Abschlussbericht

KMU-innovativ

Verbundprojekt KISEL

Kataloggestützte interdisziplinäre Entwurfsplattform für Elektrofahrzeuge

Teilvorhaben: Entwicklung einer Katalogplattform für Komponenten und Systemmodelle

Zuwendungsempfänger: TLK-Thermo GmbH	Förderkennzeichen: 16EMO0243K
Projektleiter: Dr.-Ing. Wilhelm Tegethoff	Tel.: +49 531 390 76 11 Fax: +49 531 390 76 29 E-Mail: w.tegethoff@tlk-thermo.com
Laufzeit des Vorhabens: 01.04.2017 – 31.03.2020	
Berichtszeitraum: 01.04.2017 – 31.03.2020	Datum: 30.09.2020

Inhaltsverzeichnis

I. Kurzdarstellung

1. Aufgabenstellung

Die Aufgabe des Teilvorhabens lag in der Entwicklung einer Katalogplattform für Simulationsmodelle und Systeme sowie der Schaffung einer Infrastruktur für Simulation, Variation, Optimiereranbindung und Management der Komponenten- und Systemmodelle. Folgende Ziele wurden dabei verfolgt:

- Ermöglichung der Modell- und Systemablage innerhalb der Katalogplattform
- Unterstützung unterschiedlicher Modellformate innerhalb der Katalogplattform
- Anbindung der Katalogplattform an unterschiedliche Simulatoren
- Implementierung einer Refactoringfunktion zum Versionsmanagement
- Schnittstellendefinition und Transformation über mathematisch-numerisch getriebene Modellinvertierung
- Analyse von Systemvarianten und Anbindung von Optimierern
- Bereitstellung von Meta-Informationen zur Unterstützung der Suchfunktionalität
- Aufbau eines batteriebetriebenen Elektrobusses als Demonstrator der Katalogplattform

Der hier vorliegende Abschlussbericht stellt die Bearbeitung der benannten Bereiche und die erzielten Ergebnisse dar.

2. Vorhabenvoraussetzungen

Zur erfolgreichen Bearbeitung der im Verbundprojekt KISEL gesteckten Ziele sind fundierte Kenntnisse auf dem Gebiet der Softwareprogrammierung, Simulationstechnik sowie der Modellierung thermischer Systeme notwendig. Die TLK-Thermo GmbH arbeitet auf ebendiesen Gebieten und bringt neben ihrer langjährigen Erfahrung auch eigens entwickelte Softwaretools mit, um die Voraussetzungen vollständig zu erfüllen.

3. Planung und Ablauf des Vorhabens

Die TLK-Thermo GmbH war im Verbundprojekt KISEL in allen Arbeitspaketen involviert. Diese wurden gemäß der Planung bearbeitet.

Das Förderprojekt KISEL wurde im Jahr 2016 geplant. Es wurden 11 Arbeitspakete formuliert. Dabei wurden die folgenden Meilensteine definiert, die sogleich das Projekt in die folgenden Abschnitte unterteilen:

- Ermöglichung der Modellablage innerhalb der Katalogplattform
- Entwurf einer Katalogplattform für Modelle und Systeme
- Bereitstellung von Systembeispielen und Bereitstellung der Grundfunktionalität eines Variantenmanagements innerhalb der Katalogplattform
- Bereitstellung einer Refactoring-Funktionalität für Modelle und Systeme
- Simulatorintegration und Simulation des Demontrators mit Variantenoptimierung

Die TLK-Thermo GmbH hat thermische Komponenten- und Systemmodelle zum Aufbau eines Demonstrators bereitgestellt und - wo es erforderlicher war - neu entwickelt. Diese Modelle wurden auf Basis von MongoDB in einer Datenbankstruktur überführt und über ein Web-Interface als Katalogplattform verfügbar gemacht. In enger Absprache mit den Projektpartnern wurden die für ein Versionsmanagement relevanten Informationen erarbeitet und über die Datenbank verfügbar gemacht. Das Versionsmanagement wurde mit in die Katalogplattform integrierten Systemen getestet und weiterentwickelt. In enger Zusammenarbeit mit den Projektteilnehmern wurden Schnittstellendefinitionen für Komponenten- und Systemmodelle zur Anbindung von unterschiedlichen Simulatoren innerhalb der Katalogplattform definiert und umgesetzt. Im Fokus lagen dabei Simulationen unter Dymola und Matlab/Simulink. Weiterhin wurden Funktionen zur Bereitstellung und Überführung von Modellen in verschiedenen Formaten entwickelt. Hier lag der Fokus auf die Formate Modelica, FMU und S-Function. Im Anschluss wurden die Modelle und Systeme mit zusätzlichen Meta-Informationen ausgestattet, um die Suchfunktionalität innerhalb der Katalogplattform zu verbessern. Weiterhin wurden Schnittstellen zur Anbindung von Optimierern implementiert und als Funktionalität in die Katalogplattform integriert. Damit wurde das Fitten an Messdaten sowie die dynamische Optimierung von Modellen und Systemen innerhalb der Katalogplattform ermöglicht. Die entwickelte Katalogplattform und ihre Funktionalitäten wurden am Beispiel des Gesamtsystems eines batteriebetriebenen Elektrobusses demonstriert.

4. Wissenschaftlicher und technischer Stand

Basierend auf dem aktuellen Stand wird die Katalogplattform in den nächsten Monaten in ein bestehendes TLK Softwareprodukt integriert werden. Die im Antrag formulierten Ziele sind erreicht worden. Zudem haben sich zahlreiche Anknüpfungspunkte für darüber hinausgehende Forschungen und Entwicklungen, wie beispielsweise die Topologieoptimierung ergeben.

5. Zusammenarbeit mit anderen Stellen

Neben der Zusammenarbeit mit den im Projekt beteiligten Partnern wurde an die in Aachen ansässige TLK Energy GmbH ein Unterauftrag vergeben. TLK Energy entwickelte ein Parametrierungswerkzeug mit Versuchsplanungsfunktion für nichtlineare, dynamische Modelle sowie dessen Integration in das Katalogsystem.

II. Eingehende Darstellung

Die in den Arbeitspaketen bearbeiteten Aufgaben und die erzielten Ergebnisse werden im Nachfolgenden detailliert dargestellt.

AP 1: Festlegung der Anforderungen und Grundkonzeption der Katalogplattform

AP 1.a: Anforderungen für die Architektur der Katalogplattform

Die im Verbundprojekt KISEL entwickelte interdisziplinäre Katalogplattform soll verschiedenen Anforderungen genügen und Funktionen bereitstellen (siehe Abbildung 1).

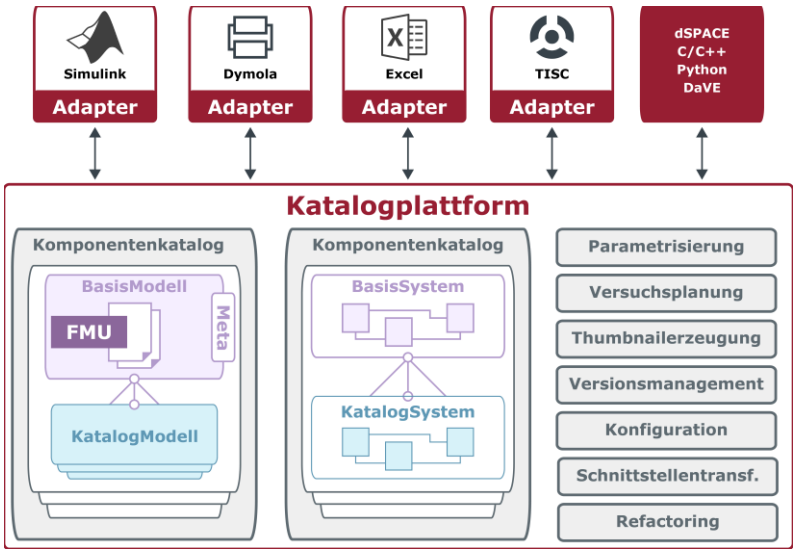

Abbildung 1: Funktionalitäten und geplanter Aufbau der Katalogplattform.

Diese sind:

- Anbindung an unterschiedliche Simulatoren (z. B. Simulink, Dymola, Excel, TISC etc.)
- Ablage von Modellen und ganzen Systemen
- Funktionen zum interdisziplinären Arbeiten mit Modellen aus dem Katalog:
  - Parametrisierung – manuell oder automatisiert über ein Fitting-Tool (ModelFitter)
  - Versuchsplanung (DOE)
  - Thumbnailerzeugung (Verbesserte Ergebnisauswahl der Suchfunktion)
  - Versionsmanagement (von Modellen und ganzen Katalogen)
  - Konfiguration (von Systemen aus Modellen des Katalogs)
  - Schnittstellentransformation (für die Analyse von Modellen)
  - Refactoring (z. B. für Modelländerung und Systemerstellung)

Elemente aus der Katalogplattform sollen dem Anwender zur Verfügung gestellt werden (vgl. Abbildung 2).

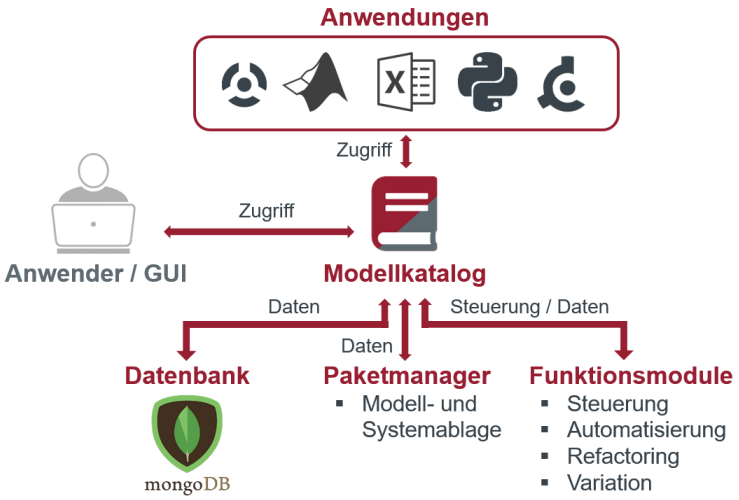

Abbildung 2: Übersicht der konzeptionierten Plattform-Architektur.

Der zentrale Modellkatalog soll als Datenbank verwaltet und der Zugriff auf Modelle des Katalogs über eine GUI realisiert werden. Im Hintergrund sollen die Modelle und Systeme über eine Art Paketmanager abgelegt werden. Weiterhin sollen Funktionsmodule als Services der Plattform angeboten werden. Des Weiteren soll der Zugriff auf Modelle unterschiedlicher Formate ermöglicht werden, um unterschiedliche Simulatoren (z. B. Simulink, Dymola, Excel, Python, TISC, DaVE etc.) zu bedienen.

Um die Anforderungen an die Katalogplattform und dessen Architektur zu konkretisieren, wurden in Zusammenarbeit mit den Projektpartnern verschiedene Use-Cases unterschiedlicher Nutzergruppen erarbeitet (vgl. Abbildung 3).

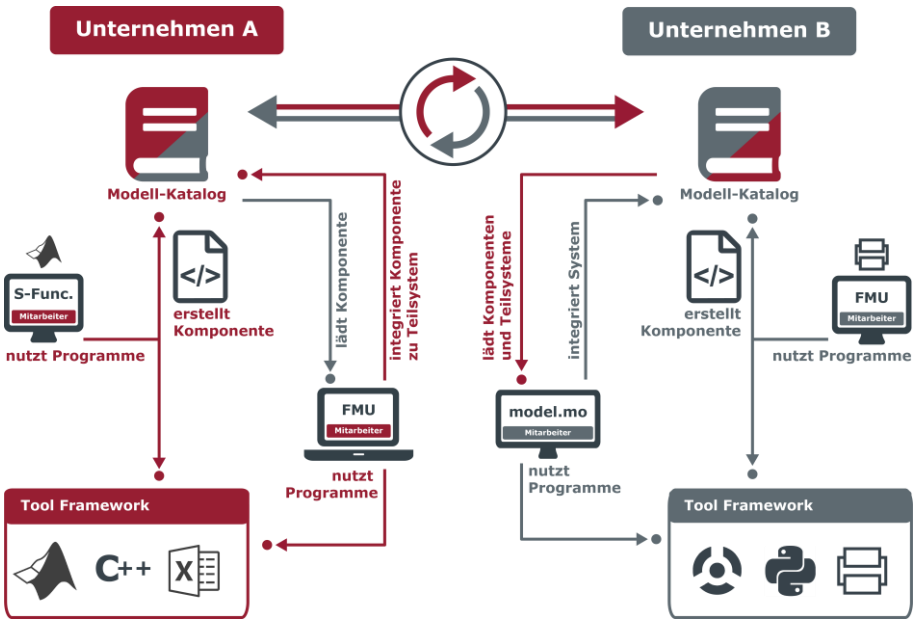

Abbildung 3: Exemplarisches Anwendungsszenario der Katalogplattform.

