des Komponenten-Herstellers. Dabei bietet Typ 2 verschiedene Varianten an. Einerseits handelt es sich lediglich um eine Bereitstellung. Es besteht jedoch auch die Möglichkeit, die FMU direkt auf dem Server zu betreiben und über eine Schnittstelle in die Simulation zu integrieren. Die Co-Simulation findet in diesem Fall direkt auf dem Server statt. Über eine API kann auf die FMU-Methoden (z.B. doStep) zugegriffen werden. Dies ermöglicht eine Integration in das VIBN-Tool. Ein gravierender Nachteil dieses Vorgehens ist die Latenzzeit bei der Kommunikation mit dem Server. Dieser Ansatz kann als langfristiges Ziel betrachtet werden. Bezogen auf die Sicherheit (vgl. Abschnitt *9.9*) bietet dieses Konzept ebenfalls Vorteile, da die FMU auf einem externen Server ausgeführt wird und somit vom Setup/Firmennetz entkoppelt läuft.

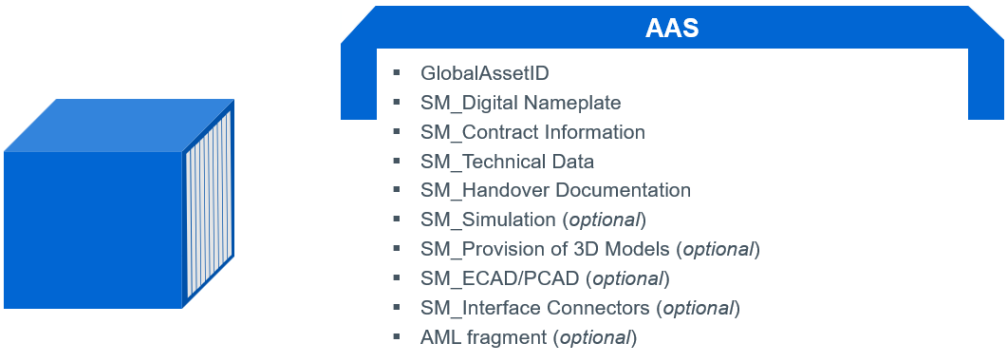

Abbildung 40: DIAMOND-Empfehlung für Komponenten-AAS (FMU im SM(Submodel)_Simulation)

Ein Teil der Informationen (z.B. EAs, FMI-Version, FMU-Versionierung, Generierungs-Tool, Datum, Autor) werden möglicherweise redundant abgelegt und befinden sich sowohl in der modelDescription.xml als auch in der AAS. Die Verwaltungsschale bietet die Möglichkeit, zusätzliche Metadaten zu hinterlegen. Beispiele hierfür sind das Tool, mit dem die FMU validiert wurde, der Zweck und der Umfang des Modelles oder die Einschränkungen bei der Simulation (z.B. Betriebssystem). Zusätzlich kann auch die Versionierung von FMUs verwaltet werden. Über die so genannten Interface-Connectors (SM_Interface Connectors, vgl. Abbildung 40) [4] erlaubt AAS, Schnittstellen

als Teil der Verwaltungsschale anzubieten. Über den Standard kann dem Anwender auch mitgeteilt werden, sobald die Komponente vom Hersteller abgekündigt wurde bzw. in welchem Geltungsbereich die Komponente einzusetzen ist (SM_Technical Data, vgl. Abbildung 40) [37]. Des Weiteren bietet AAS die Möglichkeit, digitale Zwecke zu monetarisieren und verschiedene Detaillierungsgrade von Daten bereitzustellen [3].

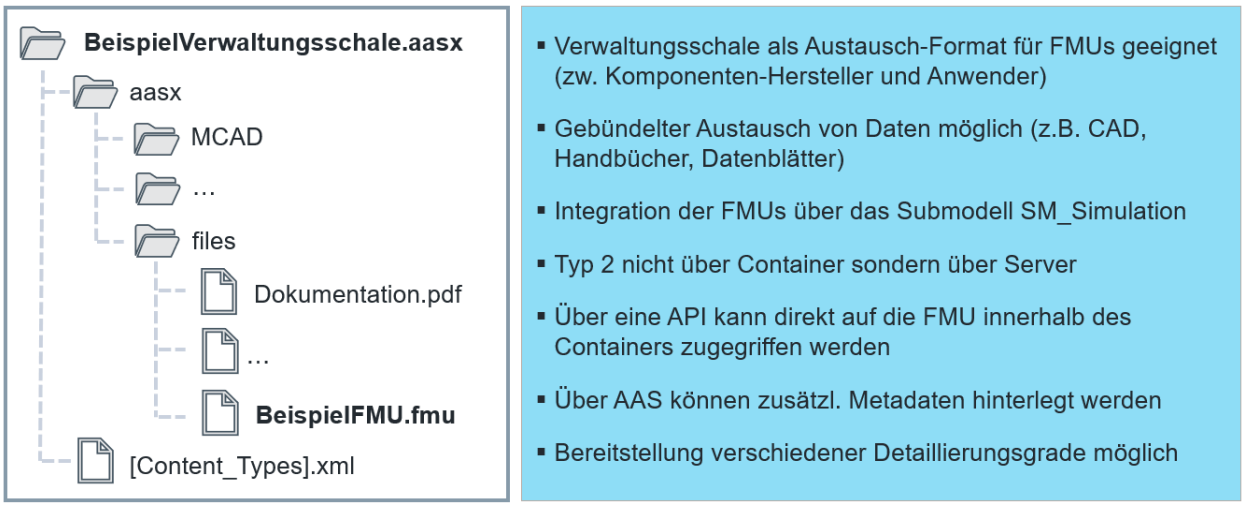

Abbildung 41: Struktur und Überblick zum Thema Integration von FMUs in die Verwaltungsschale

Generell vereinfacht die Implementierung und Nutzung von Verwaltungsschale die Integration des FMI-Standards in das Unternehmen. Über die dadurch stattfindende Standardisierung und Bündelung von Stamm-Daten, können Teile der FMU automatisiert ausgeleitet werden. Neben FMUs können auch SSPs in einer Verwaltungsschale verknüpft werden. In Verbindung mit Verwaltungsschale ist auf den AAS Reading Guide [10] zu verweisen, welcher für die jeweilige Zielgruppe entsprechende Dokumente verlinkt.

