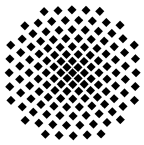


H L R S

Institut für
Höchstleistungsrechnen

FORSCHUNGS- UND ENTWICKLUNGSBERICHT

MIKROSERVICEARCHITEKTUR-BASIERTE CFD-SIMULATION VON PHÄNOMENEN DYNAMISCHER SYSTEME AM BEISPIEL MODERNER BERGBAUBEWETTERUNG

Alexey Cheptsov

ISSN 0941 - 4665 Februar 2021 HLRS-24

Höchstleistungsrechenzentrum
Universität Stuttgart
Prof. Dr.-Ing. Dr. h.c. Dr. h.c. Prof. E.h. Michael M. Resch
Nobelstrasse 19 - 70569 Stuttgart
Institut für Höchstleistungsrechnen

MIKROSERVICEARCHITEKTUR-BASIERTE CFD-SIMULATION VON PHÄNOMENEN DYNAMISCHER SYSTEME AM BEISPIEL MODERNER BERGBAUBEWETTERUNG

von der Fakultät Energie-, Verfahrens- und Biotechnik
der Universität Stuttgart zur Erlangung der Würde
eines Doktor-Ingenieurs (Dr.-Ing.) genehmigte Abhandlung
vorgelegt von

Alexey Cheptsov
aus Donezk, Ukraine

Hauptberichter:	Prof. Dr.-Ing. Dr. h.c. Dr. h.c. Prof. E.h. Michael M. Resch
Mitberichter:	Prof. Dr.-Ing. Vladimir Svjatnjy
Tag der Einreichung:	09. Juli 2020
Tag der mündlichen Prüfung:	03. Februar 2021

D93

ISSN 0941 - 4665 Februar 2021 HLRS-24

*Эту диссертацию я посвящаю моим **Родителям** в знак благодарности за их стремление сделать из меня того человека, которым я стал, а также моей супруге **Анастасии** за самый замечательный подарок в моей жизни – нашу дочь **Аннабель**.*

---

*Diese Dissertation widme ich meinen **Eltern** als Dankbarkeit für ihr unermüdliches Bestreben, mich zu einem Menschen, wie ich es geworden bin, zu bilden sowie meiner Frau **Anastasia** für ihr großartigstes Geschenk meines Lebens – unsere Tochter **Annabelle**.*

Danksagung

An dieser Stelle möchte ich mich bei allen meinen Kolleginnen und Kollegen, Freundinnen und Freunden, Familienmitgliedern aber auch all denjenigen bedanken, die mich während der Arbeit an dieser Dissertation unterstützt, motiviert, begleitet, aber auch durch gezielte Anregungen und Impulse zum erfolgreichen Abschluss meiner Arbeit geführt haben.

Mein besonderer Dank gilt dabei Herrn **Professor Resch** für seine Betreuung meiner Arbeit am Höchstleistungsrechenzentrum Stuttgart (HLRS). Herr Resch, dank der exzellenten Forschungsatmosphäre am HLRS, die durch Ihre Bemühungen als Direktors des Zentrums geschaffen wurde, hatte ich neben den mir anvertrauten Forschungsprojekten die Zeit, sowie Lust und Spaß gefunden, am Forschungsvorhaben dieser Dissertation zu arbeiten. Vielen Dank für Ihre kompetente und gleichzeitig freundliche Betreuung während der letzten Jahre, die ich als Teil der HLRS-Familie verbringen durfte! Das war mir eine große Ehre.

Ein ganz besonderer Mensch, ohne den diese Arbeit nicht zustande gekommen wäre, ist Herr **Professor Svjatnjy**, und bei ihm möchte ich mich besonders bedanken. Herr Svjatnjy, Ihr tiefes Fachwissen aber auch Ihr unermüdlicher Enthusiasmus und Ihr positives Denken haben mich ganz besonders geprägt, gestärkt und durch viele Jahre der Arbeit an dieser Dissertation geführt. Sie waren mir immer das Vorbild eines Wissenschaftlers für mich, dergestalt wie er in seiner Arbeit als auch im Privatleben sein sollte.

Außerdem möchte ich mich bei den Menschen bedanken, die eine entscheidende Rolle in meiner Karriere als Wissenschaftler an der Universität Stuttgart gespielt haben. Und zwar danke ich ganz herzlich Herrn **Professor Zeitz** für die langjährige Leitung des DAAD-Kooperationsprogramms mit der Universität Donezk seitens der Universität Stuttgart, durch welches ich die Möglichkeit bekommen habe, nach Stuttgart zu kommen. Herr Zeitz, Ihr Stadtplan von Stuttgart, den Sie mir in der ersten Woche unserer Bekanntschaft in 2002 geschenkt haben, ist zu meinem Lebensplan in dieser wunderschönen Stadt geworden. Ganz besonders möchte ich auch Herrn

Professor Göhner danken, dem Direktor des IAS-Instituts, an welchem ich während meines DAAD-Aufenthalts von 2004 bis 2005 als wissenschaftlicher Mitarbeiter sein durfte. Herr Göhner, wenn alles so gut und ordentlich funktionieren würde, wie Sie das am IAS, aber auch an der Fakultät und der Universität, in Ihren Rollen als Dekan und Prorektor, organisiert haben, hätten wir längst in einer idealen Welt gelebt. Ich wünsche Ihnen gute Gesundheit und viele positive Ereignisse in Ihrem Forschungs- und auch privaten Leben.

Einen großen Dank schulde ich auch meinem HLRS-Kollegen **Herrn Klank** für seine Korrektur des Texts und der Graphiken.

Abschließend möchte ich mich bei meiner **Familie** bedanken. Ihr seid immer ein besonders wichtiger Baustein in meiner Karriere als Wissenschaftler gewesen, ohne welchen ich nichts davon hätte erreichen können.

Inhaltsverzeichnis

Abkürzungsverzeichnis

API	(engl. Application Programming Interface) – Schnittstelle eines Softwareprogramms
AS	Automatisierungssystem
CAD	(engl. Computer-Aided Design) – rechnergestütztes Entwerfen
CFD	(engl. Computational Fluid Dynamics) – numerische Strömungsmechanik
CPS	Cyberphysisches System
CPU	(engl. Central Processing Unit) – zentrale Recheneinheit
DAG	(engl. Directed Acyclic Graph) – gerichteter azyklischer Graph
DBMS	(engl. Data Base Management System) – Datenbankverwaltungssystem
DS	Dynamisches System
DSKP	Dynamisches System mit konzentrierten Parametern
DSVP	Dynamisches System mit verteilten Parametern
FDM	Finiten Differenzen Methode
Flops	(engl. Float-point operations per second) – Fließkomma-Operationen pro Sekunde
GAS	Grubenbewetterung-Automatisierungssysteme
GBN	Grubenbewetterungsnetz
HIL	(engl. Hardware-in-the-loop) – Anschluss eines eingebetteten Systems an einen Simulator
HLRS	das Höchstleistungsrechenzentrum Stuttgart
HPC	(engl. High Performance Computing) - Höchstleistungsrechnen
HPDA	(engl. High Performance Data Analytics) – Höchstleistungs-Datenanalytics
IE	(engl. Industrial Ethernet) – ein Industrielles Ethernet-basiertes Computer-Netzwerk
IoT	(engl. Internet-of-Things) – das Internet der Dinge
IT	Informationstechnologien
MIMD	(engl. Multiple Instruction Multiple Data) – eine Architektur von Großrechnern nach flynnscher Klassifikation
MPI	(engl. Message-Passing Interface) – nachrichtenbasiertes Austauschprotokoll für parallele Rechner
MPMD	(engl. Multiple Program Multiple Data) – eine Technik des parallelen Programmierens
MS	Mikroservice
NUMA	(engl. Non-Uniform Memory Access) – eine parallele Computer-Arbeitsspeicher-Architektur
P2P	(engl. Point-to-Point) – Punkt-zu-Punkt Kommunikationsstrategie
REST	(engl. REpresentational State Transfer) – ein Internet-Austauschprotokoll

RISC	(engl. Reduced Instruction Set Computer) – eine spezialisierte, vereinfachte Mikroprozessor-Architektur
RK4	numerisches Runge-Kutta-Verfahren der 4. Ordnung
SaaS	(engl. Software-as-a-Service) – dienstbasiertes Programmiermodell
S-a-a-S	siehe SaaS oben
SOA	Service-orientierte Architekturen
SoC	(engl. System-on-Chip) – eine Technologie der ein-platinen Rechner
SIL	(engl. Software-in-the-loop) – eine Hardware-freie HIL-Technologie (siehe oben)
TO	Testobjekt
UML	(engl. Unified Modelling Language) – unifizierte Modellierungssprache
VOD	(engl. Ventilation-on-Demand) – bedarfsgerechte Bewetterung

Abbildungsverzeichnis

Tabellenverzeichnis

Kurzfassung

Die moderne industrielle Produktion verlangt einfache aber gleichzeitig effektive Modelllösungen für komplexe Problemstellungen – eine brandaktuelle Herausforderung, mit welcher die klassischen, monolithisch aufgebauten und auf der Basis weniger Messreihen entwickelten Modelle nicht länger zurechtkommen.

Die vorliegende Forschungsarbeit ist zum Großteil von den akuten Bedürfnissen der ventilationstechnischen Systeme von (Untertage-) Bergwerken motiviert – einem Industriebereich mit enormen Herausforderungen für die Simulationstechnik, die maßgeblich von zwei wichtigen Faktoren bestimmt sind: Zum Einen zeichnen sich die Ventilationsanlagen (auf der Fachsprache – **Bewetterungssysteme**) durch eine sehr komplexe **Dynamik** der zugrundeliegenden physikalischen Prozesse aus (z.B. die Methanemission und Vermischung mit dem durch die Strecken fließenden Luftstrom), was die hohe Komplexität der resultierenden Modelle bestimmt. Zum Anderen umspannen die Bewetterungssysteme realer Objekte großdimensionale Strukturen, die aus vielen (bis hunderten) topologisch vernetzten Wetterführungsstrecken bestehen können. Die Effizienz und Effektivität der Untertagebewetterung ist der entscheidende Faktor der Arbeitssicherheit in Bergbaubetrieben, was heutzutage von Automatisierten Steuerungssystemen gewährleistet wird. Ein wichtiger Bestandteil solcher Systeme sind die Modelle, welche das Verhalten des kontrollierten Systems in bestimmten Fällen voraussagen. Aufgrund der oben genannten Komplexitätsgründe werden in den aktuellen Kontrollsystemen nur statische Modelle realisiert, was ihre Funktionalität wesentlich einschränkt. So werden z.B. die wichtigen Aspekte der Gasdynamik (wie zeitabhängiger Methanaustritt aus dynamischen Quellen) vernachlässigt. Die Nutzung von dynamischen Modellen ist aufgrund der limitierten Rechenkapazität des zentralen Rechners des Steuerungssystems sowie wegen fehlender Methodik zur Echtzeitintegration zeitabhängiger Ergebnisse der Modelle unmöglich. Flexibilität und Anpassungsfähigkeit der Simulationssoftware im Sinne der **Portabilität, Verteilung, Echtzeitfähigkeit, Energieeffizienz und Leistungsstärke** werden zu den wichtigsten Anforderungen an die moderne Simulationstechnik, was

die Erforschung neuartiger Ansätze zur Entwicklung der Simulationssoftware mittels fortgeschrittener Konzepte, wie service-orientierten Architekturen, Cloud, Industrie-4.0, Fog-Computing und sonstigen aktuell erforderlich macht.

Diese Herausforderungen werden in der Arbeit aus der Perspektive der **Cyber-Physical Automatisierung (CPA)** – eines Grundkonzepts der Industrie-4.0- Methodologie – adressiert, was einerseits viele attraktive Vorteile für die Durchführung der Simulationsstudien in einem direkten industriellen Umfeld bietet, aber andererseits innovative Umsetzungskonzepte für die Simulationssoftware, wie dem in dieser Arbeit entwickelten **Mikroservicearchitektur-Ansatz,** erfordert. Nach diesem Ansatz wird eine modulare Simulationsanwendung mittels einer funktionellen Komposition vieler autonomer, hierarchisch aufgebauter und über eine heterogene Infrastruktur verteilter Simulations-Mikroservices aufgebaut, welche die Kompositionsteile des modellierten dynamischen Systems bzw. Prozesses realisieren. Durch die Möglichkeit, die einzelnen Services auf unterschiedlichen Hardwareplattformen auszuführen, wird eine Integration mit Sensor-Hostsystemen angestrebt. Im Gegenteil zu den klassischen Entwicklungsansätzen für Simulationsanwendungen, bietet der Mikroservicearchitektur-Ansatz alle vorher erwähnten Vorteile (Portabilität, Verteilung usw.), die lediglich von den technischen Möglichkeiten des Hostsystems begrenzt werden. Es wird eine Architektur des Mikroservice-basierten Systems erarbeitet und entwickelt, inkl. eines Programmiermodells, einer Ausführungsplattform sowie der dazu gehörigen Hilfsdienste. Der Ansatz kann problemlos um die dynamischen Systeme von anderen Forschungsbereichen als dem Bergbau, wie z.B. in der Konstruktion, Bio- und Verfahrenstechnik, der Energetik, Ökologie, Unternehmens- und Öffentlichkeits-strukturen u.v.m., erweitert werden. Die entwickelten Technologien werden anhand der Simulationsaufgabenstellungen realer Objekte validiert und evaluiert.

Der weitere Inhalt der vorliegenden Arbeit umfasst sechs Kapitel. **Kapitel 1** legt ein Fundament für das angestrebte Forschungsvorhaben, indem ein Überblick über die historische Entwicklung der Basisbereiche gegeben und die motivierenden Anforderungen präsentiert werden. **Kapitel 2** beschreibt und diskutiert den

derzeitigen Stand der Forschung und Entwicklung in zweckgerichteten Bereichen der Wissenschaft und Technik, die von einer großen Relevanz für das in dieser Arbeit verfolgte Vorhaben sind. **Kapitel 3** präsentiert den vorgeschlagenen Ansatz zur Entwicklung der Simulations-Anwendungen und Studien mittels Mikroservice-architektur-basierter softwaretechnischen Lösungen. **Kapitel 4** beschreibt die Grundlagen, Vorgänge und Ergebnisse der Realisierung von strömungsdynamischen (**CFD**) Modellen der Bergbaubewetterung nach dem erarbeiteten Mikroservice-Ansatz. **Kapitel 5** präsentiert und analysiert die Validierungs- und Evaluierungsergebnisse und beschreibt die wichtigsten Anwendungsszenarien für die entwickelten Simulations-services. **Kapitel 6** beinhaltet eine Zusammenfassung sowie einen Ausblick.

Abstract

Recent advances in science and technology (such as Industry-4.0) have allowed massive digitalization of industrial technological systems, paving the way towards the creation of new, innovative, simulation applications. Such applications show great potential for designing new approaches for emerging industrial tasks of highly dynamic systems that require immediate or real-time support, and for which the offline data and resulting coupling with an external simulation platform is not possible, e.g. due to time- or security-critical constraints. The currently available technologies that can help realize such simulation scenarios are largely confronted by two major challenges – integration with numerous diverse sensor devices and use in real-time use case scenarios. While the challenge of service integration is addressed by the concept of Cyber-Physical Systems, which aims to incorporate sensor data in digital applications workflows, the usage of high-performance, parallel and distributed computing technologies helps minimize the execution time to fulfill the real-time scenarios' requirements. However, the existing standard programming models — used by the majority of the simulation applications — do not allow them to take advantage of both technologies simultaneously.

As a reaction to the challenges of the reactive usage in industrial environments, this work proposes a novel, service-oriented approach to the development of simulation applications, as opposed to the standard, monolithic design approaches. The key concept of the proposed approach is microservices – functionally decoupled, interconnected composition blocks of a hierarchically organized, modular simulation application that implements a specific part of the simulation logic for the targeted physical phenomena. Microservice-based applications can take advantage of running on heterogeneous, distributed hardware architectures that might include low-power embedded hardware of industrial sensors (e.g. fog), distributed parallel (e.g. cloud), and high-performance computing infrastructures.

For the fulfillment of this vision, a multiple-instructions-multiple-data (MPMD) programming model design for microservice-based application was elaborated, which

allows definition of individual services, their hierarchical communication patterns, strategies to develop composed services based on the already available ones, and realization of the application logic, among other things. Microservices offer a simple but efficient way to leverage the “bottom-up” decomposition approach to the development of complex models, in particular from the computational fluid dynamics domain. The application of this proposed approach is demonstrated on a computational fluid dynamics simulation study of aero- and gas-dynamic processes in security critical objects of underground mine ventilation networks.

1 Einleitung

Der Inhalt dieses Kapitels legt das Fundament für das angestrebte Forschungsvorhaben an, indem ein Überblick über die historische Entwicklung der Basisbereiche – Simulationstechnik (Kapitel 1.2) und Automatisierungstechnik (Kapitel 1.3) – gegeben wird. Danach werden die Zielobjekte der Forschung – die Bewetterungsnetze (Kapitel 1.4) – präsentiert. Im Abschluss (Kapitel 1.5) werden die motivierenden Anforderungen erläutert.

1.1 Vorwort

> *"Well, in our country," said Alice, still panting a little, "you'd generally get to somewhere else—if you run very fast for a long time, as we've been doing."*
>
> *"A slow sort of country!" said the Queen. "Now, here, you see, it takes all the running you can do, to keep in the same place. If you want to get somewhere else, you must run at least twice as fast as that!"*

Der oben aufgeführte Dialog zwischen der Roten Königin und Alice aus Lewis Carrolls "Through the Looking-Glass, and What Alice Found There", bekannt auch als „Red Queen's Race" Paradox aus der gleichnamigen Novelle von Isaac Asimov, illustriert die Rolle relativistischer Effekte in der Physik und im Leben der Menschen, die in der vorliegenden Arbeit weiterhin auf die Ebene der Simulation gehoben werden.

Das Hauptdilemma der heutigen Simulationstechnik im ingenieurwissenschaftlichen Bereich – **die Brauchbarkeit von deterministischen Modellen für reale physische Objekte und Anwendungsfälle mit immer wechselnden, nicht-deterministischen (also dynamischen) Verhaltungsmustern gegenüber ihrer** Funktionsumgebung – wurde immer noch nicht eindeutig gelöst (wie z.B. von Feldmann et al. [R1] sowie vielen anderen gezeigt). Die Gründe dafür sind für jedes konkrete analysierte System unterschiedlich, belaufen sich aber auf zwei wichtige Punkte: (i) unzureichende Aufbereitung und Analyse aller anfallenden dynamischen Daten aus physischen Objekten und (ii) Nichtberücksichtigung der systemübergreifenden Erkenntnisse in den Modellen und ihrer Funktionsumgebung.

Viele moderne industrielle automatisierte Systeme entwickeln sich in Richtung der Cyber-Physischen Systemen, die über das Potenzial verfügen, die vorhandene Datenverfügbarkeitsproblematik lückenlos zu schließen. Diese Tatsache bietet viele Chancen für die moderne Simulationstechnik, stellt sie aber gleichzeitig vor mehrere Herausforderungen, die diese Arbeit zur Erarbeitung umfangreicher Lösunwogen für dynamische Simulationsprobleme motiviert.

1.2 Entwicklung der Simulationstechnik

Heutzutage ist die Simulation ein unverzichtbares Werkzeug der grundlegenden sowie der angewandten Forschung, das zu einer vertrauten Quelle der wissenschaftlichen Erkenntnisse über dynamische Systeme in Bereichen wie Astrophysik, Medizin, Atomtechnik, Klimaforschung, Automatisierungstechnik und zahlreichen anderen geworden ist. Die Entwicklung eines neuen Automodells oder eines Flugzeugs ist nur mittels Simulationstechnik möglich. Der Ursprung der heutigen Simulation liegt allerdings mehrere Jahrhunderte zurück und stammt aus den Werken bedeutender Forscher der antiken, mittelalterlichen und auch modernen Zeit.