Des Weiteren gaben die Use-Cases Aufschluss über die technische Umsetzung der Ablage von Modell- und Systemvarianten sowie benötigter Schnittstellen. Weiterhin wurden IT-Anforderungen potenzieller Kunden eingeholt und in den Planungsprozess integriert. Es konnten folgende Anforderungen an die Plattform festgehalten werden:

- Zugriff über Internet
- Synchronisation von Kataloginformationen (online/offline Modus)
- Versionierung von einzelnen Modellen und ganzen Modellkatalogen innerhalb der Plattform
- Schnittstellendefinitionen bzgl. Plattforminteraktion und Modellentwicklung

Die Anforderungen, die sich durch die Schnittstellenformulierung für die Plattforminteraktion erfüllt werden müssen, sind:

- Zugangssicherheit (Verschlüsselung)
- Benutzerverwaltung (Zugangsberechtigung)
- Lizensierung von Inhalten der Plattform
- Offline-Zugriff auf Inhalte der Katalogplattform

Die modellbezogene Schnittstellendefinitionen müssen den folgenden Anforderungen genügen:

- Entwicklung von Systemen auf Basis von Modellen
- Modellaustausch in verschiedenen Modellformaten
- Anbindung von Simulatoren und Optimierern (Interoperabilität)
- Integration von Modell- und Systeminformationen (Parametersätze, Messdaten, Versuchspläne etc.)

AP 2: Umsetzung der Katalogplattform

Die Entwicklung der Katalogplattform ist das zentrale Arbeitspaket im Projekt KISEL. In diesem Abschnitt werden die Arbeiten an der grundlegenden Struktur der Katalogplattform beschrieben. Die Arbeiten an den darauf aufbauenden Arbeitspaketen werden in den jeweiligen Kapiteln beschrieben.

AP 2.a: Strukturelle Umsetzung und Integration auf Modellebene

Für die Umsetzung der Modellablage wurde auf die vom ISSE entworfene Architektur aufgesetzt. Es können Basismodelle und davon abgeleitete konkrete Modelle in unterschiedlichen Formaten abgelegt werden. Weiterhin können hier modellbezogene Zusatzinformationen (z. B. zugrundeliegende Mess- oder CFD-Simulationsergebnisse) hinterlegt werden.

Die dokumentenbasierte Datenbank (MongoDB) ist in der Lage, die Struktur der Ablage sowie alle grundlegenden Modellinformationen (Modellname, Modellformat, zugehöriges Basismodell etc.) der Plattform zur Verfügung zu stellen. Mittels JSON-Dateien wurden Schemata für Komponententypen (Verdichter, Lüfter, etc.) entworfen, mit denen einzelne Komponententypen im Katalog beschrieben werden können (vgl. Abbildung 4).


```
"_id": "5d84df14d70476cc005ba296",
"f_datatable": null,
"transcritical": true,
"nominal capacity": null,
"nominalEvaporatingTemperature": 268.15,
"nominalCondensingTemperature": null,
"nominalCOP": 3.11,
"displacement": 0.0001379310344827586,
"fixedSpeed": true,
"drive": null,
"speed": null,
"confidential": null,
"nominalCapacity": 37700,
"vendor": "Bitzer",
"name": "4HTC-20K",
"component_type": "Compressor",
"description": " Semi-hermetic Reciprocating Compressors",
"fluid": "R744",
"nominalVoltage": 420,
"maxSpeed": 24.16666666666667,
"minSpeed": 24.16666666666667,
"model_refs": [
  {
    "$oid": "5e2efdb1c61081558451548b"
  },
  {
    "$oid": "5e2efdb1c61081558451548c"
  }
]
```


Abbildung 4: Darstellung der Datenbankinformationen eines exemplarischen halbhermetischen Kältemittelverdichters.

Für die Erstellung dieser Schemata ist thermodynamisches Fachwissen über die jeweiligen Komponenten erforderlich, um die jeweiligen notwendigen Attribute (z. B. Spannungsversorgung, Drehzahlbereich, etc.) zu definieren. Über einzelne Collections für komponentenbezogene Typendefinitionen kann die Datenbank Informationen bzgl. Name, Format, zugehöriges Basismodelle sowie Kommentare anlegen.

Weiterhin sind Verweise auf andere Dateien implementiert, die Informationen über Hash (eindeutige Identifikation innerhalb der Datenbank) und Datum liefern.

Um die Datenbank von Weboberfläche und den Interfaces (z. B. Python) zu entkoppeln und zusätzliche Funktionalitäten bereitstellen zu können, wurde eine Middleware (JS-Express) implementiert.

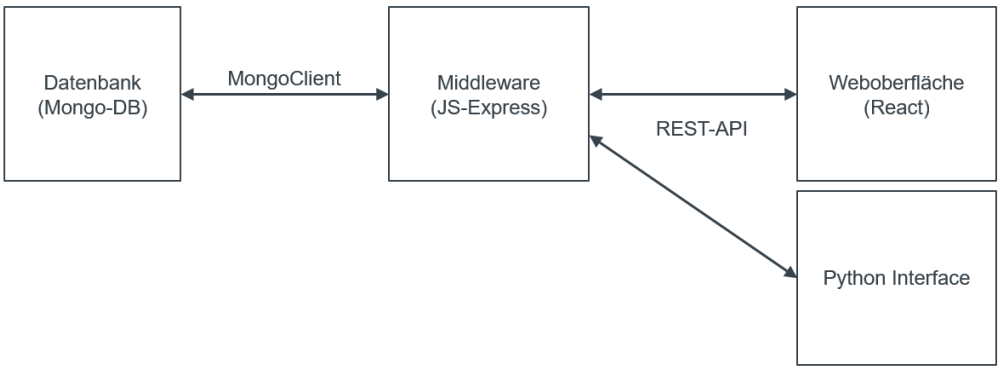

Abbildung 5: Middleware zur Umsetzung von Weboberfläche und Python Interface.

Mit Einführung der Middleware wird der Zugang für die auf Basis von React und Redux erstellten Weboberfläche (http: PUT, PUSH) ermöglicht.

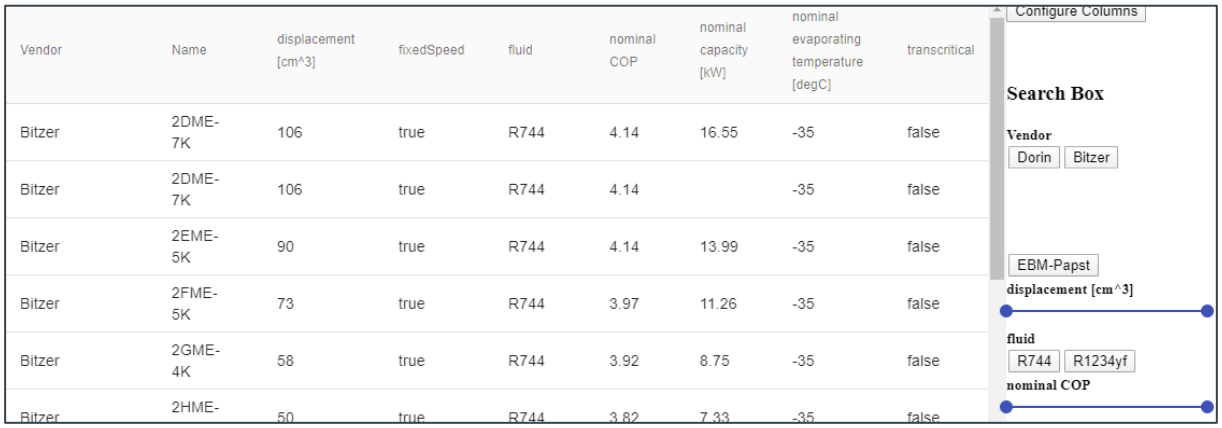

Abbildung 6: Ausschnitt der Weboberfläche mit Auswahl an auswählbaren Kältemittelverdichtern der Katalogplattform.

Innerhalb der Weboberfläche ist ebenso die Suche nach im Katalog vorhandenen Modellen anhand vorgegebener Kriterien (hier z. B. Verdrängungsvolumen und Kältemittel) möglich.

Weiterhin werden Informationen über die Elemente über eine JSON-Datei der Datenbank bereitgestellt:

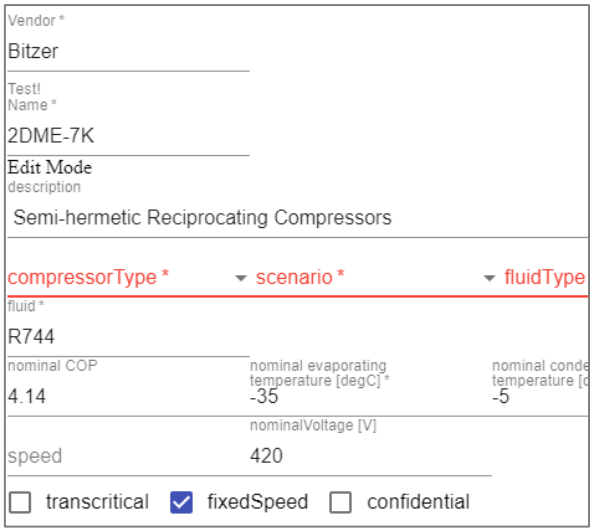

Abbildung 7: Informationen der Katalogplattform über einen ausgewählten Kältemittelverdichter.

Die Middleware stellt weiterhin diverse Funktionalitäten bereit, wie z. B.:

- Dokumentenvalidierung
- File Storage
- Backup
- Benutzerauthentifizierung

Die Dokumentenvalidierung überprüft, ob eine Komponente mit ihrer Typendefinition übereinstimmt. Über die Funktionalität „File Storage“ werden Dateien über ihren Hash als eindeutiger Dateiname in der Datenbank gespeichert. Die Funktion „Backup“ bezieht die Struktur der Datenbank als auch ihre Inhalte mit ein. Die Benutzerauthentifikation ermöglicht einen sicheren Zugang zu der Datenbank. Weiterhin ermöglicht die Middleware eine Anbindung an verschiedene Interfaces.

Für die Entwicklung einer universellen Schnittstelle zur Kommunikation zwischen einem Modell der Plattform und einer außerhalb der Plattform verwalteten Software wurde eine universelle Python-Schnittstelle entwickelt. Über diese ist das Aufrufen und Simulieren von Modellen möglich. Weiterhin können sämtliche unter Python verfügbare Bibliotheken eingebunden werden (vgl. Abbildung 8).

```
#load simulator module
from TLKSimulator import *
sim = Simulator()

#load the model
sim.loadModel("Sine.fmu")

#================================================================================

#help(sim.set)
sim.set({"fmu.freqHz":0.25, "fmu.amplitude":0.25, "fmu.offset":0.25})
#help(sim.simulate)
res = sim.simulate(0.25,4)

#================================================================================

import pylab as P
P.plot(res["time"],res["fmu.y"])
P.show()
```

Abbildung 8: Parametrierung, Simulation und Visualisierung einer FMU als Anwendungsbeispiel der Python-Schnittstelle.

Durch die Skriptbarkeit der Schnittstelle in Python ist es möglich, neben der reinen Simulation auch andere Anwendungsfälle zu bedienen. So kann beispielsweise die entwickelte Optimiererschnittstelle genutzt werden, um ein Modell mit Hilfe einer zu definierenden Kostenfunktion zu optimieren. Das könnte z. B. auch das Fitten von Modellparametern an Messdaten umfassen. Über diese Optimiererschnittstelle ist es möglich, nahezu jeden Optimierungsalgorithmus mit dem Modell zu koppeln und eine Optimierung durchzuführen. Exemplarisch seien hier der Nelder-Mead-Algorithmus, SLSQP sowie Least-Squares-Algorithmen wie der Levenberg-Marquardt-Algorithmus aufgeführt. Einige dieser Algorithmen wurden von TLK zur Steigerung der Recheneffizienz angepasst bzw. erweitert.

Ebenfalls ist es durch die skriptbare Python-Schnittstelle möglich, ein Modell in einem (weichen) Echtzeitkontext zu simulieren. Dadurch kann sowohl der Einfluss von Parametern in Echtzeit dargestellt als auch ein Modell an einen Prüfstand o. ä. angebunden werden. Die für diesen Fall entwickelte Schnittstelle zu LabVIEW wird in AP 9 beschrieben.

Um die Schnittstellenfunktionalität zu beschreiben, wird exemplarisch das Fitten eines Modells an vorliegende Messdaten beschrieben. Hierbei wird auf die Funktionen innerhalb der von TLK-Thermo entwickelten Software ModelFitter zurückgegriffen. Der ModelFitter ist in der Lage das Fitting eines Modells mit zugrundeliegenden Daten (z. B. Messdaten) vorzunehmen (vgl. Abbildung 9).

Auf der Grundlage von eigens definierten modellbezogenen abhängigen und unabhängigen Variablen versucht der ModelFitter einen Parametersatz zu finden, der ein definiertes Fitting Target mit einer festgelegten Varianz erfüllt. Damit wird innerhalb der Katalogplattform die Möglichkeit geschaffen, dass auf der Grundlage von existierenden Messdaten aus einem Basismodell ein an diese Messdaten angepasstes konkretes Modell wird. Das neue Modell mit dem entsprechenden Parametersatz sowie der zugehörigen statistischen Auswertung werden ebenso in der Baumstruktur abgelegt und der Datenbank verfügbar gemacht.