AAS bietet die Möglichkeit, FMUs zusammen mit weiteren Dokumenten und Informationen standardisiert auszutauschen. In dieser übergeordneten Schale können auch Metadaten und Signaturen hinterlegt werden.

10.3 AutomationML (AML)

AutomationML bietet im Rahmen von DIAMOND die Möglichkeit, komplexe Anlagen während des Engineerings aufzubauen und zu beschreiben. AML definiert die Anbindung der Komponenten in der Gesamtanlage mitsamt den jeweiligen Verdrahtungen und wird vom Anlagenbauer erstellt. Für den Komponenten-Hersteller bedeutet dies, dass er sich in erster Linie nicht mit AML beschäftigen und keine Daten in dieser Form bereitstellen muss. Nach aktuellem Ansatz ist AML vom Komponenten-Hersteller nur auf Anfrage zu liefern. Im Falle von Modulen und ganzen Systemen, wie z.B. einem Portal, ist es denkbar, dass der Komponenten-Hersteller dazugehörige

Informationen auch über AML bereitstellt. Logiken und Verhaltensmuster können nur bedingt mit AML beschrieben werden. Die Möglichkeiten, die AML hier bereitstellt, sind unter [2] beschrieben. Um komplexe Zusammenhänge zu beschreiben, sind FMUs zu integrieren und miteinander zu verknüpfen. Verglichen mit SSP können auch über AML einzelne Verhaltensmodelle miteinander verknüpft und parametriert werden. Während SSP nur die Verknüpfung von FMUs (Verhaltensmodellierung) erlaubt, besteht in AML die Möglichkeit, zusätzliche Metadaten sowie weitere Engineering-Daten (z.B. CAD, SPS) miteinander in Verbindung zu bringen. Auch SSPs können in die AML-Struktur eingebunden werden.
Generell wird AML im Anlagenbau bereits produktiv genutzt. So werden beispielsweise Informationen vom EPLAN über einen AML-Export ausgeleitet und im TIA Portal importiert. Auch im CAD-Umfeld findet der AutomationML-Standard bereits Anwendung zum Austausch von Daten.

AutomationML dient zur standardisierten Beschreibung von komplexen Anlagen während des Engineerings.

10.4 Integration von AAS und AML im DIAMOND-Projekt

Im DIAMOND-Projekt werden die Standards AAS und AML als Transport- bzw. Referenzierungs-Möglichkeit für FMI/SSP folgend angewendet (vgl. Abbildung 42) *[8,5]*:

1. Komponenten-Hersteller bieten das Verhaltensmodell als FMU über die Verwaltungsschale an. Die FMU wird hierbei über ein Submodell eingebunden. (vgl. Abbildung 40) [3]
2. Beim Anlagenbauer werden die Informationen der einzelnen Komponenten-Verwaltungsschalen über ein Engineering-Tool in eine AML-Anlagenengineering-Struktur (als external) integriert. Die FMU aus „1.“ wird über die AML-Struktur referenziert. Die Integration über AML findet während des gesamten Anlagen-Engineerings Anwendung. (AML-Struktur mit referenzierten Verwaltungsschalen der einzelnen Unter-Komponenten)
3. Bei der Übergabe an den Anlagenbetreiber (OEM) am Ende des Anlagenentstehungsprozesses wird die Anlage in Form einer Verwaltungsschale (AAS) übergeben. Diese referenziert die AML-Struktur und dokumentiert somit das Engineering (Verwaltungsschale für die Gesamtanlage). Dieser Prozess ist nicht Fokus von DIAMOND.

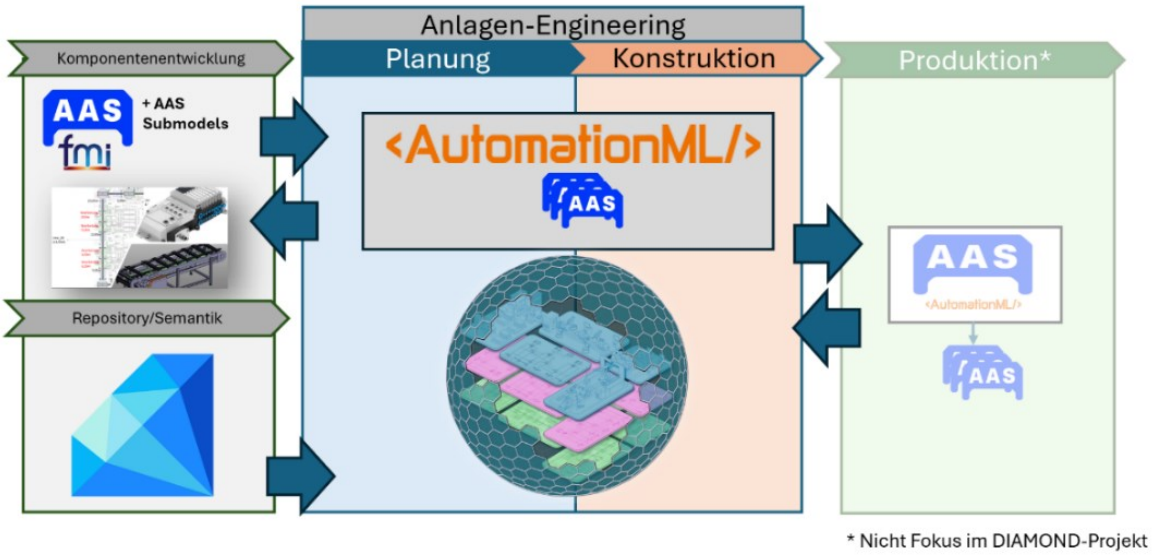

Abbildung 42: Zusammenspiel von AML und AAS während des Engineering-Prozesses im Kontext des DIAMOND-Projektes [8]

Während des Engineering-Prozesses können FMU, SSP, AAS sowie AutomationML Verwendung finden.

