

Schlussbericht – Teil II: Eingehende Darstellung

Wir haben den Schlussbericht Teil II in drei Kapitel aufgeteilt. Das erste Kapitel enthält die ausführliche Darstellung der durchgeführten Arbeiten. Die Darstellung erfolgt gegliedert nach den Hauptarbeitspaketen und entspricht damit nicht zwangsläufig der chronologischen Abfolge im Projektverlauf. Dieses Kapitel umfasst die Notwendigkeit der Tätigkeiten und auch die dabei erzielten Ergebnisse inkl. Veröffentlichungen. Zur Verwendung der Zuwendung kann allgemein gesagt werden, dass diese sich aus der Darstellung der Arbeiten ergibt, da hauptsächlich Personalkosten geltend gemacht wurden. Die Verwendung der angeführten Lizenzen wird ebenfalls kurz im Kapitel gesammelt erläutert. Der bekannt gewordene Fortschritt wird gesammelt in Kapitel 2 betrachtet, während in Kapitel 3 die Verwertungsperspektiven betrachtet werden.

1 Ausführliche Darstellung und Notwendigkeit der durchgeführten Arbeiten

1.1 B1, B2: Bibliotheksaufbau von Standardisierten Verhaltensmodellen

Die Zielsetzung bestand darin, digitale Verhaltensmodelle zu entwickeln, die als zentrale Grundlage für die effiziente und wiederholbare Projektgenerierung dienen sollten. Dabei lag ein besonderer Fokus auf der Etablierung einer Standardisierung der zugrunde liegenden Informationen, um eine einheitliche Referenz für verschiedene Projekte und Anwendungen zu schaffen. Diese Standardisierung sollte nicht nur theoretisch fundiert, sondern auch praktisch anwendbar sein, was eine intensive Validierung und Nachweisführung im realen Projektkontext erforderte. Um dieses Ziel zu erreichen, waren umfangreiche konzeptionelle Arbeiten notwendig, einschließlich der Definition von Modellierungsrichtlinien, der Auswahl und Anpassung geeigneter Technologien sowie der Entwicklung von Verfahren zur Integration der Verhaltensmodelle in bestehende Systemlandschaften. Ergänzend dazu musste ein nachhaltiger Ansatz für die Dokumentation und Weiterentwicklung geschaffen werden, um die langfristige Nutzbarkeit und Skalierbarkeit der entwickelten Modelle sicherzustellen.

Im Rahmen des Arbeitspakets hat die HEITEC eine Referenzarchitektur für Verhaltensmodelle konzipiert. Dazu wurden etablierte Industriestandards analysiert und mit den spezifischen Anforderungen abgeglichen. Unter anderem fanden dabei SysML, AutomationML und FMI, Berücksichtigung. Besondere Bedeutung hatte die Entwicklung einer Architektur, die mit der Verbundarchitektur im SDM4FZI kompatibel ist. Diese Anforderung konnte durch den FMI-Standard erfüllt werden, da dieser sich nahtlos in eine Architektur der Verwaltungsschale integrieren lässt.

In den ersten Umsetzungen wurden digitale Komponentenmodelle, beispielsweise von Druckluftzylindern, entwickelt und getestet. Ein Schwerpunkt lag auf der Überprüfung ihrer Kompatibilität mit verschiedenen Simulationsumgebungen. Dabei wurden sowohl etablierte Software-Tools wie Virtuos3, Matlab, Modelica, TwinCAT3 und Unity als auch quelloffene Lösungen wie FMPy, PyFMI und FMI4CPP einbezogen. Die Testergebnisse belegen, dass die erzeugten FMUs in sämtlichen getesteten Umgebungen fehlerfrei funktionierten und eine überzeugende Performance zeigten.

Die Architektur, welche diese Implementierungen unterstützt, ist in der folgenden Abbildung dargestellt.

Eine Gap-Analyse ergab, dass der FMI-Standard kein integriertes User-Interface vorsieht, über das Verhaltensmodelle direkt steuerbar sind. Aus den Erfahrungen von HEITEC wurde

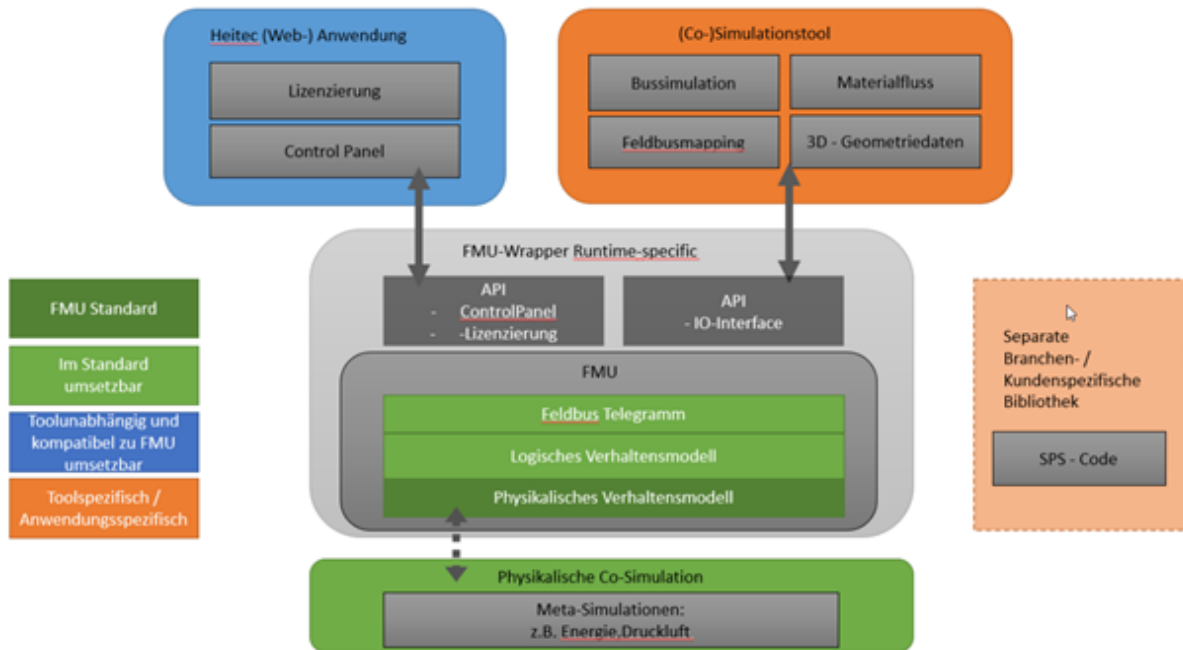


Abbildung 1: Architektur zur Einbindung von FMU-Verhaltenskomponenten.

jedoch deutlich, dass eine solche Funktionalität für den praktischen Einsatz bei virtuellen Inbetriebnahmen unerlässlich ist. Um diese Lücke zu schließen, wurde der Standard für die Architektur um einen MQTT-Connector erweitert. Dieser ermöglicht die Steuerung der Verhaltensmodelle unabhängig von der jeweiligen Laufzeitumgebung. Der MQTT-Connector wurde erfolgreich in das SDK integriert und hinsichtlich seiner Performance umfassend getestet und validiert.

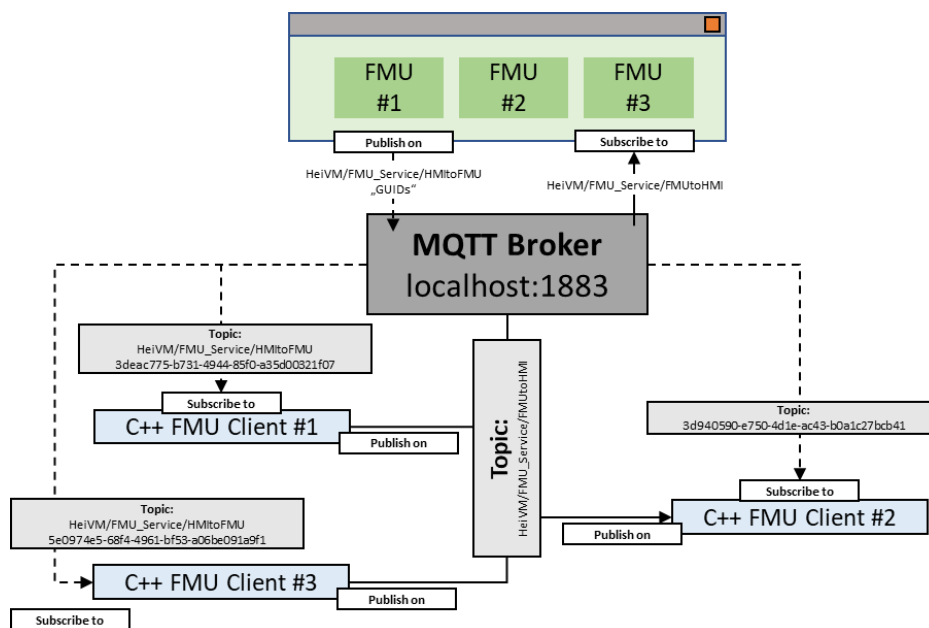
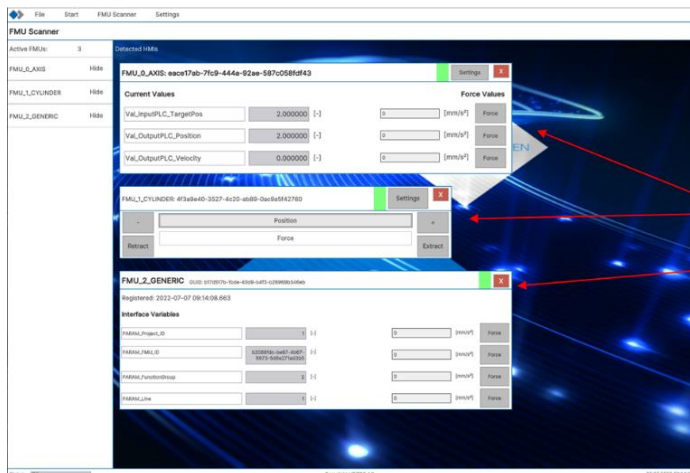


Abbildung 2: MQTT-Kommunikationsstruktur für Forcebefehle.

Es wurde eine Anwendung entwickelt, die eine Übersicht aller lokal und über das Netzwerk erreichbaren FMUs bietet. Mithilfe eines speziell entwickelten, leichtgewichtigen Protokolls können sich FMUs eigenständig in der HMI registrieren, gesteuert und parametrisiert werden

sowie ihre Abmeldung vornehmen. Für Standardkomponenten steht eine umfassende HMI-Bibliothek zur Verfügung, die automatisch ein passendes Benutzerinterface erstellt und mit der sich registrierenden FMU verknüpft. Sollte der FMU-Typ in der Bibliothek nicht vorhanden sein, erzeugt das System ein generisches HMI, das grundlegende Ein- und Ausgänge sowie Parameter der FMU bereitstellt, um eine sofortige Nutzung zu ermöglichen.



FMU Scanner

- Alle lokalen und über das Netzwerk erreichbaren FMUs werden angezeigt
- Wertvorgaben (Forcen) der Verhaltensmodelle
- Parametrierung

HMI Generierung

- Bekannte FMU Typen werden mit vorgefertigten HMIs dargestellt
- Unbekannte FMUs werden durch eine generische Liste der Ein-, Ausgänge und Parameter repräsentiert

Abbildung 3: FMU-Force HMIs in toolunabhängiger Umgebung

Zur Erweiterung der Funktionalität wurde ein umfassender C++ Wrapper einschließlich eines SDK entwickelt, das die Anbindung und Kommunikation der FMU über das MQTT-Protokoll ermöglicht und somit eine flexible Integration in verschiedene Automatisierungsumgebungen sicherstellt; ergänzend wurde ein FMU-Client in Form eines PLC-Open XML ausgearbeitet, um die Kompatibilität mit TwinCAT zu gewährleisten und eine nahtlose Einbindung in SPS-Systeme zu ermöglichen.

Da der Aufwand zur manuellen Erstellung von FMUs aufwändig und fehleranfällig ist, wurde ein Modul entwickelt, mit dem die Erstellung teilautomatisiert werden kann. Der FMU-Generator ist ein leistungsfähiges Modul zur vereinfachten Erstellung von Verhaltensmodellen im Functional-Mockup-Unit (FMU) Format.

Die Eingabe der Modellvariablen erfolgt über eine intuitive grafische Benutzeroberfläche. Hier können sowohl allgemeine Modellinformationen wie Model Name, Kurzbeschreibung, Model ID und Hersteller als auch variablenbezogene Details spezifiziert werden. Zu den variablenbezogenen Informationen zählen:

- Signalrichtung (Inputs: PLCtoFMU, SIMtoFMU; Outputs: FMUtoPLC, FMUtoSIM; Parameter)
- Datatype (bool, int, double, string)
- Name und Beschreibung
- Optional: Startwert und Variabilität

Für die Modellkonfiguration werden Ein- und Ausgänge sowie Parameter definiert. Da der FMU2 Standard keine Arrays unterstützt, wurde auch hier im Rahmen des Generators eine Automatisierung implementiert. Wird bei Array Size ein Wert größer als Eins eingegeben, erzeugt der Generator automatisch eine Reihe von Variablen mit fortlaufenden Namenssuffixen (_0, _1, _2, ...). Inputs und Parameter erfordern zusätzliche Angaben wie Startwerte und Variabilität. Bereits konfigurierte Variablen können bei Bedarf gelöscht

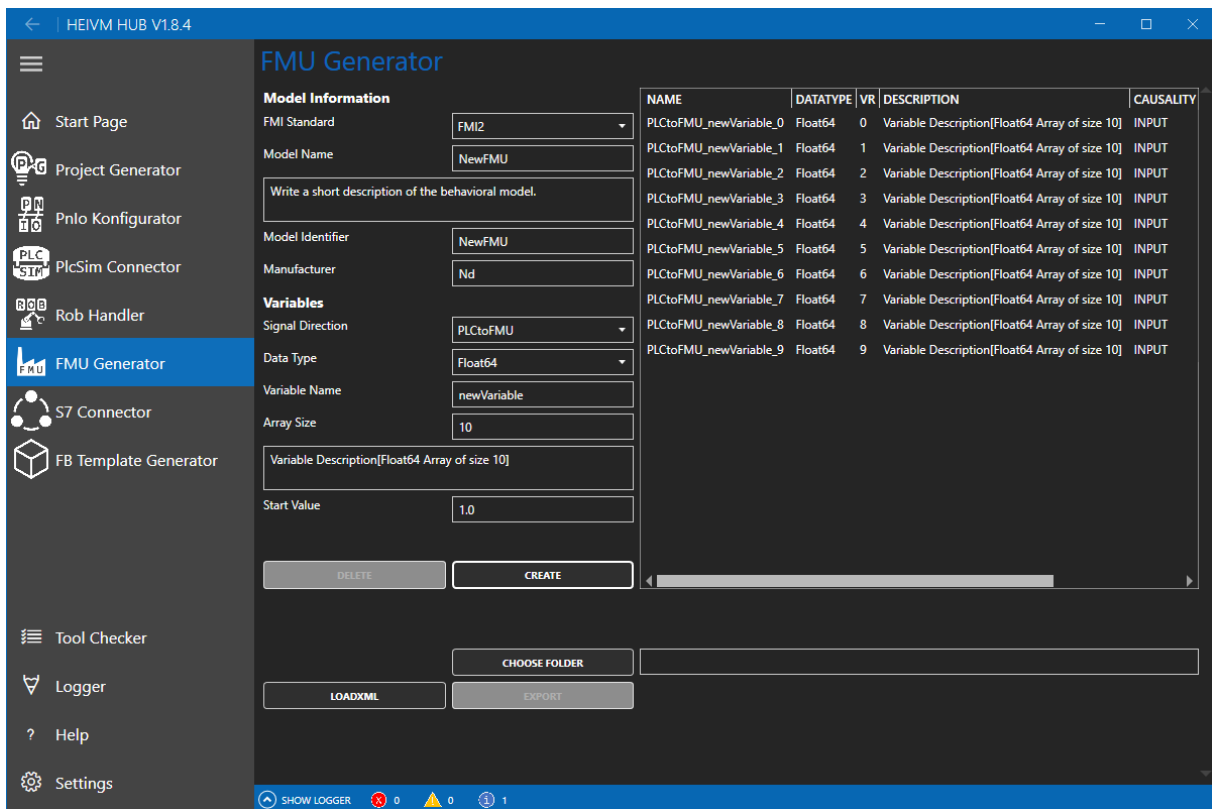


Abbildung 4: Userinterface des FMU-Generators.

werden. Bei Array-Variablen werden dabei alle zugehörigen Einträge entfernt. Nach dem Export kann das Modellverhalten in der generierten Visual Studio Solution programmiert werden. Die Kernlogik wird in der Methode DoStep der Datei `HeiVM_FMU_{Hersteller}_{ModelName}.cpp` implementiert, während die dazugehörigen Laufzeitvariablen in der Headerdatei `HeiVM_FMU_{Hersteller}_{ModelName}.h` definiert werden.