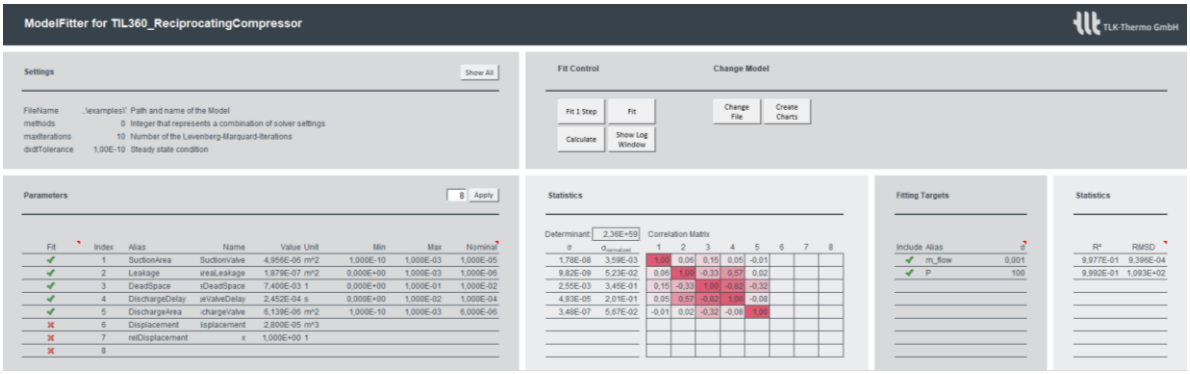

Abbildung 9: Excel-Oberfläche des ModelFitters.

Die implementierte Python-Schnittstelle der Katalogplattform ist in der Lage, alle für den ModelFitter notwendigen Funktionen und Informationen innerhalb einer Konfigurationsdatei (JSON) zur Verfügung zu stellen. Dies sind:

- Zugriff auf den Katalog
- Suche von Modellen über Attribute
- Herunterladen von Modellen
- Hochladen von Modellen
- Skriptbasierte Eingabe von Modellen aus Dritt-Datenbanken

Damit kann der ModelFitter aus der Katalogplattform heraus aufgerufen und zum Parametrieren von Modellen und dem Fitten an Messdaten genutzt werden.

Die Schnittstellendefinition kann die Konfigurationsdatei (Beschreibung des Fitting-Problems) schreiben und innerhalb der Datenbank ablegen. Dadurch wird die Reproduzierbarkeit der Ergebisse gewährleistet. Weiterhin wird damit die Basis für die in AP 8 beschriebene Refactoring-Funktionalität gelegt und die Aktualisierung bzw. Neuerstellung von Modellen ermöglicht.

Weiterhin wurde eine Funktion entwickelt, um das Fitting automatisiert ablaufen zu lassen. Hierfür wurde eine Steuerungssoftware erstellt, die den jeweiligen Optimierer aufrufen kann. Diese Funktion greift dann auf die anzupassenden FMUs innerhalb der Plattform zu. Für jeden Fitting-Durchgang wird eine Log-Datei erzeugt und innerhalb der Plattform gespeichert. Die Ergebnisse (Parametersätze) des Fittings werden zusätzlich in einer Excel-Datei gespeichert.

AP 2.b: Strukturelle Umsetzung und Integration auf Systemebene

Die Untersuchungen an den strukturellen Umsetzungen basierend auf den Anforderungen haben die Vorteile einer weitgehend gleichen Behandlung von Komponenten- und Systemmodellen verdeutlicht. Die Arbeiten der strukturellen Umsetzung wurden somit bereits vollständig im AP 2.a dargestellt.

AP 2.c: Interaktion von Modell- und Systemebene

Die in der Katalogplattform hinterlegten Modelle können zu Systemen zusammengeführt werden. Die für die Zusammenführung benötigten Schnittstellen-Informationen werden von jedem einzelnen Modell an die Datenbank über separate Dateien übergeben. Aus Modellen erstellte Systeme können in der Katalogplattform hinterlegt und verwaltet werden. Eine ausführliche Darstellung der Beschreibung von Systemvariationen innerhalb der Plattform geschieht in AP 6.

Der Austausch von Modellen innerhalb eines Systems wird über die vom Basismodell festgelegten Schnittstellen ermöglicht. Auf Komponentenebene wurden physikalische und informationstechnische Konnektoren definiert, um einen Modellaustausch zu ermöglichen. Für die physikalische Modellierung sind dies z. B. Informationen über den Energie- und Stofftransport einer Komponente. Das Modell einer Komponente (z. B. Lüfter) kann dann durch das Modell einer anderen Komponente ausgetauscht werden, wenn diese vom gleichen Typ ist und somit Basismodell und Schnittstelleninformationen übereinstimmen. Die in der Katalogplattform hinterlegte Systematik wird anhand des folgenden Anwendungsszenarios beschrieben:

Anwendungsszenario der Interaktion zwischen Modell- und Systemebene:
Modelle können aus unterschiedlichen (Modell-)Katalogen stammen. So kann z. B. Anwender A Entwickler für Kältemittelverdichter und Anwender B Entwickler von Wärmepumpensystemen sein. Anwender B bedient sich unterschiedlicher Kataloge aus der Katalogplattform, um unterschiedliche Systeme für Wärmepumpen zu erstellen. So z. B. auch aus dem Verdichter-Katalog von Anwender A. Wird nun ein Modell eines Kältemittelverdichters aus dem Katalog von Anwender A, z. B. im Zuge einer Modellentwicklung oder einer Fehlerbehebung verändert, muss eine neue Version des Modells erzeugt und in den Katalog aufgenommen werden. Um die Reproduktion der Simulationsergebnisse von Anwender B zu bewahren, darf es zu keinem automatischen „Update“ des betreffenden Modells kommen. Anwender B muss auf Basis von Änderungsinformationen von A selber entscheiden, ob er das Modell innerhalb seines Systems aktualisieren möchte. Da Anwender A in den Änderungsinformationen eine Aktualisierungsempfehlung angegeben hat (weil z. B. ein Parameter des ursprünglichen Modells falsch bedatet wurde), wird Anwender B einer Aktualisierung zustimmen.

Eine neue Systemtopologie kann auch auf der Grundlage eines Basissystems erstellt werden. Ein Basissystem stellt eine Schablone eines Systems, z. B. einer Wärmepumpe dar, welches ausschließlich aus Basismodellen besteht. Jedes konkrete System kann also auf ein Basissystem zurückgeführt werden. Eine ausführliche Erläuterung der Erzeugung von Systemen geschieht in AP 6.

AP 3: Einstellen vorhandener Modelle in den Komponentenkatalog

AP 3.a: Bereitstellung von Modellen

Für die Vorbereitung des Katalog-Prototyps wurden Modelle aus den Bereichen Lüfter, Verdichter und Pumpen der Katalogplattform hinzugefügt. Einige Modelle mussten den Anforderungen entsprechend angepasst werden. Für die späteren Demonstratoren wurden ebenfalls Modelle erstellt. Die Modelle wurden in ihren ursprünglichen Modellformaten (z. B. Modelica, csv-Datei) sowie im standardisierten FMU-Format in die Datenbank integriert. Alle Modelle wurden über JSON-Dateien in die Datenbank integriert und über eine Ordnerstruktur abgelegt.

AP 3.b: Konvertierungsmethoden und Ablage

Zur Anwendung innerhalb der Katalogplattform wurde eine in Python umgesetzte Konvertierungsmethode für Modelle entwickelt. Die Modelle, die z. B. für die Demonstratoren eingesetzt werden, sollen unabhängig vom Simulator eingesetzt werden können. Hierfür werden diese in das standardisierte FMU-Format konvertiert. Informationen über die so erzeugten Modelle werden in Form einer JSON-Datei in die Datenbank eingestellt und somit der Plattform verfügbar gemacht. Die Einträge in die Datenbank werden dabei von der Middleware übernommen.

Um eine größere Anzahl von Modellen oder ganzen Modell-Bibliotheken zeiteffizient in FMUs zu überführen, wurden entsprechende Programme erstellt. Diese Programme erlauben es,

- alle Modelle einer Bibliothek zu erfassen,
- das Modell den entsprechenden Simulatoren (z. B. Matlab oder Dymola) zuzuweisen,
- den jeweiligen Simulator automatisiert zu starten und den FMU-Export des Modells durchzuführen,
- die exportierten FMUs zusammen mit den Originalmodellen in einer gegebenen Ordnerstruktur wieder abzulegen.

Bei Bedarf kann ein automatisierter Arbeitsablauf den Abgleich der Simulationsergebnisse der exportierten FMU mit denen des Original-Modells anhand von gegebenen Randbedingungen oder Testmodellen durchführen. Der Anwender kann anhand einer automatisch generierten Übersicht erkennen, ob alle bereitgestellten Modelle den Plausibilitätstest bestanden haben. Die Automatisierung dieses Prozesses mit den erzeugten Programmen sorgt für eine erhebliche Zeitersparnis bei gleichzeitiger Sicherstellung der Qualität beim Erstellen und Validieren der simulatorunabhängigen Modellbibliotheken.

Mit dieser in die Katalogplattform integrierten Methode können somit Modelle automatisiert in das FMU-Format überführt werden. Zur Veranschaulichung dieser Funktionalität wurden zwei Anwendungsszenarien auf der Basis der Modellerstellung für Kältemittelverdichter betrachtet. Die beiden Szenarien behandeln die automatisierte Modellerstellung auf Basis von:

1. Parametersätzen (physikalische Informationen)
2. Herstellerdaten (Polynom)

Für das **Szenario 1** wurde angenommen, dass die Parameter in Form einer Excel-Datei sowie ein entsprechendes Basismodell (z. B. Dymola-Modell) vorliegen. Über die Plattform wird das Basismodell eines Kältemittelverdichters ausgewählt. In einem weiteren Schritt wird der Parametersatz ausgewählt. Über eine Python-Schnittstelle werden nun die Parameter z. B. über eine Skript-Datei (.mos) an das Basis-Dymola-Modell übergeben. In einem weiteren Schritt wird die FMU-Export-Funktion innerhalb von Dymola aufgerufen. Diese erzeugt ein Modell des jeweiligen Verdichters im standardisierten FMU-Format. Das Modell wird anschließend in die Katalogstruktur gespeichert und Informationen innerhalb einer JSON-Datei in die Datenbank eingepflegt.

Das **Szenario 2** geht von Polynomen aus, die seitens der Hersteller angegeben wurden. Im Bereich der Kältemittelverdichter werden üblicherweise 10 Koeffizienten definiert, die über ein Polynom das physikalische Verhalten der realen Komponente abbilden sollen. Um jedoch aus polynomgestützten Modellinformationen physikalische Modelle automatisiert zu generieren, kann die in die Plattform integrierte Fitting-Funktionalität genutzt werden. Im ersten Schritt werden FMUs nach Szenario 1 erzeugt. Diese Modelle können anschließend aus der Plattform heraus in den ModelFitter importiert werden (vgl. Abbildung 10).

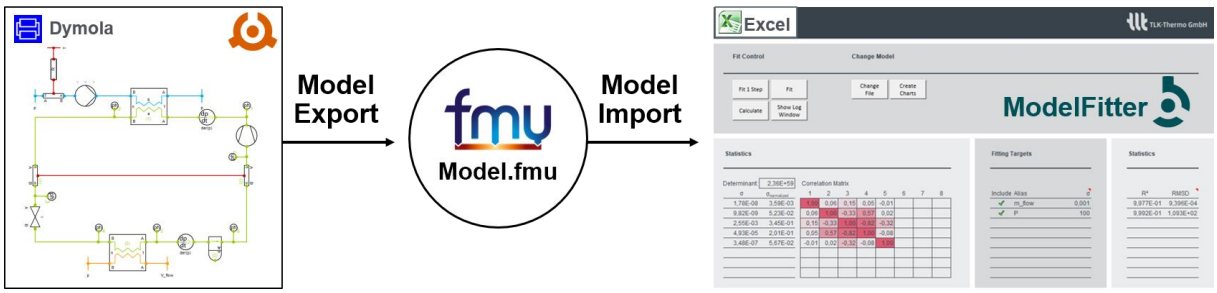

Abbildung 10: Funktion der Katalogplattform – Modell-Export aus Dymola in das FMU-Format und Import in den ModelFitter.

Der ModelFitter bekommt die Informationen über den Parametersatz sowie des zu verwendenden Dymola-Basis-Modells in Form einer JSON-Datei übergeben. Über die Simulationsergebnisse des polynombasierten Modells werden die Parameter des physikalischen Basismodells gefittet (vgl. Abbildung 11).

```
model Bitzer_2EME_5K
  extends TIL.VLEFluidComponents.Compressors.ReciprocatingCompressor(
    final displacement=8.97701149425287E-05,
    final areaSuctionValve=0.001,
    final areaLeakage=5.00301973084094E-07,
    final relativeDeadSpace=0.02758123184447,
    final dischargeValveDelay=5.94864794814054E-07,
    final areaDischargeValve=2.15557380688252E-04,
    final frictionCoefficient=20,
    final frictionCoefficientPiston=0.724139635559826);
end Bitzer_2EME_5K;
```

Abbildung 11: Exemplarisches Ergebnis des ModelFitters: In die Katalogplattform abgelegter Parametersatz des physikalischen Modells eines Kältemittelverdichters der Firma Bitzer.

Über diesen Prozess können physikalische Modelle von Kältemittelverdichtern auf der Basis von Herstellerangaben automatisiert erzeugt in die Datenbank integriert und somit der Katalogplattform verfügbar gemacht werden.