11. Ausblick: Hochdetaillierte Verhaltensmodelle als FMUs

Generalisierte Modelle als FMUs können in verschiedenen Engineering-Phasen für entsprechende Anwendungszwecke verwendet werden. Ebenfalls erlaubt es der Standard, hochkomplexe Modelle zu entwickeln. Mit der Bereitstellung von hochdetaillierten und realitätsnahen Modellen steigt jedoch das Risiko von Know-how-Schutz und die Gefahr von Reverse Engineering. Ab einer gewissen Komplexität wird der Schwerpunkt auf das Thema Simulation und nicht auf die VIBN mit ihren Echtzeitanforderungen gelegt. Es ist in allen Fällen zu beachten, dass diese zusätzlichen Aspekte die VIBN in keiner Weise beeinträchtigen dürfen. Falls Abstriche in der Modellperformance oder Stabilitätsprobleme zu erwarten sind, ist eine gesonderte FMU speziell für die erweiterten Anwendungsfälle zu definieren. Im Vordergrund steht ein Verhaltensmodell mit sämtlichen Grundfunktionalitäten, welches keine 100%ige Abbildung der realen Komponente darstellt. Dennoch sollen im Folgenden die Potentiale von hochdetaillierten Verhaltensmodellen aufgeführt werden.

Hochdetaillierte FMUs dürfen auf keinen Fall negative Auswirkungen auf die VIBN haben.

11.1 Nutzung von KI im Bereich der FMU-Erstellung

Mit höherem Detaillierungsgrad steigt auch der Aufwand in der Modellgenerierung. Die Abhängigkeiten von verschiedensten Einflussfaktoren machen die Modellierung zunehmend komplex (z.B. Einfluss von Reibung, Beschleunigung, Masse). Eine Modellierung über logische und physikalische Zusammenhänge ist zeit- und performanceaufwändig. In diesem Bereich spielt KI eine bedeutende Rolle, um kostengünstige, performante sowie realitätsnahe Modelle zu liefern. So kann beispielsweise das Verhaltensmodell anhand einer realen Komponente angelernt bzw. optimiert werden. Es besteht die Möglichkeit, ein bestehendes Modell mit der realen Komponente zu vergleichen (Gap-Analyse). Die Unterschiede sind über KI-Methodiken zu reduzieren.

Die relevanten Daten können über die reale Anlage/Komponente gesammelt werden. Zudem besteht die Möglichkeit, synthetische Daten zu generieren. Unter Umständen können auch Dokumentationen und Handbücher mit Hilfe von KI automatisiert ausgelesen werden und als Datengrundlage Verwendung finden. Zudem ist es denkbar, über KI den Source-Code der realen Komponente zu bündeln und daraus eine FMU abzuleiten. Es ist dabei allerdings fraglich, inwieweit ein Komponenten-Hersteller seinen Source-Code (und damit sein Know-how) einer KI bereitstellen will. Zu klären ist ebenfalls, in welchem Umfang KI im Kontext einer VIBN genutzt werden darf. Gerade im Hinblick auf Safety und Haftung für einen Schaden an der realen Anlage ist dies als kritisch einzustufen.

Aus technischer Sicht ist es im Zusammenhang mit dem FMI-Standard möglich, KI-Methodiken und Funktionen zu verwenden. So können beispielsweise Parameter gefunden und optimiert werden, welche in die FMU

weitergegeben werden. Ebenfalls ist es möglich, KI-Operationen direkt in der FMU zu betreiben. Das Framework TensorFlow bietet beispielsweise die Möglichkeit, KI direkt in die FMU zu integrieren. Allerdings ist zu beachten, dass derartige Operationen die Zykluszeit und das Ladeverhalten von FMUs stark beeinflussen können. Aus diesem Grund finden derartige Modelle aktuell vor allem in Forschungsprojekten Einsatz. Im produktiven Umfeld erfüllen sie nicht die Anforderungen an die erreichbaren Zykluszeiten.
Eine weitere Möglichkeit, wie die FMU-Erstellung durch KI unterstützt werden kann, bilden Chatbots. Da es sich bei FMI um einen offenen, leicht zugänglichen Standard handelt, hat die KI die Möglichkeit, von Veröffentlichungen und dem Standard selbst zu lernen. Aus diesem Grund sind in gängigen Chatbots bereits grobe Aussagen über die FMU-Erstellung möglich. Allerdings ist in diesem Zusammenhang höchste Vorsicht geboten. Die Aussagen sowie die über Chatbots erstellten Modelle (FMUs) gilt es dringend zu validieren. Ein Chatbot kann dazu führen, dass schlechte und invalide Modelle den produktiven VIBN-Betrieb beeinträchtigen.

11.2 Nutzung der FMUs für die Komponenten-Auslegung

Durch detaillierte Verhaltensmodelle wird bereits zu einem frühen Zeitpunkt eine Rückmeldung gegeben, ob die gewählte Komponente ausreichend dimensioniert wurde. So kann beispielsweise vor der Beschaffung überprüft werden, ob die Komponente für ihren Einsatzzweck geeignet ist. In dieser frühen Projektphase sind Änderungen an einzelnen Komponenten noch möglich.