Modellierung ist eine wichtige Erkenntnismethode des menschlichen logischen Denkens. Die moderne Evolutionsforschung bezeichnet die Modellierung „par ailleurs“ als die grundlegende angeborene Eigenschaft des menschlichen Gehirns (wie im Thomas Junkers Werk „Die Evolution des Menschen“ [R2]) und erweitert damit das berühmte lateinische „C*ogito ergo sum*“ – „Wir denken und somit existieren wir“ zum „Wir simulieren und somit denken wir“. Damit werden die Parallelen von der Simulation zum Denkprozess gezogen, wobei die Letztere nichts weiter sein sollte als die Simulation – die dynamische Simulation, wie es im Sinne der vorliegenden Arbeit belegt wird.

Im technischen und physikalischen Kontext sind Modellierung und Simulation die bedeutendsten Werkzeuge der Wissenschaft und Forschung, die zum Hauptziel haben, eine Vorhersage über die Zustandsentwicklung eines physikalischen Prozesses oder Systems unter vorgegebenen Einflussbedingungen zu treffen. Im Rahmen der

Modellierung wird ein **Modell** (aus lat. *Modulus* – Muster, Maket) erstellt, welches eine entweder abstrakte (anhand einer mathematischen oder symbolischen Beschreibung) oder materielle (z.B. mittels Makete, Pläne, Schemata, usw.) isomorphische Beschreibung/Repräsentation des untersuchten Zielobjekts darstellt. Die Präzision dieser Darstellung definiert die Genauigkeit des Modells. **Simulation** ist ein Prozess der Nutzung des Modells mit dem Ziel, eine Prognose über eine mögliche Entwicklung der analysierten Prozesse im untersuchten Objekt unter bestimmten Voraussetzungen (Anfangs- und Randbedingungen) zu erstellen.

Die Entstehung der heutigen Simulationstechnik (mit dem Schwerpunkt Fluid-Dynamik) als ein Komplex von Methoden, Verfahren, und Lösungen für Modellierung und Simulation hat eine Jahrhunderte alte Geschichte und wird von mehreren entscheidenden Meilensteine markiert (**Abbildung 1**).

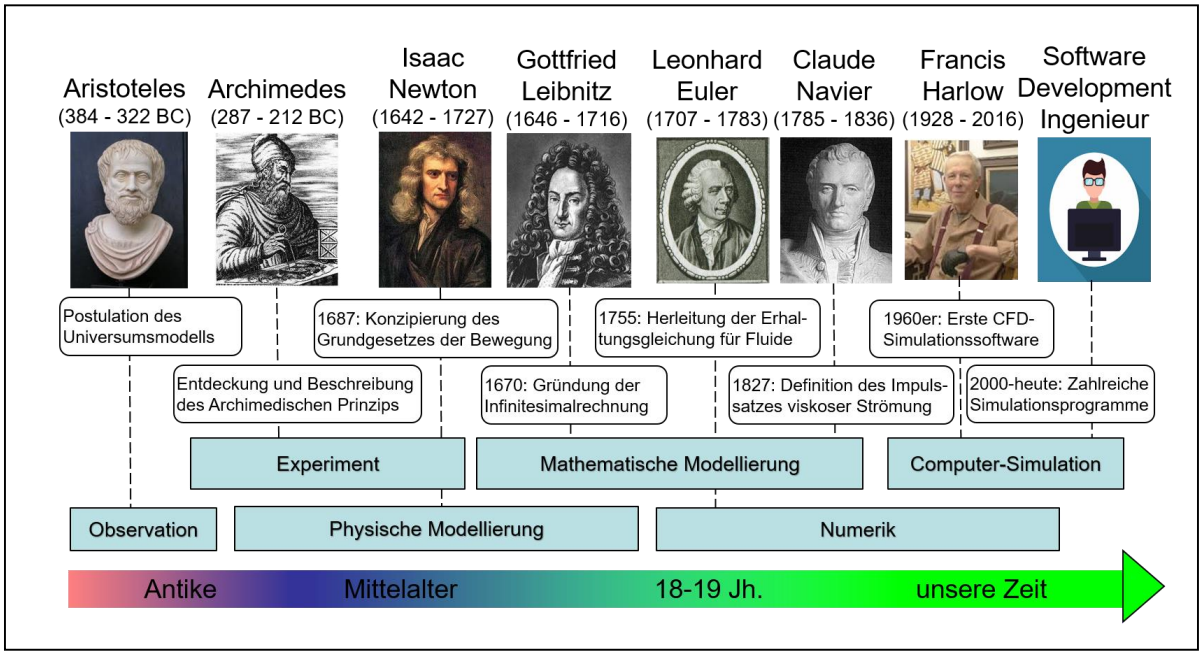


Abbildung 1. *Entwicklung, Meilensteine und Geschichte der CFD-Modellierung und Simulation.*

Die Ursprünge der systematischen Erforschung (also Modellierung) der umgebenden physikalischen Prozesse und Systeme wurden schon in der antiken Philosophie des logischen Denkens mit Konzepten wie Abstraktion, Analogie, Hypothese, sowie vielen anderen festgelegt, die bis heute die Simulation immer noch maßgeblich prägen. Die Etablierung der Modellierung als einer wichtigen Erkenntnismethode hat schon zu

den Gründungszeiten der Aristotelischen Schule begonnen. Anhand der überspringenden assoziativen Erkenntnisse, die man mittels Observation bekommen hatte, wurden Experimenten durchgeführt und die grundlegenden physikalischen Modelle konzipiert (wie das archimedische Grundgesetz der Hydrostatik, welches laut Legende aus dem Experiment von Archimedes in seiner berühmten Badewanne entstanden war). Archimedes wird übrigens auch als einer der ersten bedeutenden Numeriker ausgezeichnet, dank seiner näherungsweisen Lösungen für die Berechnung von Flächen (Integralen) oder der Kreiszahl *Pi*.

Den elaborierten Modellen wurde die Beschreibung der Prozesse zu Grunde gelegt, die schon damals ihrer Zeit weit voraus waren. Zum Beispiel haben die griechischen Philosophen Dimokritis und Epikuris bereits im 4. Jh. v. Chr. die physikalischen Eigenschaften verschiedener Stoffe durch die wechselnde Wirkung der Bauteile ihrer Materie erklärt, was zum Archetyp des erst im 19. Jh. n. Chr definierten Atoms geworden ist. Ein weiterer Repräsentant des griechischen philosophischen Denkens - Theophrast - hat im 4. Jh. v. Chr. die Grundlagen eines Rauchkanals [R3] mit Hilfe des vorher erwähnten archimedischen Gesetzes beschrieben, welche zu den Zeiten des industriellen Aufschwungs im 18. Jh. n. Chr. aktiv zum industriellen Bau der Fabrikanlagen eingesetzt wurden. Die komplexen Lösungen, die z.B. die ägyptischen Pyramidenbauer oder die altgriechischen Schiffhersteller für ihre Probleme mittels Modellierung erarbeitet hatten, werden auch in unseren Zeiten immer noch benutzt. Solche Beispiele waren auch zahlreich im Mittelalter zu finden. So hat Georgius Agricola (Georg Bauer) – der deutsche Arzt und Philosoph des 16. Jh. sowie Gründer der Mineralogie und Bergbaukunde – die Grundlagen der Bergbaubewetterung mittels Zufuhr frischer Luft (ein Prozess, den das heutige Bergbaulexikon als *Wetterführung* bezeichnet) beschrieben [R4], die bis heute als „goldener Standard“ der Untertagebewetterung gelten.

In ihrer jetzigen Form hat die Modellierung zum großen Teil der rasanten Erweiterung des mathematischen Apparats im 17.-18. Jh. zu verdanken, insbesondere der Differentialrechnung und den numerischen Verfahren (in Arbeiten von Newton und Leibnitz). Die Werke von Vorreitern der modernen Naturwissenschaften wie von

Fourier, Kelvin, Maxwell, Kekule, Butlerow und vielen weiteren haben eine essentielle Grundlage der mathematischen Beschreibung der wichtigsten physikalischen und chemischen Prozesse (also **mathematische Modelle**) geschaffen [R5]. Neben den Naturwissenschaften haben auch die Gesellschaftswissenschaften wie die Ökonomie (z.B. die Arbeiten von Kehne), Politik (Marx), die Sozialwissenschaften u.a. die mathematische Modellierung als Basiswerkzeug übernommen.

Die bekanntesten mathematischen Modelle basieren auf den Systemen algebraischer und differentialer Gleichungen und werden durch viele Kategorien klassifiziert, wie z.B. lineare und nichtlineare, mit konzentrierten und verteilten Parametern, diskrete und kontinuierliche, usw. In vielen Fällen gibt es keine explizite theoretische Lösung für die mathematischen Modelle, was im Wesentlichen an ihren Nichtlinearität liegt. Dies ist auch der Fall für die im Jahre 1822 veröffentlichte Navier-Stokes-Gleichungen (I.1) – eines der grundlegenden mathematischen Modelle der Physik, das Dynamik der viskosen Ströme in Flüssigkeiten und Gasen beschreibt:

$$\begin{aligned} &\frac{\partial \vec{V}}{\partial t} = -(\vec{V}\cdot\nabla)\vec{V} + \nu\Delta\vec{V} - \frac{1}{\rho}\nabla P + \vec{g}, \\ &\nabla\cdot\vec{V} = 0, \end{aligned} \tag{I.1}$$

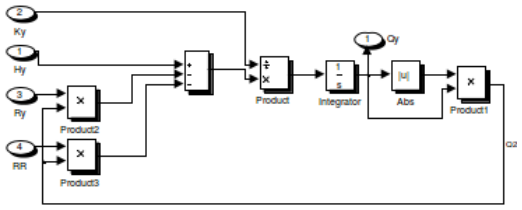
mit dem Druckterm *P*, dem Stromfeldvektor *V*, der Zeit *t*, der kinematischen Viskosität ν, Schwerkräften *g* sowie Nabla- (∇) und Laplace- (Δ) Operatoren. Diese Gleichungen liegen auch den Modellen, die im Rahmen dieser Arbeit für die akuten Aufgabestellungen des ausgewählten Forschungsgebiets (Bergbau) erarbeitet werden, zu Grunde.

In den Fällen, in denen das mathematische Modell über keine analytische Lösung verfügt, werden spezielle Techniken und Methoden der **Numerik** eingesetzt – eines Teilgebiets der Mathematik, welches sich mit approximativen Lösungen kontinuierlicher dynamischen Probleme beschäftigt. Den meisten numerischen Methoden liegt das bekannte Prinzip der linearen Interpolation zu Grunde, welches in Form eines numerischen Verfahrens (wie z.B. Finite-Differenzen, Finite-Elemente, Finite-Volumen u.v.m.) auf das ursprüngliche komplexe Modell (z.B. in der Form eines

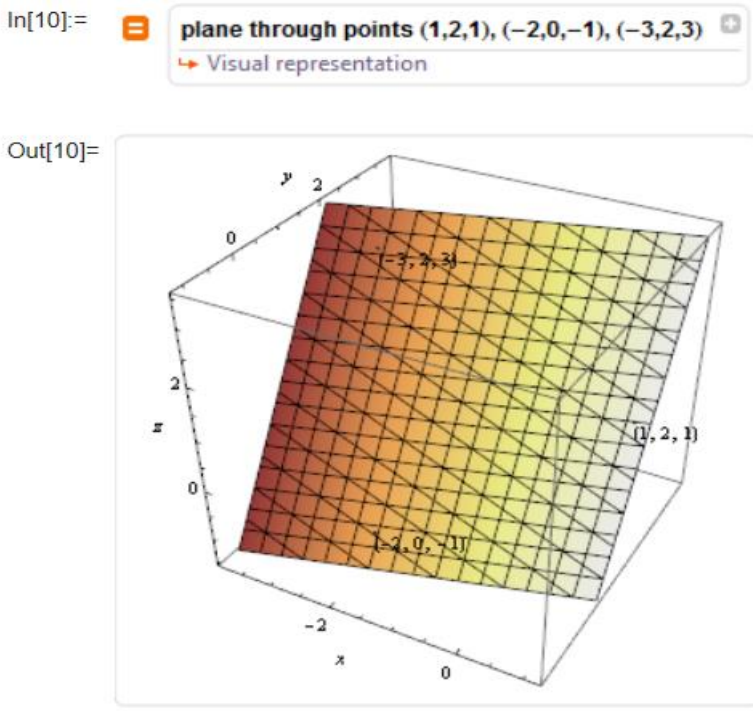
Systems nichtlinearer Differentialgleichungen) angewendet wird und zur Erstellung einer Reihe einfacherer äquivalenten Sub-Modelle führt. In der nächsten Lösungsphase werden die Gleichungen der Sub-Modelle z.B. mit Hilfe eines numerischen Integrierungs-Verfahrens (z.B. Euler-, Runge-Kutta-, Adams-Baschfort-Verfahren u.v.a.) zeitauflösend gelöst. Alle diesen Methoden gehören z.Z. zum Alltagswerkzeug der Simulationstechnik und werden in jedem technischen oder wissenschaftlichen Bereich aktiv angewendet.

Mit der Entstehung des ersten hinreichend schnellen Rechners (in 1960er Jahren) und der damit verbundenen neuen Branche der Wissenschaft – des Scientific Computings – hat die Modellierung eine neue Funktion erworben – die **Simulation**. Die Computerprogramme, die die mathematischen Modelle in eine Programmiersprache implementieren – also die Simulationsmodelle – haben die Simulation auf eine prinzipiell neue Ebene gebracht. Der Einsatz der Rechentechnik lohnte sich insbesondere für die Modelle, die eine iterative numerische Lösung (z.B. aufgrund einer Nichtlinearität der Basisgleichungen) und eine entsprechend große Anzahl an durchzuführenden Rechenoperationen benötigen. Solche komplexen Modelle sind zum Standard in Bereichen wie den Ingenieurwissenschaften geworden und haben sich zu eigenständigen Klassen von Simulationsanwendungen wie **CFD** (*Computational Fluid Dynamics*), **MD** (*Molecular Dynamic*) und anderen herausgebildet. Unter anderem wurde die Navier-Stokes-Gleichung zu einem ultimativen Ziel der CFD-Simulation, wofür zahlreiche Simulationspakete von Computersimulationspionieren wie *Francis Harlow* entwickelt wurden.

Um die Entwicklung der Simulationssoftware zu vereinfachen, wurden neue Programmiersprachen entwickelt, die dediziert auf Simulationsprobleme zugeschnitten sind – die **Simulationssprachen**. Die Simulationssprachen ermöglichen eine ganze Reihe nützlicher Techniken zur Erstellung von Simulationssoftware, wie z.B. Block-, Gleichungs- oder Objekt-orientierte Ansätze, die in die aktuell verbreiteten Paketen wie *Maple*, *Mathematika*, *Mathcad*, *Matlab*/*Simulink*, *ACSL*, *Modelica* u.v.m. implementiert sind (**Abbildung 2**).



(a) Blockorientiertes Modell von Simulink



(b) Gleichungsorientiertes Modell von Mathematica (Quelle: wolfram.com)

```
classdef Visitor < AbstractVisitor
    methods
        function run(obj,graph)
                  //Runner as in normal visitors
        end
        function result = visit(obj,element,graph)
            if (isa(element,'Subsystem'))
                result = obj.visitSubsystem(element,graph);
            elseif (isa(element,'SimpleBlock'))
                result = obj.visitBlock(element,graph);
            end
        end
        function result = visitSubsystem(obj,element,graph)
            result=0; //Will be filed in the child classes
        end
        function result = visitBlock(obj,element,graph)
            result=0; //Will be filed in the child classes
        end
    end
end
```

c) Objekt-orientiertes Modell von Matlab (Quelle: [R81])

Abbildung 2. *Beispiele der formalen Modelldarstellung in verschiedenen Simulationssprachen.*

Die 1990er Jahre zeichneten sich durch eine rasant ansteigende Komplexitäts-anforderung an die Modelle aus, die eine neue herausfordernde Anwendung auf Bereiche der Wissenschaft und Technik wie die Klimaforschung, die Ingenieurs-wissenschaften, die Biotechnologie, die Ökologie und weitere gefunden hatten. Die Komplexität der Modelle, also genauer gesagt – die Anzahl ihrer approximierenden Gleichungen, hat dazu geführt, dass die Kapazität der herkömmlichen Rechner (deren Leistungssteigerung maßgeblich vom Mooreschen Gesetz limitiert war) absolut unzureichend war. Diese Diskrepanz zwischen den Anforderungen der Simulationssoftware und der Leistung der Rechner hat die Entwicklung des Parallelen und Höchstleistungsrechnens und des dazu gehörigen „High Performance Computers (HPC)“ in Schwung gebracht.

Der erste als Höchstleistungs- anerkannte Rechner wurde von der amerikanischen Firma Cray 1962 produziert. Der erste Cray-Rechner trug den Codenamen Atlas und verfügte über die Leistung von nahezu 1 MFlop/s[1]. Der erste in Europa erschienene Höchstleistungsrechner war auch von Cray – der Cray-2 Rechner wurde im Höchstleistungsrechenzentrum Stuttgart im Jahr 1986 installiert. Die Stuttgarter Cray-2-Installation verfügte schon über 4 Prozessorkerne, die die Leistung von nahezu 2.000 MFlop/s (oder 2 GFlop/s) erbracht hatte. Seitdem hat die Leistung der neuentwickelten Systeme deutlich zugenommen (**Abbildung 3**). Die modernen Systeme[2] übertreffen diese Leistung zwar in einem vielstelligen Faktor, stehen aber im Grunde genommen vor dem selben Problemen wie ihre Vorgänger aus den 1980ern – der effizienten Nutzung der parallel-aufgebauten Hardware von (parallelen) Simulationsanwendungen, inkl. der Aufgaben der Problemdekomposition, Synchronisierung der parallelen Softwareeinheiten, Kommunikation zwischen ihnen usw.