Nach Abschluss der Implementierung wird die FMU durch Ausführen der Datei `CreateDistribution.bat` erstellt. Zusätzlich bietet das Userinterface die Möglichkeit, vorhandene `modelDescription.xml`-Dateien von FMUs einzulesen. Dabei werden die enthaltenen Modellvariablen automatisch in die Konfigurationsliste übertragen und können nach Bedarf angepasst werden.

Telegramme spielen eine zentrale Rolle in der Kommunikation zwischen Steuerungssystemen und Simulationen, da sie als standardisierte Datenpakete Informationen effizient übertragen. Nach dem FMU-Standard können Telegramme jedoch nicht direkt verwendet werden. Um die korrekte Verarbeitung und Interpretation dieser Daten zu gewährleisten, ist es notwendig, spezifische Bereiche des Byte-Arrays flexibel auf verschiedene Variablen und deren Datentypen zu mappen. Hier setzt der Structifyer an, ein im Rahmen des Forschungsprojekts entwickeltes Tool, das diese Anforderungen erfüllt und somit eine präzise und anwendungsfreundliche Nutzung von Telegrammen ermöglicht.

Der Structifyer ermöglicht es, einzelne Abschnitte von Byte-Arrays (Telegrammen) flexibel mit Variablen bestimmter Datentypen zu verknüpfen. So kann beispielsweise das erste Bit (Bit 0.0) einer Nachricht einer booleschen Variablen mit dem Bezeichner „Start“ zugeordnet werden. Durch eine intuitive grafische Eingabe lassen sich die Strukturen der Telegramme präzise nachbilden, bearbeiten und anschließend exportieren. Dies vereinfacht die Verwaltung komplexer Telegrammstrukturen erheblich und trägt zur verbesserten

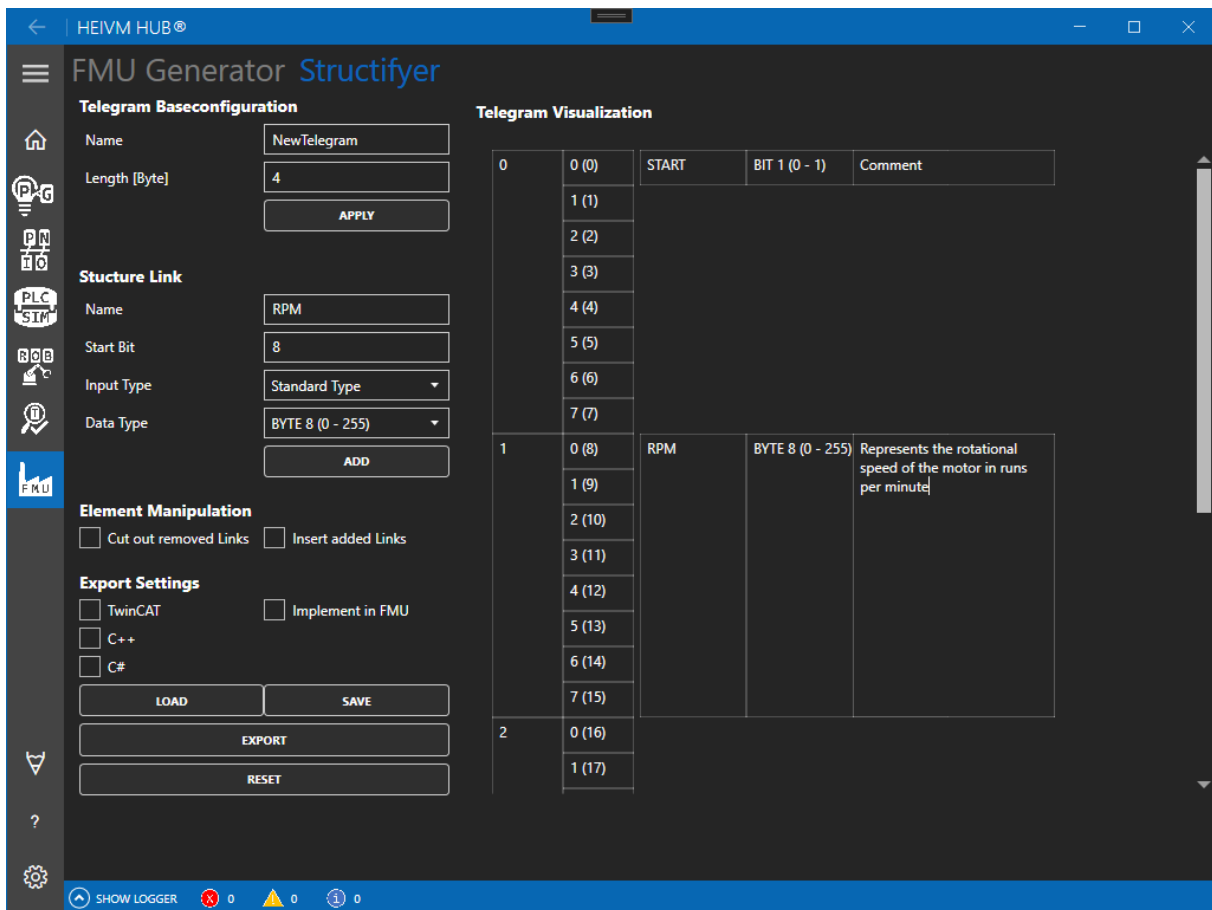


Abbildung 5: Structifyer für die einfache Berücksichtigung von Telegrammen in FMUs.

Interoperabilität zwischen Systemen bei.

In der „Telegramm Baseconfiguration“ wird zunächst der Name und die Länge des Telegramms in Byte eingegeben. Die Länge kann im weiteren Verlauf der Strukturierung nur vergrößert oder gleichbleiben aber nicht mehr verkleinert werden! Der nächste Schritt besteht darin die einzelnen logischen Bestandteile des Telegramms mit Links abzubilden. Jeder Link muss einen eindeutigen Namen sowie ein Startbit besitzen. Mit dem Dropdown Input Type kann zwischen „Standard Type“ und „Special Type“ gewählt werden. Ist „Standard Type“ ausgewählt muss für die weitere Konfiguration lediglich der konkrete Typ bei „Data Type“ gesetzt und der Link mit Add hinzugefügt werden. Die Bitlänge der Standardtypen ist in die Auswahl mit integriert. Mit „Special Type“ können Links für nicht vorhandene Datentypen erstellt werden. In diesem Modus ist die Angabe des Typnamens sowie einer Bitlänge erforderlich. Ist die Option „Insert added Links“ bei „Element Manipulation“ aktiviert, werden bei einer Überschneidung alle folgenden Variablen um den Betrag der Überschneidung nach unten verschoben! Alle frei gelassenen Stellen des Telegramms werden vor dem Export noch durch namenlose Lückenfüller reserviert und werden nicht beschrieben. Nachdem der Link hinzugefügt wurde, kann in der Visualisierung ein Kommentar hinzugefügt werden. Dieser Kommentar wird in den Exportierten Dateien zu der Variable hinzugefügt.

Für den Export der erstellten Telegrammstrukturen bietet der Structifyer zwei Varianten:

1. Export in ausgewählte Formate:

Nach einem Klick auf „Export“ kann ein Zielordner ausgewählt werden, in dem abhängig von der Auswahl in der linken Spalte (z. B. TwinCAT, C++, C#) die entsprechenden Dateien automatisch generiert werden. Zusätzlich wird das Telegramm im Hub Folder gespeichert. Wichtig ist hierbei zu beachten, dass bei einem bestehenden Telegramm mit identischem Namen, wie in der Baseconfiguration angegeben, die bestehende Datei überschrieben wird.

2. Integration in eine FMU:

Alternativ kann der Haken bei „Implement FMU“ gesetzt werden. Dadurch wird beim Export automatisch die Struktur in die Solution der FMU integriert. Dies erfolgt durch die Generierung der Strukturdefinition in der Datei „<FMU_NAME>_TypeDefs.h“, das Hinzufügen einer zugehörigen Variablen in den Runtime-Variablen der Datei „<FMU_NAME>.h“ sowie die notwendigen Anpassungen, um die Struktur innerhalb der FMU nutzbar zu machen.

Diese flexiblen Exportoptionen erleichtern die Verwendung der Telegrammstrukturen sowohl in klassischen Entwicklungsumgebungen als auch in FMU-basierten Modellen.

Durch die Kombination aus automatisierter HMI-Generierung, einem leistungsfähigen SDK und dem modularen FMU-Generator wurde der Prozess zur Erstellung von Functional-Mockup-Units (FMUs) signifikant optimiert. Die automatisierte Generierung von HMIs reduziert manuelle Konfigurationsfehler und ermöglicht eine schnelle, standardisierte Visualisierung der Verhaltensmodelle. Das SDK stellt dabei eine zentrale Schnittstelle bereit, die die Kommunikation und Integration der FMUs in verschiedene Umgebungen vereinfacht. Der Generator unterstützt durch eine intuitive grafische Benutzeroberfläche die effiziente Eingabe aller relevanten Modellparameter und minimiert durch automatisierte Prozesse und Validierungen mögliche Fehlerquellen. Diese integrierten Werkzeuge gewährleisten nicht nur eine erheblich schnellere Entwicklung, sondern steigern auch die Zuverlässigkeit und Qualität der FMUs, was den Aufwand für die virtuelle Inbetriebnahme deutlich reduziert.

1.2 S1: Vertrieb am digitalen Zwilling

Im Zuge des Arbeitspaketes wurde das Potential von Digitalen Zwillingen für den Vertrieb untersucht. Dies soll es ermöglichen den individuellen Anforderungen des Kunden besser gerecht zu werden und bereits in einer frühen Phase das Verständnis über die angebotene Anlage zusammen mit dem Kunden zu schärfen. Dies zielt auf eine Reduzierung von Änderungen in späteren Phasen, welche kostenintensiv und fehleranfällig sind. Als Grundproblem wurden sequenzielle Prozesse und eine fehlende Datenbasis identifiziert. Dies führt zu vielen manuellen Prozessen an den Schnittstellen und birgt somit einen hohen Aufwand und ein hohes Fehlerpotential. (Abbildung 6)

Auf Basis eines visualisierten Modells können Kunde und Vertrieb ein höheres Verständnis erreichen. Die Durchgängigkeit auf einer gemeinsamen Datenbasis erlaubt es, Schnittstellen zu eliminieren und somit Engineering-Aufwände zu reduzieren. (Abbildung 7)

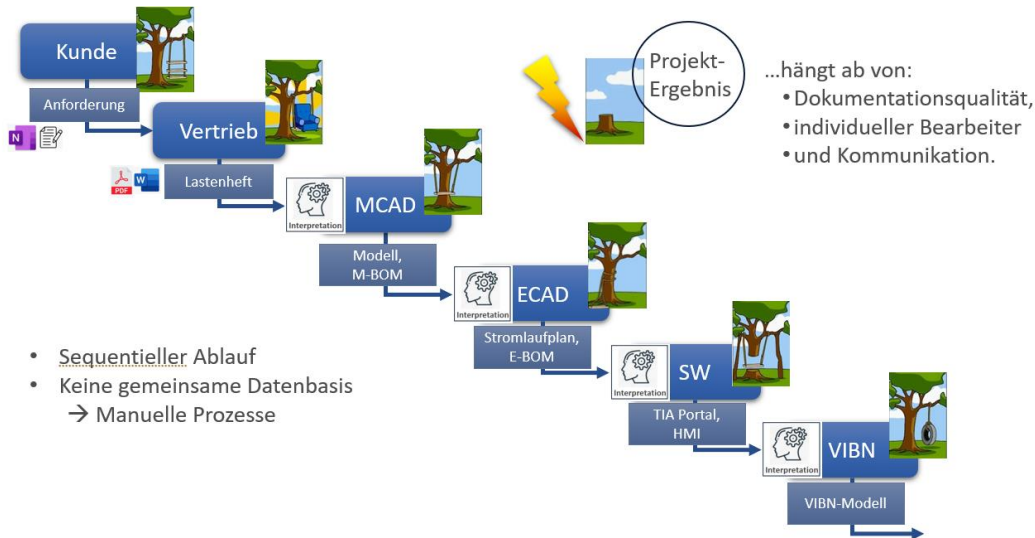


Abbildung 7: Der konventionelle Prozess

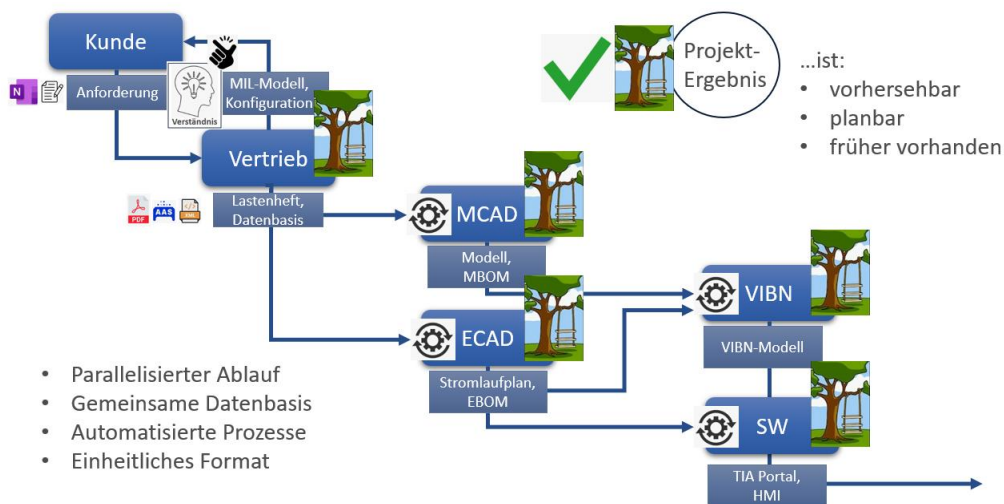


Abbildung 6: Vorteile des durchgängigen Engineering Prozesses

Prototypisch wurde als Vertriebsgenerator ein einfaches Modell einer Vial-Ablagestation umgesetzt. Hierzu wurde in Model-in-the-Loop Model besagter Anlage in ISG Virtuos umgesetzt wie in Abbildung 8 zu sehen ist. In diesem Model ist es möglich verschiedene Optionen für die Anlage zu treffen. Beispielsweise kann angegeben werden, ob eine Zuführstation an die Ablagestation angeschlossen ist, ob eine Beleuchtungseinheit mit konfiguriert werden soll oder ob eine Handentnahmestation Teil der Anlage ist.