11.3 Energieverbrauch

In Bezug auf Nachhaltigkeit sowie Einsparung von Energie und Kosten rückt der Faktor Energieverbrauch immer mehr in den Fokus. Aus dem Verbrauch können direkte Rückschlüsse auf die damit verbundenen Emissionen geschlossen werden. Es ist denkbar, dass in Zukunft Gesetze oder Richtlinien definiert werden, welche eine klare Aussage zum Energieverbrauch einer Anlage fordern. Somit ist es von Bedeutung, FMUs in Zukunft mit dem Aspekt des Energieverbrauches anzureichern. Dadurch kann die Energie einer Produktionsanlage im Vorfeld abgeschätzt, simuliert und optimiert werden. Es wird deutlich, wie viel Energie eine Produktionsanlage benötigt und wo Energieflüsse liegen. Erste Untersuchungen mit Robotern haben gezeigt, dass das Einsparpotential durch geeignete Modelle beachtlich ist. Diese Idee wird derzeit von mehreren Forschungsprojekten behandelt. Das Thema ist allerdings sehr komplex, da eine Vielzahl an Parametern und Einflussfaktoren zu berücksichtigen sind (z.B. Druck, Zeiten, Reibung). Eine detaillierte Physik muss im Verhaltensmodell und in der Visualisierung abgebildet werden. Für eine realitätsnahe Energie-Analyse sind komplexe und validierte Verhaltensmodelle nötig. Das Thema Energieverbrauch und Nachhaltigkeit lässt sich noch weiter ausbauen: So können auch die Daten aus der Produktion der Komponente und dem Lebenszyklus mit integriert werden. Energieflüsse ermöglichen es, beispielsweise die Abwärme von Komponenten (z.B. Kompressoren) zu analysieren und gezielt zu nutzen. Der FMI-Standard selbst stellt hier ausreichende Funktionalitäten zur Verfügung. In diesem Kontext ist das Thema

Erfassung von Energieverbrauch und CO2-Fußabdruck (PCF) im Zusammenhang mit der Verwaltungsschale (vgl. Abschnitt 10.2) zu nennen. So wird beispielsweise bei der Herstellung der Komponenten ein zugehöriger Wert hinterlegt. In der AAS kann dieser mit dem Verbrauch im Betrieb erweitert werden.

Insgesamt können erste Abschätzungen bereits über einfach gehaltene Modelle mit Standard-Verbräuchen getätigt werden. Die Energiebewertung muss nicht zwingend zu 100% der Realität entsprechen. Erste Analysen erlauben bereits eine grobe Optimierung.

Ebenfalls besteht die Möglichkeit, parallel zum realen Anlagen-Betrieb den tatsächlichen Energie-Verbrauch mit dem simulierten zu vergleichen. Auf diese Weise entsteht ein „Energie-Beobachter“ (vgl. Abbildung 43), welcher bei Abweichungen (analog zur Predictive Maintenance, vgl. Abschnitt 11.6) eine Rückmeldung an den Betreiber liefert. Die reale Anlage kann auf diese Art überwacht werden. Dadurch lassen sich beispielsweise Leckagen bei Druckluft-Anwendungen erkennen.