[1] Flop/s bezeichnet „Fließkomma Operationen pro Sekunde“. 1 MFlop/s = 1000 Flop/s

[2] z.B. das HPC-System Hawk vom Hersteller HP-Electronics, welches im Höchstleistungsrechen-zentrum Stuttgart 2019 installiert wurde, verfügt bereits über 27 PetaFlop/s (1.000.000 MegaFlop/s) Leistung

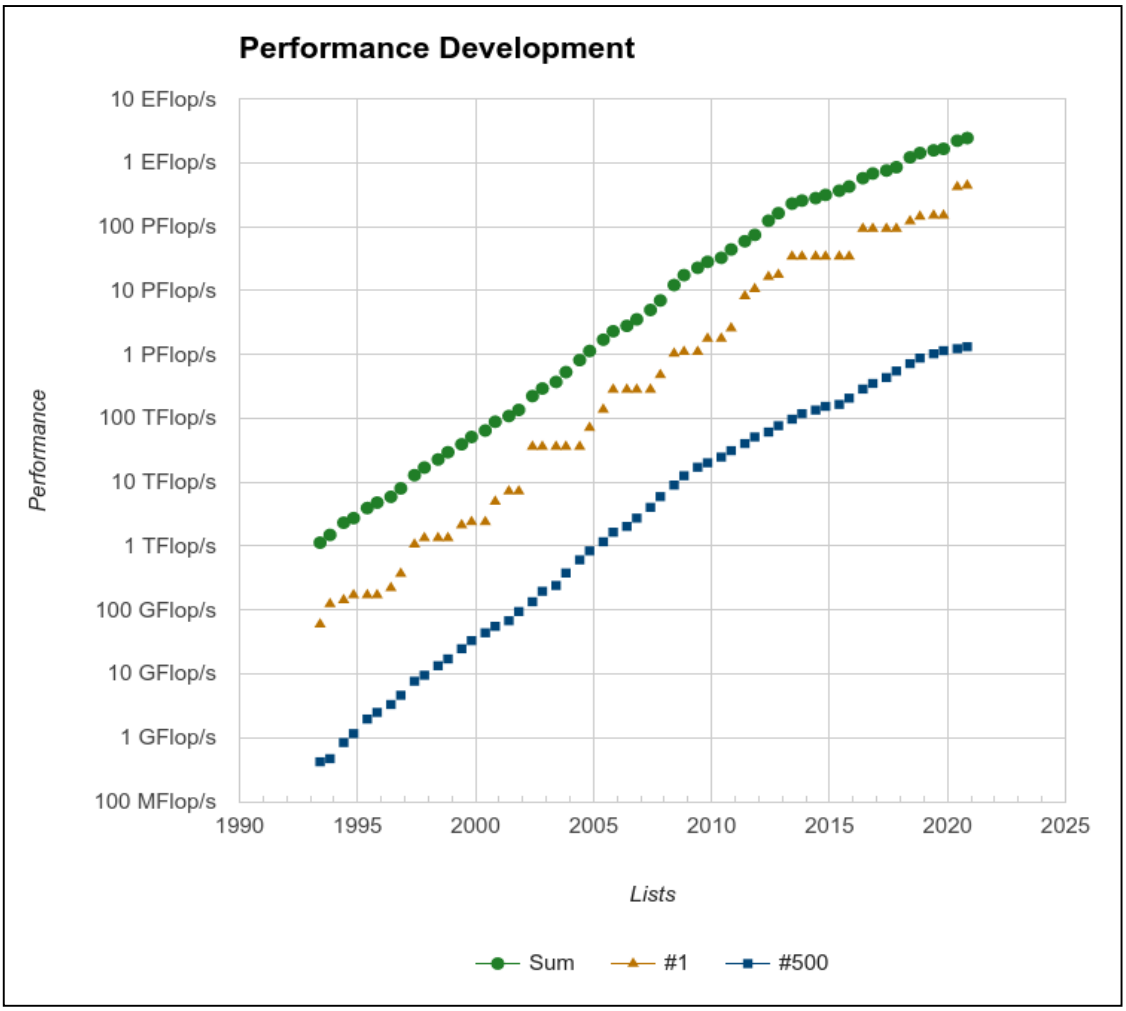


***Abbildung 3**. Leistung der größten Superrechner*[3].

Eine Alternativentwicklung der Recheninfrastrukturen für massiv-paralleles Rechnen ist im Bereich des **Cloud Computing** angesiedelt, das im Gegenteil zu HPC auf vielen verteilten Systemen mit deutlich geringerer Einzelleistung basiert. Anders als HPC, welches auf weniger Applikationen aber auf enorme Leistung orientiert ist, steuert das Cloud-Nutzungsmodell viele Nutzer mit eher geringeren Leistungsanforderungen an. Jedoch hat das Cloud-Modell in den letzten Jahren in der Simulationscommunity deutlich an Attraktivität gewonnen, was unter anderem an seinen vielversprechenden Programmieransätzen wie **Mikroservices** liegt. Die Mikroservices zeichnen sich durch eine einfache Installation auf heterogenen Cloud-Systemen aus, die zwar weniger leistungsstark als die HPC-Architekturen sind, aber eine „rund-um-die-Uhr-Verfügbarkeit“ anbieten und in unmittelbarer Nähe an die Daten (welche den

[3] Quelle: TOP500.org [W1].

für die Simulation notwendigen Input bereitstellen) ausgeführt werden können, z.B. mittels tragbaren „Ein-Platinen“ Hardware-Plattformen wie *Odroid*[4]. Weitere Vorteile der mobilen Plattformen sind die üppige Ausrüstung mit Kommunikationsinterfaces, die Verfügbarkeit spezieller Prozessor-Units (wie z.B. eine Extra-FPU) und ein geringer Energieverbrauch, was die Integration in die Betriebsbedingungen von Technologie-anlagen ermöglicht (**Abbildung 4**).

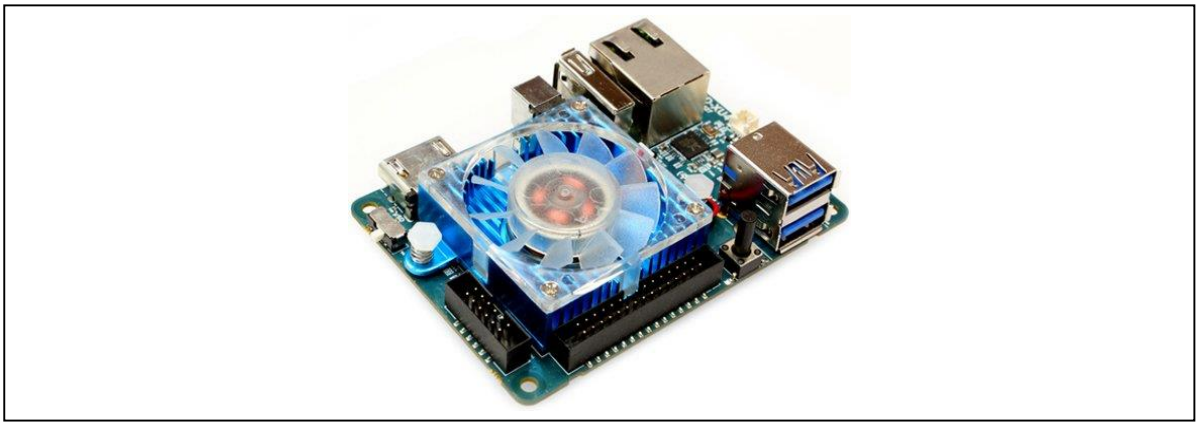


Abbildung 4. *Portables Rechnersystem Odroid-XU4.*

Die Mikroservices verfügen über eine weitere wichtige Eigenschaft für die Simulationsanwendungen – Interoperabilität, welche ihre Ausführung auf einem breiten Spektrum von Hardware ermöglicht – von den eingebetteten Systemen bis hin zu einer generischen Cloud und spezialisiertem HPC. Damit lassen sich die beiden wichtigsten Anforderungen der Nutzer erfühlen – die Wirtschaftlichkeit der Anwendungen und die hohe Leistung der rechenintensiven Simulationsabläufe, was die Simulation insbesondere attraktiv für Nutzer aus der Industrie und KMUs (Kleinen und Mittleren Unternehmen) macht.

Die Mikroservicearchitekturen bieten sich insgesamt als eine effektive Lösung zu den beiden wichtigsten Herausforderung der modernen Simulationstechnik an (wie z.B. erläutert von Bauernhansl et al. in [W2]): i) die Integration in zahlreiche Sensorik-Geräte, die in der Produktion eingesetzt werden, und ii) die Realisation in Echtzeitszenarien der modellgetriebenen Produktion.

[4] Odroid XU-4 bietet 8 heterogenen CPU-Kerne, 2 GB RAM, Gigabit Ethernet Port, WLAN, von ca. 7.8 TFLOP/s Leistung an.

1.3 Entwicklung der Prozessautomatisierungstechnik

Automatisierung ist zweifelsfrei einer der treibenden Faktoren industrieller Produktionsentwicklung und bezeichnet die Eigenschaft der technischen Systeme, selbstständig (im Sinne der Abwesenheit des menschlichen Operators) nach einem im Vorfeld vorgegebenen (also vorprogrammiertem) Algorithmus zu handeln. Das selbständige Handeln ist damit die wichtigste Eigenschaft eines Automatisierungssystems im Kontext des autonomen Handelns.

Ähnlich wie im Falle der Simulationstechnik, liegen die methodologischen Ursprünge der Automatisierung weit zurück in der antiken Zeit. Noch in Aristotelischen Werken wurde der Begriff „Automatisierung“ im Kontext von „autonomen Handeln" der zu entwickelnden technischen Systeme benutzt. Die Geschichte der Automatisierungstechnik in der Form wie wir sie jetzt kennen fängt aber erst mit der massiven Industrialisierung im 18. Jh. an. Die Betreibung komplexer Maschinen und Anlagen, die einen Vorgang der Umwandlung eines technischen Prozesses von einem Anfangsstand zum vorgegebenen Endstand) umsetzen, benötigte neue Ansätze zur Bewertung des Prozesszustandes sowie zu seiner Beeinflussung. Die Erarbeitung solcher Ansätze wurde zum Ziel von zwei wichtigen Bereichen der Forschung und Innovation – Regelungs- und Steuerungstechnik, die die Voraussetzungen dafür geschaffen haben, den Produktionsablauf immer mehr zu automatisieren. Die Automatisierungsfortschritte waren die Antreiber und Begleiter der so genannten „industriellen Revolution“ und sind von drei wichtigen Meilensteinen markiert: Einsatz der ersten mechanischen Maschinen am Ende des 18. Jh., Einführung der automatisch gesteuerten industriellen Fließbänder am Ende des 19. Jh. und Entwicklung der programmierbaren Rechen- und Speichertechnik in der zweiten Hälfte des 20. Jh..

Die Automatisierung basiert auf dem Fundament, welches von der Regelungstechnik geschaffen wurde, mit den wichtigsten methodischen Konzepten und technischen Mitteln wie Sensoren, Aktoren und Verarbeitungssystemen, die in einen gemeinsamen Digitaldatenaustauschkreis mittels eines Kommunikationssystems eingeschlossen sind. Während die Sensoren die wichtigen Informationen über den

Stand des technischen Systems (oder Prozesses) an das Verarbeitung- (Steuerung-) System liefern, werden vom Letzteren die Steuersignale gebildet und an die Aktoren zur Umsetzung am Steuerobjekt über das Kommunikationssystem übermittelt. In den modernen Automatisierungssystemen unterscheidet man zwischen mehreren Kategorien der automatischen Steuerung, die sich nach dem Regelverfahren (kontinuierlich, diskret, relais-basiert), der Nutzung der Sensorinformation (mit und ohne Regelkreis), dem Automatisierungsgrad (offline-Betrieb, online/open-loop-Betrieb, online/closed-loop-Betrieb), der Umwandlung der Koordinaten (deterministische und stochastische, lineare und nicht-lineare), dem Typ des grundlegendes Modells (deterministisch, statisch, adaptiv), usw. klassifizieren lassen [R6].

Eine entscheidende Eigenschaft des **Automatisierungssystems** (**AS**) ist die Echtzeitfähigkeit. Das heißt, dass die Zeiten, die ein Automatisierungssystem für jeden Steuerungsvorgang benötigt, deterministisch (also mit einem vorhersehbaren zeitlichen Ablauf) sind. Die Einhaltung der Echtzeitanforderungen zusammen mit der Gewährleistung der Richtigkeit des Prozessablaufs und der Ergebnisse garantiert, unter anderen Bedingungen, die AS-Zuverlässigkeit und Sicherheit. Das Konzept der Echtzeitprogrammierung, elaboriert u.a. von Halang et al. in [R7], hat heutzutage einen großen Nutzen in vielen Bereichen der kommerziellen oder der technisch-wissenschaftlichen Datenverarbeitung gefunden, insbesondere für moderne eingebettete Systeme.

Eine wichtige Nische in der Funktionierung der modernen Automatisierungstechnik besitzt die Simulationstechnik, welche zu einem unersetzlichen Werkzeug der Begleitung des AS während ihres gesamten Lebenszyklus geworden ist: von der Projektierung (z.B. zur Einschätzung der Effektivität verschiedener Regelungs-strategien für die grundlegenden physikalischen Prozesse), über die Inbetriebnahme (bspw. zur Validierung der Korrektheit der AS-Kontrolleinstellungen), bis hin zum industriellen Einsatz (z.B. zum Testen der Funktionalität entlang der gesamten Werteschöpfungskette). Die Integration der Simulationstechnik und der Automatisierungstechnik hat in den letzten Jahren rasant zugenommen, sodass man zurzeit sogar von einer modell-getriebenen Automatisierungstechnik spricht. Dabei

spielt die Echtzeitdatenerfassung aus den „intelligenten Sensoren“ und ihre Bearbeitung, die möglichst nah am Steuersystem erfolgt, die zentrale Rolle. Diese Integration soll im Gegenteil wesentliche Vorteile für die Simulationstechnik anbieten, in dem die Modellexperimente die wichtigen Erkenntnisse über den in der Realität erreichten Zustand der physikalischen Objekte gewinnen. So wird eine „Brücke“ zwischen der realen und der modellierten Welten geschaffen, die eine Querschnittsdisziplin zwischen der Automatisierungs- und der Simulationstechnik bildet (**Abbildung 5**). Zum Beispiel werden Automatisierungssysteme eingesetzt die dynamischen Modelle bereits in der Planungsphase zur Einschätzung der Eigenschaften des physikalischen Objekts verwenden (wie etwa von Svjatnjy [R8] für die Analyse von Bergbaubewetterungsnetzen).

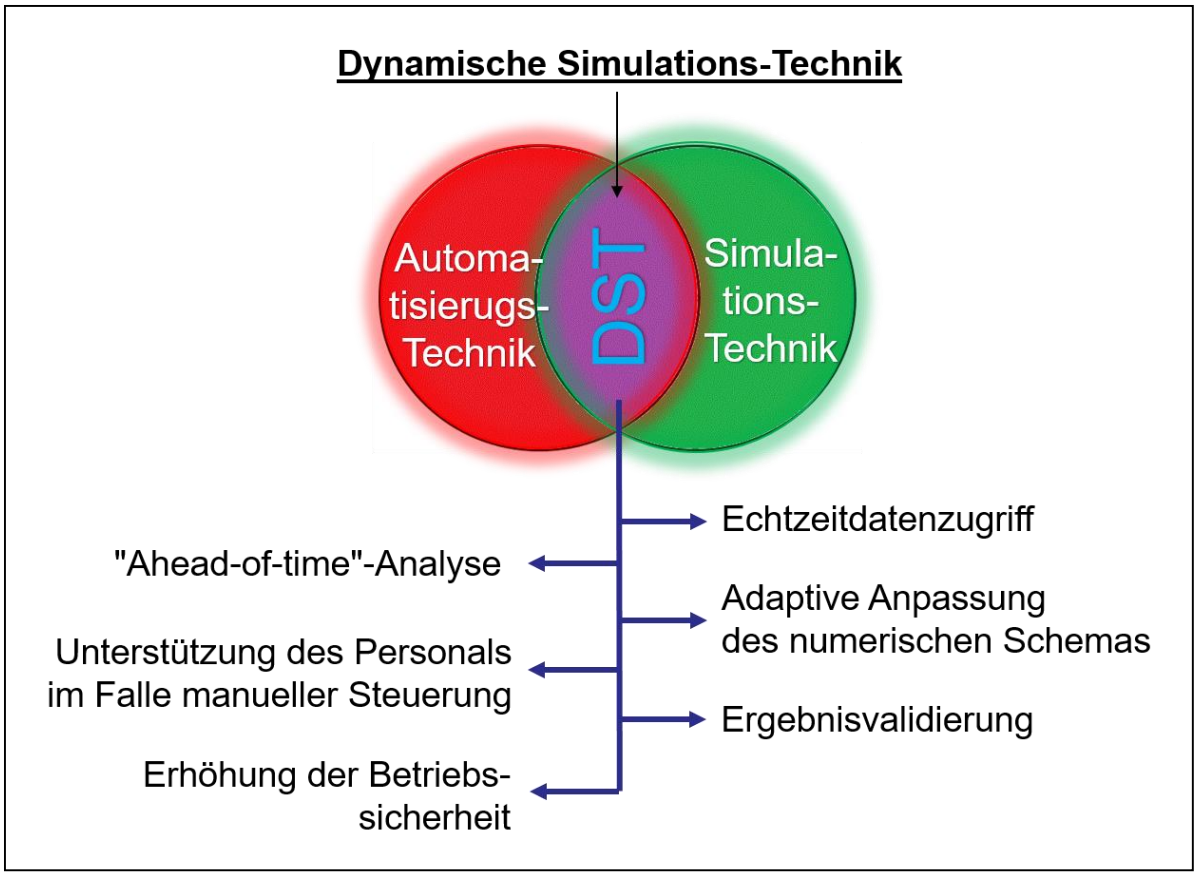


Abbildung 5. *Positionierung der Dynamischen Simulationstechnik als fachübergreifendem Bereich zwischen der Automatisierungs- und der Simulationstechnik.*

Als großes Ziel bleibt für die Zukunft eine vollständige Integration der Simulationstechnik in die AS-Funktionierung unter der Berücksichtigung der Echtzeitaspekte. Die

Echtzeitsimulation wird mit Sicherheit ein wichtiger Meilenstein für die Entwicklung neuer Automatisierungssysteme, unter anderen für sicherheitskritische Systeme verschiedenster Ordnung (von den kleineren automatisierten Fahrzeugen zu den riesigen industriellen Anlagen). Die Erreichung dieses Ziels kommt aber in Hinsicht auf die immense Steigerung der Digitalisierung (insbesondere der Rechen- und Kommunikations-Technik) im Industrieumfeld immer näher. Wissenschaftliche Disziplinen wie Mechatronik (betrachtet die Verschmelzung von Steuertechnik und Mechanik) sowie die Ansätze in „Hardware-in-the-Loop“ (HIL) und „Internet-of-things“ (mit dem Konzept von den vernetzten „intelligenten“ Sensoren – also solchen, die über eine Rechen- und Kommunikationseinheit verfügen) sorgen für eine neue Etappe der industriellen Revolution des 21. Jh., die sich vor allem durch das Konzept von Cyberphysischen Systemen auszeichnet.

Als ein **Cyberphysisches System (CPS)** bezeichnet man ein technisches System, dessen mechanische und elektronische „physische“ Teile mit informations- und software-technischen „Cyber“ -Komponenten über eine Dateninfrastruktur (Computernetzwerk) verbunden sind. Eine Anbindung der industriellen Sensornetze mit Recheneinheiten über das Internet ist momentan eine der innovativsten Forschungsrichtungen der Wissenschaft und Technik und spiegelt die zukünftige Entwicklung großer, verteilter, komplexer Systeme in vielen Industriebereichen wie Energie, Ökosysteme, Ventilation, Konstruktion usw. wider.

In Deutschland wird die CPS-Strategie von einem Zukunftsprojekt der Bundesregierung verfolgt, bekannt als Industrie-4.0 [W3]. Der Schwerpunkt von Industrie-4.0 liegt auf Erhöhung des Digitalisierungsgrads der modernen industriell-technischen Systeme durch Einsetzung von Mess-, Steuer- und Regelungsgeräte sowie von Anwendungen für sie. Ein wesentlicher Beitrag zur Erreichung dieses Ziels soll von der Simulationstechnik geleistet werden, in dem die Komplexität der Steuerungsarchitekturen durch vereinfachten und interaktiven Einsatz der analysereichen Simulationstools reduziert werden soll.

1.4 Bewetterungssysteme als Objekte der Automatisierung und Modellierung

Die Energieversorgung Deutschlands befindet sich auf dem Weg zur grundlegenden Reformierung: So sollen nach Angaben des BMWI [W4] bis 2025 rund 40% und bis 2050 mindestens 80% der gesamten Elektroenergie aus regenerativen Quellen erzeugt werden. Diese Tendenz wird auch weltweit fortgeführt. Bis aber dieses globale Ziel erreicht ist, bleiben Erdöl und Kohle die wichtigsten Energieträger industrieller Wirtschaft. Aktuelle wird der deutsche Primär-Energieverbrauch zu 80% von fossilen Energieträgern abgedeckt, ca. 28% von solchen der auf Kohle[5] fällt, so die BGR-Energiestudie für 2018 [W5].

Die Kohle bleibt zudem einer der wichtigsten fossilen Energiestoffe der Erde, mit dem Blick auf nachgewiesene Reserven von 1.055 Gt (Gigatonnen) weltweit (Angaben der Studie [W5]). Auch angesichts der angestrebten „Energiewende“ bleibt der Bergbau aus mindestens drei Gründen eine wertvolle Industriebranche Europas und der Welt und somit ein wichtiges Forschungsgebiet:

- Der jährliche Weltverbrauch der Kohle lag 2018 bei ca. 5,5[6] Gt und weist eine steigende Tendenz auf.
- Die Abschaffung eigener Kohleförderung führt schrittweise zur Wandlung der großen Kohleexporteure zu Importeuren. Auch Deutschland bleibt immer noch einer der größten Hartkohlenimporteure weltweit – es wurde 2018 ca. 41 Mt Steinkohle nach Deutschland importiert, so Angaben aus der Statistik der Kohlenwirtschaft e. V. [W6]. Dieser Stand soll mindestens bis 2040 beibehaltet werden, bis der volle Umstieg auf CO_2-neutrale Energieträger abgeschlossen ist.
- Hinsichtlich der zu erwartenden langfristigen Steigerung des globalen Primärenergieverbrauchs der Weltindustrie sowie angesichts möglicher Versorgungsengpässe durch eine international-instabile geopolitische Lage bei den wichtigsten Produzenten der Rohstoffe für öko-neutrale Stromgewinnung,

[5] Anteile von 10,9 % für Steinkohle und 11,1 % für Braunkohle im primären deutschen Energieverbrauch in 2018

[6] Angaben des Internationalen Energie Programms (IEA) [W7]

sollen fossile Energieträger weiterhin eine wichtige Rolle spielen dürfen. Auch wenn der komplette Umstieg auf die erneuerbare Energieversorgung erfolgreich durchgesetzt ist, gehen die Experten der EU-Kommission [W8] von ihrer unzureichenden Zuverlässigkeit und Preisstabilität aus, was ein **„duales Energiesystem**“ in den kommenden Jahrzenten erfordern wird, wobei fossile und erneuerbare Energien gemeinsam die Energieversorgung der EU-Länder gewährleisten werden. Die Beibehaltung des dualen Systems wird auch auf der nationalen deutschen Ebene gefordert, wie z.B. von Schlappei in [R9] analysiert.