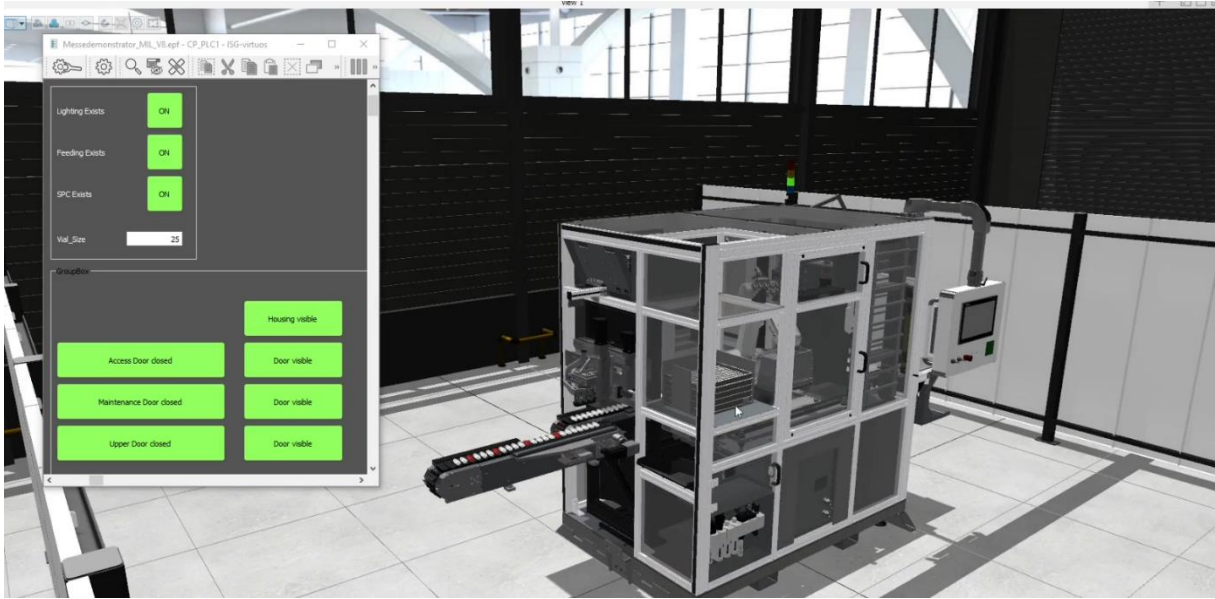


Abbildung 8: MIL-Model mit Vertriebs-GUI

Die Konfiguration kann dann als XML-Datei gespeichert werden. Die XML-Datei dient als Basis für Eplan, welches damit die Hardwarekonfiguration für das SPS-Projekt im TIA-Portal bildet und automatisiert EPLAN Dokumente der gespeicherten Konfiguration bereitstellt. Des Weiteren wird die XML-Datei als Basis für das Maschinenverhalten in der HeiVM verwendet und kann dort als Grundstruktur des Anlagenmodells für die ViBN geladen werden. (Abbildung 9)

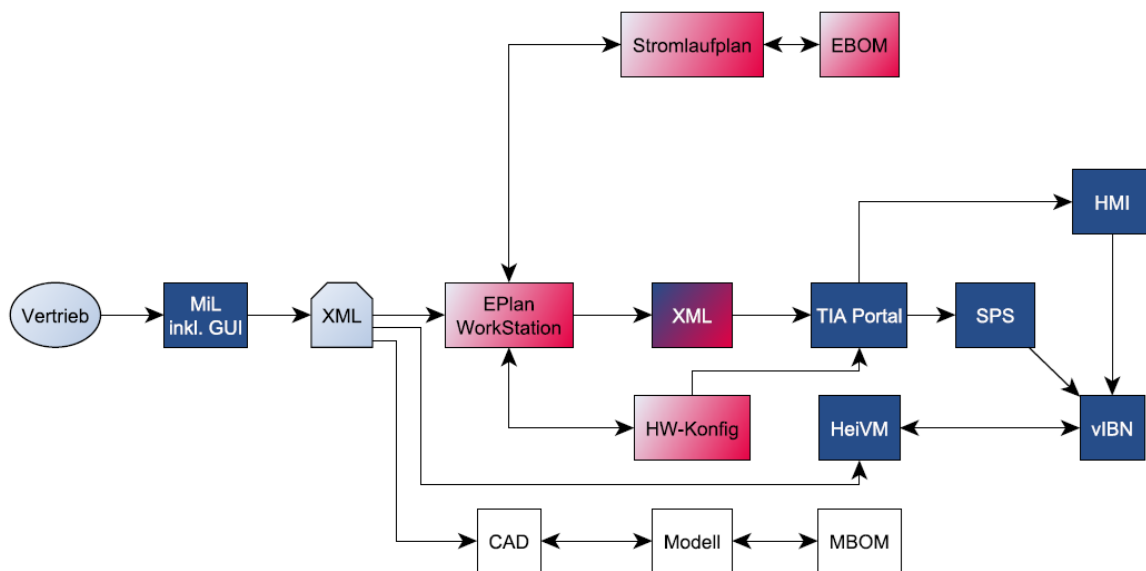


Abbildung 9: Architektur des Produktkonfiguratorprototypens

Nach dem proof-of-concept wurde mit Unity ein Vertriebsgenerator geschaffen, welcher es ermöglicht verschiedene Konfigurationen einer Heller Multiflex Zelle zu konfigurieren. Die Funktionen des Vertriebskonfigurator sind im Folgenden beschrieben. Über ein UI können Standardvarianten der zu konfigurierenden Anlage geladen werden, welche als Basis für die weitere Konfiguration dienen. Hierbei gibt es drei Optionen, eine für eine preiseffiziente Konfiguration für einfache Bauteile, eine fortgeschrittene Option mit mehr Ablagefläche und eine Performance Variante, welche eine zweite Bearbeitungsstation an die Roboterzelle anbindet.

Anschließend können, wie in Abbildung 10 gezeigt, Detailkonfigurationen vorgenommen werden, welche die Position verschiedener Bauteile betreffen oder die Maschinentypen im Einzelnen definieren. Auch weitere Optionen wie eine Inbetriebnahme oder Bedienschulungen können gewählt werden, sodass aus allen gewählten Optionen automatisch ein Angebot mit Echtzeitpreisberechnung erstellt werden kann. Hierbei war es von besonderer Bedeutung durch ein Variantenmanagement die Optionen so zu beschränken, dass es jederzeit nur möglich ist eine bestellbare Anlage zu konfigurieren. Optionen, die nicht miteinander kombinierbar sind, sind dementsprechend nicht auswählbar, wenn ein Ausschlusskriterium an einer anderen Stelle bereits gewählt wurde.

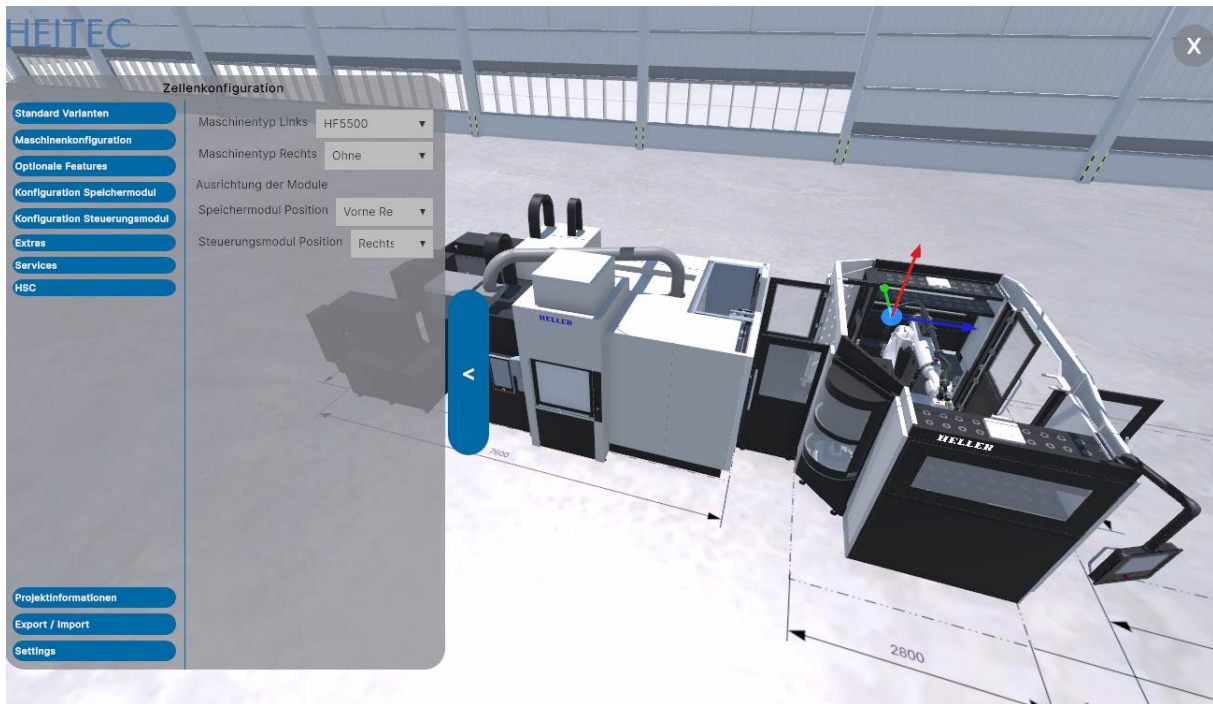


Abbildung 10: Produktkonfigurator in Unity

Für die Erstellten Funktionen wird in vielen Fällen auch eine Model-in-the-Loop Funktionalität geboten, so lassen sich vorkonfigurierte Bewegungen auf Knopfdruck starten, um noch ein besseres Verständnis im Vertriebsgespräch über die konfigurierte Anlage zu bekommen. Hier können zum Beispiel Lagerplatten ein und ausgefahren oder Türen geöffnet und geschlossen werden. Auch Roboter können manipuliert werden, um zum Beispiel Fragen der Erreichbarkeit von Ablagen zu klären. Beim Klicken auf den Roboter erscheint ein Kontextmenü, mit welchem die Achsen des Roboters eingestellt werden können. (Abbildung 11)

Der Produktkonfigurator bietet außerdem die Möglichkeit Notizen in der 3D-Umgebung anzubringen um spezielle Änderungswünsche direkt in der Anlage mit zu Verwalten. Die konfigurierte Anlage kann des Weiteren in einer VR Umgebung auf einer Apple Vision Pro geladen und betrachtet werden um ein noch besseres Gefühl für die Dimension der Anlage zu bekommen. Die erstellten Konfigurationen können gespeichert und in Nachgesprächen mit dem Kunden erneut geladen werden. Dies ermöglicht es noch leichter weitere

Anforderungen an die Anlage nachträglich einzuarbeiten.

Der Produktkonfigurator ist sowohl in Windows lauffähig, wobei eine Steuerung über Maus

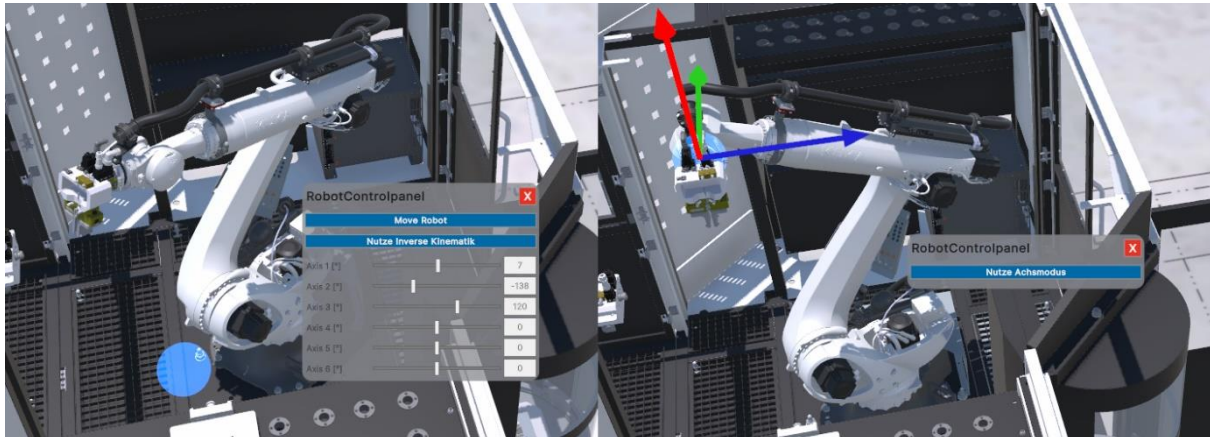


Abbildung 11: Manipulationsmöglichkeiten des Roboters

und Tastatur vorgenommen wird, als auch als Version für Android Tablets. Die Bedienung findet hierbei über verschiedene 1-, 2- und 3- Finger-Gesten statt und ermöglicht eine einfache Bedienung, die sich stark an bekannten Systemen wie Google Earth orientiert und somit auch für den potenziellen Kunden leicht zugänglich ist.

Eine Umsetzung als Webapp ist mit Unity nicht vollumfänglich möglich, da für die Verwendung im Browser WebGL notwendig ist, welches von mobilen Endgeräten in den meisten Fällen nicht unterstützt wird.

Im Zuge der Prototypenentwicklung stellte sich heraus, dass vor allem das Variantenmanagement als ausprogrammierte Variante eine Herausforderung darstellt, deren Komplexität exponentiell mit der Anzahl der Optionen zunimmt. Ein zentrales Merkmal davon ist die Fähigkeit, nur erlaubte Konfigurationen zu ermöglichen, indem Regeln und Abhängigkeiten zwischen den verschiedenen Produktkomponenten definiert werden. Dies geschieht durch ein Regel- und Constraint-Management, das sicherstellt, dass keine technisch oder wirtschaftlich unzulässigen Kombinationen angeboten werden. Variantenmanagement in einem CPQ-Tool bezieht sich auf die Fähigkeit des Systems, verschiedene Produktvarianten effizient zu verwalten, zu konfigurieren und anzubieten. Hierzu wurde ebenfalls prototypisch ein System entwickelt, welches das Variantenmanagement so organisiert, dass verbotene Kombinationen über eine Matrix in einer Exceltabelle eingelesen werden und live verändert werden können. Randbedingung hierbei ist, dass es nur Constraints erster Ordnung sind, also keine tiefergehenden Einschränkungen mit mehreren verschachtelten Bedingungen vorliegen. Diese verschachtelten Constraints können jedoch vorweg aufgelöst werden, indem man den Nutzer des Produktkonfigurator anschließend geführt durch mehrere Seiten in der Konfiguration leitet.