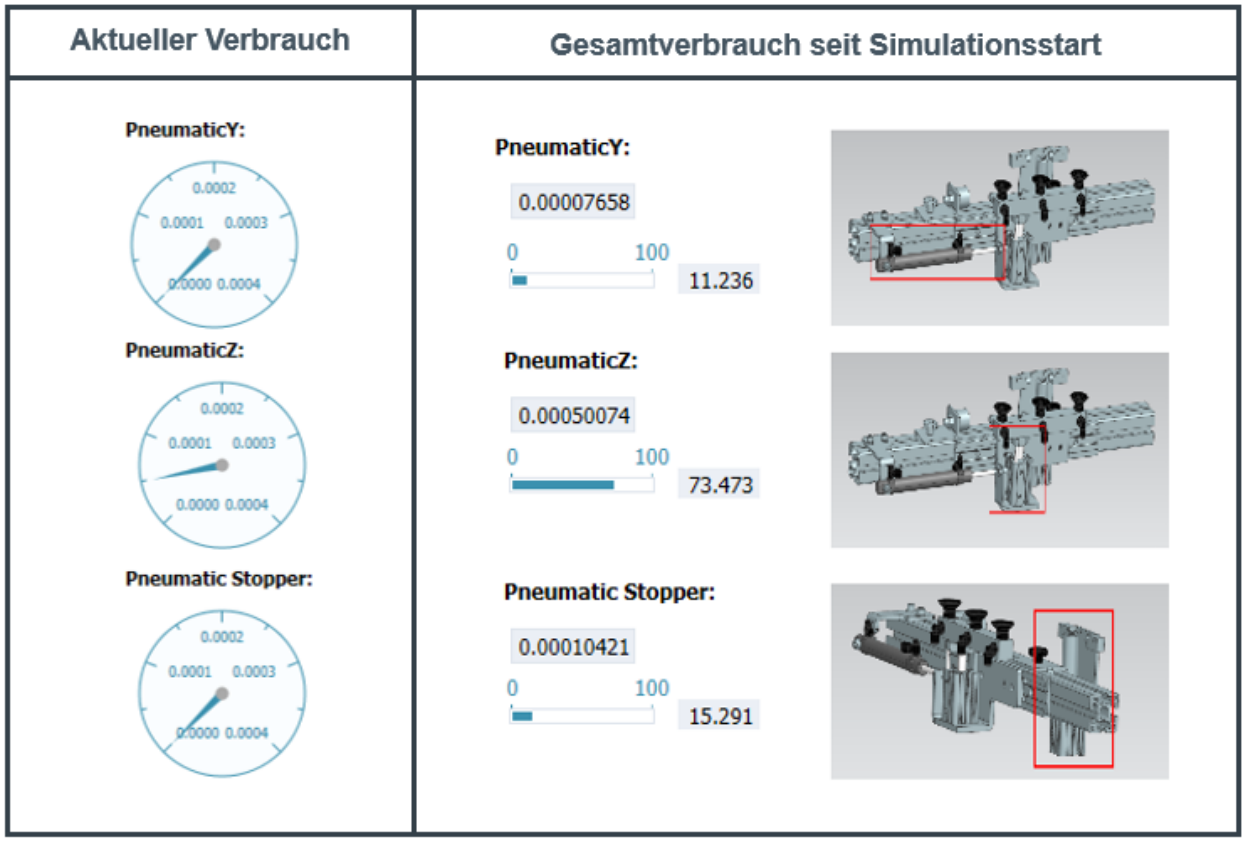

Abbildung 43: Energiebetrachtung am Beispiel Luftverbrauch eines Pneumatik-Zylinders. Zu sehen sind der aktuelle Verbrauch (links) sowie der Gesamtverbrauch seit Simulationsstart (rechts)

11.4 Detaillierte Physiksimulation (Schleppabstand von Antrieben)

Detaillierte Simulationen sind für spezifische Anwendungsfälle von Interesse. Hierzu zählen beispielsweise Pumpen-, Klebe- oder Dosierprozesse. Auch im Bereich von Motor- und Getriebemodellen sind Detailsimulationen von Bedeutung. Durch den FMI-Standard wird eine detaillierte Modellierung von physikalischen Zusammenhängen ermöglicht. So können beispielsweise Temperatur-Untersuchungen getätigt werden. Es besteht die Möglichkeit, Überlast-Zyklen zu analysieren sowie bei der Auslegung zu unterstützen. Verhält sich die Komponente (z.B. ein Antrieb) anders als erwartet, kann darauf reagiert werden.

Bei Positionierantrieben ist die Simulation von Schleppabständen von Bedeutung. Dadurch können Regel- und Einstell-Parameter bereits vorab abgeschätzt werden. Neben dem vorausgesetzten detaillierten Verhaltensmodell ist hier auch eine aussagekräftige Physiksimulation mit Trägheit notwendig. Hier besteht im Kontext der VIBN derzeit Handlungsbedarf, da häufig die Physik stark abstrahiert wird, um die nötige Stabilität und Zykluszeit zu erreichen. Aktuell geschieht die Parametrierung meist über ein so genanntes „Auto-Tuning“, welches über eine Referenzfahrt an der realen Komponente stattfindet.

11.5 Nutzung des FMI-Standards zur Steuerungssimulation

Für den Fall, dass bei bestimmten Steuerungen (z.B. SPS, RC) keine Simulations- bzw. Emulations-Software zur Verfügung steht, kann der FMI-Standard ebenfalls verwendet werden, um Steuerungscode zu simulieren. Aus den Programmier-Tools können FMUs ausgeleitet werden, welche dann über die Schnittstellen (evtl. auch über Robot Operating System (ROS)) an die Simulation (digitaler Zwilling) angebunden werden. Eine FMU als virtuelle Steuerung kann auch parallel zur realen Steuerung simuliert werden und bestimmte Informationen über die Input-/Output-Beziehungen bereitstellen (z.B. über OPC-UA).

11.6 FMUs beim Anlagenbetreiber (Predictive Maintenance)

Durch die Nutzung von FMUs während des Betriebes der realen Anlage bekommt der Anlagenbetreiber zusätzliche Daten und Informationen über die Komponenten. Diese können dem Komponenten-Hersteller zurückgespielt werden. Die Herausforderung besteht darin, den digitalen Zwilling parallel zur realen Anlage zu pflegen und weiterzuentwickeln. So müssen beispielsweise Parameter (z.B. Reglerparameter) und Konfigurationen aus der realen IBN in den digitalen Zwilling zurückgespielt werden. Dieser Aufwand würde sich deutlich reduzieren, sobald die FMU selbständig Daten aus der realen Anlage bezieht und eigenständig die Parameter angleicht. Im Idealfall lernt das Verhaltensmodell (FMU) direkt von der realen Anlage und optimiert sich selbst. Eine Anbindung der FMU kann beispielsweise über OPC-UA erfolgen.

Wird der Zwilling parallel zur realen Anlage simuliert, kann der Instandhalter bei Fehlerfällen eine Wiederholung am Zwilling einspielen und die Ursache gezielt analysieren. Je detaillierter hierbei die Verhaltensmodelle sind,

desto wahrscheinlicher ist es, den Fehler nachstellen zu können. Durch die Anbindung von FMUs können die realen mit den simulierten, idealen Daten verglichen werden. Dies ermöglicht Aktivitäten in Richtung Predictive Maintenance. Die Auswertung selbst kann beispielsweise über KI-Ansätze auch direkt in eine FMU integriert werden (vgl. Abschnitt 11.1). Die Betrachtung und Rückmeldung von Live-Daten einer realen Automatisierungskomponente über einen Server wurde von Komponenten-Herstellern bereits implementiert und könnte als Gegenstück zur idealisierten FMU dienen. Zusätzlich kann über die Verwaltungsschale beispielsweise auch zurückgespielt werden, wo und wie die jeweiligen Komponenten beim Anlagenbauer eingesetzt werden. Derartige Lebenszyklus-Informationen der realen Komponenten dienen zu Verbesserungen bei Neuentwicklungen.