Die implizierte Verstärkung der europäischen und internationalen Forschungs-aktivitäten im Fachbereich „Bergbau“ wird nicht nur angesichts möglicher Fortsetzungen der Gewinnungstätigkeiten im Steinkohlentiefbau in Europa von den Experten gefordert, sondern ist auch wichtig im Hinblick auf große Probleme bei der Entwicklung der Kohlereserven in den Ländern, die noch nicht bereit sind, auf den Einbau fossiler Energieträger völlig zu verzichten (wie z.B. China). Die volle Umsetzung der Prognose über das duale Versorgungssystem in der EU und weltweit kann nur durch ausreichende Erweiterung bestehender und Entwicklung neuer Technologien zur verbesserten Untertagegewinnung in Steinkohlebergwerken, u.a. was die Aspekte untertägiger Kohlevergasung oder Methannutzung anbelangt [R10], [R11] erreicht werden. Eine der größten technologischen Herausforderungen liegt bei der fossilen Energierohstoffförderung an der tiefen geologischen Einlagerung der Rohrstoffabbaugebiete. Im Fall der Steinkohle können Untertagebauwerke von bis zu mehreren Tausend Meter unter Tage hinunterdrängen.

Damit sowohl die Arbeitsbedingungen unter Tage für die funktionierenden Objekte als auch die Sicherheitsanforderung für die bereits stillgelegten Werke gewährleistet werden können, sollen die Betriebspunkte eines Bergwerks ständig mit Frischwetter (Luft) von der Oberfläche versorgt werden, was in der Bergbauterminologie als **Bewetterung** bezeichnet wird. Die Bewetterung erfolgt mittels Wettertechnik, wie z.B. Grubenventilatoren. Die Entwicklung des untertägigen Abbaus von Rohstoffen ist abhängig von Eigenschaften der zunehmenden Abbauteufen, komplexer werdender Grubengebäude sowie wachsendem Wettermengenbedarf. Hinsichtlich höherer

Betriebskosten (bis zu 50% des Gesamtenergiebedarfs des Bergwerks, z.B. nach Angaben von Clausen [R12]), ist die Effizienz und Effektivität der Bewetterungsmaßnahmen von einer besonders großen Bedeutung. Neben der Funktion der bedarfsgerechten Frischluftversorgung zur Aufrechterhaltung eines angenehmen Arbeitsklimas (für die Bergleute sowie Maschinen) an den Stützpunkten des Bergbaus, wo die durch Diffusion entstehende natürliche Luftzirkulation unmöglich ist, trägt die Bewetterung maßgeblich zur Arbeitssicherheit unter Tage bei, indem die auftretenden Schadstoffe (wie leichtentzündendes CH_4/Methangas) aus den Abbaugebieten abgeleitet werden. Eine Überschreitung der zulässigen Höchstkonzentrationen des CH_4 kann zu Vergiftungen des Bergbaupersonals führen und gewaltige Explosionen der Gas-Luft-Gemische verursachen. Die Letzteren sind die häufigste Quelle aller Unfälle in Bergbauwerken, öfters mit zahlreichen Todesopfern – die durch Methanexplosion entstandenen Feuerwalzen werden mit hohem Druck kilometerweit in den Schächten verbreitet und verursachen dadurch großen Schaden. Der letzte große Unfall in einem Steinkohle-Bergwerk ereignete sich im ukrainischen Lugansk (Angaben aus [W9]) aufgrund einer Schlagwetterexplosion, was zu 19 Todesopfern geführt hatte. Weltweit rechnen Experten mit mehr als 10.000 Todesfällen in Bergwerken durch das Phänomen der Schlagwetterexplosion in der Geschichte des Bergbaus [R13].

Die Entgasung der Abbauorte ist daher vor Herausforderungen der **Dynamik der Gasbildung** gestellt: Das in großen Mengen in den Schächten- und Strecken-Ablagerungen der Grube konzentrierende Methan wird während des laufenden Kohleabbauprozesses freigelassen und kann daher kaum im Voraus prognostiziert werden. Die wichtigste Erkenntnisquelle über die Entwicklung der aero- und gas-dynamischen Prozesse in Bergbauwerken bleibt daher die Modellierung. Die Strömungsdynamikmodellierung in Elementen des Bewetterungsnetzes beruht auf der Betrachtung der Eigenschaften der grundlegenden, gegenseitig wirkenden Prozesse an Abbauorten wie z.B. instationäre Luftströmung entlang der Grubenräume, Filtration der Luftströmung durch das Kohleabbaugebiet, Wechselwirkung zwischen dem im Kohleabbaugebiet gebildeten Methan mit Luftleckstrom aus angrenzenden

Ventilations- und Förderstrecken, usw. (siehe Arbeit von Hanf [R14]). Mathematische Modelle der Bergbaubewetterungs-Strömungsdynamik berücksichtigen Zusammenhänge zwischen Strömungsgeschwindigkeit, Drücken, Dichte und Temperatur der Luft und Gase (also einer dispersen Mehrphasenströmung), wie z.B. von der grundlegenden für alle Strömungsmodelle Navier-Stockischen-Gleichung (I.1) vorgegeben. Die Gleichung wird durch Definition der Randbedingungen und Durchführung einer Approximation zur Spezifik der Bewetterungsprozesse angepasst und dadurch essenziell vereinfacht, wie sie z.B. im Werk von Abramov, Feldmann und Svjatnyj [R15] konzeptionell erarbeitet wird. Trotz dieser Vereinfachung bleibt eine analytische Lösung des Modells jedoch unmöglich, was den Einsatz numerischer Methoden fordert. Die Forschungen, die sich mit Fragestellungen der Untertagesicherheit durch Bewetterung beschäftigt, haben sich in eine besondere Forschungsrichtung entwickelt – die Bergbauaerologie.

Einen praktischen Nutzen haben die Bergbauaerologie-Modelle mit der Einführung der Prozessautomatisierung in den 1980er Jahren bekommen. Sie wurden den Funktionierungsalgorithmen der Kontrollsysteme zu Grunde gelegt, welche die großen Methangaskonzentrationen unter Tage sowie die Verbreitung des Kohlenstaubs durch ein mehrstufiges Sicherheitsverfahren überwachen und steuern sollen. Die Grubenbewetterung-Automatisierungssysteme (GAS) beinhalten folgende Basiskomponenten:

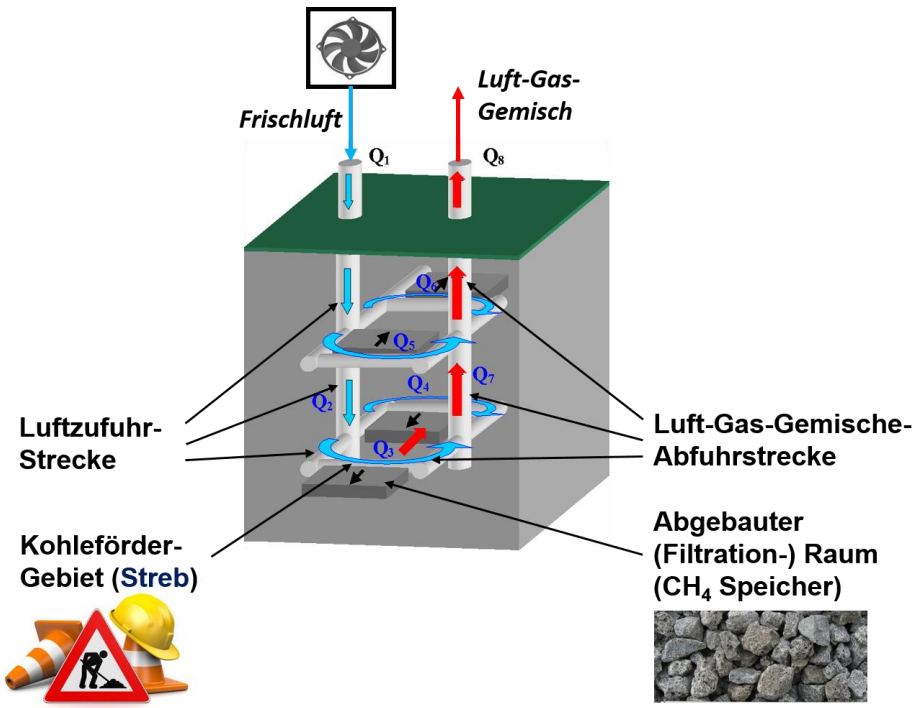
- Messstationen – Sensorgeräte, die ständig den Gasgehalt in der Grubenanlage überwachen und kritische Gaskonzentrationen an das zentrale Kontrollsystem über Tage melden.
- Steuerungssystem – eine zentrale Anlaufstelle für alle Messstationendaten sowie ein Rechensystem für die Ausführung der Algorithmen, die eine sichere Funktionierung der Bewetterung (wie die Menge der vom über Tage angesaugten Luftstroms) anhand des aktuellen Wetterstands im Ventilationsnetz mittels der Modelle bestimmt und die notwendigen Kontrolleinstellungen an die Aktuatoren leitet.

- Aktuatoren/Regler – die Steuerungstechnik der Hauptelemente des Belüftungssystems, die physikalische Vorgänge im Ventilationsnetz einwirken und damit die Bewetterung bestimmen. Die Regulierung kann auf verschiedenen Ebenen angewandt werden, wie z.B. global auf der Grubenebene mit Hilfe von einem oder mehreren Grubenventilatoren (globale Regulierung), oder lokal auf der Ebene einer Ventilationsabteilung, z.B. mit Hilfe einer Belüftungsklappe, die den Luftstrom durch einzelne Strecken des Bewetterungssystems reguliert (lokale Regulierung).

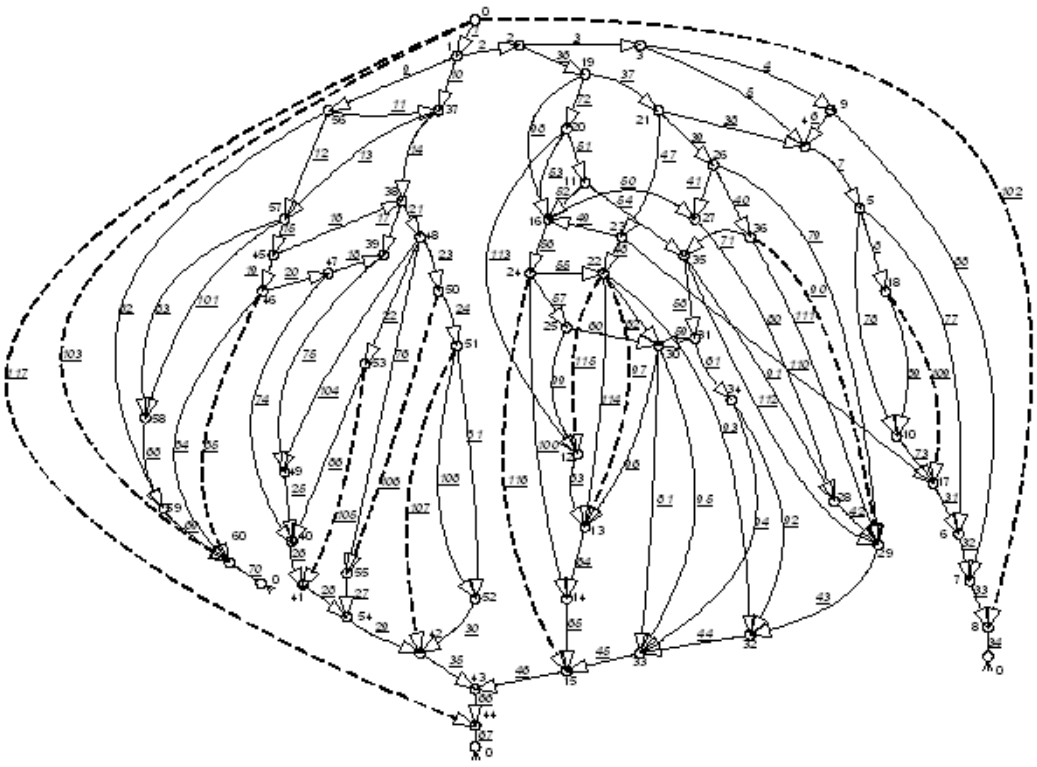
Die Ziele der GAS-Regelung umfassen die Aufgaben der Gewährleistung der ausreichenden Durchlüftungskapazität in verschiedenen Betrieben (wie z.B. Kohleabbau, Entgasung, usw.), Senkung des gesamten Energieverbrauchs und anderen, die von einer großen Bedeutung bei der Projektierung der neuen sowie Erweiterung der bestehenden Bewetterungsanlagen sind. Die Regelung basiert auf quantitativen Charakteristiken der kontrollierten dynamischen Prozesse in Grubenanlagen, die durch Messungen aber zum großen Teil aus Modellen stammen, die dynamische Prozesse in Bewetterungssystemen abbilden. Die Vielfalt der physischen Problemstellungen in verschiedenen Bergbau-Tätigkeitsrichtungen (wie Bewetterung, Elektroversorgung, Transportierung, usw.) fordert eine umfangreiche Umfassung bei den Modellen, deren Komplexität sich auf zwei Hauptfaktoren beläuft:

- Umständlichkeit der grundlegenden mathematischen Beschreibung, die den Einsatz von Numerik und die daraus folgende Notwendigkeit der Nutzung der leistungsstarken Rechenressourcen fordert,

die hohe Dimensionalität des Bewetterungssystems als Ganzes - die vernetzte Struktur der Untertageleitungen (mit bis zu Hunderten von Elementen) und die Länge ihrer einzelnen Elemente (mehrere Hunderte von Metern), die ein verzweigtes Streckennetz von Stollen, Schächten, und anderen Objekten der Bergbaubewetterung – also das Grubenbewetterungsnetz (***GBN***) – bilden (**Abbildung 6**)



(a) Hauptelemente und Struktur



(b) Netz des Bergwerks Süd-Donbass-3 als Graph mit m=117 Luftwegen und n=61 Verbindungen

Abbildung 6. *Grubenbewetterungsnetz.*

Deutschland gehört traditionell zu den Führern auf dem Markt der GAS-Technologien. Auch falls sie demnächst keinen Einsatz mehr im Inland finden sollten, dienen die in Deutschland hergestellten Sicherheitstechnologien (bspw. TLT-Turbo[7], DMT-Group[8], Woellke[9] u.v.a.) als Vorbild für den Steinkohlebergbau weltweit.

1.5 Motivierende Anforderungen und Grundideen der Echtzeitsimulation zur Unterstützung Dynamischer Systeme

Dynamik ist die wichtigste Eigenschaft des Universums, weil alles bis auf die winzigsten Bauteile unserer Existenz auf dynamische Weise verläuft und sich im Laufe der Zeit dauernd ändert. Dynamik bestimmt auch die Kontinuität aller physikalischen Prozesse, die als **Dynamische Systeme** bezeichnet werden (siehe z.B. die Definition von Denker in [R16]). Dieser fachübergreifende Begriff bezeichnet technische Systeme, in denen mit der Zeit veränderliche Zustandsänderungen ablaufen, bedingt durch natürliche Naturerscheinungen und/oder den Einfluss externer Regelungsfaktoren.

Wichtige Erkenntnisse über den Prozessverlauf in einem Dynamischen System können durch Modellierung gewonnen werden. Eine besondere Herausforderung stellen für die Modellierung und Simulation die Dynamischen Systeme, die Teile funktionierender industriell-technischen Systeme sind dar – sie bestimmen nicht nur die Komplexität der Modelle, sondern erfordern auch die aktive Beteiligung der Modelle an dynamischen Steuerungsabläufen in Echtzeit. Die Echtzeit-Anbindung der Simulation an die laufenden Dynamischen Systeme verspricht nicht nur einen Mehrwert für Automatisierungssysteme, die ihre Steuerungsabläufe durch aus den Modellen gewonnen Vorhersagen verbessern können, sondern bietet auch Vorteile für die Modelle, die meistens auf statischen Datensätzen basieren und somit nur beschränkt einsetzbar sind. Die Dynamik-Eigenschaft ist damit eine ganz wichtige Funktionalität der Modelle von Dynamischen Systemen, insbesondere für die, welche im direkten industriellen Umfeld eingesetzt werden sollen.

f[7] https://www.tlt-turbo.com/
[8] https://www.dmt-group.com/
[9] https://www.woelke-gmbh.eu/

1.5. Motivierende Anforderungen und Grundideen der Echtzeitsimulation zur Unterstützung Dynamischer Systeme

Die Modellierung der industriell-technischen Dynamischen Systeme in verschiedenen Gegenstandsgebieten beruht, trotz ihrer Vielfalt, auf einheitlichen methodologischen Prinzipien, wie z.B. von Feldmann in [R1] erläutert. Dabei folgt der Modellaufbau dem Prinzip der strukturellen Dekomposition der Teilelemente des Systems, nach welchem die Modelle einzelner Teile durch eine Topologie verbunden werden, die der Struktur der Verbindungen zwischen den technischen Systemen bzw. Prozessen entspricht (**Abbildung 7**).

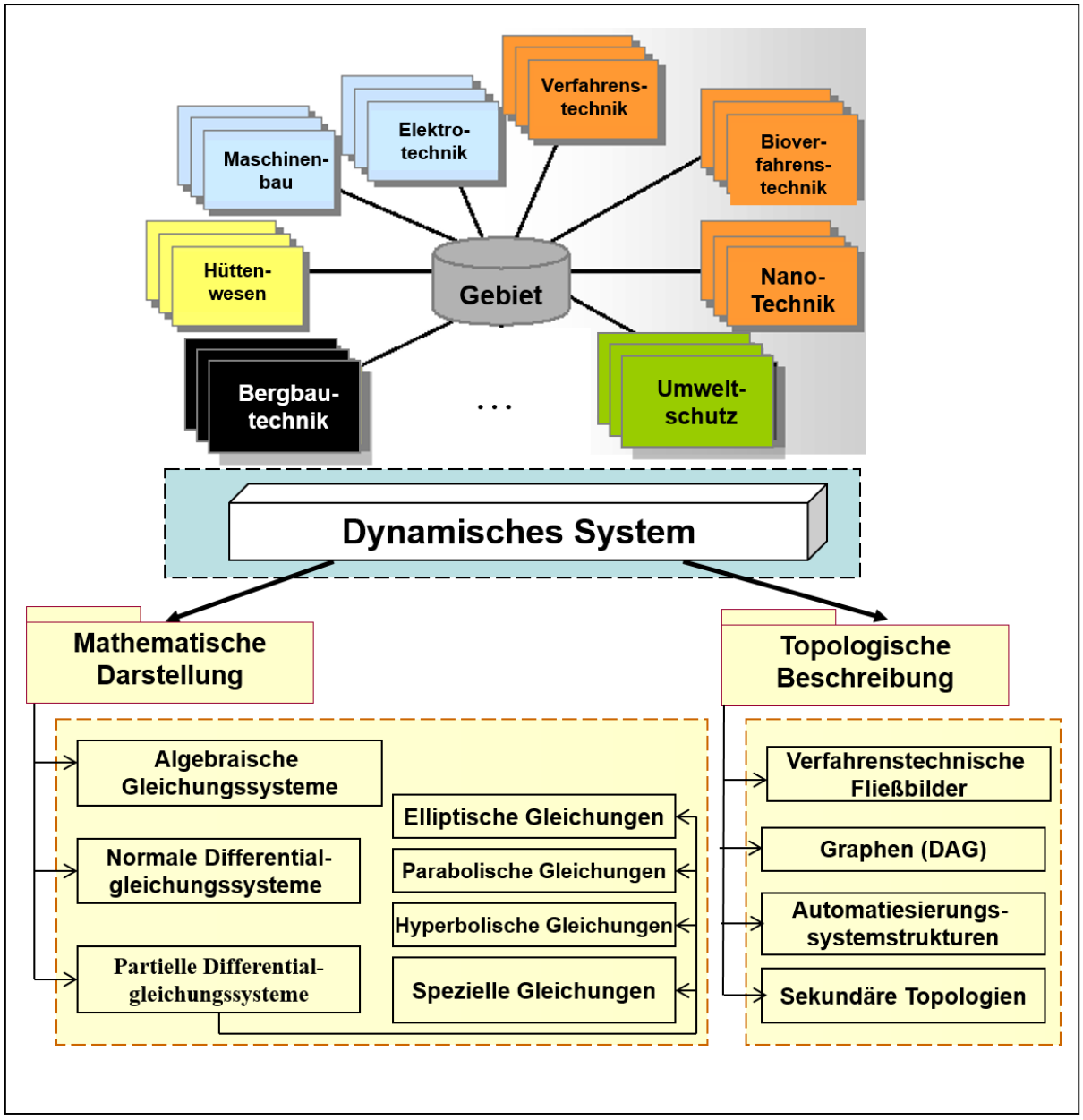


Abbildung 7. *Eigenschaften der dynamischen Systeme verschiedener Gegenstandsgebiete*[10].