Der Prototyp das Interface und die Optionen automatisch aus der Excel Tabelle und lässt markierte verbotene Konfigurationen nicht zu. (Abbildung 12)

Des Weiteren wurde eine Architektur entwickelt, welche die Asset Administration Shell als Zentrale Datenbasis für den Produktkonfigurator nutzt. Die Architektur wird in Abbildung 13 aufgezeigt. Hierbei dient der Produktkonfigurator als Einstiegspunkt in einen Produktlebenszyklus. In Im Produktkonfigurator existiert eine Asset Administration Shell Template Bibliothek, welche die Inhalte bereithält, die in der Anlage konfiguriert werden können. Diese Bibliothek wird einerseits durch einen Import aus Eplan-Daten erzeugt, als manuell durch einen Projektingenieur, der besonders komplexe Modelle, oder jene die ein

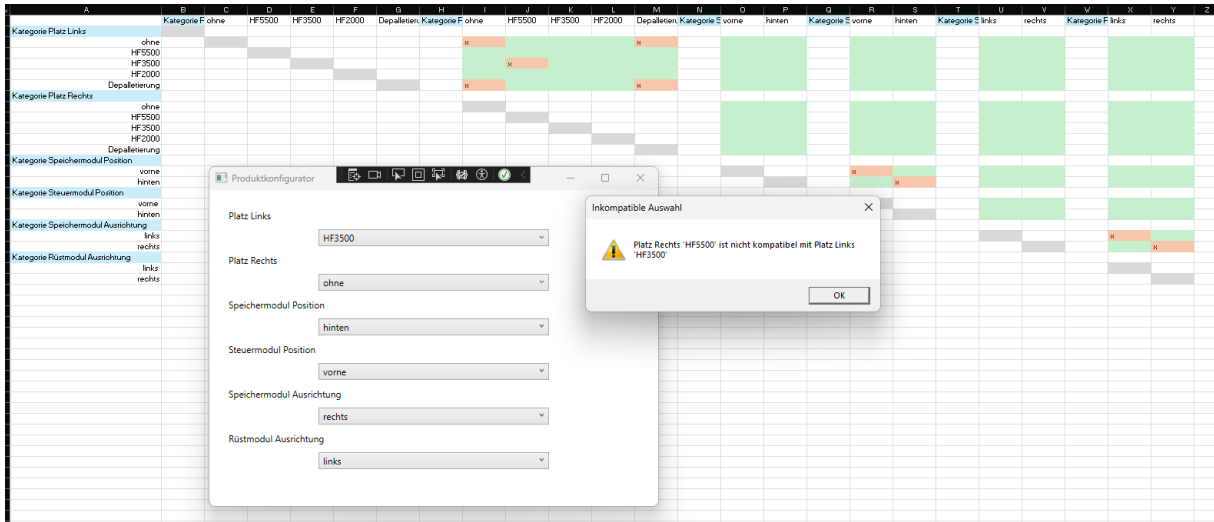


Abbildung 12: Variantenmanagement im Prototyp

hohes implizites Wissen beinhalten händisch in die Bibliothek einpflegt. Mit Hilfe des Produktkonfigurator wird im Anschluss das Anlagenmodell konfiguriert. Hierbei werden die zugehörigen Asset Administration Shell Templates in eine Verwaltungsschale instanziiert, welche vom Asset Administration Shell Server bereitgestellt wird. Die so erstellte Verwaltungsschale stellt die Modellinformationen für den Aufbau des Software-in-the-Loop Projektes bereit. Mittels dieser Simulation der realen Anlage wird die reale Annahme inbetriebgenommen. Die reale Anlage leitet ihre Anlagendaten an den Digitalen Schatten weiter, welcher daraus Analysedaten generiert, welche wieder zurück in die Verwaltungsschale gespeist wird.

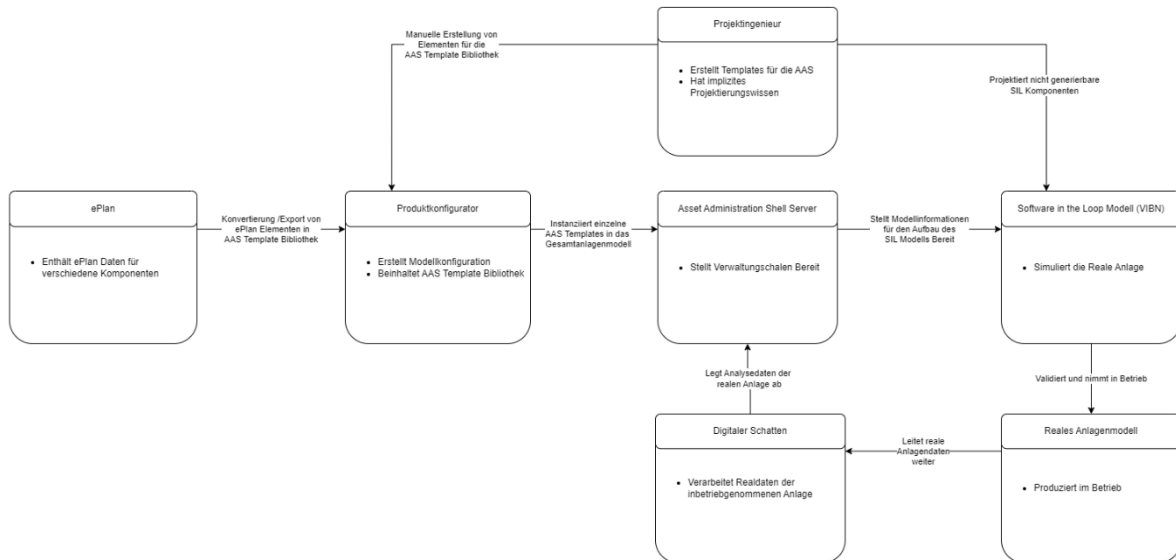


Abbildung 13: Asset Administration Shell als Datenbasis für den Produktkonfigurator

1.3 S3.2: Automatisiertes Testen vom und am Digitalen Zwilling

Eine von Software gesteuerte automatisierte Erstellung, oder auch Komposition von Steuerungssoftware und Maschinenverhalten, erfordert ein besonders hohes Maß an automatisierter Qualitätssicherung. Dieser Punkt wurde im Rahmen des Projekts im Arbeitspaket S3.2 ausführlich beleuchtet.

Es wurden mehrere Aspekte beleuchtet, die in zwei Hauptkategorien aufgeteilt werden können. Diese dienen zusammen der Qualitätssicherung der Endsoftware.

- Automatisiertes Testen von Digitalen Zwillingen.
- Automatisiertes Testen am Digitalen Zwilling.

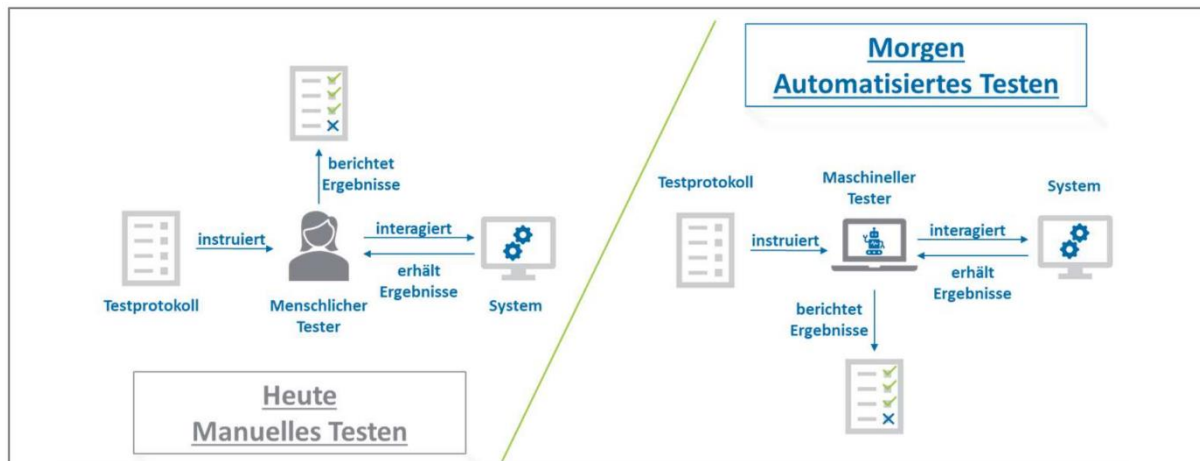


Abbildung 14: Softwaretest Heute und Morgen

Zum Projektbeginn wurden vorhandene Tests von Automatisierungssoftware und von virtuellen Modellen analysiert. Heute basieren Steuerungstests auf Testanweisungen, die einen menschlichen Tester instruieren, Aktionen an einem System durchzuführen und das beobachtete Systemverhalten anhand von Akzeptanzkriterien zu überprüfen. Die hierdurch gewonnenen Ergebnisse werden manuell dokumentiert und ausgewertet.

Werden Fehler gefunden oder Änderungen durchgeführt muss ein Test nochmals durchgeführt werden.

Es zeigte sich eine sehr hohe Abhängigkeit vom durchführenden Bearbeiter. Dies betrifft sowohl die Ausformulierung wie auch die Ausführung von Tests, da eine verbindliche Spezifikation von Testtiefe und Testumfang häufig fehlt. Hier bieten automatisierte Tests, verbunden mit einem modularen, wiederverwendbaren Aufbau der Software, einen vielversprechenden Ansatz.

Eine Analyse vorhandener Testdokumente, zeigte dass sich nahezu alle Testfälle mit einem einfachen Satz an Schlüsselworten vollumfänglich formalisiert notieren lassen.

Weiterhin wurde der aktuelle Stand der Wissenschaft bewertet. Hierzu wurde insbesondere die Dissertation von Karl Kübler an der Universität Stuttgart zum Thema „Methodik für eine ganzheitliche Testautomatisierung beim Systemtest von automatisierten Fertigungssystemen“ als Darstellung des Status Quo herangezogen. Die von Kübler identifizierte Forschungslücke ist in Abbildung 16 dargestellt. In Anbetracht des Fehlens einer vorhandenen kommerziellen Lösung, welche automatisiertes Testen an durchgängigen Systemen von MIL bis zu HIL ermöglicht, wurde die vorhandene firmeninterne Suite „HeiMAX“ als geeignete Basis für den Aufbau einer Testumgebung identifiziert. Diese wurde für alle praktischen Tests und Demonstrationen verwendet und im Rahmen des Forschungsprojekts weiterentwickelt und verbessert.

Die Softwarelösung „Dirigent“ des Konsortialpartners ISG Steuerungstechnik wurde ebenfalls ins Auge gefasst. Diese ist aber eng mit der zugehörigen VIBN-Software Virtuos verstrickt, so dass es im Widerspruch zum toolunabhängigen Ansatz der HEITEC stand. Das Thema Testautomatisierung wurde aber sowohl bilateral als auch in diversen Arbeitsgruppen, offen und gemeinschaftlich durch beide Firmen getrieben und diskutiert.

In einem ersten Schritt wurde die Test-Suite aufgebaut und die Unterstützung für den notwendigen Satz an Schlüsselworten implementiert. So kann zum Beispiel der Wert von Variablen gesetzt (SET) und auf den vorgegebenen Wert einer Variable (AWAIT) gewartet werden. Da es im Fokus stand möglichst keine Bindung zu externen Systemen zu bekommen wurde der Kommunikationsweg OPC UA ausgewählt. Der Standard wird mittlerweile von nahezu allen gängigen Steuerungssystemen unterstützt und es sind diesbezüglich keine Einschränkungen der, zu testenden, Systeme zu erwarten. In Proof-of-Concept Aufbauten wurden Steuerungen zu Beginn Steuerungen von Siemens herangezogen. Im Projektverlauf wurden noch Steuerungen von Allen Bradley und Beckhoff verwendet, auch um schon auf der Steuerungsebene ein erstes Abstraktionslevel zu ziehen. Ein einfacher Unit-Test auf einer Steuerung mittels der Testumgebung ist in Abbildung 15 zu sehen.

Schritt	Befehl	Operation	Variable	Erwartet	Istwert	Ergebnis	Ergebnisdetails	Zeitpunkt
1	Set	Set	ns-11;s=57-1200.Memory.Input 1	True	True	Ok		21:21:34.033
2	Set	Set	ns-11;s=57-1200.Memory.Input 2	True	True	Ok		21:21:34.087
3	Await	Await	ns-11;s=57-1200.Outputs.Output 1	True	True	Ok		21:21:34.170
4	Await	Await	ns-11;s=57-1200.Outputs.Output 2	True	True	Ok		21:21:34.233
5	Set	Set	ns-11;s=57-1200.Memory.Input 1	False	False	Ok		21:21:34.293
6	Await	Await	ns-11;s=57-1200.Outputs.Output 1	False	True	Fehler	Errorf Parameter: ns-11;s=57-12	21:22:34.370
7	Set	Set				Fehler		21:22:34.370

Abbildung 15: Automatisierter Unit Test in der HeiMAX Test-Suite

Der Aufwand für die Testung von Automatisierungssystemen ist häufig hoch. Häufig ist die erstellte Steuerungssoftware individuell und Maschinenspezifisch. Daher müssen auch die zugehörigen Tests anlagenspezifisch erstellt werden. Hier setzt das SDM4FZI Projekt mit seinem ganzheitlichen Ansatz an. So wurde in den Arbeitsgruppen stets das Augenmerk auf die Themen Standardisierung und Modularisierung gelegt. Testfälle müssen passend zu der modularen Software vorliegen und abhängig von der jeweiligen Orchestrierung der Module angesteuert werden. So kann der Aufwand für die Erstellung von Tests für eine Automatisierungslösung stark reduziert werden. Auch wenn der initiale Aufwand nicht vernachlässigt werden darf, um die Modulbibliothek zu erstellen und mit Tests zu versehen. Ein weiterer Ansatz, um das Verhältnis von Aufwand und Nutzen zu verbessern stellt die wiederholte Verwendung der Tests in sogenannten Regressionstests dar. Regressionstests dienen der umfassenden Testung einer Softwarelösung nach Änderungen. Diese können schon bei der Erstellung der Software eingeführt werden und werden bei jeder Änderung, oder bei jedem Bau der Software automatisch angestoßen. So können ungewollte Querabhängigkeiten ausgeschlossen werden, da eine Funktion immer wieder getestet wird, sobald sich eine Änderung ergibt. Hierfür ist eine automatisierte Durchführung der Testfälle essenziell. So können Aufwände für die Testdurchführung auf ein Minimum gesenkt werden und nur so macht das wiederholte umfassende Testen Sinn. Automatisierte Regressionstests zu ermöglichen, wurde zu einer Kernanforderung an ein Testsystem festgelegt. Die Umsetzung konnte im frühen Projektverlauf in Proof-of-concepts, und später

auch am Demonstrator erfolgreich gezeigt werden.