AP 4: Entwicklung einer optimalen Versuchsplanung für nichtlineare dynamische Modelle

AP 4.a: Schnittstellenentwicklung

Die mathematische Beschreibung eines Simulationsmodells innerhalb der Schnittstelle erfolgt in Form eines hybriden Algebro-Differentialgleichungssystems bzw. DAE („Differential algebraic equation“) (vgl. Abbildung 12).

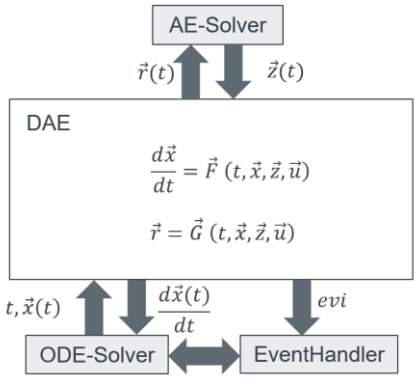

Abbildung 12: Schematische Darstellung der mathematischen Zusammenhänge innerhalb der Schnittstelle.

Die gewählte Darstellung ermöglicht die Modellierung physikalischer Sachverhalte inklusive mathematischer Unstetigkeiten (Events). Durch die zeitliche Integration des Modells können dynamische Zeitverläufe bestimmt werden. Hierfür werden Löser aus der Sundials-Bibliothek sowie ein mehrdimensionaler Newtonlöser zur Nullstellensuche verwendet.

Mittels dieser Schnittstelle können Optimierer das Modell beispielsweise hinsichtlich einer Kostenfunktion optimieren oder Modellparameter an Messdaten anfitten. Dieses Verfahren wird u. a. mit dem ModelFitter angewendet (vgl. AP 2). Über die Optimierungsschnittstelle ist es möglich, nahezu jeden Optimierungsalgorithmus mit dem Modell zu koppeln und eine Optimierung durchzuführen. Exemplarisch genannt seien hier der Nelder-Mead-Algorithmus, SLSQP sowie Least-Squares-Algorithmen wie der Levenberg-Marquardt-Algorithmus. Einige dieser Algorithmen wurden von TLK zur Steigerung der Effizienz angepasst bzw. erweitert.

Ebenfalls ist es durch die skriptbare Schnittstelle möglich, ein Modell in einem (weichen) Echtzeitkontext zu simulieren. Dadurch kann sowohl der Einfluss von Parametern in Echtzeit dargestellt als auch ein Modell an einen Prüfstand o. ä. angebunden werden. Die Skriptbarkeit der Schnittstelle in Python kann ebenfalls für die Anbindung von Simulatoren verwendet werden (vgl. AP9).

AP 4.b: Entwicklung Parametrisierungswerkzeug

Für nichtlineare dynamische Modelle wurden Algorithmen zur optimalen Versuchsplanung entwickelt und implementiert. Als Schnittstelle zwischen Nutzer oder anderen Programmen und den Algorithmen wurde eine objektorientierte Python-Bibliothek (DOE-Tool) entwickelt (vgl. Abbildung 13).

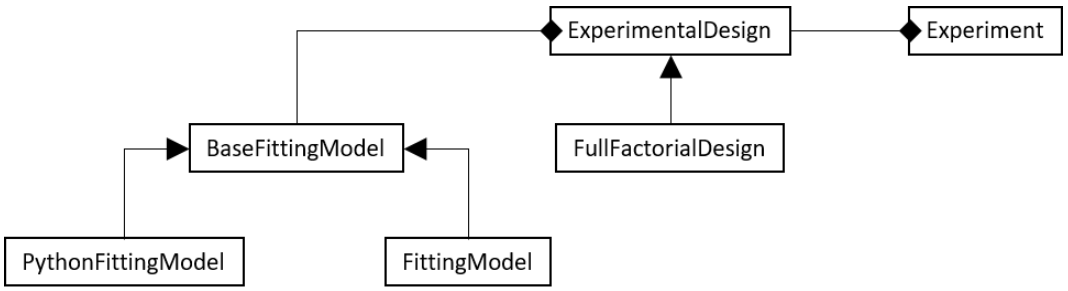

Abbildung 13: Klassenstrukturdiagramm des DOE-Tool-Python-Pakets.

Das Simulationsmodell wird durch eine Klasse repräsentiert und kann entweder direkt als Python-Funktion implementiert oder über den FMI Standard importiert werden; wobei als Python-Funktion nur statische Modelle unterstützt werden. Für dynamische Modelle muss das standardisierte FMI-Format verwendet werden.

Der eigentliche Versuchsplan wird ebenfalls durch eine Klasse repräsentiert. Darin sind neben dem zugehörigen Simulationsmodell alle während eines Experiments anzufahrenden Betriebspunkte gespeichert. Außerdem sind in dieser Klasse alle Methoden zur Berechnung der notwendigen statistischen Informationen implementiert. Für diese Berechnungen wird auf die in den AP 7 entwickelten Methoden zur Berechnung von stationären Arbeitspunkten zurückgegriffen.

Für die Optimierung eines Versuchsplans wurde ein Austauschalgorithmus implementiert. Dabei wird aus einem großen Versuchsplan mit sehr vielen Punkten eine vorgegebene Anzahl an Punkten ausgewählt, die den maximalen Informationsgehalt für die spätere Parameteridentifikation liefern. Optional können Wiederholungen erlaubt werden. Der Algorithmus wurde mit einfachen Modellen, deren optimale Lösung bekannt ist, validiert.

Für die Visualisierung von Versuchsplänen wurden Plot-Routinen in Python implementiert. Die Darstellung der Punkte im Versuchsraum erfolgt zum Beispiel über eine Projektion der Punkte in die von jeweils zwei Freiheitsgraden aufgespannten Ebenen (vgl. Abbildung 14).

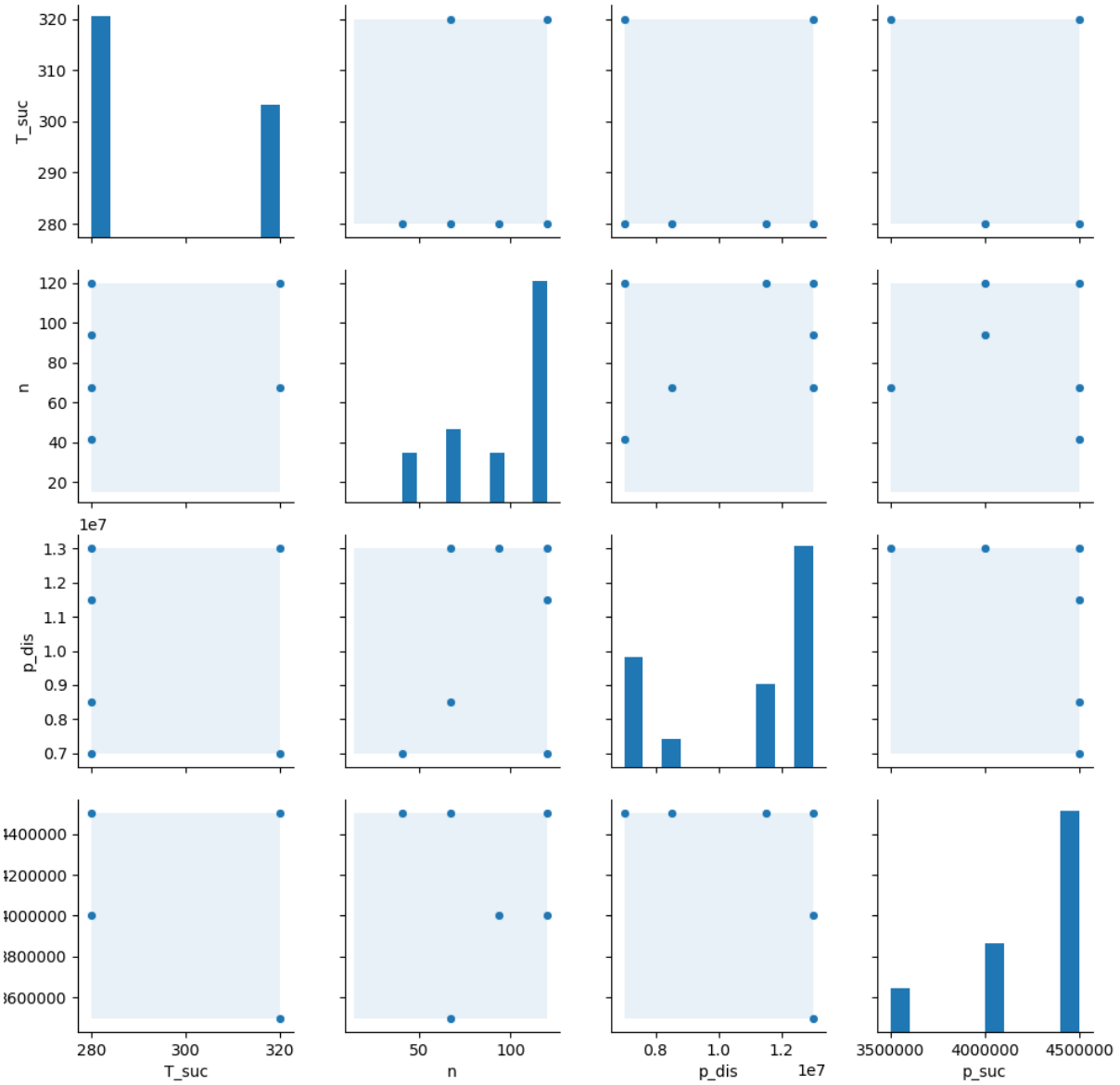

Abbildung 14: Optimaler Versuchsplan für das verlustbasierte Modell eines Kältemittelverdichters.

Als reale Anwendung wurde der Versuchsplan für einen Kältemittelverdichter optimiert. Die Freiheitsgrade bei den Messungen sind:

- Temperatur des angesaugten Kältemittels (T_suc)
- Verdichterdrehzahl (n)
- Hochdruck (p_dis)
- Saugdruck (p_suc)

Der Versuchsraum ist durch obere und untere Grenzen für alle Freiheitsgrade definiert. Aus einem vollfaktoriellen Versuchsplan, der den kompletten Versuchsraum abdeckt, werden 16 Punkte bestimmt, mit

denen die unbekannten Parameter am sichersten geschätzt werden können. Das Ergebnis ist in Abbildung 14 dargestellt. Man erkennt, dass die optimalen Punkte eher am Rand des Versuchsraums liegen und vor allem Punkte mit hoher Drehzahl und hohen Saugdrücken bevorzugt werden.

Die Daten durchgeführter Messungen werden in einer eigenen Klasse beschrieben. In dieser Klasse sind auch Methoden zur Parameteridentifikation mit einem spezialisierten Least-Squares-Optimierer enthalten. So werden die folgenden, in der Praxis üblichen mehrstufigen Prozesse unterstützt:

- Versuchsplan mit wenig Punkten erstellen (Screening)
- Messungen durchführen
- Parameter schätzen
- Zweiter Versuchsplan unter Berücksichtigung der neuen Parameterschätzungen und der durchgeführten Messpunkte
- Messungen durchführen
- Parameter schätzen

AP 5: Einstellung vorhandener und Erstellung neuer interdisziplinärer Systembeispiele

Am Beispiel eines Elektro-Omnibusses wurde eine Systemschablone entwickelt, anhand derer Komponenten, die als variative Bereiche beschrieben wurden, ausgetauscht werden können. Über die Modellablage können diese durch andere Komponentenmodelle ausgetauscht und damit neue Systemvarianten erzeugt werden. Das Schablonenlayout wurde für den Zugriff von ausgewählten Optimierern vorbereitet, um Untersuchungen der Systemverschaltungsvariation zu ermöglichen.

AP 5.a: Bereitstellung vorhandener Systemmodelle

Am Beispiel eines batterieelektrischen Stadtbusses wurde eine Schablone über ein BUS-System bereitgestellt, über die verschiedene Systemmodelle erstellt werden können.

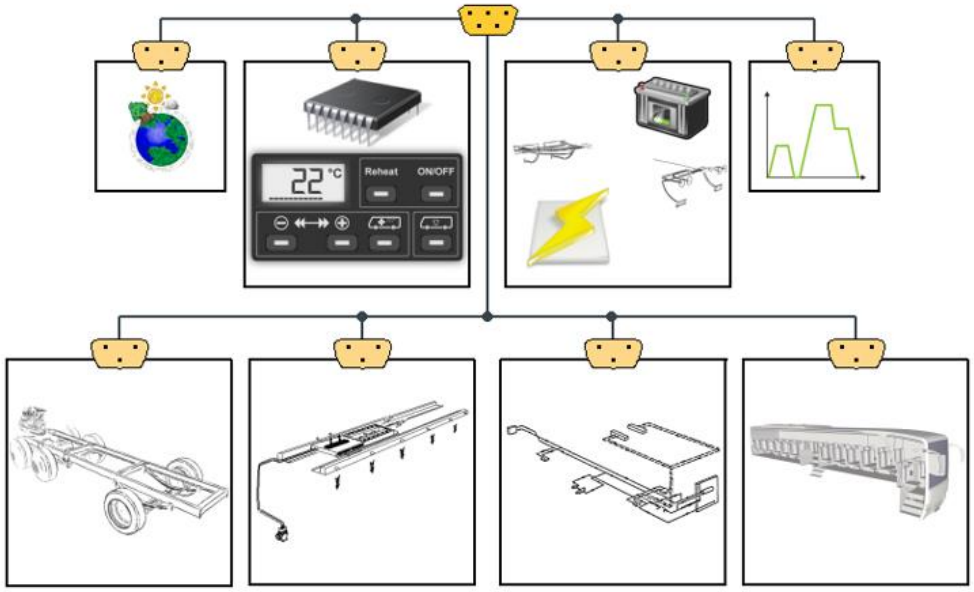

Abbildung 15: Systemschaubild eines batteriebetriebenen Stadtbusses mit BUS-Struktur.