[10] in Anlehnung an [R1]

Dabei kann der Modellierungsprozess potenziell auf die folgenden Problematik-Faktoren stoßen:

- Komplexe physikalische Prozessabläufe im analysierten Systemen (wie z.B. nichtlineare wechselwirkende Prozesse und Komponenten/Module mit multi-dimensionalen Massen-, Energie- und/oder Informations-Austauschflüssen) benötigen einen aufwendigen mathematischen Apparat zu ihrer umfangreichen Darstellung. Das sollte entweder in der Form von Differentialgleichungssystemen mit verteilten (beinhalten zeitliche und örtliche Differenzierung) oder konzentrierten (nur zeitliche Differenzierung) Parametern erfolgen, erweitert durch Systeme algebraischer Gleichungen.
- Nichttriviale Geometrie einzelner Strukturkomponenten des technischen Systems und viele topologische Verbindungen zwischen denen, welche die Berücksichtigung komplexer Randbedingungen bei den Modellen fordert.
- Einsatz numerischer Verfahren zur Approximation der grundlegenden Differentialgleichungssysteme (eine Zeitdiskretisierung für Modelle mit konzentrierten Parametern und sowohl eine Zeitdiskretisierung als auch Ortsapproximation für Modelle mit verteilten Parametern).
- Einsatz der Modelle für Echtzeit-Probleme benötigen eine dynamisch bedingte Rechenzeit bei einer maximal-möglichen Auflösung für ihre Algorithmen, die auf jeden Fall die Dauer der Übergangsprozesse nicht übersteigen sollte.

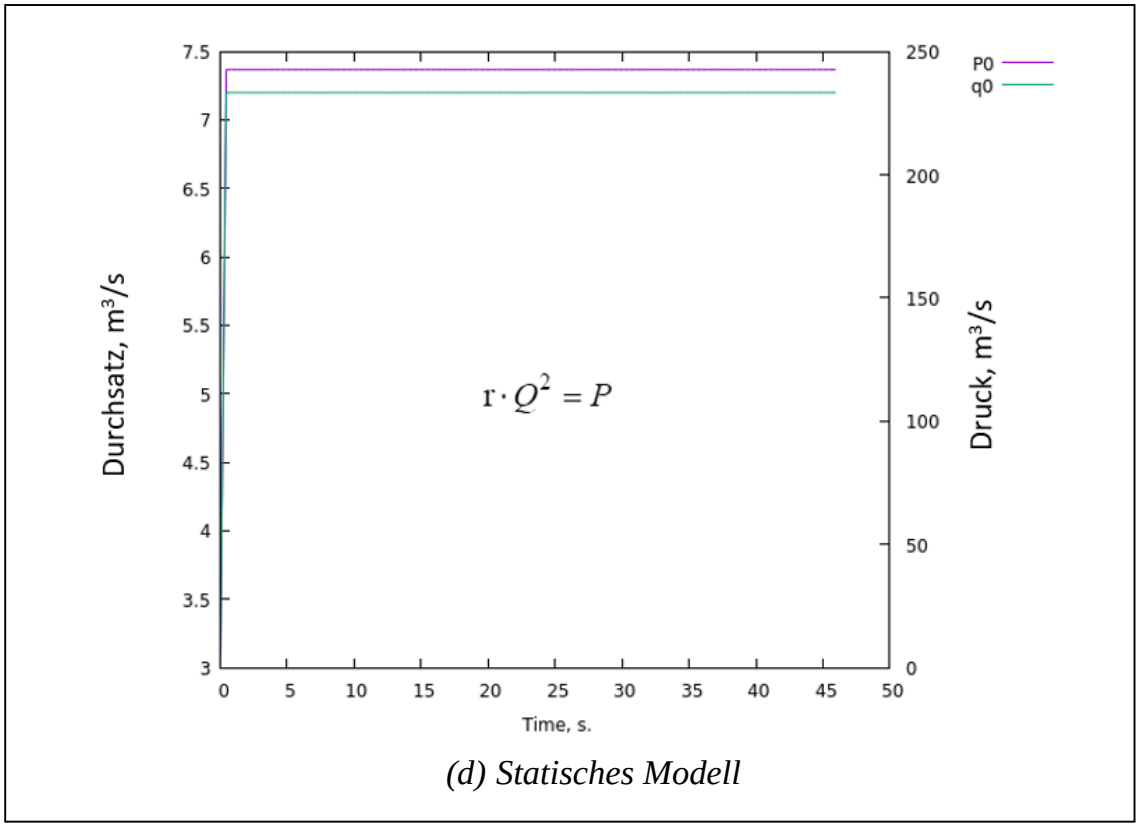
Ein klassisches Beispiel eines dynamischen Systems sind die oben beschriebenen Grubenbewetterungsnetze (Kapitel 1.4): Sie bestehen aus vielen (bis zu mehreren Tausenden) hierarchisch vernetzten Elementen (Strecken, Streben, Schähten, Schleusen u.a.), deren Konfiguration und Eigenschaften sich ständig im Laufe des Kohleabbaus verändert. Dies bezieht sich z.B. auf die wichtigsten Parameter, wie die Längen der Ventilationsstrecken, aerodynamische Resistenzen der Luftschleusen, Drucke in Verbindungsknoten u.v.m. Vor diesem Hintergrund soll die Konfiguration der Regelelemente der Automatisierungssysteme entsprechend der aktuellen Lage aktualisiert und angepasst werden. Eine solche Anpassung soll in regelmäßigen

Zeitabständen (wie z.B beschrieben von Busche in [R17]), unter Berücksichtigung der Lage der sicherheitskritischen Gasbildungsprozesse durchgeführt werden.

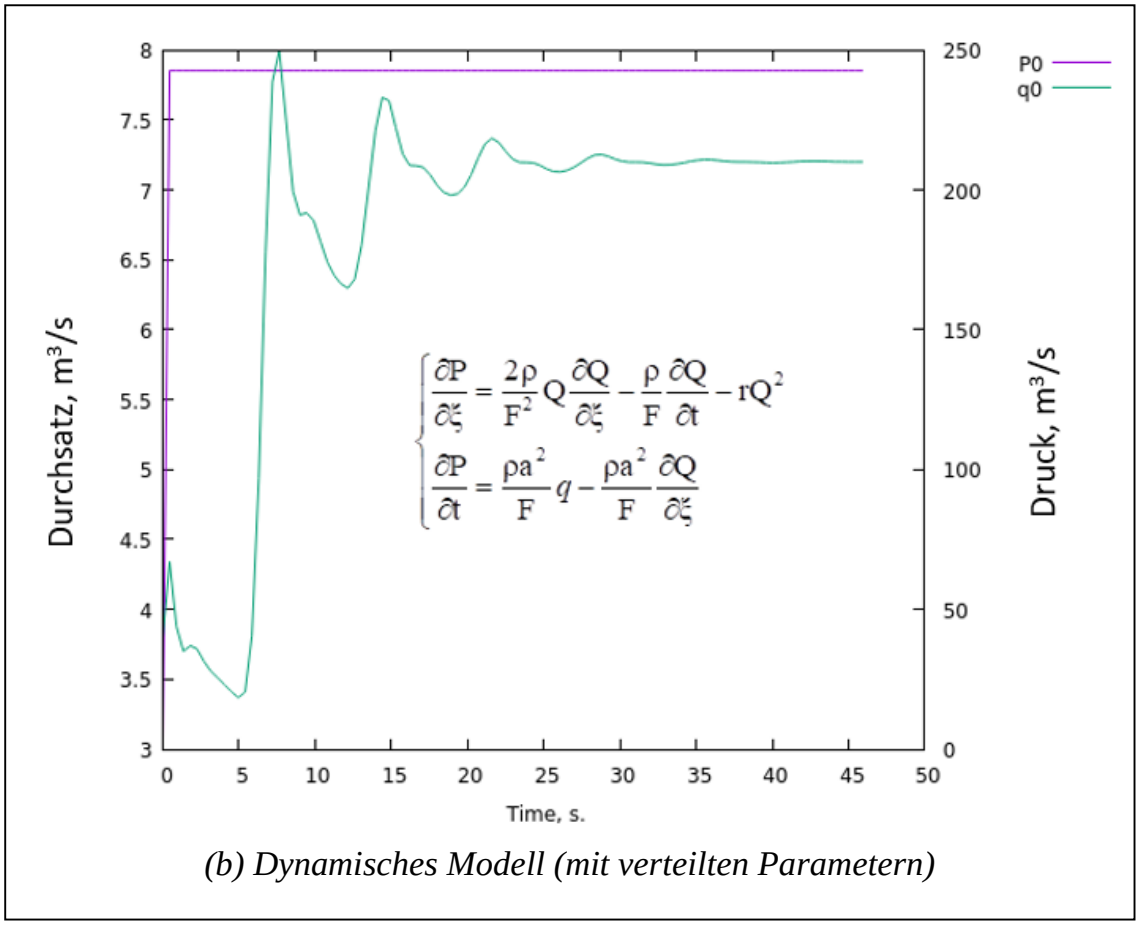
Die Erfüllung der Sicherheitskriterien bei dieser Anpassung wird in "state-of-the-art"-Steuerungssystemen nur aufgrund statischer Modelle überprüft, was zu einer mangelhaften Sicherheitsgewährleistung führen könnte. Die Anpassung der Datensätze für dynamische Modelle, also solche, die die Dynamik des Übergangsprozesses berücksichtigen (siehe Vergleich in **Abbildung 8**), werden nur selten durchgeführt (in der Regel ein oder zwei Mal pro Jahr). **Die Datenaktualität hat aber das Potenzial, einen entscheidenden Beitrag für eine bessere Erfüllung der Sicherheitsanforderung der sicherheitskritischen industriell-technischen Systeme wie der Grubenbewetterungsnetze zu leisten.**

Die Problematik der umfangreichen Simulation der dynamischen Systeme nimmt im Kontext der Evolution von klassischen automatisierten Systemen zu cyber-physischen Plattformen deutlich zu und geht weit über die Möglichkeiten bestehender Softwarelösungen hinaus: Im Nachgang zu den „traditionellen" Problemstellungen der Simulationstechnik, wie der automatischen Generierung und Lösung der groß-dimensionalen Gleichungssysteme anhand der topologischen Beschreibung, Entwicklung der parallelen und hochskalierbaren numerischen Methoden, Erstellung von Anwendungs- und Problemgebiet-orientierten Simulationsumgebungen u.v.m. sollen zusätzlich die Aspekte der kontinuierlichen Planung und Ausführung von Simulationsexperimenten, der Echtzeit-Verfügbarkeit von Simulationsergebnissen, der dezentralisierten Steuerung von Simulationsservices, der Ausführung auf verteilter, hierarchisch-organisierten Hardwareinfrastruktur, u.v.m. berücksichtigt werden.

Eine vollständige Integration der Simulation in die CPS-Infrastruktur kann als die größte Herausforderung für die Weiterentwicklung der Simulationstechnik im Rahmen von Industrie-4.0 betrachtet werden. Die Anforderungen, die bei der Erstellung der Simulationssoftware von einer besonders großen Bedeutung sind, sind wie folgt:



(d) Statisches Modell



(b) Dynamisches Modell (mit verteilten Parametern)

Abbildung 8. Vergleich der Ergebnisse von statischen und dynamischen Modellarten.

(i) **Wirtschaftlichkeit der Simulationstechnik.** Trotz steigender Förderausgaben für Höchstleistungsrechner (der schnellste Rechner 2017 soll laut Wikipedia die chinesische Regierung 272 Milliarden US-Dollar gekostet haben) bleibt die Bestrebung nach Preissenkung der einzelnen Simulationen so hoch wie nie. Dieses Ziel wird im Wesentlichen aus zwei wichtigen Perspektiven erreicht –der Erhöhung der Produktivität der Simulationssoftware (durch Optimierung und Verbesserung der Parallelisierungstechnologien) sowie der Senkung des Energie-verbrauchs der Infrastrukturen. Das Letztere bekommt gerade Aufschwung im Sinne des Einsatzes der Low-Power-Technologien – also der Hardware mit reduziertem Energieverbrauch (wie z.B. die in der Mobilgerätewelt dominierenden ARM-Prozessoren). Die Betreiber von Höchstleistungsrechnern haben ihr Interesse an der Nutzung von den energieschonenden Infrastrukturen mit der Einführung der Green500[11]-Liste signalisiert. Manche Hersteller nehmen diese Signale bereits wahr: Zum Beispiel hat die Firma Fujitsu Ende 2017 einen Umstieg auf ARM für ihre Großrechner angekündigt. Der Einsatz neuer Hardware-Architekturen fordert im Gegenzug die Anpassung der Anwendungen, was auch die Portabilität der Simulationstechnik-Lösungen stark betrifft.

(ii) **Serviceorientierung und Nutzerfreundlichkeit**. Die rasant expandierende Verfügbarkeit der Rechenressourcen sowie die neuen IT-Organisations-strategien (wie Cloud Computing) sorgen für eine steigende industrielle Nutzung der Simulationstechnik, die zunehmend als Dienstleistung über flexible kommerzielle Tarifbedingungen angeboten wird. So hat das "*S-a-a-S*" ("*Software-as-a-Service*")-Konzept in der Simulationstechnik eine Erweiterung als "*Modelling-and-Simulation-as-a-Service*" ("*MS-a-a-S*") bekommen, indem die Simulationslösungen den (industriellen) Kunden über einen Online-Dienst zur Verfügung gestellt werden. Manche Hersteller der Modellierungssoftware wie Mathworks haben mit dem Umstieg auf Cloud-

[11] Angaben von [W10]

Plattformen bereits begonnen und bieten ihre Services über die Infrastruktur eines externen IT-Dienstleisters (wie z.B. Amazon) an. Der Schwerpunkt liegt allerdings meistens auf den Modellierungsplattformen und in deutlich geringerem Umfang auf den Simulationen (Anwendungen) selbst, wie Cayircis in seiner Analyse [R18] zeigt. Eine erfolgreiche Simulationsserviceerbringung benötigt eine domänenübergreifende Modellspezifikation (für beides – Topologie sowie mathematische Modelle) und viele benutzerfreundliche Tools zur Erstellung, dem Debuggen, Testen, sowie Nutzen der Simulatoren. Spezielle service-orientierte Modellspezifikationen für die Struktur, dem Verhalten und den Eigenschaften der zu modellierenden Objekte sollen entwickelt und die Wege für ihre Einbettung in eine Cloud-IT-Umgebung erarbeitet werden.

(iii) **Engere Kopplung mit den Eingabedatenquellen**. Das am Anfang des 21. Jahrhunderts erarbeitete Konzept des "*Ubiquitous Computings*" hat die Entwicklung der Kommunikationsschnittstellen der mit den Sensoren ausgerüsteten physikalischen Objekte in Schwung gebracht. Mittels eines kleinen Software-Kommunikationsagenten haben die physikalischen Objekte eine Möglichkeit bekommen, Teil der betrieblichen IT-Infrastruktur zu werden. Eine weitere Realisierung hat dieses Konzept in der Technologie des Internets der Dinge ("*Internet-of-Things*" – "*IoT*") gefunden, welches die physikalischen Objekte und ihre Sensoren als intakte Teile einer privaten Internet-Cloud einbindet. Für die Simulationstechnik bedeutet das die Möglichkeit, relevante Informationen aus dem modellierten Abbild des Objekts erfassen und mit den Modelldaten verknüpfen zu können. Diese Kopplung hat mehrere Ziele – Verifizierung der Simulationsergebnisse, Anpassung der Parameter der Modelle, Erhöhung der Qualität der numerischen Löser und Schemata u.v.m. Im Höchstleistungsrechentechnikbereich hat dieses Konzept eine Weiterentwicklung in die Richtung der Electronic Science gefunden: Die Sensordaten konnten in die Workflow-basierte HPC-Anwendungen eingebaut werden, wie z.B. Cheptsov, Keller et al. in [R19] am Beispiel einer Elektronensynchrotron-Anlage gezeigt haben. Die besonders großen Daten ("*Big Data*"), deren

Bestände aus zahlreichen Sensoren verschiedenster Art kommen, können sogar den Einsatz der Höchstleistungsrechentechnik erfordern – die Problematik, die sich in einen Sonderbereich namens HPDA (High Performance Data Analytics) abgespaltet hat. Grundsätzlich sind die Konzepte wie "*IoT*" oder "*BigData*" eine wichtige Treibkraft der zukünftigen Evolution der Rechen- und insbesondere der Simulations-Technik. Eine koordinierte Ausführung der Simulationsexperimente mit gleichzeitiger Akquisition der Daten aus heterogener Sensorquellen bietet ein großes Potenzial an – wie im Falle des Ende 2017 angekündigten Experimentes von MPG, die einen sensationellen Beweis für die Existenz der Gravitationsquellen geliefert hat. Die Simulationstechnik soll in Zukunft offener für die Integration der externen Daten werden, was die Anforderungen an die Entwicklung neuer daten-basierten Programmiermodelle, Kommunikationsschnittstellen und Aus-führungsplattformen für „dynamische" Anwendugsszenarien fordert.

(iv) **Unterstützung der unkonventionellen und eingebetteten Hardware.** Die hohe Komplexität der Simulationsalgorithmen benötigt eine leistungsfähige Hardware. Die vollständige Simulation von großen Technologieobjekten erfordert sogar den Einsatz von Höchstleistungsrechnern. Aber in manchen Fällen sollen die Simulationen direkt bei den Kontrollobjekten durchgeführt werden, so etwa wie während eines Notfalleinsatzes durch spezielle Sicherheitskräfte. Im Falle des Gegenstandsgebiets „Ventilationsnetze" sind solche Einsätze bei einem Untertageabbruch oder einer Methanexplosion unbedingt notwendig. In solchen Fällen kann zwar keine höchste Qualität der Simulationsergebnisse erreicht werden, es wird aber ein wichtiges Kriterium zur schnellen und risikofreien Notfallbehebung geschaffen – die Verfügbarkeit einer Simulationsplattform, die anhand von aktuellen aero- und gasdynamischen Verhältnissen im Untertagebau eine Vorhersage über den weiteren Verlauf der sicherheitskritischen Prozesse, z.B. nach der Einleitung eines Notfallplanes, machen kann.

Diese Arbeit unternimmt ein Versuch, eine methodologische Grundlage zur Einbeziehung des Konzepts der Simulation auf den Bereich der cyber-physische Systeme zu schaffen und diese zur Lösung aktueller Probleme des Forschungsbereichs Bergbauaerologie in Form eines zu entwickelnden Prototyps einer echtzeit-fähigen Simulationsplattform anzuwenden.