12. Fazit

Wie eingangs erwähnt, bietet die virtuelle Inbetriebnahme ein großes Potenzial zur Verbesserung und Beschleunigung der Entwicklung von Produktionsanlagen. Um diesen Prozess zu ermöglichen, bedarf es umfangreicher Bibliotheken mit Verhaltensmodellen. Neben dem Aufwand, welcher aktuell für die Verhaltensmodellierung betrieben wird, zählen unvollständige sowie fehlerhafte Modelle zu einer großen Herausforderung bei der Erstellung eines digitalen Zwillings für die VIBN. Da jeder Anwender bzw. Tool-Hersteller eigene Modelle zu entwickeln hat, sind der Aufwand und die daraus entstehenden Kosten für die Erstellung und Pflege entsprechend hoch. So macht die Verhaltensmodellierung in vielen Fällen die Erstellung eines digitalen Zwillings (gerade bei Neueinführung) und im Mittelstand unwirtschaftlich.
In Zukunft wird zudem die Komplexität von Produktionsanlagen weiter steigen. Die Anbindung und Verschaltung von hochintelligenten Automatisierungskomponenten erfordert einen höheren Aufwand bei der Programmierung, welcher nur durch die VIBN beherrschbar wird. Analog steigen auch die Komplexität und die Anforderungen an die Verhaltensmodelle der jeweiligen Komponenten. Die Modell-Erstellung anhand von Handbüchern und SPS-Bausteinen wird immer zeitaufwändiger und fehleranfälliger. Aus diesem Grund ist es unerlässlich, dass sich der Komponenten-Hersteller mit dem Thema Verhaltensmodellierung beschäftigt und standardisierte Modelle bereitstellt. Der FMI-Standard erfüllt in diesem Zusammenhang aus technischer Sicht alle Anforderungen. Um den sicheren Austausch von FMUs im Hinblick auf Cyber Security und Nachverfolgbarkeit von Änderungen am Modell zu gewährleisten, wurde im Zusammenhang mit dem DIAMOND-Projekt eine Best Practice [27] veröffentlicht (vgl. Abschnitt 9.9.6). Diese beschreibt, wie mit Signaturen im Kontext des FMI-Standards umzugehen ist.
Der hier bereitgestellte Leitfaden soll die Einarbeitung in den FMI-Standard minimieren und die Qualität der bereitgestellten Modelle erhöhen. Die Einführung von FMI wird rollenübergreifend betrachtet, wodurch bisherige Hemmnisse und Fragestellungen beseitigt werden sollen. Das Ziel besteht darin, dass zu jeder realen Komponente ein Verhaltensmodell als FMU bereitgestellt wird. In Zukunft wird dies beim OEM auch in den Freigabeprozess mit integriert und gefordert werden. Die Entwicklung sollte sich an der etablierten Bereitstellung von CAD-Daten orientieren. Standardisierte Verhaltensmodelle (FMUs) ermöglichen es, eine durchgängige Digitalisierung zu realisieren. Ohne einen digitalen Zwilling kann die steigende Komplexität der Komponenten in Zukunft nicht mehr gehandhabt sowie Liefertermine nicht mehr eingehalten werden.

Das Forschungsprojekt DIAMOND wird von der Europäischen Union (EU) finanziert und vom Bundesministerium für Wirtschaft und Energie (BMWE) gefördert.

Abbildung 44: Logo des Förderträgers und DIAMOND

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

3D	Dreidimensional
AAS	Asset Administration Shell (Verwaltungsschale)
AML	AutomationML
API	Application Programming Interface (Programmierschnittstelle)
BDE	Betriebsdatenerfassung
BMK	Betriebsmittelkennzeichen
BMWE	Bundesministerium für Wirtschaft und Energie
CAD	Computer-Aided Design (computergestützte Konstruktion)
CC-BY-SA 4.0	Creative Common Attribution-Share Alike 4.0
CPU	Central Processing Unit
DIAMOND	Digitale Anlagenmodellierung mit neutralen Datenformaten
DIN	Deutsches Institut für Normung
DLL	Dynamic Link Library
EA	Eingang Ausgang
ECE	Economic Commission for Europe
ECU	Electronic Control Unit
FMI	Functional Mock-up Interface
FMU	Functional Mock-up Unit
GSDML	Generic Station Description Markup Language
GUI	Graphical User Interface
HiL	Hardware in the Loop
HMI	Human-Machine Interface
IBN	Inbetriebnahme
IDTA	Industrial Digital Twin Association e.V.
IO	Input Output
IODD	IO Device Description
IoT	Internet der Dinge (Internet of Things)
KI	Künstliche Intelligenz
LoD	Level of Detail (Detaillierungsgrad)

MDE	**Maschinendatenerfassung**
MiL	**Model in the Loop**
MKS	**Mehrkörpersimulation**
NDA	**Vertraulichkeitsvereinbarung (Non Disclosure Agreement)**
OEM	**Original Equipment Manufacturer**
OPC	**Open Platform Communications**
OPC UA	**OPC Unified Architecture**
PCF	**Product Carbon Footprint**
PDM	**Produktdaten-Management**
PKCS	**Public Key Cryptography Standards**
PKI	**Public Key Infrastructure**
PLC	**Programmable Logic Controller (SPS)**
RC	**Robot Controller**
RFID	**Radio Frequency Identification**
ROS	**Robot Operating System**
RSA	**Rivest-Shamir-Adleman, kryptographisches Verfahren**
SHA	**Secure Hash Algorithm**
SHM	**Shared Memory**
SiL	**Software in the Loop**
SO	**Shared Object Library**
SPS	**Speicherprogrammierbare Steuerung**
SSP	**System Structure and Parameterization**
SVN	**Subversion**
TCP/IP	**Transmission Control Protocol/Internet Protocol**
TIA Portal	**Totally Integrated Automation Portal**
UI	**User Interface (Benutzerschnittstelle)**
VIBN	**Virtuelle Inbetriebnahme**
XML	**Extensible Markup Language**

Literaturverzeichnis

[1] Modelica Association (2025). “ssp System Structure & Parameterization”. URL: https://ssp-standard.org/ (besucht am 30.07.2025)