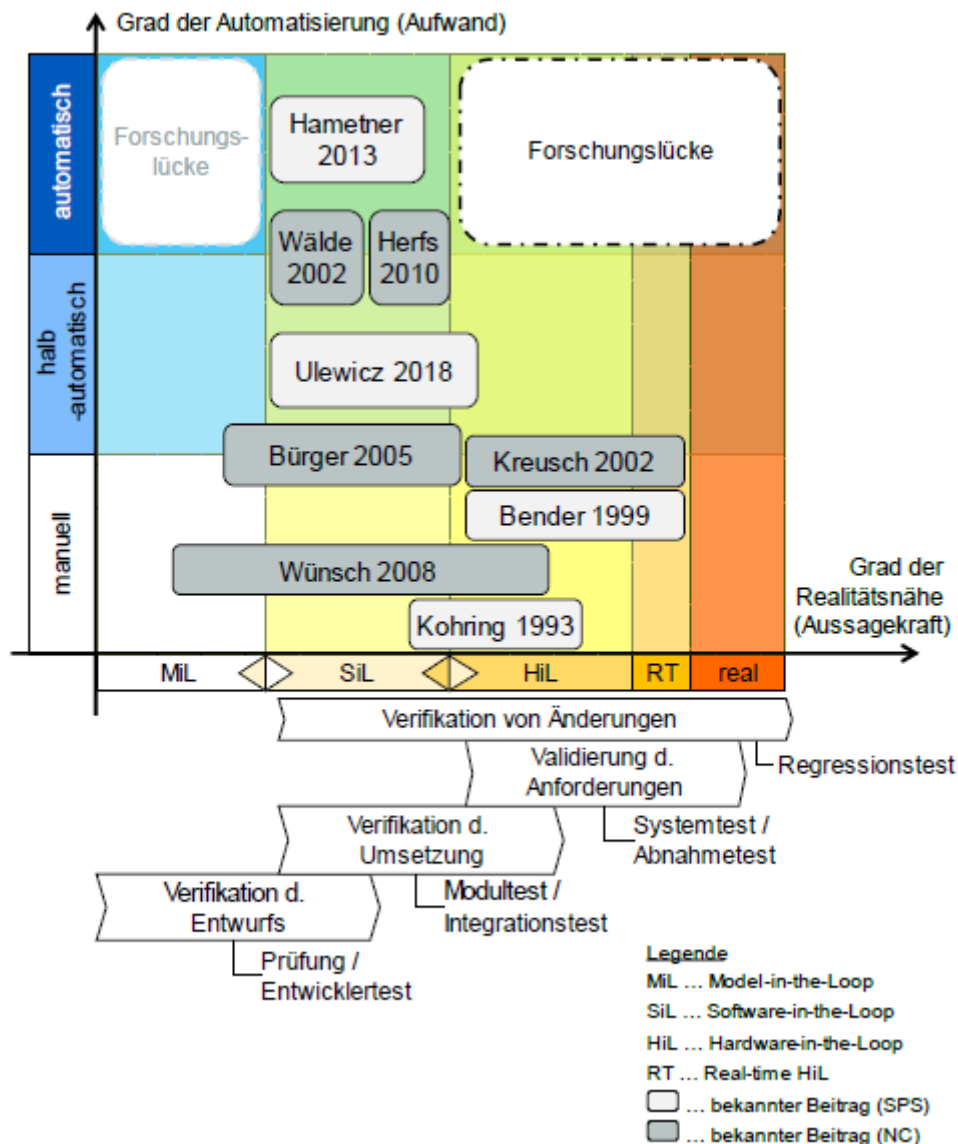


Abbildung 16: [K.Kübler, Methodik für eine ganzheitliche Testautomatisierung beim Systemtest von automatisierten Fertigungssystemen, 2022] Einordnung der bekannten Forschungsarbeiten zwischen Aufwand und Aussagekraft der Testmethodik; Identifikation der Forschungslücke.

Steigerung der Testtiefe

Im späteren Projektverlauf wurde das Thema beleuchtet, wie man die geschaffenen Grundlagen zu einer höheren Testtiefe ausbauen kann. Hierbei wurden insbesondere zwei Aspekte betrachtet.

- Einbeziehen des Digitalen Zwillings in die automatisierte Testung
- Echtzeitfähige und Zyklusgenaue Testung

Beide Aspekte bedienen die schon angeführte und in Abbildung 16 dargestellte Forschungslücke zu der realitätsnahen automatisierten Testung.

Einbeziehen des Digitalen Zwillings

Bestehende Testlösungen für Software oder auch Automatisierungssysteme umgehen diese Lücke und setzen den Fokus anders. So wird in der Regel nur ein einzelnes System für sich getestet. So sind Unit-Tests, auch automatisiert, in aller Regel gut durchführbar. Soll aber zum Beispiel steuerungsübergreifend automatisiert ein Gesamtsystem getestet werden

geraten die Testlösungen an ihre Grenzen. Diese sind häufig proprietär und auf ein System, oder eine Systemlandschaft ausgelegt, beispielsweise die Siemens TIA Testsuite, die auf die namensgebende Steuerung ausgelegt ist. Ist im zu testenden System nun neben einer Siemens Prozesssteuerung eine Motionsteuerung eines anderen Herstellers kann die Interaktion nicht automatisiert abgetestet werden. In diesem Fall müssen beide Systeme getrennt voneinander Unit Tests unterzogen werden, in denen jeweils die Schnittstelle zum anderen System von einer Testumgebung bedient wird. Dies führt zu einem erhöhten Aufwand, da mehr Tests angelegt, verwaltet und gepflegt werden müssen. Außerdem ist ein Risiko gegeben, dass die Bedienung der Schnittstelle durch das Testsystem, insbesondere im zeitlichen Verhalten, nicht dem Verhalten, bzw. dem steuerungstechnischen Zusammenspiel in Realität entsprechen. Letztendlich bleibt man auf die Möglichkeiten und Qualitätsaussagen eines Unit-Tests beschränkt. Um dem „Software-Defined“ Paradigma Rechnung zu tragen war der konsequente Schritt in der Testumgebung Integrationstests zu ermöglichen, unabhängig davon wie der Hardware-Aufbau aussieht. Ein Kommunikationsweg, der mit nahezu allen gängigen industriellen Steuerungssystemen erfolgreich aufgebaut werden kann, ist OPC UA. Dementsprechend wurde in der HeiMAX Kommunikationsplattform ein Support für OPC UA, sowohl als Server, als auch als Client realisiert, um mit allen relevanten Steuerungen kommunizieren zu können. Nochmals beziehungsweise auf die Abbildung 16 konnten wir so, die auf der X-Achse unterhalb des Diagramms aufgetragene Testtiefe bis zu den Hardwareunabhängigen Integrationstests erfolgreich verproben.

Das Ziel automatisierte Systemtests zu ermöglichen war das nächste Ziel, dem sich in der zweiten Hälfte des Projekts gewidmet wurde. Automatisierte Tests an realer Hardware und/oder echten Anlagen sind sicherlich in Bezug auf die Realitätsnähe ein anstrengenswertes Ziel. Betrachtet man jedoch die Sicherheitsaspekte, also das Risiko, das Menschen oder auch technische Einrichtungen zu Schaden kommen durch automatisiert ausgelöste Test- und Störfälle, erkennt man, dass hier nicht in allen Fällen die reale Maschine ein geeignetes Testobjekt darstellen kann. Gerade wenn man den Aspekt betrachtet, dass man automatisierte Regresstests einführen möchte, die jede Nacht überprüfen ob der aktuelle Softwarestand die Spezifikation erfüllt. Benötigt man hier Personal, welches die automatisierten Tests überwacht, hat man zwar eine gesteigerte Qualität und Testtiefe, jedoch zu einem Preis eines stark erhöhten Personalaufwands. Auch die Möglichkeiten Systeme oder Systemkomponenten zu testen, die so in der Montagehalle noch gar nicht real aufgebaut sind, führen zum Test am virtuellen Modell. Mit einem qualitativ hochwertigen virtuellen Modell kann man eine Testtiefe erreichen, die einem sämtliche Systemtests ermöglicht. Limitierender Faktor ist die Simulationstiefe, die das virtuelle Modell erreicht. Häufig sind zum Beispiel Prozessparameter nicht Teil eines virtuellen Inbetriebnahmmodells. Schnelle Prozesse, bei denen es auf das exakte Verhalten der verwendeten Materialien ankommt (z.B. die Farbdicke beim Lackieren von Fahrzeugteilen mit dem Roboter), sind in der Regel nur sehr abstrahiert in virtuellen Modellen abgebildet. Dementsprechend können auch keine prozessspezifischen Systemtests am Modell gefahren werden. In der Fahrzeugindustrie trifft man allerdings sehr häufig auf Roboterbasierte Montageprozesse, welche sehr umfassend abgebildet werden können.

Um das virtuelle Modell in die automatisierte Testung einzubeziehen, benötigt man eine Schnittstelle zwischen Simulationssoftware und Testplattform. Hier ist die Lösung nicht mehr so einfach wie bei den Steuerungssystemen, bei denen es mit OPC UA einen de facto Standard gibt, den jeder unterstützt. Im Projektverlauf haben wir uns mit den Simulationsplattformen ISG Virtuos, machineering iPhysics, Siemens MCD und Unity tiefer auseinandergesetzt. Alle Systeme bieten unterschiedliche proprietäre Schnittstellen und Interaktionsmöglichkeiten. ISG Virtuos hat mit dem ISG Dirigent eine eigene integrierte Testlösung. Eine zusätzliche .NET Schnittstelle entstand im Tool während der Projektlaufzeit. iPhysics bietet eine .NET Schnittstelle über die Funktionen steuerbar sind. In Unity lässt sich durch die Natur als Framework, jegliche Schnittstelle integrieren und programmieren. Ein einheitlicher Standard, welche Funktionalitäten, welche Befehle und welche Events aber in einer Simulationsumgebung ausgelöst werden können oder sollten, existiert aber nicht und ist auch nicht absehbar. Für das Projekt wurde daher mit einem

proof-of-concept Ansatz gearbeitet, mit dem Ziel die erreichbare Testtiefe in den Vordergrund zu stellen, wohlwissend, dass diese nicht nahtlos von einer Simulationsumgebung auf die nächste übertragen werden kann. Der Prototyp wurde für die Simulationsumgebung iPhysics umgesetzt, der Systemaufbau ist in Abbildung 17 dargestellt. Insbesondere die Möglichkeit am virtuellen Modell Materialflusselemente in den Testablauf integrieren zu können, erschließt eine große Menge Tests, die sonst nur an der echten Anlage möglich wären. Dadurch, dass die Materialflusselemente im virtuellen Modell analog zur Realität mit Sensoren und Aktoren interagieren, konnte eine starke Reduktion der Komplexität in der Testerstellung festgestellt werden. Um einen komplexeren Systemtest

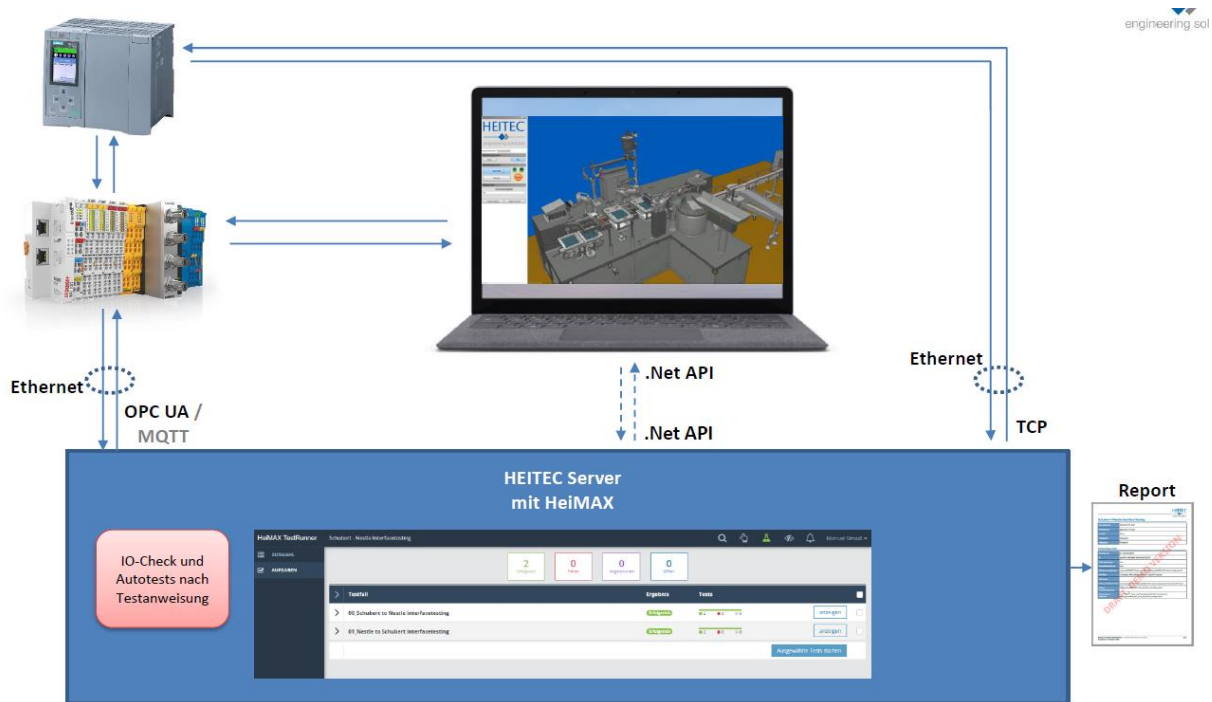


Abbildung 17: Systemaufbau für Tests am virtuellen Modell

abzubilden, würden ohne virtuellen Materialfluss viele Signale der Maschine künstlich erzeugt werden und zeitlich orchestriert werden. So muss jeder bedämpfte Sensor auf dem Weg des Materials künstlich erzeugt werden, da sonst ggf. Fehlermeldungen die Maschine stoppen, zum Beispiel durch einen Fehler „Verlorenes Teil“, da die Automatisierung erwartet, nachdem ein Förderband gestartet wurde, innerhalb von zwei Sekunden eine steigende Flanke am Infrarotsensor zu sehen und eine weitere Sekunde später wird erwartet das eine RFID-Pille erfolgreich ausgelesen werden kann. Abhängig von den Informationen der ausgelesenen Informationen unterscheidet sich die weitere Bearbeitung. Durch ein virtuelles Modell, welches den Materialfluss simuliert kann von diesen Zwischenbedingungen abstrahiert werden. Über die API zur Simulationsumgebung wird lediglich ein virtuelles Materialflusselement am Übergabepunkt der vorgelagerten Maschine erzeugt. Der Weg des zu bearbeitenden Teils durch die Anlage wird durch die Automatisierung im Verbund mit den erzeugten Sensorsignalen aus dem Modell erzeugt und muss nicht händisch in der Testvorschrift berücksichtigt werden. Für einen Test, der überprüft ob ein Materialflusselement vom Typ A auch wie gewollt im Lager C eingelagert wird reichen also sehr wenige Testschritte:

1. Starte Anlage (Trigger HMI Knopf über Testumgebung)
2. Erzeuge Materialflusselement vom Typ A am Eingang (API der Simulationsumgebung)

OK Bedingung: Materialflusselement vom TYP A wird am Übergabepunkt zu Lager C gescannt.

NOK-Bedingungen:

- Materialflusselement vom Typ A wird an einem anderen Lagerübergabepunkt gescannt.
- Dauer von Teststart liegt länger als X Sekunden zurück und es ist kein Materialflusselement vom Typ A am Übergabepunkt zu Lager C gescannt worden.

Alle dazwischenliegenden Abhängigkeiten, Sensoren und Aktoren können ignoriert werden, so dass sehr einfach auch übergreifende Tests formuliert werden können. So ist es auch möglich länger laufende Tests anzulegen, bei denen das Verhalten der Anlage abgeprüft wird, das sich einstellt, wenn Materialflusselemente unterschiedlichen Typs in bestimmten, oder auch zufälligen Reihenfolgen eingelastet wird.

Im Projekt konnten so im Verbund mit dem virtuellen Modell erfolgreich Systemtests gezeigt werden. Alle notwendigen Schnittstellen zu den beteiligten Steuerungen und der verwendeten Simulationsumgebung wurden in die HeiMAX Testumgebung integriert. So dass diese Systemtests am virtuellen Modell auch als automatisiert angestoßene Regressionstests umgesetzt werden konnten.

Ein zweiter großer Punkt, mit dem man sich im Projektverlauf beschäftigt hat, war die Echtzeitfähigkeit und Zyklusgenauigkeit der Testumgebung. Wie oben dargelegt wurde sich für eine Kommunikation über das OPC UA Protokoll mit den Steuerungen entschieden. Viele der OPC UA Servern, die auf gängigen Steuerungen laufen kommen typischerweise auf Zykluszeiten von 50-500ms. Dies ist offensichtlich nicht geeignet, um Tests an Steuerungen auszuführen, bei denen es auf die zyklusgetreue Betrachtung von Signalen ankommt. Beispielsweise könnte es sein, dass Flanken an einem Bausteinausgang vom Testsystem gar nicht erfasst werden, wenn sie nur für einen Steuerungszyklus aktiv sind. Andere Programmbausteine im Systemverbund können aber potenziell auf die Flanke reagieren. So entsteht aus Sicht der Testlösung nicht deterministisches Verhalten, da bei mehreren Durchläufen desselben Tests unterschiedliche Signalverläufe entstehen können. Abhilfe könnte hier die flächendeckende Verbreitung von OPC UA mit TSN (Time-Sensitive Networking) im Steuerungsumfeld bieten. Hierbei wird die standardisierte OPC UA-Kommunikation mit deterministischer Ethernet-Technologie kombiniert. Dadurch wird eine zuverlässige und zeitkritische Datenübertragung mit Zykluszeiten im Sub-Millisekunden Bereich (<1ms) ermöglicht. So entstehen nochmal umfassendere Testmöglichkeiten auch über Edge-Computing oder eine Cloud-Integration. Im Forschungsprojekt wurden dem Thema mangels aktueller konkreter Umsetzbarkeit auf Steuerungsebene aber keine tiefergehenden Aufwände gewidmet.