Die bereitgestellte Schablone bietet Platzhalter für Modelle von:

- Umgebung (Temperatur, Luftfeuchtigkeit, Solarstrahlung, etc.)
- Regler (z. B. für Klimaanlage, Motor, etc.)
- Elektrische Komponenten für Nieder- und Hochvoltseite
- Fahrprofil
- Längsdynamik
- Kältekreis
- Wasserkreis
- Innenraum

AP 5.b: Bereitstellung von Systemmodellen mit variativen Bereichen

Für die Erzeugung von Systemvarianten zur Topologieoptimierung wurde die in AP 5 beschriebene Schablone eines Elektrobusses verwendet und variante Bereiche definiert (vgl. Abbildung 16).

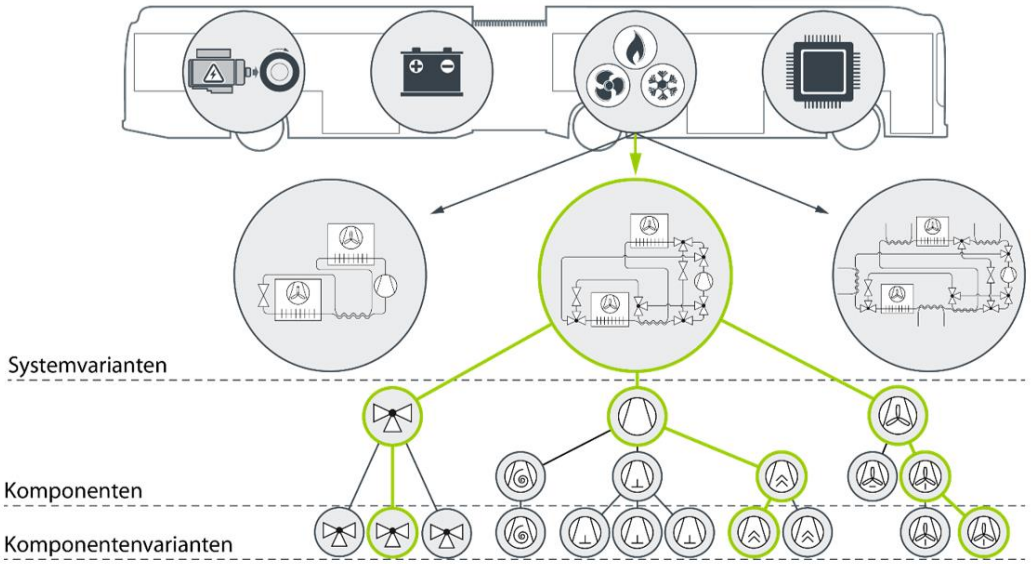

Abbildung 16: Variative Bereiche des untersuchten Gesamtmodells eines batteriebetriebenen Elektrobusses.

Es entstanden insgesamt 5 Bereiche:

- Antriebsstrang
- Batterie
- Heiz- und Kühlsystem
- Regelung
- Innenraum

Das Schablonenlayout bildet das Systemdesign des Elektrobusses und definiert variante Subsysteme. Diese sind über die Plattform austauschbar. Auf die Modelle des Systems kann über die Python-Schnittstelle von ausgewählten Optimierern zugegriffen werden. Dies dient als Vorbereitung, um Untersuchungen der Systemverschaltungsvariation zu ermöglichen.

AP 6: Entwicklung eines Variationsmanagements von Systemen und Subsystembeschreibung

AP 6.a: Umsetzung von Variantenmanagement

Zur Verwaltung der Varianten werden Dateien im Modelica-Format für die jeweilige Verschaltung sowie ein flacher Graph automatisch generiert (vgl. Abbildung 17).

```
within MyPackage;
Model System_8f9fc8c6-4933-41d3-9080-a61bfc6d0ef9
        valve_specialization1 v1;
        sep_specialization1 condrec_1_sep;
        hxfl_specialization1 xor_chiller_1_hx;
        compressor_specialization1 c1;
        hxfg_specialization1 condrec_1_hx2;
        dp_specialization1 condrec_1_dp1;
equation
        connect(condrec_1_dp1.B, condrec_1_sep.A);
        connect(condrec_1_hx2.BF, c1.B);
        connect(condrec_1_hx1.BF, condrec_1_dp1.A);
        connect(condrec_1_sep.B, condrec_1_hx2.AF);
        connect(v1.A, condrec_1_hx1.AF);
        connect(c1.A, xor_chiller_1_hx.BF);
        connect(xor_chiller_1_hx.AF, v1.B);
end Model System_8f9fc8c6-4933-41d3-9080-a61bfc6d0ef9;
```

Abbildung 17: Verwaltung einer Systemverschaltung innerhalb der Datenbank über einen flachen Graph.

Hierüber lassen sich Informationen über den eindeutigen Namen (Hash), die verwendeten Modelle und deren Schnittstellen festhalten. Die Zuordnung der von Systemverschaltungen abgeleiteten Varianten geschieht über JSON-Dateien, welche von der Datenbank interpretiert werden können. In diesen Dateien befinden sich u. a. Informationen über das zugrundeliegende Basissystem und dessen Basismodelle.

AP 6.b: Umsetzung von variativen Bereichen

Im Zuge des AP 6 wurden verschiedene Methoden zur Umsetzung der automatisierten Verschaltung und zum Austausch variativer Bereiche untersucht. Dabei war es das Ziel, Teile eines über einzelne Modelle verschalteten Gesamtsystems austauschbar zu machen. Als Voraussetzung hierfür müssen alle topologischen Freiheitsgrade des Systems allgemein beschrieben werden. Eine Anforderung, die an eine Implementierung von Systemen mit varianten Bereichen und deren Simulation gestellt werden könnte, wäre das strukturierte Auffinden von für den Austausch geeigneten Modellen. Jeder variante Bereich, ob einzelne Komponente oder Sub-System, stellt gewisse Anforderungen. Eine solche Anforderung könnte z. B. der Arbeitsbereich (Druckniveau) eines Verdichters sein. Die hierfür notwendigen

Informationen müssten der Datenbank zur Verfügung gestellt werden. Diese könnten beispielsweise über Metadaten in die Datenbank eingebracht werden. Dieses Thema wurde seitens ISSE ausführlich betrachtet.

Methodenbeschreibung DSL (Domänenspezifische Sprache)

Ein Lösungsansatz, der für Modelle des Modelica-Formats untersucht wurde, ist die abstrakte Beschreibung der Systemverschaltung mit virtuellen Ports über einen DSL-Ansatz.

Für die Umsetzung des DSL-Ansatzes beschreibt der Nutzer seine vorliegenden Komponenten und die Verschaltung in abstrakter Form. Am Beispiel eines Expansionsventils für Kälteanlagen wird in Abbildung 18 formal beschrieben, wie Komponenten und Gruppen definiert werden.

```
ComponentGroup Valve ports A B;
ID_0815 is Valve;
ID_0816 is Valve;
ID_4711 is Valve;
```

Abbildung 18: Abstrakte Beschreibung von Komponentengruppen und einzelnen Komponenten vom Typ "Valve".

Weiterhin werden variative Gruppen definiert (VariationGroup), die zur Ermittlung verschiedener Systemvarianten ausgetauscht werden können (vgl. Abbildung 19):

```
VariationGroup ValveGroup ports A B;
```

Abbildung 19: Abstrakte Beschreibung variativer Gruppen und Konnektoren.

Die Instanzen von Variationsgruppen beschreiben einen möglichen Inhalt, also eine Verschaltung von Komponentengruppen (vgl. Abbildung 20).

```
BaseValve implements ValveGroup{
        Valve 0816;
        BaseValve.A at 0816.A;
        0816.B at BaseValve.B;
}
```

Abbildung 20: Beschreibung einer Instanz einer VariationGroup und Verschaltung einer Komponentengruppe.

Diese abstrakte Beschreibung ermöglicht es, die Darstellung der Freiheitsgrade für das Ersetzen variativer Bereiche und die Ermittlung der Systemstruktur zu definieren. Als Ergebnis können Verschaltungsvariationen erzeugt werden.

Methodenbeschreibung XOR/SEQ

Weiterhin wurde eine Verschaltung von Komponenten und Subsystemen mittels Entweder-Oder-Gruppen (XOR) und Sequentiellen Gruppen (SEQ) erfolgreich zur Nutzung innerhalb der Katalogplattform

implementiert. Zur Definition des System-Bauplans zur Erzeugung weiterer Systemkonfigurationen werden zu Beginn zwei Gruppen definiert (vgl. Abbildung 21).

- Entweder-Oder-Gruppe (**XOR**): Komponenten oder Subsysteme werden ausgetauscht
- Sequentielle Gruppe (**SEQ**): Die Reihenfolge von Komponenten oder Subsystemen ist variabel

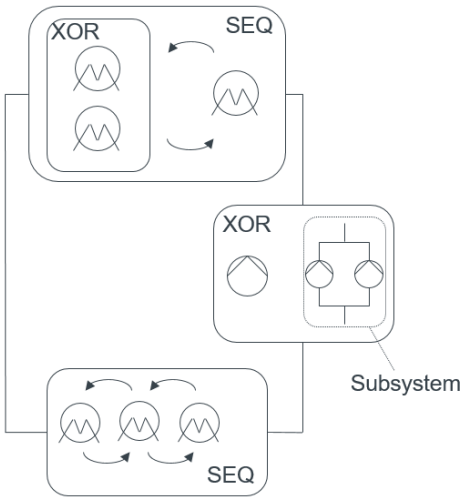

Abbildung 21: Strukturplan zur automatisierten Erzeugung von Varianten über XOR und SEQ.

Mit dieser Methode können mögliche Systeme automatisiert erzeugt und anschließend simuliert werden. Die Zuordnung der Modelle in eine Gruppe erfolgt über Informationen aus JSON-Dateien. Daraufhin können ausgehend vom Strukturplan echte Systembeschreibungen erzeugt werden. Anschließend wird überprüft, ob die Datei- und Konnektortypen der Schnittstellengrößen zusammenpassen (Boolean, Double, Integer, etc.).

Mit dem oben aufgezeigten Beispiel können sich schnell mehrere hundert möglicher Systemtopologien ergeben. Über unterschiedliche Parametersätze kann dies schnell zu mehreren tausend möglicher Systeme führen. Um die große Anzahl der Systeme zu reduzieren, erfolgt eine Plausibilitätsprüfung. In einem weiteren Schritt kann die Optimierer-Schnittstelle genutzt werden, um z. B. über eine Kostenfunktion eine optimale Systemkonfiguration zu finden.

Methodenbeschreibung skriptbasierter Modellaustausch unter Modelica

Da im Bereich der thermischen Simulation sehr häufig auf das Modelica-Format zurückgegriffen wird, wurde speziell eine für diesen Dateityp geeignete Funktion entwickelt. Diese ist in der Lage Modelica-Systemmodelle zu variieren, indem die Klassen der einzelnen Komponenten ausgetauscht werden.

Die Funktion kann über eine Excel-Tabelle bedatet werden. Die Excel-Tabelle kann manuell geschrieben oder aus einem Variations-Tool automatisiert erstellt werden. Die Funktion erstellt aufbauend auf einer Grundverschaltung (Basissystem) pro Zeile der Excel-Tabelle ein Systemmodell.

Erprobt wurde die Variation anhand einer mehrstufigen Kälteanlage. Jede Stufe der Kälteanlage verfügt über mehrere Verdichter, deren Zusammenstellung optimiert werden muss. Der erprobte Entwicklungsprozess zum Modelaustausch ist schematisch in Abbildung 22 dargestellt:

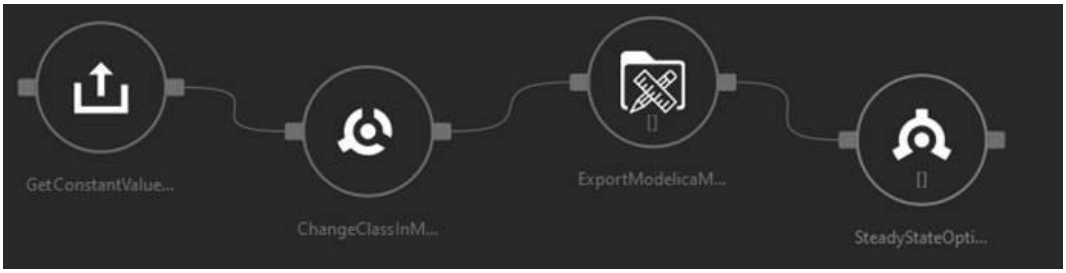

Abbildung 22: Entwicklungsprozess zur Findung einer optimalen Komponentenkonfiguration mit vier Prozessschritten. Von links nach rechts: Datenimport, Modellaustausch, Systemerzeugung und Analyse.