[2] AutomationML consortium (2020). „<AutomationML/> The Glue for Seamless Automation Engineering, Whitepaper AutomationML Part 4: AutomationML Logic”. In: *WP Logic,V 1.6.0*. URL: https://www.automationml.org/wp-content/uploads/2024/01/AML_Whitepaper_Part4_Logic_V1.6.0.zip (besucht am 04.07.2025)

[3] IDTA (2022). „Provision of Simulation Models, Submodel Template of the Asset Adminsitration Shell”. In: *Specification IDTA 02005-1-0*. URL: https://industrialdigitaltwin.org/wp-content/uploads/2023/01/IDTA-02005-1-0_Submodel_ProvisionOfSimulationModels.pdf (besucht am 04.07.2025)

[4] IDTA (2025). „Interface Connectors, Submodel Template of the Asset Adminsitration Shell”. In: *Specification IDTA 02062-1-0*

[5] Dörner L., Hundt L., Mersch T., Schleipen M., Freund M., Paul M., Lüder A., Kose K. (2025). „Synergien durch Standards im Anlagenentstehungsprozess – DIAMOND – AML und AAS im Engineering“. In: *Fachzeitschrift atp magazin* (Stand 15.07.2025: eingereicht und akzeptiert)

[6] Modelica Association (2025). “fmi Functional Mock-up Interface”. URL: https://fmi-standard.org/ (besucht am 01.07.2025)

[7] PKWARE Inc. (2022). „.ZIP File Format Specification - Version: 6.3.10”. URL: https://pkware.cachefly.net/webdocs/casestudies/APPNOTE.TXT (besucht am 30.06.2025)

[8] Mersch T., Schleipen M., Kuhlenkötter B., Hypki A., Burlein J., Schlögl W., Ach M., Göbeler C., Freund M., Paul M., Dörner L., Lüder A., Hünecke P. (2025). „Datendurchgängigkeit im Anlagenentstehungsprozess – DIAMOND - Modell und Realisierung“. In: *Fachzeitschrift atp magazin* 67.04. DOI:10.17560/atp.v67i4.2775

[9] ECLASS (2025). „Technical specification: Structure and structural elements: Unit”. URL: https://eclass.eu/support/technical-specification/structure-and-elements/unit (besucht am 28.05.2025).

[10] Neidig J., Orzelski A. & Pollmeier S. (2022). „Asset Administration Shell Reading Guide”. URL: https://www.plattform-i40.de/IP/Redaktion/EN/Downloads/Publikation/AAS-ReadingGuide202201.pdf?__blob=publicationFile&v=1 (besucht am 27.05.2025)

[11] Modelica Association (2024). “Functional Mock-up Interface Specification v3.0.2”. URL: https://fmi-standard.org/docs/3.0.2/ (besucht am 20.05.2025)

[12] Modelica Association (2025). „Tools that support SSP”. URL: https://ssp-standard.org/tools/ (besucht am 20.05.2025)

[13] Preston-Werner, T. (2025) „Semantic Versioning 2.0.0“. URL: https://semver.org/#semantic-versioning-200 (besucht am 20.05.2025)

[14] ITEA (2011). “MODELISAR: From System Modeling to S/W running on the Vehicle”. URL: https://itea4.org/project/modelisar.html (besucht am 20.05.2025)

[15] ITEA 2 Office (2012). „Project Results: Speeding automotive ECU development”. URL: https://itea4.org/project/result/download/5533/MODELISAR%20Project%20results%20leaflet.pdf (besucht am 20.05.2025)

[16] Wolf, C., Schleipen, M., Paul, M., Krsticevic, T., Neveu, F., Wincheringer, C., Wünsche, H., Ziegenhagel, V., Zellmaier, T., Moghaddam Nejad, A., Hammerl, F. & Frey, G. (2025). „FMI in der VIBN: Ein Leitfaden zur Umsetzung“ In: *VDI-Kongress AUTOMATION 2025* (Stand 15.07.2025: eingereicht und akzeptiert)

[17] Puntel Schmidt, P. & Fay, A. (2015). „Konsistente Simulationsmodelle für die virtuelle Inbetriebnahme fertigungstechnischer Anlagen mit Hilfe regelbasierter Modellverbinder“. In: *16. ASIM Fachtagung Simulation in Produktion und Logistik*

[18] Sauer, O., Schleipen, M. & Ammermann, C. (2010). „Digitaler Fabrikbetrieb - Virtual Manufacturing“. In: *Fachtagung Simulation in Produktion und Logistik 2010*

[19] Bergert, M. & Diedrich, C. (2008). „Durchgängige Verhaltensmodellierung von Betriebsmitteln zur Erzeugung digitaler Simulationsmodelle von Fertigungssystemen“. In: *Fachzeitschrift atp magazin* 50.07. DOI: 10.17560/atp.v50i07.2058

[20] Kiefer, J., Wack, K.-J. & Manns, M. (2010). „Cross-functional digital production validation framework for automotive industry”

[21] Kiefer, J., Ollinger, L. & Bergert, M. (2009). „Virtuelle Inbetriebnahme - Standardisierte Verhaltensmodellierung mechatronischer Betriebsmittel im automobilen Karosserierohbau“. In *Fachzeitschrift atp magazin* 51.07. DOI: 10.17560/atp.v51i07.92

[22] Hämmerle, H., Strahilov, A. & Drath, R. (2016). „AutomationML im Praxiseinsatz: Erfahrungen bei der virtuellen Inbetriebnahme“. In: *Fachzeitschrift atp magazin* 58.05. DOI: 10.17560/atp.v58i05.567

[23] Kiefer, J., Prieurs, M., Schmidgall, G. & Bär, T. (2009). „Digital planning and validation of highly flexible manufacturing systems in the automotive body shop”. In: *Proceedings of CIRP MS*

[24] Mewes, J. & Wegener, F. (2009). „Virtuelle Inbetriebnahme von Förderanlagen mit Feldbusemulation und Materialflusssimulation“. In: *GMA-kongress „automation“*