Zuerst wurde getestet, direkt aus dem Siemens Steuerungsprogramm heraus Werte in einer SQL-Datenbank zyklusgetreu zu loggen. Hier traten aber hin und wieder zeitliche Probleme bei den Latenzen auf, was den Ansatz nicht verlässlich macht. Er war zwar ein Schritt in die richtige Richtung, versprach aber keine nachhaltige technische Basis zu liefern.

Vielmehr wurde der Ansatz verfolgt die VIBN-Software als Plattform zu verwenden, um zyklusgetreue SPS-Signale zu lesen und zu setzen. Simulationsmodelle werden in unserem Haus über eine Plattform auf Beckhoff Basis an Steuerungen und Co-Simulationen angebunden. Dies Plattform ermöglicht es mit real vorhandenen, oder auch in Software emulierten, Steuerungen auf der Feldbusebene zu kommunizieren. Diese VIBN-Plattform wurde im Proof-of-concept als Middleware zwischen die Testlösung und das zu testende System geschaltet. Ein schematischer Systemaufbau ist in Abbildung 18 dargestellt, hier für den Fall einer mittels PLCSimAdvanced emulierten Soft-SPS aus der Siemens Produktfamilie. Die Signale konnten so über die Plattform zyklusgenau gesetzt werden. Die Ausgangsdaten wurden im Echtzeitfähigen System gesammelt und in eine Datenbank gespeichert. Die Auswertung fand dann auf Basis der aufgezeichneten Daten statt. Hierfür wurde InfluxDB verwendet, da es sich um eine leistungsstarke, zeitreihenbasierte Datenbank handelt, welche speziell für das Speichern und Analysieren von zeitabhängigen Daten optimiert ist.

So konnte auch das automatisierte Testen unter Echtzeitbedingungen am virtuellen Modell erfolgreich in einer Konzeptstudie gezeigt werden. Allerdings muss gesagt werden, dass es ein spezifischer Aufbau ist, der nicht ohne Weiteres auf alle gängigen Steuerungen und

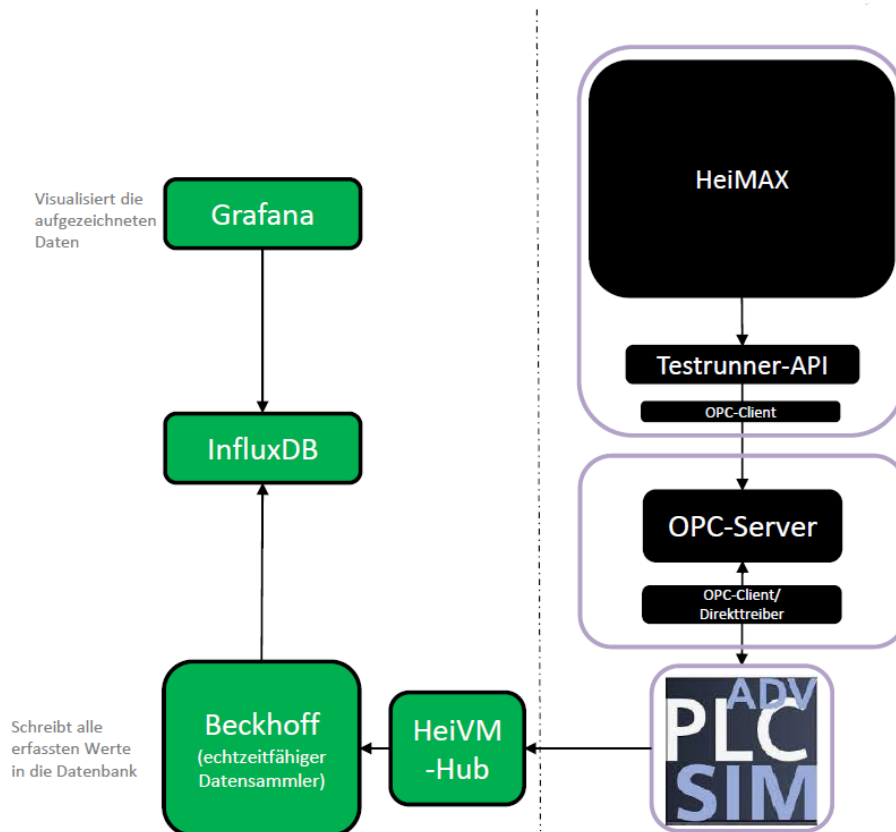


Abbildung 18: Systemaufbau für echtzeitfähiges Testen

Feldbustypen übertragbar ist. Außerdem sind für den Testaufbau Änderungen an den Steuerungsprogrammen notwendig um alle Signale, die für die Tests relevant sind auf der Feldbusebene abzubilden. Dadurch entstehen trotz der prinzipiellen Machbarkeit in der praktischen Anwendung noch Probleme. Denn durch die notwendigen Anpassungen kann der Automatisierte Tests nicht reibungslos in eine CI/CD Pipeline eingebunden werden. Das Feld automatisiertes Testen unter realitätsnahen Bedingungen in Echtzeitverhältnissen weist somit auch weiterhin eine Forschungslücke auf, auch wenn erste Schritte gegangen wurden.

Insgesamt konnte im Projektverlauf sowohl erfolgreich der automatisierte Test vom virtuellen Modell, als auch der automatisierte Test am virtuellen Modell gezeigt werden. Für die in Teilarbeitspaket B1 erstellten Verhaltensmodelle im FMU-Standard konnte gezeigt werden, wie die Komponenten in der aufrufenden Simulationsumgebung automatisiert testbar sind. Wie also ein Unit-Test nicht nur losgelöst auf Code-Ebene, sondern fundiert vom zeitlichen Verhalten in der aufrufenden Simulationsumgebung gestaltet werden kann.

Weiterhin wurde gezeigt, wie durch die Einbindung des virtuellen Modells in den Testaufbau auf Steuerungsebene auch übergreifende Integrations- und Systemtests automatisiert und damit strukturiert ausgeführt werden können. Dies kann zu einer signifikanten Qualitätssteigerung bei gleichbleibenden oder nur moderat steigenden Aufwänden für die Testerstellung führen.

Ergebnisseitig wurde in der Testumgebung ein anpassbares Ausgangsformat entwickelt, dass die Testergebnisse darstellt. Auf diesem sind sowohl Trace Aufzeichnungen einzelner Variablen vorgesehen als auch die Aufschlüsselung einzelner abgetesteter Use-Cases des zu testenden Systems. Die Protokollierung spielte aber in der Gesamtbetrachtung gegenüber der Testung an sich eine untergeordnete Rolle.

1.4 S3.5: Demonstratorentwicklung

Viele Prinzipien der softwaregetriebenen Fertigung und damit auch viele Ideen und Prinzipien, die im SDM4FZI Projekt zum Projektbeginn diskutiert wurden, basieren auf der Standardisierung von Komponenten und Schnittstellen und daraus resultierend auf Generierbarkeit und Orchestrierfähigkeit von Software. Diese Prinzipien haben wir an Demonstratorapplikationen in einzelnen Schritten umgesetzt und getestet.

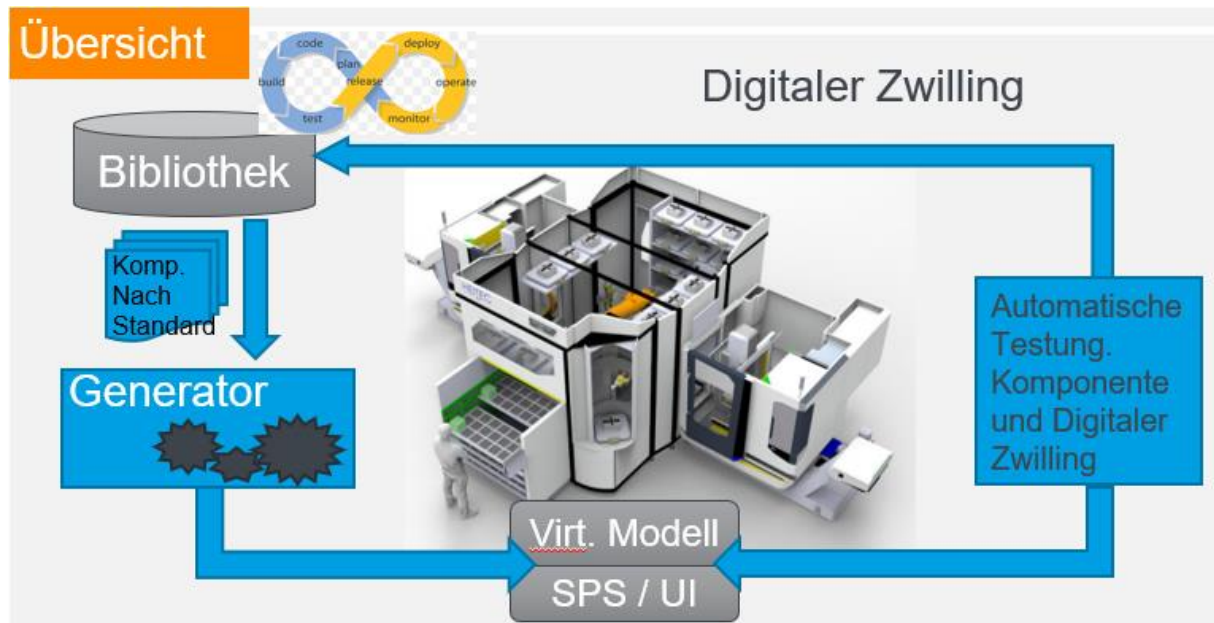


Abbildung 19: Übersicht Demonstrator-Konzept

In der Abbildung 19 ist das Demonstrator-Konzept in der Übersicht dargestellt. Datenbasis bildet die in Kapitel 1.1 aufgebaute Bibliothek aus standardisierten Verhaltensmodellen. Wie in dem zugehörigen Abschnitt beschrieben liegen diese über das FMU-Format in einer einheitlichen standardisierten Form vor. Notwendige Parameter zum Aufruf des Verhaltensmodells sind in dem Paket enthalten und können direkt von der Simulation verwendet werden. Über das Submodel Simulation der Verwaltungsschale lassen sich die Verhaltensmodelle durch weitere Rahmendatenpunkte geeignet katalogisieren, verwalten und softwaregestützt orchestrieren.

Diese Orchestrierung wurde im Demonstrator durch den im Übersichtsbild mit „Generator“ dargestellten Block umgesetzt. Durch den Reifegrad des Verwaltungsschalen-Submodells und den Projektfortschritt wurde der Aspekt in einer abstrahierten Proof-of-Concept Variante umgesetzt, in der die losgelösten FMUs verwendet wurden und auf das Einbetten in eine Verwaltungsschale verzichtet wurde.

In der ersten Demonstratorstufe wurden drei virtuelle Modelle einer Bearbeitungszelle aus dem HEITEC eigenen Maschinenbau erstellt. In dieser wurden die standardisierten Verhaltensmodelle verwendet, aber in drei unterschiedlichen Simulationsumgebungen verwendet. Dies waren ISG Virtuos, machineering iPhysics und HeiVM Unity. So konnte erfolgreich gezeigt werden, dass über den FMU-Standard simulationsumgebungsübergreifend erfolgreich standardisierte Verhaltensmodelle erstellt und verwendet werden können. Die einheitliche Verwendung als Voraussetzung für eine Generierbarkeit wurde dabei ebenso nachgewiesen wie die einheitliche Ansteuerung über das zugehörige User Interface.

Im nächsten Schritt des Demonstrators ging es darum das virtuelle Modell (in Teilen) zu generieren. Hierbei wurde sich auf das Verhaltensmodell konzentriert und der 3D-CAD Aspekt vorerst außenvor gelassen. Hierfür wurde sich für den weiteren Projektverlauf auf die Simulationsumgebung HeiVM + Unity fokussiert und das manuell im ersten Schritt erstellte

Modell analysiert. Zusammen mit der Projekterfahrung einiger langjähriger Projektgenieure wurden einige Teilgeneratoren erschaffen und im sogenannten HeiVM Hub zusammengebracht. So konnte der Grad der generierbaren Inhalte immer weiter erhöht werden, ohne an der Komplexität zu scheitern direkt das komplette Modell generieren zu müssen. Durch den iterativen Ansatz konnten auch sukzessive die notwendigen Voraussetzungen geschaffen werden. So ist für die Generierung eine vorherige Standardisierung unerlässlich. Um diesen Standard aufzubauen, ist es notwendig, schon beim Anlegen von neuen Projekten eine standardisierte Struktur aufzubauen, die dann in jedem Projekt gleich oder zumindest ähnlich ist. In jedem Projekt gibt es eindeutige projektspezifische Werte, die immerwährend unveränderlich das Projekt definieren. Das sind z.B. Projektnummer, Firma des Kunden oder auch der Projektname. Weitere Meta-Daten, die aber nicht zur eindeutigen Projektidentifikation herangezogen werden, sind z.B. der Projektleiter, das Projekt-Team oder Kontaktpersonen. All diese Informationen fließen beim Erstellen eines neuen Projektes in die Projektdaten mit ein, dienen sogar, das Projekt eindeutig auf einem Speichermedium an- und abzulegen. Die einfache Anwendung von Generatoren setzt auch auf dem physischen Speicher eine einheitliche Ordnerstruktur voraus. Zielordner können dann vom Generator erwartet werden, Quellordner liegen an erwartbaren Stellen und Verweise müssen nicht dynamisch angepasst werden.

Mit dem, für das Anlegen eines neuen Projektes entwickelten „**Project Generator**“ wurde jedem Mitarbeiter die Möglichkeit geschaffen, komplexe Projekte einheitlich anzulegen.