Der angestrebte Integrationsansatz der Simulation in die cyber-physische Infrastrukturen soll viele Vorteile für die noch intelligentere Unterstützung der industriellen technischen Objekte und Anlagen, die dynamische Systeme beinhalten, erschaffen. Unter anderem soll das Modellierungspotenzial der erhöhten funktionalen Sicherheit der Automatisierungsprozesse dienen, indem diskrete Bauteile herkömmlicher Steuerprozessoren durch vielfältige Daten und Ergebnisse der dynamischen Modellierungsexperimente erweitert werden, z.B. zur Aufdeckung der auftretenden sicherheitskritischen Konfigurationen der Kontrollobjekte (wie in **Abbildung 9** skizziert).

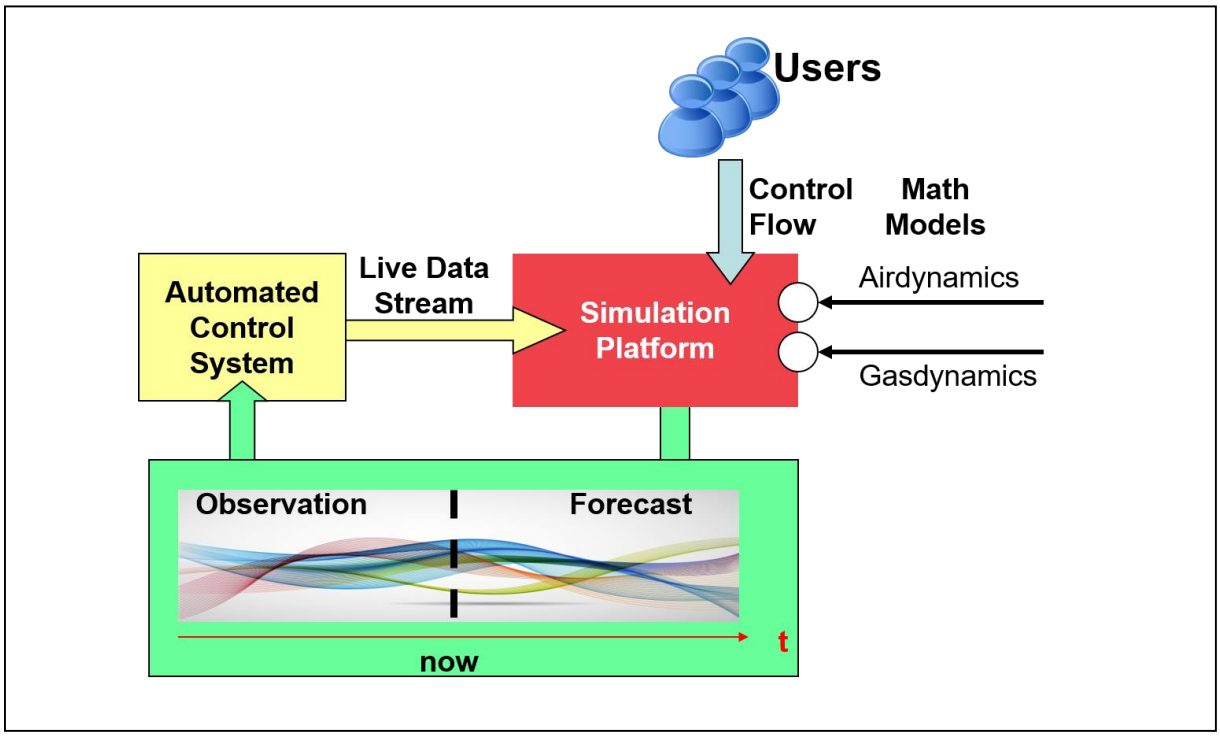


Abbildung 9. *Dynamische Simulation in Echtzeit.*

Das nachfolgende Kapitel 2 beschäftigt sich mit der Analyse der Technologien, die von maßgeblicher Bedeutung für die Erreichung dieser Vision sein könnten.

2 Stand der Technik und Forschung

Dieses Kapitel gibt einen Überblick über den derzeitigen Stand der Technik in zielweisenden Bereichen der Wissenschaft und Technik, die von einer großen Relevanz für das in dieser Arbeit verfolgte Vorhaben sind.

Der weitere Inhalt bietet zuerst einen Einblick in Cyber-Physische Systeme, die Eigenschaften der eingebetteten Systeme und digitaler Datenverarbeitungs-infrastrukturen vereinbaren, und diskutiert anschließend die Rolle der Simulation als eine mögliche Anwendung für solche Systeme (Kapitel 2.1). Es werden dann Ansätze zur Modellierung der dynamischen Systeme in der Grubenbewetterung erläutert (Kapitel 2.2) sowie die vorhandenen Lösungen zur Verteilten und Parallelen Simulationstechnik für CFD-Problemstellungen analysiert (Kapitel 2.3). Eine Übersicht der Programmiermodelle zur Entwicklung effizienter CFD-Simulationssoftware wird im Kapitel 2.4 gegeben. Abschließend werden die zielführenden Aktivitäten zur Umsetzung des Arbeitsvorhabens und seine Ergebnisse (Kapitel 2.5) dargelegt.

2.1 Simulation im Kontext von cyber-physischer Automatisierung

Moderne industrielle **Automatisierungssysteme (AS)** werden zunehmend mit fortgeschrittenen Digitalisierungstechnologien aus- und umgerüstet und befinden sich grundsätzlich auf dem Evolutionsweg zu **Cyber-Physischen Systemen (CPS)**. Dieser Begriff bezeichnet die gekoppelte und koordinierte Steuerung mechatronischer Betriebskomponenten eines physikalischen Objekts oder Systems mittels integrierbarer (bis auf die Feldebene) digitalen Rechen-, Sensor- und Kommunikationseinheiten. Nach der Definition von Broy [R20], adressieren die CPS „die enge Verbindung eingebetteter Systeme zur Überwachung und Steuerung physikalischer Vorgänge mittels Sensoren und Aktuatoren über Kommunikations-einrichtungen mit den globalen digitalen Netzen“.

Charakteristisch für die CPS-Entwicklung moderner technischen Systeme sind zunehmende Forderungen nach zusätzlichen Funktionalitäten zur Erfassung, Analyse und Verwertung der Produktionsdaten, wie z.B. von Vogel-Heuser in [R21] für

verschiedene klassische AS-Anwendungsbereiche untersucht. Die daraus folgende erhebliche Steigerung des Anteils der digitalen Komponente und ihrer Software öffnet einerseits neue Perspektiven zur Entwicklung des Unternehmens (wie z.B. von der PWC-Studie [W11] gezeigt) und fördert anderseits die Implementierung neuer Ansätze zur Produktionssteuerung. In der Praxis betrifft das insbesondere die Sensor-Anlagen, die zur Übermittlung der erfassten Daten über allgemeine Computernetzwerke (z.B. *Ethernet*) mittels standartmäßiger Internetprotokolle (wie z.B. *TCP*[12], *UDP*[13], usw.) zunehmend umgerüstet werden, mithilfe von Lösungen wie dem Industrial Ethernet.

Industrial Ethernet (IE) ist eine Anpassung des Ethernet-Standards für Vernetzung industrieller Geräte und Anlagen. IE bietet Vorteile von geringeren Kosten, größerem Datendurchsatz und schnellerer Übertragungsrate gegenüber seinen Feldbus-Vorgängen, ist aber auf der anderen Seite völlig kompatibel mit herkömmlichen Internet-Lösungen, vor allem im Software-Bereich (z.B. *RESTful*[14]), und dadurch sehr flexibel, was die Programmierung der Anwendungen angeht (wie z.B. von Wollschlaeger et al. [R22] gezeigt). Im Vergleich zu Feldbussen bietet das IE eine Möglichkeit an, die IT-Funktionen in den AS-Geräten direkt zu nutzen. Durch die besondere Berücksichtigung der wichtigsten industriellen Anforderungen an Verfügbarkeit, Echtzeitfähigkeit und Robustheit hat sich IE zu einem festen Bestandteil eines modernen industriellen Netzwerkes etabliert und dient als de-facto- Standard für Anbindung und intelligente Steuerung von Feldkomponenten und Antriebstechnik.

Eine umfassende Übersicht von IE-Prinzipien, inkl. der Systembasis, Design- und Einsatzmöglichkeiten in der Industrie wird von Kay in [R23] gegeben. Metter und Bucher präsentieren in [R24] PROFINET – eine echtzeit-fähige IE-Implementierungs-technologie, die im Umfeld von Siemens-Produkten für intelligente Produktions-steuerung eingesetzt wird und Buszyklen von rund 100 µs erreicht. Die Profinet-Hubs

[12] TCP – Transmission Control Protocol
[13] UDP – User Datagram Protocol
[14] RESTful – Sammelbegriff für Services, die auf dem REST (REpresentational State Transfer) Protokoll basieren

(siehe Beispiel in **Abbildung 10a**) verfügen sowohl über digitale Eingänge (für die Anbindung der Feldbus-Endpoints wie z.B. *VHF Digital Link* [W12]) als auch über Ethernet- (z.B. *RJ45* [W13]) Kommunikationsschnittstellen und ermöglichen somit die Übertragung der Feldbusdaten von den Sensoren (zum Beispiel einer CH_4-Messanlage, **Abbildung 10b**) in digitalisierter Form über das industrielle Computernetzwerk zu angeschlossenen Internet-fähigen „back-end" Geräten (z.B. SoC, siehe Begriffserklärung unten). Ähnliche wie die Profinet-Lösungen werden auf dem Markt von Rockwell Automation (*Ethernet/IP* [W14]), Schneider Electrics (*Modbus/TCP* [W15]), Beckhoff (*EtherCAT* [W16]) u.v.m. angeboten. Leyrer und Adzan geben in [R25] eine detaillierte Übersicht von IE-basierten Kommunikationsprotokollen und insbesondere vom TCP-IP – dem herkömmlichen Internetprotokoll, die nicht anders als "Technology for the future of smart factories" bezeichnet werden. Das White Paper von Lin und Pearson [W17] kommt zu dem Ergebnis, dass in den meisten Anwendungsdomänen der Maschinensteuerung IE-Technologien wie *Modbus* und *Profinet* schon in der nahen Zukunft die gängigen Kommunikationstechnologien wie *Control Area Network* (*CAN*) ablösen und somit den Weg zu einer umfassenden Umsetzung der Industrie-4.0[15]-Lösungen in der Praxis freisetzen werden, von kleineren Systemen wie der Steuerung eines Autos bis hin zur kompletten Automatisierung großer Werke, wie einem Bergbauwerk.

Auch viele Forschungsarbeiten widmen sich der Fragestellung der Nutzung von IE-Schnittstellen „*Sensor→Datenprozessor→Akt*uator" zur intelligenten Steuerung industrieller Automatisierungssysteme. So schlagen Gillies et al. in [R26] ein echtzeitfähiges sensor-basiertes Überwachungssystem für Untertagebauten vor. Auch Liu und Li präsentieren in [R27] ihren Ansatz zur Umstellung eines Steuerungssystems eines Kraftwerks auf *Modbus/TCP*. Auf der Implementierungsebene wurde dabei eine Software von *Scada/GE* angewandt – *iFix,* welche komplexe technologische Vorgänge

[15] Industrie-4.0 – Rahmenbezeichnung aller Aktivitäten zur umfassenden Digitalisierung industrieller Produktion – einer wichtigen Voraussetzung für den Eintritt der Industrie in die neue (4.) Produktionsrevolution.

in der Form der Modelle abbilden lässt (ähnlich zum Block-basierten Ansatz vom *Simulink*). Wienbruch zusammen mit dem Team des BMBF-Projekts ADAPTATION [W18] beschäftigt sich in [R28] mit der Transformation der Werkanlagen zu den cyber-physischen Produktionssystemen. Sie zeigen, dass die CPS-Werteschöpfungskette nicht nur auf großen Industrieanlagen Anwendung findet, sondern auch ein großes Potential für kleine und mittelständige Unternehmen (KMUs) hat. Das BMBF-Projekt BASYS [W19] hat ein Simulations-basiertes Basissystem für Wandlung der Produktionsanlagen zu Industrie-4.0/CPS entwickelt. Der Ansatz vom BaSyx zur Kovergenz von AS- und IT-Technik basiert auf Umstellung der Produktionsprozesse auf Software-gesteuerte Realisierung (mithilfe von Programmiersprachen wie *C, C++* oder *C#*), wie im Konferenzbeitrag von Schneider u.a. [R29] ausführlich beschrieben.

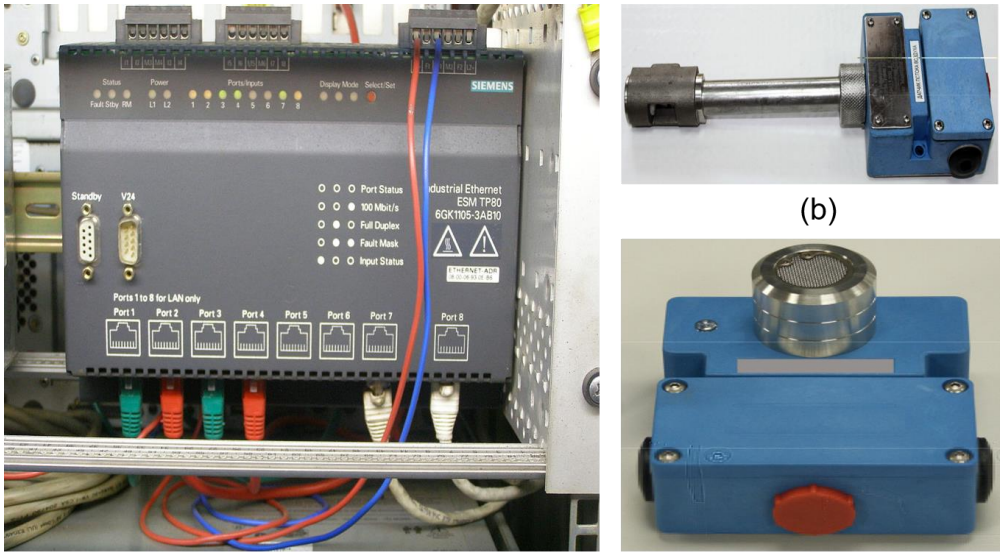


Abbildung 10. *Infrastruktur eines CPS-Smartsensors: (a) Industrial-Ethernet-Hub*[16]*, (b) Digitaler Luftdurchsatzsensor*[17]*, (c) Digitaler CH_4-Sensor*[18]*.*

An den Produktionsstellen, wo die Verkabelung der Sensorik aufgrund technologischer Bedingungen unmöglich ist, kommen zusätzlich drahtlose Netztechnologien mit erweiterter Reichweite zu Hilfe. WLAN, Bluetooth, WISA und andere im 2,4-GHz Spektrum angesiedelte Standards wurden zu gängigen Standards auch im industriellen

[16] ESM TP80 vom Siemens, Bildquelle: https://de.wikipedia.org/wiki/Industrial_*Ethernet*
[17] TX1322D vom UTAS
[18] TX3263D vom UTAS

CPS-Umfeld. Diese Netze sind in der Lage, erforderliche Übertragungsraten bzw. nötige Reaktionszeiten zu erreichen. Wollschläger u.a. geben in ihrer Survey-Veröffentlichung [R22] eine detaillierte Übersicht der Technologien, die im Bereich der drahtlosen Kommunikation für industrielle AS-Systeme angesiedelt sind. Praktische Anwendungsaspekte drahtloser Kommunikationstechnologien für Industrieanlagenautomatisierung werden z.B. vom Paavola und Leiviska in [R30] anhand des Steuerungssystems eines Dampfkraftwerks, oder für ein Steinkohlegrubenventilation-AS in den Arbeiten von Basu u.a. [R31] sowie Song u.a. [R32] diskutiert. Wie et al. präsentieren in [R33] eine WMN- (Wireless Mesh Network) basierte Kommunikationstechnologie für Kohlebergwerke, die entsprechend der Topologie eines Grubenbewetterungsnetzes aufgebaut werden kann. Diese und viele andere Forschung- und Umsetzungsarbeiten belegen sowohl die Nützlichkeit der drahtgebundenen als auch der drahtlosen IE-Technologien bei sicherheitskritischen dynamischen Systemen wie den Basisforschungsobjekten dieser Arbeit – den Untertagebewetterungssysteme (siehe Kapitel 1.4).

Im Laufe der CPS-Umgestaltung werden im Sinne von Indutstrie-4.0 viele Automatisierungssysteme mit intelligenter Sensortechnik um- und nachgerüstet. Das betrifft auch die rasant entwickelte CH_4- und Sauerstoff-Messtechnik, die ihren Ansatz heutzutage unter teils widrigen Technologieprozess-Bedingungen in der Leicht- und Schwerindustrie, in der Eisenbahn- und Schiff-Infrastruktur als auch im Bergbau finden, mit einem dazugehörigen temperatur- und zustands- (wie Vibrationen, Lärm, usw.) kritischen Arbeitsumfeld. Inzwischen sind viele Bergbauwerke mit modernen Sensoren ausgestattet, die an ein gesamtheitliches Netzwerk angeschlossen sind und ständige Überwachung von Wettergeschwindigkeiten, Drücken und verschiedenen ausfließenden Gasen gewährleisten. Clausen in [R12] und Stewart et al. in [R34] analysieren den Fortschritt moderner IKT-Technologien für die Realisierung des “Ventilation-on-Demand“-Konzepts im untertägigen Bergbau und kommen zu dem Fazit, dass viele heutige wettertechnische Systeme zur bedarfsgerechten Bewetterung und ihre Steuerungssysteme als CPS betrachtet werden. Die angewandten Beispiele der flächendeckenden sensortechnischen Überwachung eines Bergwerks finden sich

z.B. in den Arbeiten des EU-geförderten Projekts „Blue Mining“ (wie im Artikel von Müller et al. [R35] detailliert wird), aber auch im industriellen Umfeld (z.B. in der Veröffentlichung von Brady [R36] für Untertagebau sowie von Walter [R37] für ein elektrisches Netz).

Die meisten Kohlebergwerke halten heutzutage ein weitreichendes Überwachungs- und Steuerungssystem vor, das auf verteilten IE-basierten Sensornetzwerken gebaut ist. Der Bergbausektor befindet sich insgesamt auf dem Weg der Reformierung in die Richtung von Bergbau-4.0 [W20] – einem Digitalisierungsprogramm das für den Rohstoff-Sektor der Industrie-4.0 gegründet und zum jetzigen Zeitpunkt von der Firma DMT [19] , dem VDMI-Verband und der RWTH-Aachen koordiniert wird. Die marktführenden Sensor-Hersteller wie Trollex (GB) und Woelke (DE) stellen heutzutage kompakte IE-Gassensoren her, die unter kritischen Arbeitsbedingungen eines Bergbaus Einsatz finden. Ihnatovych u.a. beschreiben in ihrer Veröffentlichung [R38] *UTAS* (osteuropäischer Raum) – ein sensor-basiertes AS-Kontrollsystem zur Überwachung der Methan-Konzentrationen und Steuerung der Kohlegruben-Ventilationsnetze mit IE-unifizierten Schnittstellen. Zdanovskyi analysiert in seiner Veröffentlichung [R39] die Erfahrung der *UTAS*-Nutzung an ukrainischen Kohlebergbauwerken und verzeichnet eine insgesamt positive Nutzungsbilanz.