Dieser besteht zunächst aus dem Einlesen der Excel-Tabelle und einer anschließenden Klassenvariation. Aus der Variation ergeben sich mehrere Systemmodelle, die automatisiert in das standardisierte FMU-Format exportiert werden. In einem letzten Schritt kann die entwickelte Schnittstelle zur Anbindung von Optimierern genutzt werden.

Methodenbeschreibung FMU-Verschaltung

Die Verbindung mehrerer FMU-Instanzen über Einzelverbindungen, z. B. die Verbindung eines einzelnen Modellausgangs der ersten Instanz mit dem zugehörigen einzelnen Modelleingang der zweiten Instanz ist ineffizient und fehleranfällig. Bei der Verbindung zweier physikalischer Modelle muss beispielsweise zwischen Zustandsgrößen und strömenden Erhaltungsgrößen unterschieden werden, da letztere sich an einem Verbindungspunkt zu Null addieren, wohingegen erstere gleichzusetzen sind. Die Gruppierung dieser unterschiedlichen Einzelgrößen zu physikalischen Konnektoren erhöht zudem die Übersichtlichkeit, da zwei Komponenten des Systems mittels einer Verbindungslinie verknüpft werden. Hierzu können domänenspezifisch mehrere Ein- und Ausgänge des Modells zu einem physikalischen Konnektor zusammengefasst werden und Verbindungslogiken in einer Datei (z. B. XML) in der Datenbank abgelegt werden (vgl. Abbildung 23).

Softwareseitig wird bei automatisiert erstellten Systemen die Definition der Systemtopologie daraufhin überprüft, inwiefern die vorgegebenen Verbindungslogiken erfüllt sind und die Verbindung zweier physikalischer Konnektoren gültig ist. Bei Verwendung der grafischen Benutzeroberfläche und entsprechend manueller Verknüpfung der Modellinstanzen wird eine fehlerhafte Verbindung softwareseitig nicht zugelassen.

```
<ConnectionDefinition name="FluidConnection">
  <Description>Standard Fluid Connection, using m_flow as flow component, pressure as potential and
  enthalpy as stream.</Description>
  <Name x="flow" y="potential"/>
  <SignalDefinitions>
    <Signals>
      <SignalDefinition type="Real">
        <!-- none -->
        <Name>p</Name>
        <Direction>YX</Direction>
      </SignalDefinition>
    </Signals>
    <FlowSignals>
      <SignalDefinition type="Real">
        <!-- flow -->
        <Name>m_flow</Name>
        <Direction>XY</Direction>
      </SignalDefinition>
    </FlowSignals>
    <StreamSignals>
      <SignalDefinition type="Real">
        <!-- stream -->
        <InstreamName>h_inStream</InstreamName>
        <OutstreamName>h_outflow</OutstreamName>
      </SignalDefinition>
    </StreamSignals>
  </SignalDefinitions>
</ConnectionDefinition>
```

Abbildung 23: Exemplarische Definition der Verbindungs-Definition innerhalb einer XML.

Die Definition eines Systems, respektive einer Systemtopologie, geschieht in Form des Basissystems, welches Instanzen von Basisklassen und ggf. deren Verbindung untereinander definiert. Daraus lassen sich im nächsten Schritt konkrete Systemvarianten ableiten, die im Katalog hinterlegte, konkrete Modelle instanziieren. Anhand objektorientierter Prinzipien wie Vererbung und Polymorphie ist sichergestellt, dass nur die Modelle des Katalogs, die die in den Basisklassen definierten Eigenschaften erfüllen, innerhalb einer Repräsentation des Systems verschaltet bzw. gegeneinander ausgetauscht werden können.

AP 7: Entwicklung mathematisch-numerischer Modellinvertierung

Im Folgenden werden die Arbeiten zu AP 7 beschrieben. Aufgrund thematischer Überschneidungen innerhalb der jeweiligen Unteraufgaben, werden AP 7a und AP 7b zusammen beschrieben.

AP 7.a und 7.b: Modellinvertierung für einfache signalflussbasierte Komponentenmodelle

Für dynamische Modelle lassen sich die Signalflüsse wie folgt beschreiben:

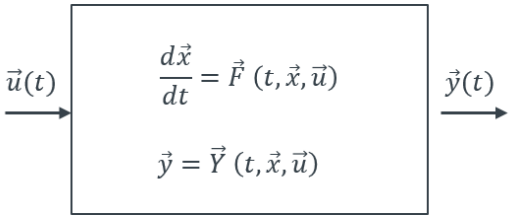

Abbildung 24: Allgemeine Darstellung der Signalflüsse für dynamische Modelle.

Um das Modell bzgl. $\vec{u}$ und $\vec{y}$ zu invertieren, wird der stationäre Zustand des Modells in der Form

$$\frac{d\vec{x}}{dt} = \vec{0} \tag{7.1}$$

vorausgesetzt.

Um den stationären Zustand eines Modells zu ermitteln, existieren mehrere Möglichkeiten:

- Ableitung eines algebraischen Gleichungssystems und Ermittlung des Residuums
- Kombination von AE-Solvern mit ODE-Solvern (Startwertsuche)
- Simulation über eine große Zeitspanne ausführen
- Einbinden eines Optimierers und den differentiellen Zustand als Optimierungsparameter über die Ableitung als Kostenfunktion ermitteln.

In der Praxis existieren jedoch einige Hürden, die bewältigt werden müssen. So existieren bei z. B. physikalischen Systemen etliche Zustände, die das Gleichungssystem sehr groß werden lassen. Weiterhin können Zustände über Minimal- und Maximalwerte verfügen (z. B. Temperaturwerte in Kelvin starten bei 0) oder ungültige Werte (z. B. negative Werte für die Angabe von Absolutdrücken), die bei der Lösung beachtet werden müssen. Weiterhin müssen die Einheiten und Nominalwerte der Zustandsgrößen berücksichtigt werden. Hierfür ist in der Praxis viel Feintuning und teilweise Expertenwissen gefragt.

Um diesen Anforderungen gerecht zu werden, wurden Schnittstellenmethoden entwickelt und bei TLK-Thermo existierende Tools weiterentwickelt.

Die entwickelten Methoden leiten, z. B. aus Einheiten und Variablennamen der Zustände, mögliche Nominal-, Minimal- und Maximalwerte ab. Weiterhin wurde eine „autodetectNominalValues“ entwickelt, die über eine dynamische Simulation über eine längere Zeit die Ergebnisse als Nominalwerte setzt. Weiterhin können über eine weitere Methode die für das Erreichen des stationären Zustands irrelevanten Zustände herausgefiltert werden.

Die Ergebnisse der entwickelten Methoden wurden in dem Tool Steady State Finder vereint. Der Steady State Finder ist ein Software-Werkzeug zur robusten und exakten, sowie schnellen Lösung von Algebro-Differentialgleichungssystemen und dient zur Berechnung von stationären Simulationsergebnissen. Die Robustheit, Exaktheit und Schnelligkeit der Lösungsfindung werden durch eine Kombination von DAE-Lösern und algebraischen Lösungsverfahren erreicht. DAE-Löser integrieren ein Simulationsmodell über der Zeit und können durch Methoden wie der flexiblen Schrittweite und der Event-Behandlung robust bis in den stationären Zustand simulieren. Algebraische Löser, die auf einer Nullstellensuche der Zustandsableitungen basieren, liefern exakte Ergebnisse und sind bei der Berechnung vieler ähnlicher Betriebspunkte, z. B. für verschiedene Messpunkte, sehr schnell. Je nach Modell wird das passende Lösungsverfahren oder eine Kombination ausgewählt. Die Berechnung von Simulationsergebnissen für verschiedene stationäre Betriebspunkte ist eine Kernanwendung von Simulatoren.

Der Steady State Finder bildet außerdem die Basis für weitere Anwendungen:

1. Für das stationäre Fitten von Modellen
2. Für die Berechnung von optimalen Versuchsplänen (DoE)

Die grafische Verknüpfung von Modellinstanzen zu physikalischen Systemen erfolgt signalflussorientiert oder anhand von physikalischen Konnektoren schaltplanorientiert. Physikalische Konnektoren kommen entweder direkt in der Sprache Modelica zum Einsatz (gleichungsbasiert) oder fassen mehrere charakteristische Ein- und Ausgänge einer FMU (oder andere Blöcke mit Ein- und Ausgängen) zusammen. Dabei kann ein Konnektor ggf. gleichzeitig sowohl Ein- als auch Ausgänge zusammenfassen. In der nachstehenden schaltplanorientierten Abbildung 25 repräsentiert jede Verbindungslinie mehrere gerichtete Einzelverbindungen von physikalischen Größen, den charakteristischen Größen eines advektiven Transportprozesses zwischen dem Verdichter und den Wärmeübertragern.

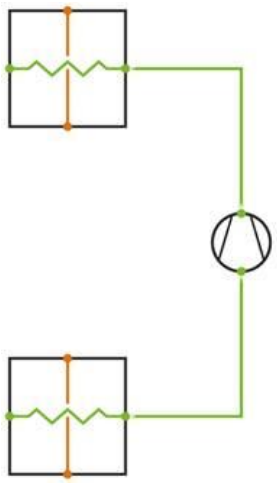

Abbildung 25: Schaltbild eines Ausschnitts eines Kältekreises mit physikalischen Verbindungen zwischen Verdichter und den Wärmeübertragern.

Die in den physikalischen Konnektoren aggregierten physikalischen Größen und Signalflussrichtungen sind exemplarisch anhand des Verdichtermodells in Abbildung 26 dargestellt.

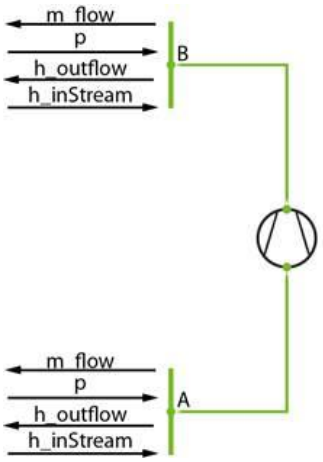

Abbildung 26: Aufschlüsselung der Größen und Signalflussrichtungen eines physikalisch modellierten Kältemittelverdichters.

Die Signalflussrichtungen dieser einzelnen physikalischen Größen für jede einzelne FMU, respektive die Zuordnung der Kausalitäten „input“ ($\vec{u}$) und „output“ ($\vec{y}$), sind zum Zeitpunkt des Modellexports festgelegt. Das FMU-Modell entspricht einer signalflussbasierten Modellbeschreibung. Das signalflussbasierte Pendant der obigen schaltplanorientierten Sichtweise mit der Aggregation von Modelleingängen und -ausgängen zu physikalischen Konnektoren zeigt Abbildung 27.

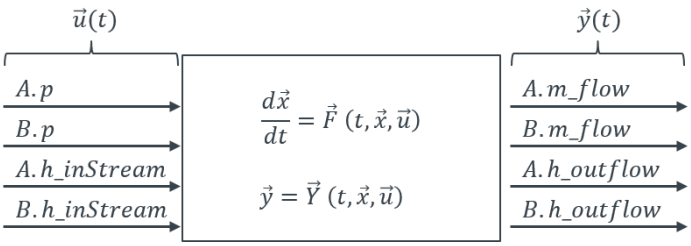

Abbildung 27: Aggregierte Darstellung der Signalflüsse.

Die implementierten mathematisch-numerischen Methoden zur Modellinvertierung erlauben es, auch im Anschluss an den Export eines Modells als FMU diese vordefinierte Signalflussrichtung bei Bedarf umzukehren. Anstelle der Vorgabe der Eingänge $\vec{u}$ und der Berechnung der Modellausgänge $\vec{y}$ ist es ebenfalls möglich, durch Vorgabe der Ausgänge $\vec{y}_{soll}$ die zugehörigen Eingänge $\vec{u}$ für stationäre Modellzustände zu berechnen. Dies ist für den Einsatz innerhalb verschalteter Systeme gleichbedeutend mit einer Invertierung der Schnittstellen der jeweiligen FMU-Instanz. Allerdings wird wie oben beschrieben vorausgesetzt, dass sich das Modell in einem quasistationären Zustand mit

$$\frac{d\vec{x}}{dt} = \vec{0} \tag{7.2}$$

befindet. Der umgesetzte numerische Algorithmus der Entwurfsplattform löst entsprechend die Gleichung

$$\vec{0} = \vec{F}(x, \vec{t}, \vec{u}) \tag{7.3}$$

Die mathematische Invertierung erfolgt durch Formulierung und Lösen einer Residuumsgleichung für den stationären Modellzustand. Die Modelleingänge $\vec{u}$ bilden die Iterationsvariablen/algebraischen Zustandsvariablen

$$\vec{r} = \vec{y}(\vec{u}, \vec{x}) - \vec{y}_{soll} \tag{7.4}$$

AP 8: Entwurf eines Ansatzes zum Refactoring innerhalb des Modell- und Systemkatalogs

Um das Refactoring zu ermöglichen, wurden im AP 8.a die Schnittstellen zu den Modell- und Meta-Informationen untersucht. Die für das Refactoring entworfenen Schnittstellen erlauben es dem Nutzer notwendige Modell- und Meta-Informationen bereitzustellen. Änderungen an der Entwurfsplattform können von verschiedenen Nutzern vorgenommen werden. Die entwickelte Schnittstelle liefert notwendige Informationen zur Aktualisierung der eigenen Entwurfsplattform sowie der Implementierung in vorhandenen Systemen. Zur Umsetzung der notwendigen Nutzerschnittstelle wurde eng mit dem ISSE kooperiert.