[25] Kegel, G., Rauscher, B., Otto, M.-A., Gehlen, S., Nagel, J & Weber, H (2024). „Die Bedeutung der Sensorik für die vierte industrielle Revolution“. In: *Handbuch Industrie 4.0: Band 2: Automatisierung*, DOI: 10.1007/978-3-662-58528-3_142

[26] Wolf C., Schleipen M.; Frey G. (2023). „Secure Exchange of Black-Box Simulation Models using Functional Mockup Interface in the Industrial Context". In: *Proceedings of the Modelica Conference 2023*, DOI: 10.3384/ecp204487. URL: https://ecp.ep.liu.se/index.php/modelica/article/view/959/900 (besucht am 21.05.2025)

[27] Wolf C. (2025). „christian-wolf-eks / ls-sign". URL: https://github.com/christian-wolf-eks/ls-sign (besucht am 21.05.2025)

[28] Unified Nations Economic commission for Europe (2010). "Recommendation No. 20: Codes for units of measure used in international trade" URL: http://www.unece.org/fileadmin/DAM/cefact/recommendations/rec20/rec20_Rev7e_2010.zip (besucht am 27.05.2025)

[29] Creative Commons (2025). „CC BY-SA 4.0 Namensnennung-Share Alike 4.0 International". URL: https://creativecommons.org/licenses/by-sa/4.0/deed.de (besucht am 22.05.2025)

[30] AutomationML (2018). „Best Practice Recommendation: Units in AutomationML". URL: https://www.automationml.org/news/bpr-units-in-automationml-available/ (besucht am 27.05.2025).

[31] Verband Deutscher Maschinen- und Anlagenbau e. V. (VDMA) (2020). „Leitfaden Virtuelle Inbetriebnahme - Handlungsempfehlungen zum wirtschaftlichen Einstieg". In: VDMA Verlag GmbH. URL: https://vdma.eu/c/document_library/get_file?uuid=66118d8d-68fc-55a7-a25b-a90ffe89f1f2&groupId=34570 (besucht am 30.07.2025)

[32] VDI-Richtlinie 3693 (2025). Blatt 1: Virtuelle Inbetriebnahme - Modellarten und Begriffe. Berlin: Beuth Verlag. URL: https://www.vdi.de/richtlinien/details/vdivde-3693-blatt-1-virtuelle-inbetriebnahme-modellarten-und-begriffe (besucht am 30.07.2025)

[33] VDI-Richtlinie 3693 (2018). Blatt 2: Virtuelle Inbetriebnahme - Einführung der virtuellen Inbetriebnahme in Unternehmen. Berlin: Beuth Verlag. URL: https://www.vdi.de/richtlinien/details/vdivde-3693-blatt-2-virtuelle-inbetriebnahme-einfuehrung-der-virtuellen-inbetriebnahme-in-unternehmen-1 (besucht am 30.07.2025)

[34] Modelon (2025). „FMI Standard: Understand the two types of Functional Mock-up Units – CS v. ME". URL: https://modelon.com/blog/fmi-functional-mock-up-unit-types/ (besucht am 31.07.2025)

[35] Modelica Association (2025). „Tools that support FMI". URL: https://fmi-standard.org/tools/ (besucht am 04.08.2025)

[36] Industrial Digital Twin Association e.V. (2025). „IDTA – Der Standard für den Digitalen Zwilling" URL: https://industrialdigitaltwin.org/ (besucht am 04.08.2025)

[37] Federal Ministry for Economic Affairs and Energy (BMWE) (2020). „Specification: Submodel Templates of the Asset Administration Shell: Generic Frame for Technical Data for Industrial Equipment in Manufacturing (Version 1.1)) URL: https://www.zvei.org/fileadmin/user_upload/Presse_und_Medien/Publikationen/2020/Dezember/Submodel_Templates_of_the_Asset_Administration_Shell/201117_I40_ZVEI_SG2_Submodel_Spec_ZVEI_Technical_Data_Version_1_1.pdf (besucht am 04.08.2025)

[38] DIAMOND Projekt 2025. *„Leitfaden zur Anwendung des FMI-Standards im Kontext der virtuellen Inbetriebnahme. DIAMOND-Forschungsprojekt 2025“* URL: https://diamond-project.de/downloads/files/DIAMOND_Leitfaden_zur_Anwendung_des_FMI_Standards_im_Kontext_der_virtuellen_Inbetriebnahme.pdf (besucht am 19.11.2025)

Autoren

Franz Hammerl,	EDAG Production Solutions GmbH & Co. KG
Ali Moghaddam Nejad,	EDAG Production Solutions GmbH & Co. KG

Mitwirkende

Thomas Zellmeier,	BMW Group
Franz Hammerl,	EDAG Production Solutions GmbH & Co. KG
Ali Moghaddam Nejad,	EDAG Production Solutions GmbH & Co. KG
Miriam Schleipen,	EKS InTec GmbH
Christian Wolf,	EKS InTec GmbH
Toni Krsticevic,	Festo SE & Co. KG
Walter Kuhlbusch,	Festo SE & Co. KG
Manuel Paul,	Festo SE & Co. KG
Viktor Ziegenhagel,	Mercedes Benz AG
Lorenz Hundt,	Murrelektronik GmbH
Francois Neveu,	Siemens AG
Christoph Wincheringer,	Siemens AG
Holger Wünsche,	WINMOD GmbH

Der Ursprung dieses Leitfadens ist im Kontext des Forschungsprojekts DIAMOND entstanden. Die EDAG Production Solutions GmbH & Co. KG führte im Auftrag der BMW AG neutrale Umfragen bei DIAMOND-Konsortialpartnern durch, konsolidierte die Ergebnisse fachlich und erstellte den FMI-Leitfaden auch mit eigenen Inhalten.