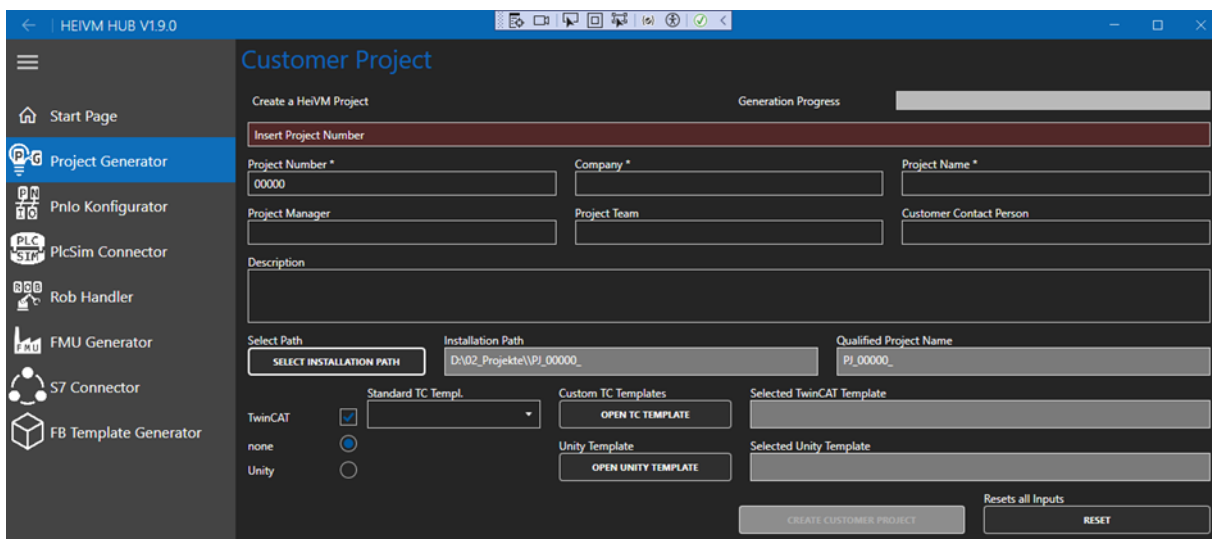


Abbildung 20: Project Generator im HeiVM Hub

Über die statischen Daten und die Ordnerstruktur hinausgehend wird durch den Projektgenerator ein Template Projekt der HeiVM Simulationsumgebung angelegt. Ein korrespondierendes Projekt in der Unity 3D Simulationsumgebung wird ebenfalls automatisch angelegt. Auch kundenspezifische vorkonfektionierte Projekte können hinterlegt werden und als Vorlage herangezogen werden.

Für die Verbindung/Kommunikation der beiden Teilprojekte zur Laufzeit wurde ebenfalls ein Generator geschaffen. Dieser legt basierend auf angelegten Simulationselementen und einer Variablenliste automatisiert die Verbindung basierend auf dem Beckhoff ADS Protokoll an. Mit dem Vorgang wird ebenfalls eine Projektbeschreibungsdatei angelegt. Diese wurde zu Projektbeginn in einem pragmatischen Format, für einen Proof of Concept tauglich, konzeptioniert. Eine Überführung ins Format der Verwaltungsschale wurde später nur noch theoretisch betrachtet (erfolgreich), sie wurde aber nicht mehr praktisch angegangen, da der Mehrwert für das Projektergebnis nicht gegeben gewesen wäre.

Müssen neue Verhaltensmodelle angelegt werden, für die es in der geschaffenen Bibliothek noch keine Umsetzung gibt, können die Vorlagen dafür ebenfalls über im HeiVM Hub enthaltene Generatoren erzeugt werden. Zum einen, im in Abschnitt 1.1 ausführlich

beschriebenen FMU-Format, aber auch native Bausteine in der HeiVM Umgebung können standardisiert erzeugt werden über den „**Function Block Template Generator**“. Hier können einige Modellspezifische Konfigurationen vorgenommen werden. Auch diese

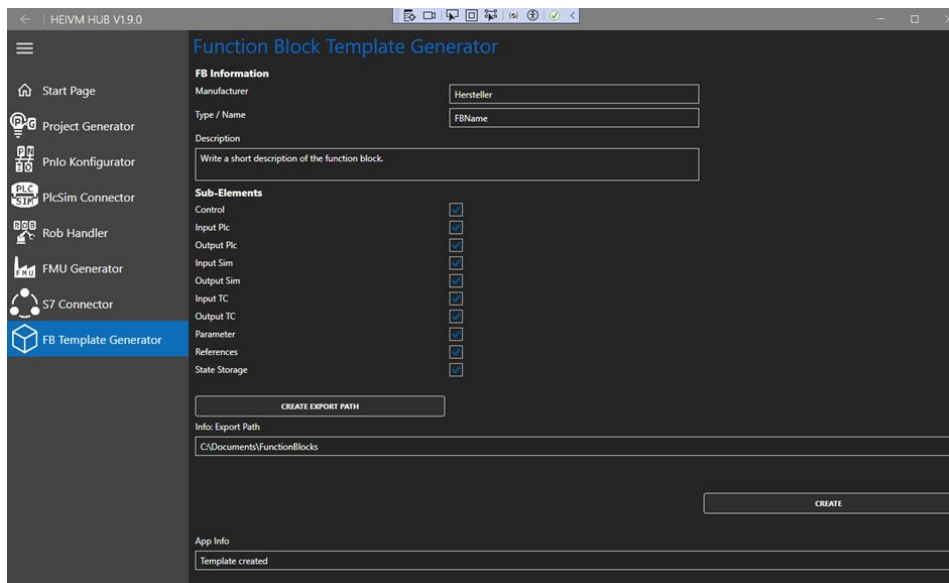


Abbildung 21: Function Block Template Generator im HeiVM Hub

Standardisierung dient letzten Endes dem übergeordneten Ziel ganze Modelle generieren zu können.

Das übergeordnete Big Picture wurde in dieser Projektphase nochmals geschärft und überarbeitet, nachdem viele Teilaufgaben standardisiert waren und über Teilgeneratoren viel händische Arbeit automatisiert war. In dem übergeordneten Blick wurde das Thema Durchgängigkeit vordergründig betrachtet. Der Use Case ist in der nachfolgenden Abbildung dargestellt.

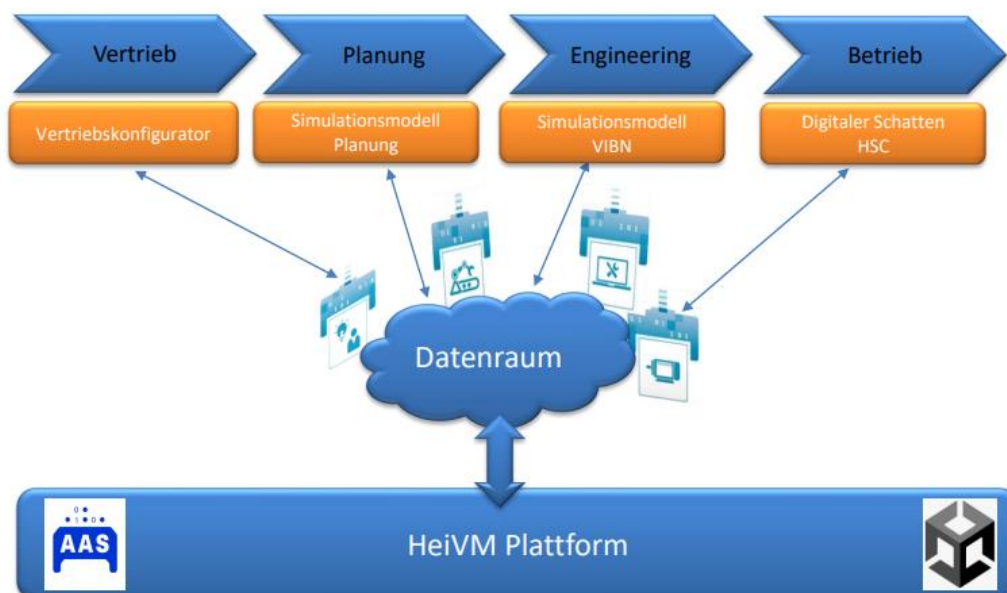


Abbildung 22: UseCase Durchgängigkeit im Digitalen Engineering vom Vertrieb bis zum Betrieb

Ziel war es Informationen schon im Vertriebsprozess aufzunehmen, durchgängig vorzuhalten und gemäß dem „Single Point of Truth“ Prinzip einheitlich zu speichern. Diesen Datenraum kann unseres Erachtens nach die Verwaltungsschale schaffen. Diverse Submodelle decken verschieden Bereiche ab, waren aber zur Projektlaufzeit noch in Entwicklung und Freigabe.

Kleine Konzeptstudien konnten aber erfolgreich mit Verwaltungsschalen und Verwaltungsschalenservern gefahren werden. Der Einstieg in den durchgängigen Prozess erfolgt mit dem Vertriebskonfigurator, indem die Anlage schon (teilweise) mit hinterlegten technischen Daten konfiguriert wird. Dieser Bereich ist ausführlich in Abschnitt 1.2 beschrieben.

Der nächste definierte Schritt ist ein simulatives Planungsmodell. Dieses fügt sich nicht nur vom Prozess, sondern auch technologisch zwischen dem VIBN-Modell und dem Vertriebsmodell ein. So sind die Ergebnisse und Aussagen aus einem Planungsmodell häufig noch im Vertriebsprozess relevant. Beispielsweise erreichbare Taktzeiten, die notwendige Anzahl an Ablageplätzen, um eine vorgegebene Anzahl an unterschiedlichen Werkstücken zu bearbeiten. Häufig führen die Ergebnisse also noch zu direkten Änderungen an der Maschine / Anlage, die angeboten wird. Daher sollte im Idealfall ein Planungsmodell ohne zusätzlichen Mehraufwand direkt aus den bekannten Engineeringinformationen und den kundenindividuellen Merkmalen automatisiert erstellt werden können. Gleichzeitig unterscheidet sich ein Planungsmodell von dem VIBN-Modell nur in der Simulationstiefe. Reicht es vielleicht einen Antrieb in der Planungsphase als sehr idealisierten Rampengenerator zu betrachten, so muss in der VIBN-Phase das komplette Feldbusteleggramm zwischen Antrieb und Steuerung nachgestellt und bedient werden. Kann ein Greifer in der Planungsphase als Alles greifende Sphäre umgesetzt werden, kommt es in der VIBN-Phase vielleicht auf exakte Greifpunkte, Kraftschlüsse und Störkonturen an. Die Beispiele zeigen die Schwierigkeit eines generalistischen Ansatzes für den Punkt Simulationstiefe. Das Thema wurde auch ausgiebig in Arbeitsgruppen mit den Konsortialpartnern diskutiert. Theoretische Ansätze, wie man verschiedene Verhaltensmodelle (z.B. als FMU) in einer Verwaltungsschale der entsprechenden Komponente vorhalten könnte wurden gefunden. Für die Entscheidung welches Verhalten in der aktuellen Situation gewünscht ist müssen beschreibende Informationen beigefügt

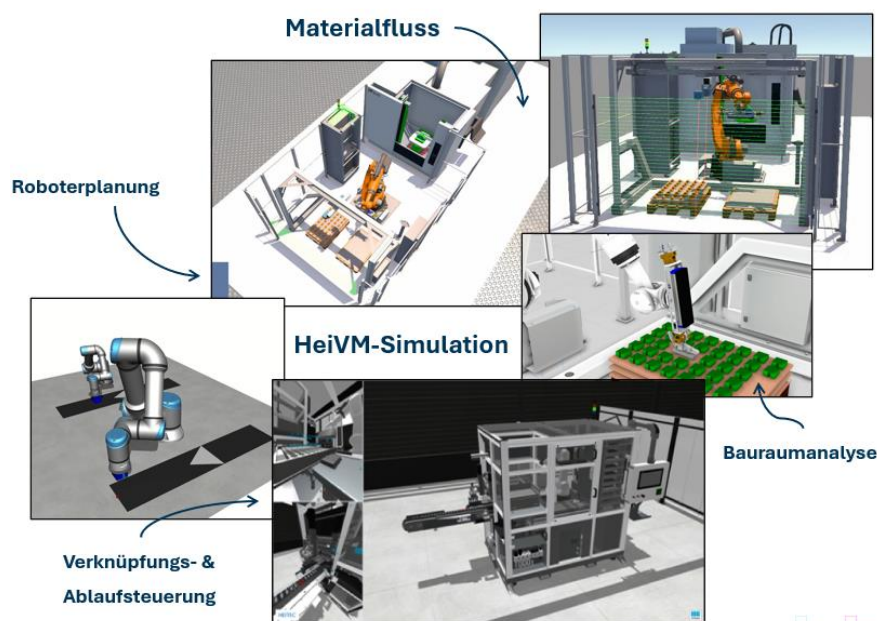


Abbildung 23: Übersicht Planungsphase Demonstrator

werden. Hier konnten aber keine einheitlichen UseCases identifiziert werden und die Einflüsse vom Kontext waren zu groß, um etwas generelles zu definieren. Für unseren Demonstrator haben wir deshalb für die Komponenten jeweils eine Variante „Planungsverhalten“ definiert und eine Variante „VIBN“ Verhalten, diese wurden so geschnitten, dass sie in unseren UseCase gepasst haben. So konnten wir den Ansatz verproben und auch das Thema Ablaufsteuerung bearbeiten. Dies wurde als weiteres Problem identifiziert. Die Orchestrierung von Verhaltensmodellen aus einer BOM (Bill of material) aus der mechanischen und elektrischen Konstruktion schafft ohne Prozess und

Ablaufinformationen kein Modell der Anlage. Hier konnte die prinzipielle Machbarkeit gezeigt werden, das solche Prozesslogiken durch Low Code Ansätze mit niedrigem Aufwand erzeugt werden können. Wie das Verhalten einer Anlage strukturiert im Engineeringprozess verschriftlicht werden könnte stellt aus unserer Sicht aber noch eine Forschungslücke dar, welche aber durch ein entsprechendes Aufwand zu Nutzen Verhältnis gerechtfertigt werden müsste.

Der VIBN-Bereich ist zu Beginn dieses Abschnitts schon beschrieben, da diese Phase für uns der Einstieg war und wir uns von da aus mit den vor- und nachgelagerten Lebensphasen der Anlage beschäftigt haben. In dem durchgängigen Konzept werden bisher komplett abstrahierte Bereiche durch Verhaltensmodelle ergänzt, oder Planungsmodelle mit niedriger Simulationstiefe durch entsprechende Pendanten mit hoher Simulationstiefe ersetzt. In dem bereits angeführten Beispiel des Antriebs, der dadurch vom einfachen Rampengenerator zur vollwertigen Feldbusstelegrammabbildung wechselt, ändert sich so auch die Schnittstelle. Dies führte häufig zu notwendigen manuellen Anpassungen im Modell, auch wenn der grundsätzliche Ansatz schlüssig gezeigt werden konnte aber nicht auf eine generelle volle Automatisierbarkeit gebracht werden konnte.