Der Trend in der industriellen Prozessautomatisierung geht in eine eindeutige Richtung – mehr Sensoren, mehr Datenübermittlungsschnittschellen, mehr Connectivity – und im Endeffekt mehr Daten, mit denen die Rechenprozessoren konfrontiert werden. Dementsprechend nahm der für die CPS-Datenverarbeitung benötigte Rechenaufwand dramatisch zu, sodass die standartmäßigen AS-Steuerungskontroller, die nur über eine begrenzte Leistung verfügten, die Echtzeitanalyse-Anforderungen nicht mehr erfüllen konnten. In der Zwischenzeit galt als effektivste Lösung dieses Problems zunächst die Anbindung der Feldbus-Systeme und ihrer Ethernet-basierten IE-Nachfolger an eine Cloud- oder HPC-Infrastruktur, die über ausreichende Kapazitäten für Datenverarbeitung verfügen. Der Einzug der Cloud-basierten Lösungen in die

[19] https://www.dmt-group.com

Produktion ergab skalierbare und service-orientierte Anwendungen, wie z.B. aus dem Bereich des Internets-of-Things (IoT) bekannt. Beispiele dafür sind vernetzte Aufzüge von Thyssenkrupp mit einer zentralen Steuerung mittels einer externen Cloud-Infrastruktur (wie von Rodig [W21] beschrieben), eine Fertigungsanlage (wie von Zhongs u.a. [R40] dargestellt), aber auch ein komplettes Bergbauwerk, wie in der Veröffentlichung von Byungwan et al. [R41] beschrieben ist. Der Jos-Ansatz basiert auf Verkopplung sensor-basierter Messstationen mit einer IoT-Datenverarbeitungsplattform mittels ZigBee-Protokolls für drahtlose Kommunikation. Zur Datenanalyse werden dann Algorithmen des Maschinellen Lernens von Azur (einer Microsoft-Cloud-Plattform) angewandt, die zur Vorhersage der Wetter-Qualität in untertänigen Arbeitsbereichen mit Hilfe neuronaler Netze trainiert wird.

Allerdings geht das Cloud-Modell für viele industrielle AS-Anlagen zu weit: die vertikale Kommunikation zu einem übergeordneten System ist durch produktionsbedingte Limitierung der Konnektivität nach draußen erschwert. Darüber hinaus stellen die Sicherheitsanforderungen eine weitere Hürde dar, auch wenn die Forschung (Arbeiten von z.B. Stergiou [R42] mit Erforschung der Einsatzperspektive der Encryption für Datenübertragung, Sathavara [R43] mit Entwicklung einer Security-Middleware mittels Software-Agenten oder Koelemeijer u.a. [R44] mit Umsetzung eines Hardware-in-the-Loop-Ansatzes) große Fortschritte auch in dieser Richtung gemacht haben.

Aufgrund der noch nicht vollständig gelösten vertikalen Skalierungsproblematik bei der zentralisierten Datenverarbeitung und hinsichtlich der rasant steigenden Leistung der Rechensysteme mit niedrigem Energieverbrauch gewannen in den letzten Jahren die dezentralen Datenbearbeitungsansätze für die Aufgaben der intelligenten Steuerung an Bedeutung. Zu solchen Systemen zählen die meisten ARM-basierten RISC-Architekturen, die als Basis für mobile Geräte mit hoher Rechen- und Kommunikationsleistung den Embedded-Marktbereich erobert haben. Die ARM-basierten Ein-Platinenrechner mit umfangreichen Anschlussmöglichkeiten (wie z.B. im Kapitel 1.2 präsentierte Odroid, **Abbildung 4**) findet man mittlerweile in vielen Bereichen, von Mobiltelefonen bis hin zu Höchstleistungsrechnern, aber insbesondere in der Industrie. Das Lokalitätsprinzip bei der Datenverarbeitung lässt in vielen Fällen

Vorteile sofortiger Rechenressourcenverfügbarkeit, schneller und sicherer Datenübertragung sowie andere ausnutzen, wie z.B. vom Zdanovskyi in [R45] für UTAS-basierten Bewetterungsnetze analysiert. Manche modernen Sensoren bieten sogar eingebettete Recheneinheiten zur sofortigen Datenvorverarbeitung, was unter den Begriff "Edge-Computing" fällt. Kluge bietet im [R46] die Technologie der neuen Kontroller-Generation, die die Eigenschaften eines speicherprogrammierbaren Steuerungssystems und eines PCs in einem Gerät vereinbaren. Diese Technologie findet in der aktuellen Produktreihe der Firma Lenze[20] Einsatz, die sich auf industrielle Automatisierung spezialisiert. Edge-Systeme werden auch aktiv in Fertigungsanlagen angewendet, wie z.B. Böhler für eine Stihl-Fabrik in [R47] präsentiert. Für den Fall, dass die Integration der Edge-Ressourcen mit dem Sensor aus technischen oder Leistungsgründen unmöglich ist, finden portable Systems-on-Chip (SoC) Einsatz, die in direkter Kopplung mit Sensornetzen die Vorteile eines programmierbaren integrierten Rechensystems mit umfangreichen Kommunikationsschnittstellen (inkl. Ethernet) bieten. Aufgrund großer Integrationsfähigkeit mit verschiedenen Schnittstellen von Sensoren, Aktuatoren, Netzwerken und anderen Bestandteilen eines industriellen Automatisierungssystems, sowie dank angebotenen Eigenschaften der Portabilität, Mobilität und einer hohen aggregierten Leistung, gewinnen die SoCs zunehmend Beliebtheit für die Industrieautomation, wie z.B. von Riemenschneider in seinem Beitrag [R48] für verschiedene Anwendungen exemplarisch gezeigt oder von Bergbauer in [R49] anhand eines breiten Spektrums der Nutzungsszenarien für das AMD-EPYC basierte Server-Klasse-Rechensystem *COM-Express* erklärt wird.

Das gesamte Spektrum möglicher Verteilungen der Datenverarbeitungssysteme in der IT-Infrastruktur eines Automatisierungssystems, von der lokalen Bearbeitung mittels eingebetteter Rechner bis zur Anbindung an eine zentrale Cloud-Instanz, wird mit Begriff "Fog-Computing" bezeichnet. Die Vielfalt der an ein Fog angeschlossenen Ressourcen erfordert besondere Programmierungsansätze für die Entwicklung der Software, die die Vorteile aller vorhandenen Architekturen umfassen könnte. Fernández-Caramés u.a. präsentieren in [R50] einen service-basierten Ansatz zur

[20] https://www.lenze.com

Softwareentwicklung für eine vielschichtige, verteilte, Fog-basierte IT-Infrastruktur einer automatisierten Schifffertigungsanlage. Einen ähnlichen Ansatz folgen Fraga-Lamas u.a. in ihrem Entwicklungsvorschlag [R51] für eine smarte Bahnanlage.

Auf ihrer „digitalen Ebene“ betrachtet, sind CPS vernetzte Computerteilsysteme, die in einer Wechselwirkung mit hierarchisch-organisierten und topologisch-gekoppelten realen Objekten und Systemen durch umfassende Sensorik und Steuerungstechnik mit der „physikalischen Ebene“ stehen, unter der Voraussetzung einer industriellen Netzverbindung (IE). Aufgrund dieser Kopplung ist es möglich, eine diskrete Darstellung kontinuierlicher physikalischer Prozesse zu erhalten, was in der Literatur oftmals als “Digital Twin“ bezeichnet wird (wie z.B. im Beitrag von Zeman u.a. [R52] erläutert). Im Verbund mit CPS-Infrastrukturen und verfügbaren aktuellsten Daten steht die Simulationstechnik vor einer neuen herausfordernden Anwendungs-perspektive – der Nutzung der von CPS abgebildeten Erkenntnisse zum Verschaffen eines Aufschlusses über das Verhalten der industriellen Objekte unter den aktuellen (gegebenen) oder ggf. virtuellen (vor Anwendern voreingestellten) Bedingungen für den notwendigen Zeitrahmen. Die Funktion der Modelle, die sich im Ingenieur- und Naturwissenschaftsbereich für die Funktionen etablierten, die meistens auf Planung (im Sinne der Funktionalität, Sicherheit, Qualität, Wirtschaftlichkeit) industrieller Prozesse beschränkt sind (wie z.B. Beiträge für bergbautechnische Aufgabestellungen von McPherson [R53], Brake [R54] u.a.), wird im Laufe der CPS-Einführung auf die Online-Begleitung solcher Prozesse, also zu ihrer Laufzeit, erweitert. Die bekannten Forschungsansätze, die auf die angegebene Problematik der dynamischen Analyse ausgerichtet sind, erkennen zwar das Potenzial, die statische Natur herkömmlicher Simulationsmethoden zu brechen, gehen aber nicht weit genug, um im industriellen Umfeld einen umfangreichen Einsatz zu finden. Besonders ausgezeichnet sollen dabei die Arbeiten im Bereich von “Digital Twin“-Implementierung für sicherheitskritische Systeme in der Luftraumtechnik von Grieves und Vickers [R55], die von Yan et al. analysierten Lösungen für Integrierte Simulation mittels Matlab, Simulink, Modelica und anderer verbreiteten Lösungen [R56], der HIL (“Hardware-in-the-loop“) basierte Ansatz zur Automatisierung industrieller Anlagen von Kleijn [W22]. Unter anderem soll

die betriebsspezifische Hardware (z.B. low-power Geräte wie ARM Odroid oder eingebettete Systeme wie Myriad [W23] oder MPSoCs wie Zynq [W24]) effizient und im Kontext einer verteilten Recheninfrastruktur genutzt werden, entsprechend der hierarchischen (sowie topologischen) Organisation industrieller Prozesse und Objekte.

Die Vorteile, die den industriellen Systemen durch die Integration von dynamischer Simulation ins CPS ermöglicht werden, sind eindeutig:

- Überprüfung eines CPS auf Erfüllung der Funktionalitätsaufgaben und Einhaltung der Zuverlässigkeitsanforderungen;
- Erhöhung der Betriebseffektivität und Produktivität durch Erarbeitung der Optimierungsvorschläge nicht nur bei der Produktionsplanung, sondern auch während des Produktionsablaufs, z.B. mittels Künstlicher Intelligenz, Data Mining und anderer innovativer Verfahren, wie z.B. in Papers von Vogel-Heuser [R21], Clausen [R12] u.a. gefordert und diskutiert wird;
- Senkung der Betriebskosten durch prädiktive Experimentplanung und Nützlichkeitsvorhersage für anstehende Steuerungsmaßnahmen (wie z.B. einer reversierten Belüftung eines Bewetterungsnetzes, die nach Vorschriften in regelmäßigen Zeitabständen durchgeführt werden soll);
- Verbesserung der Sicherheit durch frühzeitige Erkennung potenzieller sicherheitskritischen Situationen (sowie der zu ihnen führenden Ursachen), Entwicklung der Vorbeugungsmaßnahmen sowie Erarbeitung der Beseitigungspläne für die bereits ereignete Notfälle in Echtzeit.

Trotz aller sichtbaren Vorteile ist große Vorsicht bei der Nutzung der Modelle unter realen industriellen Bedingungen geboten. Insbesondere gilt das für Modelle dynamischer Systeme mit ihrem komplexen mathematischen Apparat, welches „per se“ einen gewissen inneren Determinismus birgt (wie z.B. die Analyse von Lee [R57] zeigt), der sich spätestens beim iterativen numerischen Lösungsansatz sichtbar machen oder sogar durch Hardware-Fehler verursacht werden könnte.

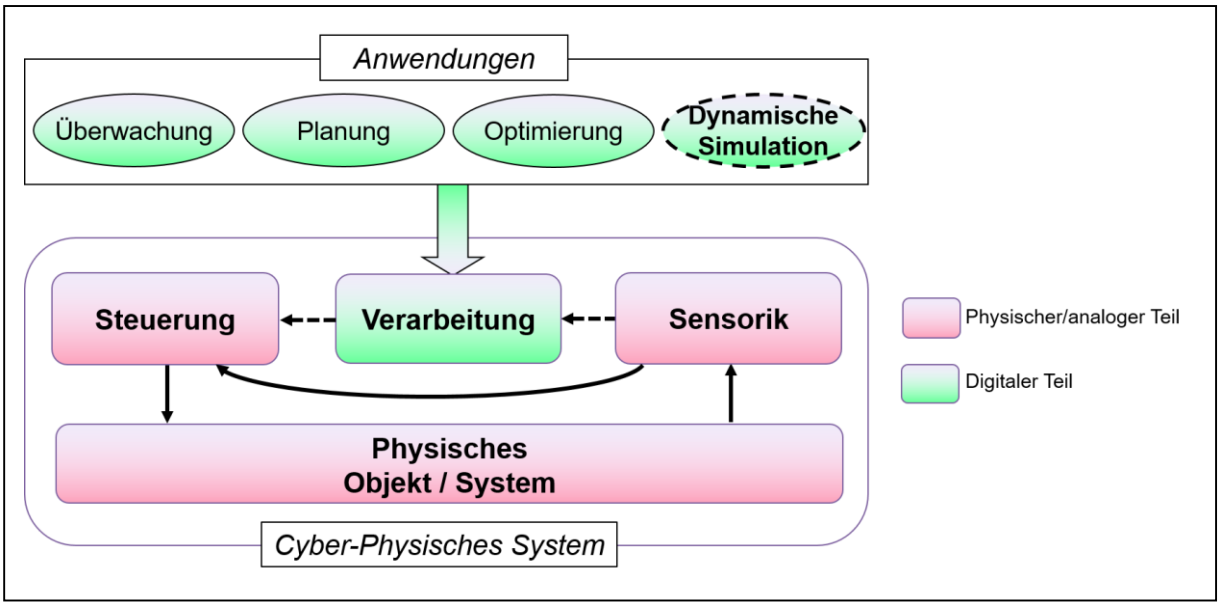


***Abbildung 11**. Physische und digitale Bestandteile eines Cyber-Physischen Systems.*

Der Grund dafür liegt an der Methodologie der Modellierung – oftmals werden die Modelle anhand eines physischen Prototyps validiert, der ein Abbild eines Teilsystems des realen Objekts darstellt, wie z.B. von Gries et al. [W25] gezeigt wird.

Die Erfüllung der Modellaufgaben und somit die Zuverlässigkeit der Modelle werden daher innerhalb von einer beschränkten Umgebung untersucht, die die Eigenschaften der Wechselwirkung verschiedener physischer und informationstechnischer Prozesse im Objekt selbst, in der gesamten Netzwerktopologie von verschiedenen weiteren Objekten, im Netzverkehr wie auch mit der Umwelt nivellieren oder sogar weglassen. Daher beschränkt sich die Rolle der dynamischen Simulation aktuell auf eine begleitende Funktion für die absolut deterministischen (statischen) Kontrollabläufe der Steuerungssysteme.

Zu einem ihrer Hauptziele setzt sich diese Arbeit, die Brauchbarkeit der bergbautechnischen CFD-Modelle auf Basis der Szenarien von realen Kohle-bergbauwerken mittels CPS-Technologien zu untersuchen.

Der Einsatz von digitaler Simulationstechnik im bergbautechnischen Umfeld ermöglicht im Verbund mit CPS die Durchführung von umfassenden virtuellen Experimenten, was allerdings in sich einige wichtige Herausforderungen birgt, wie zum Beispiel:

- Umgang mit hochgradig dynamischen CPS-Daten
    - *Potenzielles Problem*: Modelle basieren meistens auf veralteten Daten, was ihre Qualität mindert.
    - *Gängige Lösungen*: Manuelle Abfragung der Parameter und ihre Eingabe in Modelle, was in unregelmäßigen Zeitabständen erfolgt. Sonstige Eigenschaften (wie z.B. die Schrittgröße numerischer Methoden) werden meistens auch manuell angepasst (z.B. der Ansatz von Moldovanova [R58], Solonin [R59]). Andere bekannten Ansätze beruhen auf der Anpassung der Modelleigenschaften anhand vorhandener Daten für Objektkonfigurationen, die an einem gewissen Punkt in der Vergangenheit aufgefasst wurden. Die Datenaktualität wird z.B. durch ein Extrapolationsverfahren abgeschätzt, wie z.B. von Schmeyer in [R60] beschrieben oder von Gillies et al. [R26] für die Bestimmung der Bewetterungseigenschaften für Ventilationsbereiche mit unzureichender Sensorumfassung im Bergbau.
    - *Der Lösungsweg dieser Arbeit*: Abfragung der aktuellen Parameter von Sensoren vor dem Anfang jeder Simulation, Validierung der Eingaben auf Korrektheit und Anpassung der Simulationsparameter nach einem heuristischen Verfahren.
- Umgang mit topologisch organisierten und vielschichtigen CPS-Objekten
    - *Potenzielles Problem*: Horizontale Kopplung und vertikale Einbindung der Modelle verschiedener technologischen Objekte und/oder Ebenen.
    - *Gängige Lösungen*: Approximation verschiedener Elemente durch ein Basiselement und die weitere Betrachtung des Netzes aus homogenen Objekten anhand des Basiselements, wie z.B. in der Arbeit von Clausen und Agasty [R61].
    - *Der Lösungsweg dieser Arbeit*: Ein service-orientierter Ansatz soll die Entwicklung heterogener Simulationssoftware ermöglichen, die sich als Baublöcke für hierarchisch organisierte und durch einheitliche

Kommunikationsschnittstellen (entsprechend der physikalischen Struktur der topologischen Vernetzung) verbundenen Modelle anbieten. **Ferner gilt es zu erforschen, wie die Echtzeit-Messdaten in hybride Modelle in Form der Anfangs- bzw. Randbedingungen und/oder Checkpoints einfließen sollen.** Auch die Bereitstellung der Simulationsergebnisse in einem akzeptablen Zeitrahmen, z.B. durch den Einsatz effektiverer numerischer Methoden während der Simulation, ist eine wichtige Forschungsaufgabe, für welche keine überzeugende aktuelle Lösung vorliegt. Eine zu erforschende Aufgabestellung, die insbesondere für industrielle Systeme beachtet werden soll, ist die Energiesparsamkeit des Modellierungs-prozesses – die Simulationen sollen unter teils geringfügigen Energiekapazitäten der eingebetteten Systeme durchgeführt werden. Nicht zuletzt soll auch die Erstellung nutzer-orientierter Schnittstellen für die Visualisierung der durch Simulations- und Messdatenübergreifende Ergebnisse gelöst werden.

2.2 Ansätze und Lösungen zur Modellierung der Bewetterung

Um die Anforderung der sicheren und rentablen Rohstoffförderung zu erfüllen, kommt im Bergbau die Modellierungs- und Simulationstechnik als unverzichtbare Technologie zum Einsatz, die vor dem Hintergrund aktueller Trends der cyber-physischen Systemorganisation einen maßgeblichen Beitrag zur weiteren Steigerung der Produktions-Effektivität und Sicherheit im 21. Jh. leisten soll. Aktuelle Ansätze und Entwicklungen für Modellierung der Grubenbewetterung konzentrieren sich auf die Anwendung numerischer Strömungsmodelle zur Wetternetzberechnung mit dem Ziel, die bestmöglichste und effektivste Kontrolle und Steuerung von Wetterverhältnissen (Luft-Gas-Gemischen in GBN-Strecken) während der Gewinnungsaktivitäten unter Tage zu gewährleisten. In diesen Modellen werden Eigenschaften aller grundlegenden physikalischen und technologischen Bewetterungsprozesse, wie Stromgrößen, Dichten, Drücke, Gasaustrittquellen, usw. berücksichtigt. Je umfangreicher sämtliche charakteristischen Parameter in die Modelle einbezogen werden, umso größer ist ihre Genauigkeit und Nutzbarkeit für verschiedene Anwendungsszenarien aber auch die daraus resultierende Komplexität

der Modelle. Vor diesem Hintergrund ist eine Klassifikation der Modelle nach ihrer Komplexität sinnvoll, wie in **Tabelle 1** dargestellt.

***Tabelle 1**. Klassifikation der Grubenbewetterungsmodelle.*

Klasse der Modelle	Komplexität	Anwendungsart	Beispiele der Software-Lösungen
Statische Systeme	Niedrig (Algebraische Gleichungen)	Planungsaufgaben für stationäre Konfigurationen, Regelungsalgorithmen	*MineVent*[21], AEROSET[22]
Dynamische Systeme mit konzentrierten Parametern (DSKP)	Mittel (gewöhnliche Differential-gleichungen)	Berechnung instationärer Luftverteilung und bedarfsgerechter Wetterführung, adaptive Steuerung	*VNET*[23], *Smart Ventilation*[24], Bestech *NRG1-ECO*[25]
Dynamische Systeme mit verteilten Parametern (DSVP)	Hoch (partielle Differential-gleichungen)	Berechnung der Ausbreitung von Gasen aus dynamischen Quellen	*VentSim*[26]

Bewetterungsberechnungs-Software basiert meistens auf statischen (und damit im Sinne der Implementierung) einfachsten Modellen, die auch als “Steady-State-Simulation“ bekannt sind. Für das gesamte Bewetterungssystem werden stationäre Strömungseigenschaften vorausgesetzt und die durch eine Ventilationsstrecke verlaufenden Luftströme mit Hilfe einer allgemeinen Formel (II.1) definiert:

[21] http://oa-mining.com/minevent.htm
[22] https://aeroset.net/en/about/
[23] https://www.mvsengineering.com/index.php/products/software
[24] https://new.abb.com/mining/mineoptimize/digital-applications/advanced-process-control/abb-ability-expert-optimizer
[25] https://www.bestech.com/
[26] https://ventsim.com/

$$P = RQ^2 . \qquad \text{(II.1)}$$

Dabei steht *Q* für den Volumenstrom *[m^3/s]*, *P* für den Druck an Randen der Strecke [*Pa* (*N*/m^2)] und *R* für die aerodynamische Resistenz *[Ns^2/m^8]*.