AP 8.a: Entwicklung von Schnittstellen zur Integration von Refactoring-Methoden

Um topologische oder parametrische Variationen durchführen zu können, ist es nötig in einem Modell Komponenten oder deren Parameter auszutauschen. TLK hat zur Demonstration der Refactoring-Funktionalität eine Refactoring-Funktionen beschrieben, die ein Modell in einem System bestehend aus Modellen des Dymola-Formats austauschen und auch bedaten kann.

Um die Funktion korrekt ausführen zu können, müssen die notwendigen Angaben aus der Datenbank ausgelesen werden. Auf der Basis von Parametersätzen, die beispielsweise innerhalb einer Excel-Datei existieren können, wird das Modell innerhalb des bestehenden Systems bedatet. Weitere Daten sind z. B. der Ort der Modelle und Konnektor-Informationen (vgl. Abbildung 28).

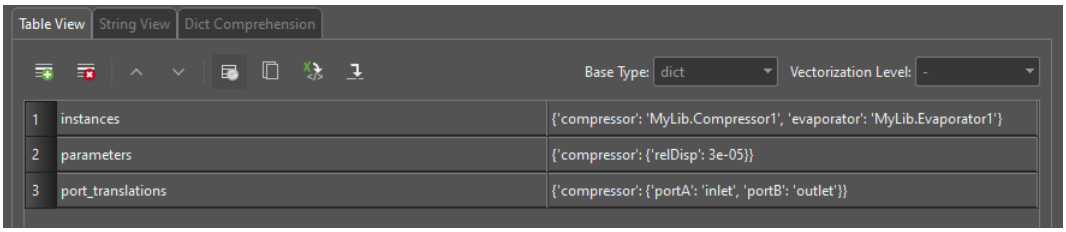

Abbildung 28: Parameterdialog der Refactoring-Funktion.

Dabei ist es sehr wichtig, dafür zu sorgen, dass das resultierende Modell syntaktisch sowie semantisch korrekt ist und alle Referenzierungen auf die Komponente bzw. deren ein- und ausgangsseitigen Schnittstelle korrekt übersetzt werden. Im Detail sind dafür folgende Teiloperationen nötig:

- Entfernen der Ursprungskomponente und Erfassung der vom Rest des Modells erwarteten Schnittstelle
- Einfügen der neuen Komponente in das Zielmodell
- Transformation der Schnittstelle des Modells zu der von der neuen Komponente erwarteten, um z. B. Namensänderungen an Ein- und Ausgängen zu berücksichtigen
- Transformation des Modifizierers der Komponente, um die sie richtig parametrieren zu können
- Speichern und testen des neuen Modells

Da für die Anwendung dieses Refactorings ein tiefes Verständnis für die Funktionsweise der Quell- und Zielkomponente nötig ist, kann dieses nur durch einen Modellierungsexperten durchgeführt werden.

AP 9: Anbindung der Katalogplattform an ausgewählte Simulatoren

AP 9.a: Prototypische Umsetzung von Simulatorschnittstellen

Für die Anbindung der Plattform an verschiede Simulatoren sind zwei mögliche Wege analysiert worden. Zum einen ein dezentraler Ansatz, indem die Plattform innerhalb verschiedener Simulatoren verfügbar gemacht wird, z. B. über entsprechende Add-ons, die für den jeweiligen Simulator installiert werden, um eine Schnittstelle zur Anbindung an die Plattform zu ermöglichen. Eine weitere untersuchte Möglichkeit ist die zentrale Ansteuerung der Simulatoren direkt aus der Plattform heraus. Dieses Verfahren bringt den Vorteil, dass keine Individuallösungen zur Implementierung in verschiedenen Simulationsumgebungen entwickelt werden müssen. Weiterhin kann auch Wartung der Schnittstellen direkt über die Plattform geschehen. Aus den genannten Gründen wurde im weiteren Verlauf des Projekts der zentrale Ansatz verfolgt.

Für die Nutzung der Schnittstellen wurde eine graphische Oberfläche unter Qt erstellt, die dem Anwender ein Kontextmenü zur Konfiguration bietet. Hierüber können alle nötigen Informationen eingetragen, wie beispielsweise Modelpfad, Simulationszeit oder Löser-Einstellungen (vgl. Abbildung 29).

Name	Value	Type
modelFilePaths	**["StandardTaskLibrary://Resources/Automotive_ACCycle.fmu"]**	string.path.file[]
useCacheDir	False	boolean
constantValueDictList	**[{"n_compressor_Hz": 50, "m_flow_air_HVAC_in_kgs": 0.15, "m_flow_air_frontend...**	dict[]
timeSeriesCSVFilePathList	[]	string.path.file[]
timeSeriesDictList	[]	dict[]
startTimes	[0]	number[]
stopTimes	**[100.0]**	number[]
outputVariableFilter	**"(?<!hxGeometry)\\.summary\\.(?!.*arrays\\.|geometry)|.*statePoint.(\\.p|\\.h)"**	string
outputVariableList	[]	string[]
caseNames	[]	string[]
outputRootDir	""	string.path.dir
outputRelDir	[]	string[]
outputIntervallLength	**0.5**	number
simulationTimeOut	-1e+300	number
outputFileType	**"csv"**	string.choice
CSV_output_Delimiter	";"	string
CSV_output_DecimalSeparator	"."	string

Abbildung 29: Kontextmenü einer Simulator-Schnittstelle zur Simulation einer FMU unter Dymola.

Die Ankopplung an die Simulatoren erfolgt über eine Python-Schnittstelle. Als Modell-Austauschformat wird der FMI-Standard verwendet. Dadurch können Modelle domainübergreifend in einer Vielzahl von Simulatoren importiert werden. Es wurden Schnittstellen zu den Simulatoren Matlab/Simulink, TISC, Excel und DaVE entwickelt (vgl. Abbildung 30).

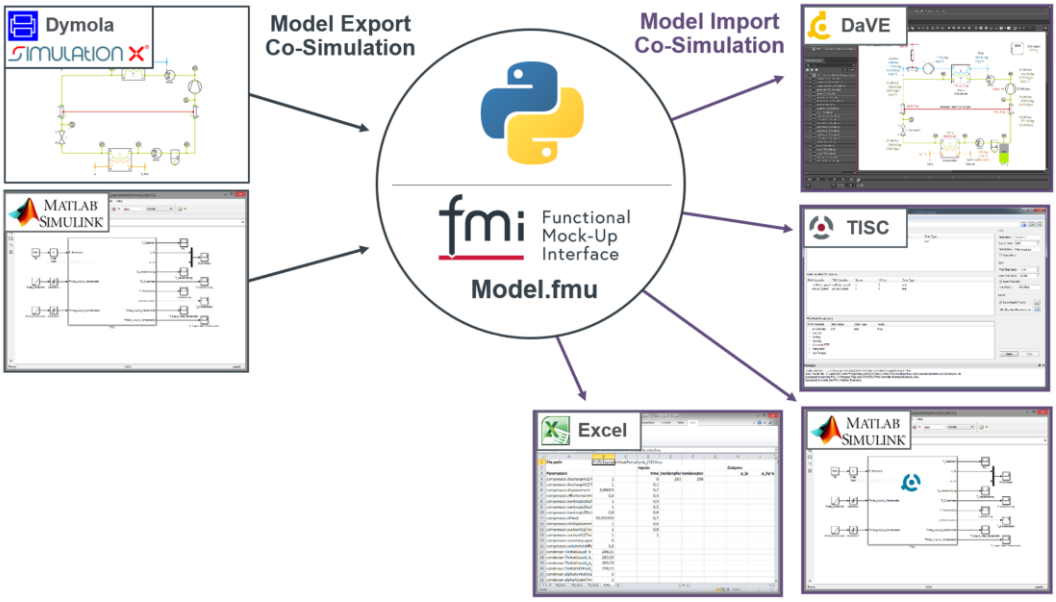

Abbildung 30: Über eine zentrale Python-Schnittstelle können Modelle im FMU-Format in unterschiedliche Simulatoren importiert werden.

Weiterhin wurde eine Schnittstelle zu Simulink entwickelt. Diese ermöglicht die vollständige Anbindung der Matlab-API. Hierüber können dann z. B. die im Workspace gespeicherten Variablen (z. B. benötigte Inputs) ausgelesen und weiterverarbeitet werden.

Eine weitere Python-Schnittstelle wurde zur Co-Simulationsplattform TISC entwickelt. Die benötigten Informationen befinden sich in der von TISC benötigten tssx-Datei. Diese Daten können direkt in die Plattform geladen werden. Ein Vorteil zur Anbindung von TISC ist die Nutzung von nicht-standardisierten Modellformaten. TISC bietet wiederum Schnittstellen zu einer Vielzahl von Simulatoren, die über ein Server-Client-System ausgeführt werden können. Eine Voraussetzung ist jedoch, dass die Co-Simulationsplattform TISC auf dem eigenen System installiert und eingerichtet wird.

Die Python-Schnittstelle zu Excel ermöglicht die Bedatung, Simulation und Auswertung innerhalb von Microsoft Excel.

Eine weitere Schnittstelle wurde für die Visualisierungssoftware DaVE umgesetzt. Mit dieser Anbindung der Plattform können die Ergebnisse bereits während der Simulation visualisiert und ausgewertet werden. Die Konfiguration der Visualisierungsoberfläche geschieht über die Generierung von dvcx-Dateien (Dateiformat von DaVE). Die dvcx-Datei beinhaltet diverse Konfigurations-Informationen, die beliebig bearbeitet werden können. Somit wird eine Visualisierungsoberfläche erschaffen, die ganz auf die Wünsche des Anwenders angepasst ist.

Zusätzlich zu den bereits genannten und implementierten Schnittstellen wurde eine prototypische Schnittstelle zu Dymola entwickelt. Diese funktioniert über die Generierung einer Skript-Datei (mos). Dieses Skript wird ausgeführt, um das Modell in den Simulator zu laden. Um eine FMU in Dymola zu

simulieren, wird ein Template-File herangezogen (mo-Datei). Diese mo-Datei stellt eine Schablone des eigentlichen Modells dar. Die mo-Datei lädt die FMU und kann dann z. B. über Refactoring-Funktionalität (AP 8) angepasst oder ausgetauscht werden.

Weiterhin wurde eine Funktion zur Report-Generierung (PPT und pdf) entwickelt, um automatisierte Dokumentationen zu erzeugen. Hier können z. B. Modellinformationen (Name, Version etc.), Ergebnisse (z. B. in Tabellenform oder unter Verwendung der Instrumente aus DaVE) oder allgemeine Projektbeschreibungen enthalten sein.

AP 10: Entwurf und Optimierung eines elektrifizierten Beispielfahrzeugs der Kompaktklasse

AP 10.a: Gesamtfahrzeugmodell mit thermodynamischen Komponenten

Für das thermische Gesamtfahrzeug wurden physikalische Modelle des Heiz-, Kühl- und Kältekreises sowie des Innenraums erstellt. Es entstand eine Schablone für ein Basissystem, aus dem sich Varianten ableiten lassen (vgl. Abbildung 31).

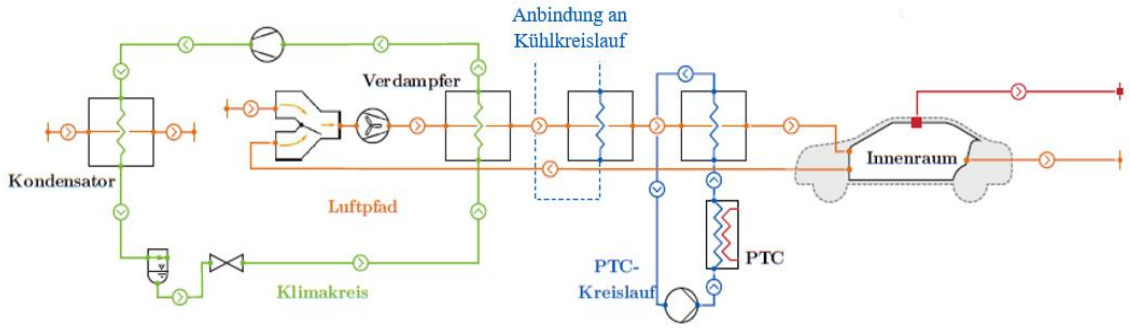

Abbildung 31: Struktur des thermischen Basissystems des betrachteten Demonstrators.

Es wurden weitere Modelle implementiert, um das vom IMAB bereitgestellte E-Maschinen-Modell in die thermische Gesamtfahrzeugsimulation einzubinden. Dies sind Modelle für Fahrzyklus, Fahrer, Batterie, Wandler, Hoch- und Niedervoltverbraucher, Innenraum, Getriebe, Rad und Fahrwiderstände (vgl. Abbildung 32).