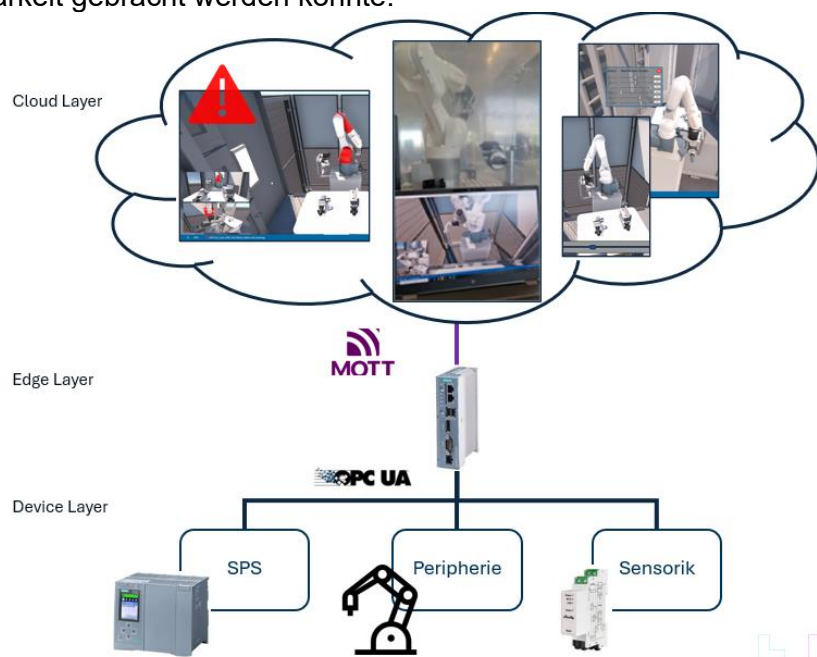


Abbildung 24: Digitaler Schatten Aufbau und Übersicht

Im letzten Abschnitt des SDM4FZI Projekts wurde sich dann auch mit der letzten Phase der Anlage beschäftigt, der Betriebsphase. Hier sollte das VIBN-Modell parallel zum Betrieb der

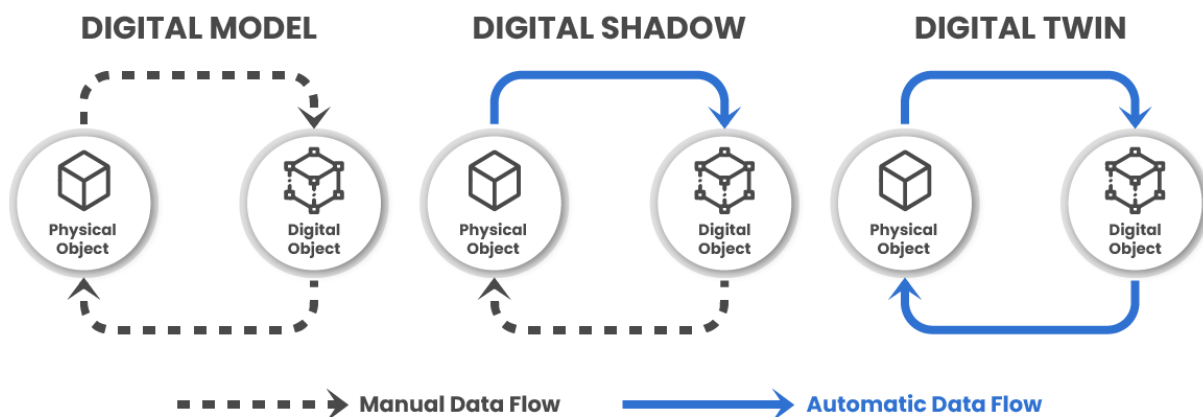


Abbildung 25: Die Abstufungen des Digitalen Zwillings

Anlage weitergenutzt werden. Zuerst muss hier die Ebene und Ausprägung des Zwillings definiert werden. Die Abstufungen sind in Abbildung 25 dargestellt. Wir haben uns für den Mittelweg des Digitalen Schattens entschieden, da wir hierfür am ehesten einen realistischen Mehrwert des Anlagenbetreibers identifizieren konnten. Dann muss sich entschieden werden, ob es sich um einen betriebsbegleitenden oder betriebsparallelen Zwilling handeln soll. Hier haben wir uns aus denselben praxisorientierten Gründen für den betriebsbegleitenden Zwilling entschieden. Es wird also dieselbe Hardware sowohl mit dem realen Asset, als auch mit dem digitalen Abbild verknüpft. Dies führt dazu, dass insbesondere die Kommunikationswege aus dem VIBN-Modell angepasst werden müssen. Auch das Verhalten muss für die Ausprägung als Schatten angepasst werden. Ein einfaches Beispiel ist ein Sensor einer Lichtschranke. Für das VIBN-Modell muss das Modell eine Kollision mit dem Lichtstrahl detektieren und den Sensor auslösen und diese Information an die Steuerung schicken. Im Fall des digitalen Schattens kommt nun die Information, dass der Sensor bedämpft ist aus der Steuerung und muss im Modell kenntlich gemacht werden. Dies kann zum Beispiel über einen Farbumschlag erfolgen. Gleichzeitig kann eine Plausibilitätskontrolle durchgeführt werden. Wäre eine Lichtschranke im Modell nicht unterbrochen, aber in der Steuerung steht der Sensor an ist dies nicht plausibel. Nun kann entweder ein Fehler am Sensor vorliegen, oder aber der Zustand des Modells entspricht nicht dem Zustand der Anlage in der Realität. Hier könnten allgemeine Algorithmen in Zukunft den Mehrwert eines Digitalen Schattens deutlich steigern. Gleichwohl kann der Abgleich des Materialfluss zwischen Modell und Anlage nur dann erfolgen, wenn hinreichend viele Sensoren verbaut sind, die den Materialfluss detektieren. Dies spielt in der Regel in der Konstruktion der Anlage eine untergeordnete Rolle, es werden üblicherweise nur die für den realen Prozess notwendigen Sensoren verbaut.

Wir haben uns daher im Demonstrator auf die Generierbarkeit und einfache Verschaltung von Features konzentriert die wir als Leistungsumfang für einen Digitalen Schatten gesammelt hatten.

- Live-View des Anlagenzustands in 3D (ohne Materialfluss)
- History-View: Abspielen von aufgezeichneten Daten aus der Vergangenheit in der 3D Ansicht.
- Overlay Anzeigen von aktuellen Prozesswerten (Druck, Temperatur oder Ähnliches)
- Optische Darstellung von Binären Sensorzuständen
- Statusdarstellung zum Schutzkreis (z.B. Hervorheben einer geöffneten Schutztür)

Der Prototyp des Digitalen Schattens konnte im Projektverlauf nicht vollständig fertiggestellt werden. Allerdings konnte die Machbarkeit gezeigt werden und die prinzipiell notwendigen Schritte und Infrastrukturen konnten aufgezeigt und geschaffen werden.

1.5 S4: Datenbasierte Dienste

Das Arbeitspaket S4 stellte für die HEITEC AG nur ein kleineres Arbeitspaket im Gesamtvorhaben dar. Ziel war es zu untersuchen, wie eine intuitive Benutzersteuerung eines Digitalen Zwillings gestaltet sein muss, um eine gute User Experience zu liefern. Als zu betrachtendes System wurde das Interface des Virtuellen Modells gewählt, das auch für den Anwendungsfall des digitalen Schattens Anwendung findet, in dem betriebsbegleitende Mehrwerte wie Fehlersuche, Optimierung, Predictive Maintenance und Condition Monitoring realisiert werden können. Es wurden die verschiedenen bestehenden Bedienmöglichkeiten erfasst, die die Plattform bot:

- Bedienung über Pop-Ups direkt im Modell.
- Bedienung über direkte Interaktion mit dem Modell.
- Bedienung über applikatives Control Panel abseits des Modells.

Es wurde eine Befragung zur UI/UX Experience der Demonstrator Applikation unter Fachkollegen, aber auch unter Fach- und Firmenfremden Personen durchgeführt.

So wurde untersucht, welche Bedienart effizient wahrgenommen wurde und wie der Benutzer intuitiv zur jeweiligen Bedienart geführt werden kann. Hervorhebungen bei Mouse-

Over-Events erwiesen sich als ebenso hilfreich, wie halbtransparente Indikatoren / Sphären im Modell, welche eine Interaktionsmöglichkeit diskret, aber gut wahrnehmbar darstellen. Jegliche Interaktion direkt im Modell wurde intuitiver wahrgenommen als eine Einstellmöglichkeit über HMI-Panels.

In einem kurzen Proof-of-Concept wurde die immersive Wirkung betrachtet die AR-Brillen wie die Apple Vision Pro haben. Insgesamt zeigte sich hier ein beeindruckender Effekt, dieselbe Applikation wurde in einer AR-Umgebung wesentlich beeindruckender wahrgenommen, als auf einem normalen 2D Monitor. Die technischen Hürden solche Systeme nachhaltig zu integrieren sind aktuell aber noch recht hoch und rechtfertigen den Aufwand nur in Einzelfällen. Eine tiefere Betrachtung des Aspekts war nicht Teil des Projekts.

1.6 Nutzung der Sachkosten / Lizenzen

Wie im einleitenden Satz zu Beginn des Schlussberichts Teil II schon erwähnt, bestand der absolute Hauptteil der Aufwände der HEITEC AG in Personalkosten. Es wurden aber auch einige Sachkosten geltend gemacht, die für Lizenzkosten verwendet wurden. Diese werden hier kurz aufgeführt und erläutert:

Lizenz	Dauer	Beschreibung und Nutzen
TwinStore	2022-2024	Mitgliedschaft in der TwinStore Community des Konsortialpartners ISG. Notwendig für die Arbeiten in B1 und S3 um die Erstellung von standardisierten Verhaltensmodellen kompatibel zum TwinStore Standard zu erstellen. Weiterhin bei der Generierung von Modellen aus Komponenten genutzt.
ISG Virtuos	2022-2024	Die Lizenz wurde genutzt um Modelle zur virtuellen Inbetriebnahme in ISG Virtuos zu erstellen. Verwendet sowohl in B1 für standardisierte Verhaltensmodelle, als auch in S3 für Demonstratoren und die Generierung
MS Visual Studio	2022-2024	Notwendige Lizenz für die hauptsächlichen Bearbeiter des Projekts für sämtliche Tätigkeiten. In jedem Arbeitspaket musste Quellcode erzeugt, verwaltet und übersetzt werden.
Unity Dev	2023-2024	Notwendige Lizenz für die Arbeiten an den Demonstratoren und den Prototypen des Produktkonfigurators und des Digitalen Schattens die in S3 entwickelt wurden.
TwinCAT FMI Importer	2022	Notwendig um FMUs ins Beckhoff TwinCat Echtzeit System zu importieren. Dies wurde in B1 verprobt.

2 Bekannt gewordener Fortschritt auf dem Gebiet des Vorhabens

Es gab innerhalb der Projektlaufzeit bekanntgegebene FE-Ergebnisse, die Auswirkungen auf die Durchführung des Vorhabens hatten. Die HEITEC, wie auch das gesamte Konsortium strebten es an bestehende Standards zu verwenden und keine konkurrierenden neuen Standards aufzubauen. Daher waren folgende Veröffentlichungen in der Projektlaufzeit von besonderem Interesse für uns:

2.1 FMI-Standard

Die HEITEC AG hat sich für die Erstellung von Verhaltensmodellen auf den FMI-Standard festgelegt. Dieser kann mit der im Projekt entwickelten Referenzarchitektur und Submodel Simulation der Verwaltungsschale ergänzend kombiniert werden. Zu Projektbeginn war hier

der Standard in der Version 2.0 aktuell. Entsprechend wurde auf diesem Standard aufgebaut. Während der Projektlaufzeit wurde mit der Version 3.0 eine neue Hauptversion veröffentlicht.

Insbesondere wurde hier die Schnittstelle hinsichtlich der Datentypen stark erweitert, was uns hinsichtlich der Verwendung des Standards auch im Projekt bestärkt hat. Durch binäre Datentypen wird der effizientere Austausch von komplexen Datenstrukturen erleichtert. Diese treten häufig in der Bus-Kommunikation zwischen Steuerungen und Peripheriegeräten wie Antrieben, Robotern oder Kamerasystemen auf. Da genau diese häufig das Vorbild für ein standardisiertes Verhaltensmodell sind, wie sie in B1 entwickelt wurden, war dies ein wichtiger Schritt für die praktische Anwendbarkeit der Technik.

Das in der ersten Projekthälfte erstellte SDK für Verhaltensmodellentwicklung wurde entsprechend im weiteren Projektverlauf weiterentwickelt, um kompatibel zur neuen Version 3.0 des FMI-Standards zu sein.

2.2 Verwaltungsschale / Submodelle

Die Asset Administration Shell (AAS) oder, auf Deutsch, Verwaltungsschale war von Beginn an ein in vielen Arbeitsgruppen diskutiertes Thema. In der frühern Projektphase wurde sich auch festgelegt, dass die Verwaltungsschale nach Möglichkeit das standardisierte Datenhaltungsformat kann und auch sein soll. Im Bereich der digitalen Zwillinge hat die Industrial Digital Twin Assoziation (IDTA) in der Zwischenzeit zahlreiche Submodelle spezifiziert, die teilweise öffentlich zugänglich, teilweise allerdings auch nur IDTA-Mitgliedern vorbehalten sind. Die HEITEC AG ist während der Projektlaufzeit Mitglied bei der IDTA geworden und hat direkten Zugang zu allen Modellen. Insbesondere das Submodell Simulation, welches in Version 1.0 freigegeben wurde bildet für uns eine Grundlage für die Referenzarchitektur von Simulationskomponentenmodellen. Weiterhin sind wir in der Arbeitsgruppe Automation Engineering gemeinsam mit weiteren Konsortialpartnern aktiv am Schaffen eines neuen Standardisierten Submodells zur durchgängigen Datenhaltung im Engineering.

Die Veröffentlichung der Spezifikation Version 3.0 im Juli 2023 und damit mitten in der Projektlaufzeit stellte für uns einen weiteren Schritt in Richtung Reife und industrielle Anwendbarkeit der Verwaltungsschale dar. Da wir in der ersten Projekthälfte noch keine konkreten Umsetzungen gestartet hatten, für welche die Datenhaltung schon fertig spezifiziert sein musste, traten hierdurch für uns keine besonderen Hemmnisse auf.

3 Wirtschaftliche und wissenschaftliche Verwertungsperspektiven nach Projektende

Wissenschaftliche und/oder technische Erfolgsaussichten nach Projektende:

Durchgängige Engineering Methoden:

Die erprobten Techniken zur Standardisierung und Generierung können nahtlos nach Projektende im hauseigenen Maschinenbau zur Anwendung gebracht werden. Die im Projekt geschaffenen Grundlagen werden nach Adaption für die Bedürfnisse des HEITEC Maschinenbaus zu einer Qualitäts- und Produktivitätssteigerung durch reduzierte Durchlaufzeiten führen. Hierbei helfen auch die standardisierten, unabhängigen Verhaltensmodelle, welche durch die, im Projekt erschaffene, Referenzarchitektur realisiert werden können. Diese Komponenten werden Simulationstoolunabhängig über verschiedene Niederlassungen und Kundenbranchen hinweg zum Einsatz gebracht werden können. Durch die interne vielfache Verwendung bei reduziertem Aufwand für das Engineering, wird eine Produktivitäts- und Effizienzsteigerung im Digitalen Engineering firmenweit erreicht. Dies verschafft HEITEC einen Wettbewerbsvorteil zu anderen Digitalen Engineering Dienstleistern. Weiterhin ermöglicht es den Marktzugang zu Komponentenherstellern die Beratungsleistungen bei der Erstellung unabhängiger Simulationskomponenten benötigen.

Lebensdauer Digitalen Zwilling:

Durch die im Projekt erprobten Techniken zur Steigerung der Durchgängigkeit, gerade im

Verbund mit der Technologie der Verwaltungsschale, lassen sich neue Geschäftsbereiche erschließen. Ein einmalig erstelltes oder teil-generiertes virtuelles Modell kann über verschiedene Lebensphasen einen Mehrwert liefern. Einstiegspunkt stellt der im Projekt entwickelte Produktkonfigurator dar, der im Vertriebsprozess einen erheblichen Mehrwert bietet. Dieser wird als Dienstleistung externen Kunden angeboten, weiterhin wird er für den eigenen Maschinenbau als Vertriebstool verwendet. Über die Wertschöpfung der VIBN-Phase hinaus, wird im Anschluss ein Digitaler Schatten angeboten werden können. Dieser verbindet das Simulationsmodell mit gesammelten IST-Daten der Anlage und kann dort zur Fehleranalyse und Optimierung eingesetzt werden. Auch hier kann die Lösung als Dienstleistung extern angeboten werden und zur Wertsteigerung im hauseigenen Maschinenbau genutzt werden.

Wissenschaftliche und wirtschaftliche Anschlussfähigkeit:

Die Firma HEITEC ist beginnend zum 01.01.2025 im Förderprojekt LLM-SE (Large Language Model unterstütztes Systems Engineering) vertreten. Förderer ist das Bayerische Staatsministerium für Wirtschaft, Landesentwicklung und Energie / VDI VDE IT. Die im SDM4FZI erlangten Erkenntnisse zu standardisierten Verhaltensmodellen, der automatischen Generierung von Digitalen Zwillingen und auch die automatisierte Testung der generierten Systeme können nahtlos in dieses Projekt eingebracht und vertieft werden.