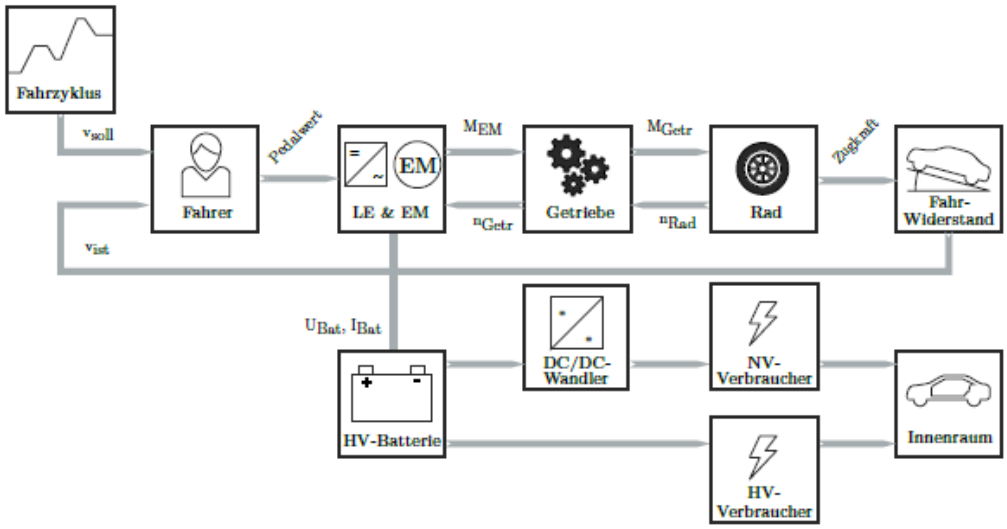

Abbildung 32: Schematische Darstellung des Signalflusses zwischen den Modellen des Antriebsstrangs.

Das Modell „Fahrer“ erhält vom Modell „Fahrzyklus“ eine Sollgeschwindigkeit und leitet diese an einen Regler weiter. Dieser versucht über die Pedalstellung den Sollwert einzuregeln. Die Stellung von Fahr-

und Bremspedal werden in eine Drehmomentanforderung an das E-Maschinen-Modell übergeben. Das Moment wird über das Getriebe, an das Rad und schlussendlich an den Boden mit den jeweiligen Verlusten übertragen. Hieraus ergibt sich eine Ist-Geschwindigkeit des Fahrzeugs, welche an den Regler im Modell des Fahrers zurückgegeben wird. Über ein Kennfeld wird aus Drehzahl und Drehmoment der E-Maschine die benötigte elektrische Leistung ermittelt, die vom Batterie-Modell bereitgestellt werden muss. Aus dem Kennfeld ergibt sich auch die Abwärmemenge, die vom Kühlkreislauf abgeführt werden muss. Weitere elektrische Komponenten sind der DCDC-Wandler sowie Hoch- und Niedervoltverbraucher, wie z. B. Lüfter, Gebläse (NV) sowie Verdichter und PTC (HV) als Komponenten des Kältekreislaufs. Diese Modelle stehen über Heiz-, Kühl- und Kältekreislauf mit dem Innenraum im energetischen Austausch.

Das vom IMAB bereitgestellte Modell einer E-Maschine wurde über eine thermische Kopplung zum Kühlkreislauf in das Gesamtfahrzeug aufgenommen (vgl. Abbildung 33).

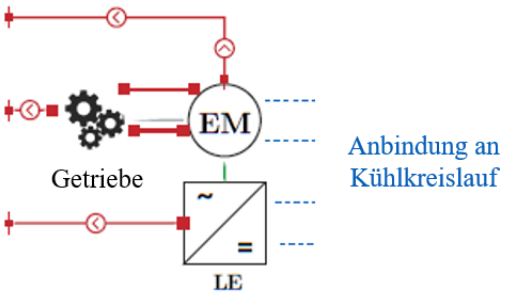

Abbildung 33: Mechanische, elektrische und thermische Integration des E-Maschinen-Modells in das Gesamtfahrzeug.

Zusätzlich zu der Kopplung zum Kühlkreislauf ist die E-Maschine weiterhin elektrisch mit dem Modell der Leistungselektronik sowie mechanisch als auch thermisch mit dem Getriebe und der Umgebung verbunden.

AP 11: Entwurf und Optimierung eines batteriebetriebenen Elektrobusses

AP 11.a: Gesamtfahrzeugmodell mit thermodynamischen Komponenten

Für das Gesamtfahrzeug eines batterieelektrischen Stadtbusses wurden physikalische Modelle des Kälte- und Kühlkreislaufs erstellt und in die Katalogplattform eingestellt. Neben den Modellen des Kälte- und Kühlkreislaufs wurden ebenso Modelle für Innenraum, Längsdynamik, elektrische Komponenten (Zusammenarbeit mit dem IMAB), Regelung, Umgebung und verschiedene Fahrprofile erstellt und in die Plattform integriert (vgl. Abbildung 34).

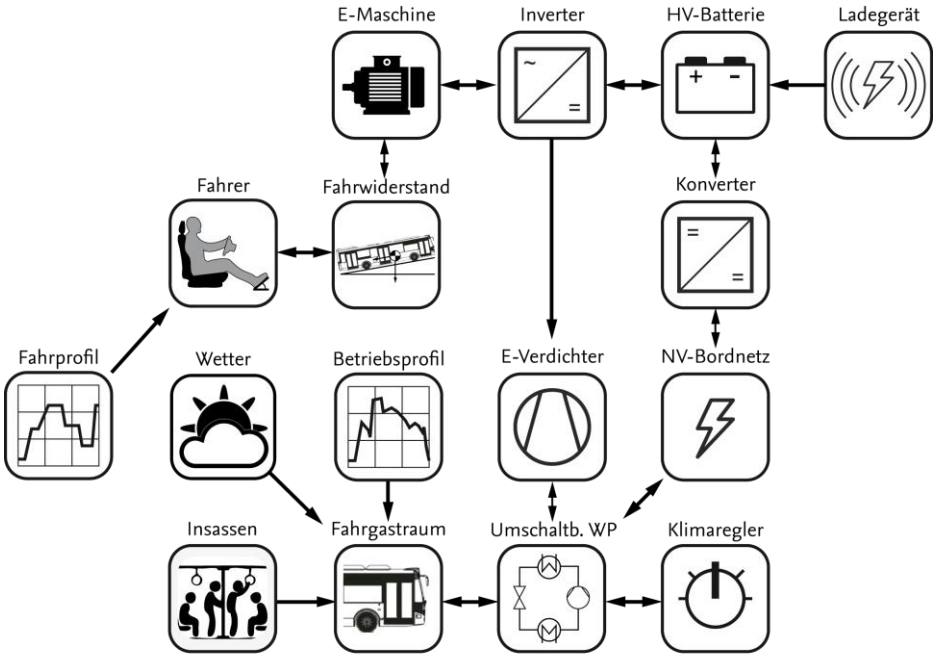

Abbildung 34: Schematische Darstellung des Gesamtfahrzeugmodells für den Elektrobus.

Für die exemplarisch zu demonstrierende Topologieoptimierung wurde der Kältekreislauf als varianter Bereich untersucht. Die als Referenzsystem herangezogene Aufdachanlage ist eine umschaltbare Wärmepumpe (enthält den Kältekreislauf). Das Heizungs- und Klimatisierungssystem wird mit dem natürlichen Kältemittel R-744 (CO_2) betrieben. Die Verschaltung des entstandenen Basissystems ist in Abbildung 35 dargestellt.

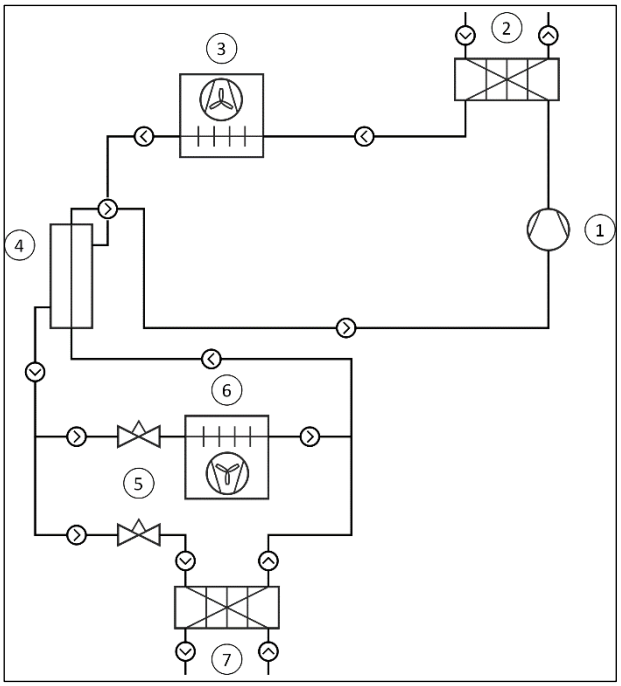

Abbildung 35: Struktur des Basissystems Kältekreislauf.

Die Anlage beinhaltet einen elektrisch angetriebenen CO_2-Verdichter (1) und einen Wärmeübertrager in Plattenbauweise für die Versorgung mit Heißwasser (2). Weiterhin wurden zwei Gaskühler (3) mit insgesamt vier Gebläsen, ein Wärmeübertrager für den internen Wärmeaustausch (IHX) (4), Expansionsventile (5) sowie zwei Verdampfer (6) mit drei Ventilatoren modelliert. Ein weiterer Plattenwärmeübertrager (7) sorgt für die Kaltwasserversorgung.

Ausgehend vom Basissystem, welches aus Basismodellen des Katalogs erstellt wurde, wurde eine konkrete CO_2-Kälteanlage eines Stadtbusses abgeleitet. Dies geschah über die Verwendung von Herstellerangaben der einzelnen Komponenten. Dadurch konnten Systemtopologien von Aufdachanlagen unterschiedlicher Hersteller erzeugt werden. Über Parametersätze konnten wiederum Systemvarianten dieser Topologien und somit physikalische Modelle realer Aufdachanlagen erzeugt werden

Das Gesamtfahrzeugmodell wurde hinsichtlich des Einsatzes von Ejektoren zur energieeffizienten Druckrückgewinnung innerhalb der Plattform untersucht. Die verschiedenen Systemvarianten wurden auf Basis von unterschiedlichen Ejektor-Modellen erzeugt. Diese unterscheiden sich in unterschiedlichen Treibdüsenquerschnitten. Die hierfür benötigten physikalischen Ejektor-Modelle entstammen aus dem Modellbestand von TLK. Diese wurden im Modelica-Format der Plattform hinzugefügt. Über die prototypische Austausch-Funktion von Dymola-Modellen wurden mehrere Systemvarianten erzeugt. Mit der Verwendung der entwickelten Python-Schnittstelle zur Anbindung von Optimierungsalgorithmen wurden die erzeugten Systeme mithilfe einer Kostenfunktion hinsichtlich des Gesamtenergieverbrauchs bzgl. eines vorher definierten Szenarios (Umgebung und Fahrprofil) bewertet (vgl. Abbildung 36).

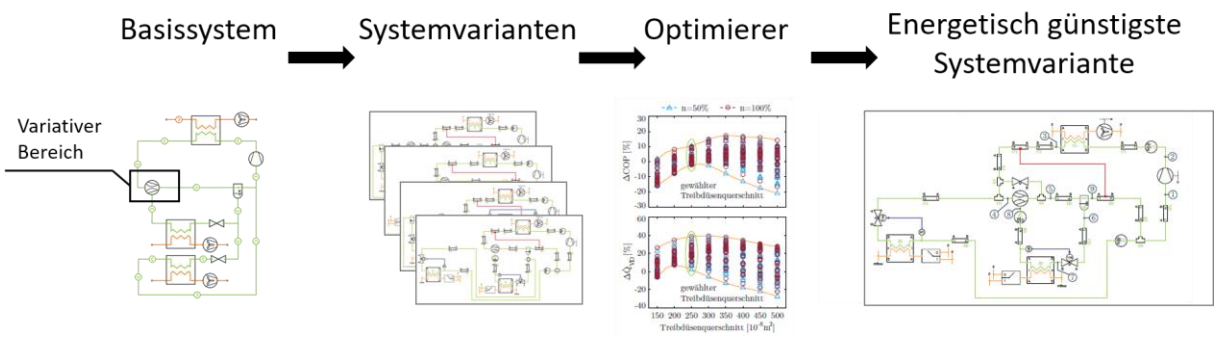

Abbildung 36: Erzeugung von Systemvarianten und Ermittlung einer energetisch Optimalen Systemvariante.

In einem weiteren Schritt wurde die Anbindung des Regelungssystems an die Optimiererschnittstelle genutzt, um eine energieeffiziente Regelung des Kältemittelhochdrucks zu untersuchen. Hierfür wurden Strecken- und Umgebungsbedingungen festgelegt, die als Randbedingungen der Simulation dienen. Für die Ermittlung des optimalen Hochdrucks diente das Gesamtmodell in Form eines modellprädiktiven Reglers. Der Regler wurde über die Modellreduktion in Form eines neuronalen Netzes erstellt und in zwei Modellformaten (Matlab und Python-Skript) in die Plattform integriert. Das neuronale Netz ermittelt eine energieoptimale Trajektorie des Hochdrucks für einen festgelegten zukünftigen Zeitraum. Der so ermittelte Hochdruckverlauf wird dem Regler des Gesamtmodells als Sollwert zugeführt.

Die in der Plattform hinterlegten Modelle können mit zusätzlichen META-Daten verknüpft werden. Hier können beispielsweise auch Informationen bzgl. Anschaffungs-, Betriebs- und Wartungskosten hinterlegt sein. Mit diesen Informationen und der Nutzung der Optimierer-Schnittstelle können so kosteneffiziente Systemvarianten ermittelt werden.