Die Berechnung der Wetterführung im gesamten Netzbereich erfolgt aufgrund der Modelle einzelner Netzelemente (Strecken, Wege, usw.). Dabei werden Methoden der Topologieanalyse verwendet, die Zusammenhänge zwischen den Elementen berücksichtigen – z.B. die auf dem Ansatz der elektrischen Analogien von Gärtner [R62] basierende Konturenmethode, welche sich an die Berechnung der Strom- und Spannungsverteilung in einem elektrischen Leitungsnetz mithilfe der Kirchhoffschnen Gesetzen ähnelt. So beispielweise der Ansatz von Hu und Longson [R63], der für die Überprüfung ausreichender Kapazität von zu installierenden Grubenventilatoren verwendet wurde. Shcundin und Ivannikov präsentieren in [R64] einen Alternativansatz zum Aufbau des Gleichungssystems für das gesamte Netz, der anstatt von Konturengleichungen auf Knotendrücke-Sätzen basiert.

Die Tools, die statische Modelle implementieren, ermöglichen die Erstellung der Topologien der Bewetterungsnetze (ähnlich zu **Abbildung 6b**) mit Hilfe von 3D-Visualisierungsverfahren, Erarbeitung verschiedener Strategien zur Anbindung der Ventilatoren und Sensoren, Berechnung optimaler Luftverteilung in allen Netzstrecken sowie Projektierung der Kontrollsysteme (z.B nach dem Hardy-Cross-Verfahren). Beispiele der auf dem Markt verfügbaren Softwarelösungen für Planungsaufgaben sind *MineVent* und *AEROSET* (siehe **Tabelle 1**). Die meisten in der Produktion eingesetzten automatischen Kontrollsysteme, wie das bereits im vorigen Kapitel 2.1 erwähnte *UTAS*, basieren wegen ihrer einfachen Realisierung und geringeren funktionellen Anforderungen auch auf statischen Modellen. Darüber hinaus ist dieser Ansatz bei der Erarbeitung von Lösungen für statische Aufgabestellungen hilfreich, wie zum Beispiel für die Erstellung optimaler Evakuationspläne für Notfallsituationen (siehe als praktisches Beispiel die Veröffentlichung von Ji et al. [R65]).

Eine ausreichende Genauigkeit kann aber von statischen Modellen nur für Systeme gewährleistet werden, in welchen entweder keine relevanten Veränderungen innerhalb des Planungshorizonts auftreten, oder diese ausreichend mit Sensoren abgedeckt sind, sodass eine komplexe Simulation nicht notwendig ist. **Bewetterungssysteme sind allerdings höchstdynamische Objekte, die zur Zeit noch über keine flächendeckende Sensorabdeckung verfügen**, wie die Analyse von Stewart et al. [R34] zeigt. Statische Modelle sind vor diesem Hintergrund nur auf eine beschränkte Anzahl von Produktionssystemen anwendbar und finden damit am häufigsten Einsatz für Planungszwecke. Darüber hinaus kann die Größenordnung des Systems eine unüberwindliche Hürde für die Konvergenz zugrundeliegender Konturenmethode darstellen, wie z.B. Tantsov [R66] in seiner Analyse von großen (mit über 4.000 Elementen) Netzen angibt.

Dynamische Modelle kommen immer häufiger für Bewetterungstechnik-Systeme mit ihren dazugehörigen kurzzeitig wechselnden Eigenschaften zum Einsatz. Die Grundlage für die Analyse dynamischer Wetterverhältnisse wird von mathematischen Modellen geschaffen, die strömungsdynamische Prozesse in den Hauptelementen eines Bewetterungsnetzes in Form von Systemen verschiedener Differential- und algebraischen Gleichungen abbilden, indem die Abhängigkeiten zwischen Strömungsgeschwindigkeit, Fluidmenge, Dichte und Drücke berücksichtigt werden. Die Komplexität der daraus resultierenden Gleichungen, die grundsätzlich über keine analytische Lösung verfügen, fordert die Anwendung der Approximation – die Vereinfachung der Initialgleichungen durch Vernachlässigung der Einflüsse, die keinen nennenswerten Vorteil für das Endergebnis ermöglichen, wie z.B. Diffusion, dichteabhängiger Trennung von Gasen, inhomogener Vermischung, Verhalten an Grenzflächen u.v.m.. Ebenso muss ein Diskretisierungs-Verfahren zur numerischen Lösung der daraus resultierenden Gleichungssysteme angewandt werden (wie z.B. Finite-Differenzen, Finite-Volumen usw.). Die Letzteren werden zur Aufteilung des zugrundeliegenden gesamten Volumenbereiches in endlich viele vernetzte Bestandelemente genutzt. Die aus der Diskretisierung entstandene Menge vereinfachter Modelle wird nachfolgend mittels eines (iterativen) numerischen

Verfahrens gelöst (wie z.B. Euler, Runge-Kutta, usw.), das die zeitliche Entwicklung der gesuchten Parameter (z.B. Luftverteilung und Gasausbreitung) in jedem Punkt des betrachteten Raums bestimmt.

Typische mathematische Modelle der Bewetterung basieren auf einer makroskopischen Beschreibung des Verhaltens komplexer, mehrphasiger newtonscher Fluide durch die Navier-Stokes-Gleichungen (also Impulserhaltung- und Kontinuitätsgleichung). Dieser in der Modellierung als klassisch bezeichnete Ansatz lumfasst verschiedenste Aspekte der Strömungsdynamik, wie beispielweise instationäre Luftführung unter Berücksichtigung der Kompressibilität und thermodynamischen Effekte, Transport und Ausbreitung von Gasen, Verhalten von dynamischen Gasquellen in schwierigen Geologien, usw. Solche dynamischen Systeme werden in der Literatur als Systeme mit verteilten Parametern (DSVP) bezeichnet – ihre Modelle beschreiben die Wirkungsanordnungen mit ortsabhängigen Parametern mittels partieller Differentialgleichungen. Neuere Modellierungsstrategien verfolgen auch einen molekular-kinetischen Ansatz, wie beispielsweise die neuartige mikroskopische Lattice-Boltzmann-Methode (LBM), die im Gegensatz zu der makroskopischen Beschreibung der Navier-Stokes-Gleichungen die Verteilungsfunktion der Moleküle und deren zeitliche Entwicklung berücksichtigen (wie z.B. im Beitrag [R67] von Feuchter ausführlich beschrieben). Allerdings findet der LBM-Ansatz, abgesehen von den Längen der Bewetterungs-strecken und ihrer topologischen Komplexität, einen eingeschränkten Gebrauch bei der Bergbau-Simulation.

Um die Komplexität der Modelle zu reduzieren und damit den Implementierungs-aufwand für die Simulationssoftware zu senken, verzichten manche Bergbau-Strömungsmodelle auf die Kompressibilitätseigenschaften der Strömung, unter Berufung auf die relativ niedrige Geschwindigkeit des Wetterstroms (Mach-Zahl < 0,3) im normalen Bewetterungsbetrieb. Also werden die Modelle als Systeme mit konzentrierten Parametern (DSKP) betrachtet und mittels gewöhnlicher Differential-gleichungen ohne Berücksichtigung der Ortsabhängigkeit der Parameter beschrieben.

Das Modell inkompressibler Wetterströmung wird in einer Form definiert, die der allgemeinen Formel (II.2) ähnelt:

$$K\frac{dQ}{dt} + RQ^2(t) = P(t)\,. \qquad \text{(II.2)}$$

Dabei sind *Q, P, R* – Parameter der Gleichung (II.1), *K* – der Trägheitskoeffizient der Strömung, der die Geometrie sowie andere Charakteristiken eines Wetterweges berücksichtigt.

Die Netz-Berechnung erfolgt ähnlich wie bei den statischen Modellen, also mittels der auf Graphentheorie basierten Ansätze (wie der Konturen- oder Knoten-Methode), wobei die daraus resultierenden Gleichungssysteme für die Basis-Differential-gleichungen der DSKP erfasst werden, die dann mit Hilfe eines numerischen Verfahrens iterativ gelöst werden. Aufgrund der Eigenschaft einer vereinfachten Modellaufstellung ist der DSKP-basierte Ansatz die meist verbreitete Methode zur Erforschung dynamischer Luftverteilung in Bewetterungsnetzen. Der DSKP-Ansatz wird beispielsweise zur Realisierung des Konzepts einer bedarfsgerechten Wetterführung (auch als “Ventilation-on-Demand“ oder einfach “VOD“ bekannt) in sensor-basierten echtzeit-fähigen Kontrollsystemen eingesetzt. Der Multiopti-mierung-Ansatz von VOD hat das Ziel, die Bewetterung effizient, also mit geringstem Energieaufwand für die Betreibung der Grubenventilatoren zu gestalten. Das Konzept wird in Software wie *SmartVentilation*, *NRG1-ECO*, *SmartEXEC* u.v.m. implementiert (siehe **Tabelle 1**).

Die auf Eigenschaften inkompressibler Strömung basierten DSKP-Modelle bieten zwar die Vorteile eines relativ einfachen Aufbaus des Gesamtgleichungssystems, sind aber zur Lösung gasdynamischer Aufgabestellungen, aufgrund kompressibler Eigenschaften der Gase in GBN-Abbauorten ungeeignet. Zwar gibt es Realisationen (z.B. von Stewart at al. in [R34]), die die Verbreitung von durch Sensoren gemessenen Gaskonzentrationen mit der DSKP-Wetterströmung berücksichtigen, nur versagen solche Methoden auf der Größenordnung eines realen Systems aufgrund der stark verzweigten, vieldimensionalen Topologie. Darüber hinaus können keine schlag-

artigen Wetteränderungen (zum Beispiel aufgrund des Ausfalls eines Grubenventilators) von DSKP-Modellen berücksichtigt werden. Vor diesem Hintergrund werden fortgeschrittene Modelle mit Hilfe des kompressiblen Navier-Stockischen-Gleichungssatzes in der DSVP-Form erstellt. Nach dem DSVP-Ansatz werden viele der verbreiteten Bewetterung-Simulationssoftware gebaut, wie beispielsweise *VentSim* (siehe eine ausführliche Beschreibung im Beitrag von Brake [R68]). Stewart et al. zeigen in ihrer Veröffentlichung [R34] die Nutzung der *VentSim*-Simulationssoftware zur qualitativer Analyse der Wetterverhältnisse großer Netzsysteme (mit über 1.500 Luftführungswegen) und schlagen einen adaptiven Ansatz zur Berücksichtigung der Gasausbreitung durch ein Interpolationsverfahren, das die Verbreitung der durch Sensoren erfassten Gaskonzentrationen von den Messstellen in die topologisch verbundene diskrete Gitterelemente analysiert, mit dem an die Luftverteilung gekoppelten Zeitablauf, vor. *VentSim*, so wie andere bekannte Softwarepakete (z.B. *MFIRE*), kann allerdings keine schlagartigen Methanfreisetzungen berücksichtigen, die z.B. aufgrund der Abbauprozesse auftreten oder von Änderungen in der Wettermasseverteilung durch Filtrationsprozesse aus dem porösen Medium der Abbauorte verursacht werden bzw. durch weitere dynamischen Quellen entstehen können, die beispielweise von Svjatnjy im Werk [R15] grundlegend untersucht werden. Alle diese dynamischen Gasquellen erfordern gezielte Verdünnungsmaßnahmen entlang des gesamten Wetterstroms, deren Effizienz im direkten Zusammenhang mit dem Umfang der analysierten dynamischen Gasquellen steht. Darüber hinaus erarbeitet Svjyatnjy eine Methodik zur Darstellung komplexer Geometrie der Wetterwege durch eine 1D-Approximation, die eine große Vereinfachung des daraus resultierenden DSVP-Gleichungssystems ohne irgendeinen Genauigkeitsverlust von Simulationsergebnissen und damit einen akzeptablen Kompromiss zwischen der Komplexität der Modelle und der Qualität der Simulationsergebnisse darstellt. Moldovanova präsentiert in seiner Arbeit [R58] eine praktische Realisierung von Svjatnjys Modelle mit Hilfe eines Finite-Elemente numerischen Verfahrens konzentriert sich dabei allerdings nur auf aerodynamische Modelle. Agasty erarbeitet

in [R61] einen Ansatz zur Integration dynamischer Simulationsdaten mit statischen Daten der Planungssoftware (anhand des oben erwähnten kommerziellen *VentSim*-Pakets). Nach diesem Ansatz wird die Simulation von dem verbreiteten (auch kommerziellen) Simulationswerkzeug *Ansys-CFX*, mittels vorhandener Turbulenzströmungsmodelle durchgeführt. Die Gasverdünnungs-prozesse werden vom Agasty inbegriffen, indem die Transportierung der durch Diffusion in den Wetterstrom gelangenden Gase berücksichtigt wird. Das Agasty- Modell beschränkt sich auf die Modellierung der CO_2-Verdünnungsprozesse und bietet momentan keine Analyse für dynamisch auftretendes CH_4 Gas (Methan), welches von größter Bedeutung für die Gewährleistung der Arbeitssicherheit ist.

Wie oben gezeigt, verfügt die Simulation der aerogasdynamischen Prozesse in Bewetterungsnetzen über ausreichende methodologische Grundlagen, die von einer breiten Vielfalt fundamentaler Basismodelle geschaffen ist. Die DSVP-Modelle sind die aussichtsreichste Klasse der Modelle, da sie teils sehr komplexe, z.B. gasdynamische, Prozesse nachbilden. Gleichzeitig sind die DSVP-Modelle die anspruchsvollsten im Sinne der Umsetzungsmöglichkeiten mittels moderner IT-Ressourcen. Die meisten Simulationstools, die die DSVP-Modelle implementieren, ziehen keine Vorteile aus der in den letzten fünf Jahren stark veränderten IT-Landschaft, die sich von schwarm-mäßig vernetzten SoC-Systemen und Clouds (also so genanntem “Capacity Computing“) bis hin zu Höchstleistungsrechnern (“Capability Computing“) sowie Verbünden aus den beiden Klassen paralleler Rechenressourcen spannt. Die Bergbau-Simulationstechnik wird von einer Methodik der Modell-erstellung auf der Basis vieler interoperablen und hierarchisch-organisierten (entsprechend der physikalischen Struktur der Objekte) Services stark profitieren können. Eine als “open-source“ verfügbare Bibliothek aller bergbautechnischen Modelle würde folgende Funktionalitäten ermöglichen: i) Erstellung neuer Modelle auf Basis der bereits vorhandenen, ii) vereinfachte Integration mit übergeordneten Planung- und Steuerungstools durch einheitliche Netzwerkprotokolle, iii) Nutzung durch Experten des Gegenstandsgebiets in unmittelbarer Nähe an Produktionsstellen u.v.m.

In dieser Hinsicht ist ein Blick auf Technologien und Erfahrungen der verteilten und parallelen Simulationstechnik, wie sie im nächsten Kapitel 2.3 diskutiert wird, sinnvoll.

2.3 Verteilte und Parallele Simulationstechnik

Die Numerik stellt eine wichtige Grundlage und einen unverzichtbaren Bestandteil der modernen Simulationstechnik dar. Sie bietet die Verfahren an, die die Bestimmung einer annähernden (approximativen) aber zur gleichen Zeit akkuraten Lösung für kontinuierliche physikalische Probleme ermöglichen, für welche sonst keine analytische Lösung vorliegt. Zu solchen Problemen zählen auch die meisten strömungsdynamischen Gleichungen, inkl. inkompressibler Fluid-Strömungen in Bewetterungsnetzen (Kapitel 4.1), die mit Hilfe der elliptischen nichtlinearen Navier-Stokes-Gleichung (I.1) beschrieben werden. Zur Problematik der Analyse solcher Gleichungen zählen: die komplexe Strömungsstruktur, die großen Reynolds-Zahlen $Re>>1$ mit Turbulenzeffekten, die dominanten nichtlinearen Effekte, die sensitiven Quantitäten sowie viele andere, wie beispielsweise von Rannacher in [W26] analysiert.

Die Anwendung eines numerischen Verfahrens auf das kontinuierliche Strömungsgebiet ermöglicht die Aufstellung eines diskreten Gleichungssatzes für die daraus resultierenden Elemente der Approximation, was mit Hilfe eines Diskretisierungsverfahrens, wie z.B. der Finite-Elemente-Methode (FEM), Finite-Differenzen (FDM), Finite-Volumen (FVM), die Spektrale-Galerkin-Methode und vielen anderen erfolgt. Sämtliche dieser numerischen Methoden führen trotz ihrer unterschiedlichen Spezifik und Nutzungsaspekte zu demselben Ziel – der Transformation der ursprünglichen Modelle in die Form eines diskreten Cauchy-Anfangswertproblems, welches nach der Anwendung eines numerischen Integrationsverfahrens (wie z.B. dem iterativen Euler-Verfahren, Runge-Kutta, usw.) das ursprüngliche komplexe Problem mit ausreichender Genauigkeit lösen kann. Die Existenz, Stabilität, Konvergenzgeschwindigkeit und andere wichtige Eigenschaften der numerischen Lösung hängen dabei von mehreren Faktoren ab, aber in erster Linie von dem Typ und der Feinheit des verwendeten Gitters (z.B. eines anisotropen Gitters an Randgebieten der Strömung), der Größe der Schrittweite des

Zeitintegrationsverfahrens, sowie der Korrektheit der Implementierung der numerischen Verfahren in Form einer Software. Mit dem Letzteren (Implementierung der numerischen Verfahren als Software) beschäftigt sich ein spezielles Teilgebiet der Informatik – die Numerische Programmierung, welche für die Entwicklung der Softwareansätze für die wichtigsten Bestandteile der numerischen Lösung – wie z.B. den Gittergenerator, Löser, Postprocessor, usw., zuständig ist. Die hohen Rechenintensivitäts- und Speicher-Anforderungen der numerischen Lösung können für höchstkomplexe Probleme realer Systeme nur von einem Rechensystem getroffen werden, das über eine genügende Leistungsstärke verfügt.

Wie auch in der Analyse von Kapitel 1.2 gezeigt wird, ist die Nutzung paralleler und verteilter Rechenplattformen die einzige umsetzbare Option zur Durchführung zeiteffizienter und kostengünstiger Simulationen. Die Parallelisierung erfolgt grundsätzlich durch die Verteilung der iterativ zu lösenden Gleichungen auf alle verfügbaren Recheneinheiten (CPU-Kerne, CPUs, usw.), die über ein Feldbus (z.B. zwischen den CPU-Kernen) bzw. ein Computernetzwerk (zwischen den CPUs) gekoppelten sind. Die zwischen den diskreten Gleichungssätzen vorhandenen Parameterabhängigkeiten werden mittels des Datenaustausches durch die verfügbaren Verbindungsschnittstellen realisiert. Je nach dem Grad der Kopplung zwischen den Recheneinheiten in einer parallelen Simulationsinfrastruktur unterscheidet man grundsätzlich zwischen einem höchstleistungs- (starke Kopplung leistungsfähiger Rechenhardware mittels hocheffektiver Netze wie Infiniband) und einem verteilten (schwache Kopplung gewöhnlicher PCs mittels herkömmlicher Netzwerke wie Ethernet) System, wie in **Abbildung 12** dargestellt ist, in Anlehnung an Fujimotos Analyse in [R69]. Die Systeme beider Klassen sind von der Problematik der Synchronisierung, der Kommunikation und anderen grundlegenden Fragestellungen des parallelen und verteilten Rechnens im gleichen Masse betroffen, auf die aber softwarelösungstechnisch aus unterschiedlichen Perspektiven eingegangen wird.

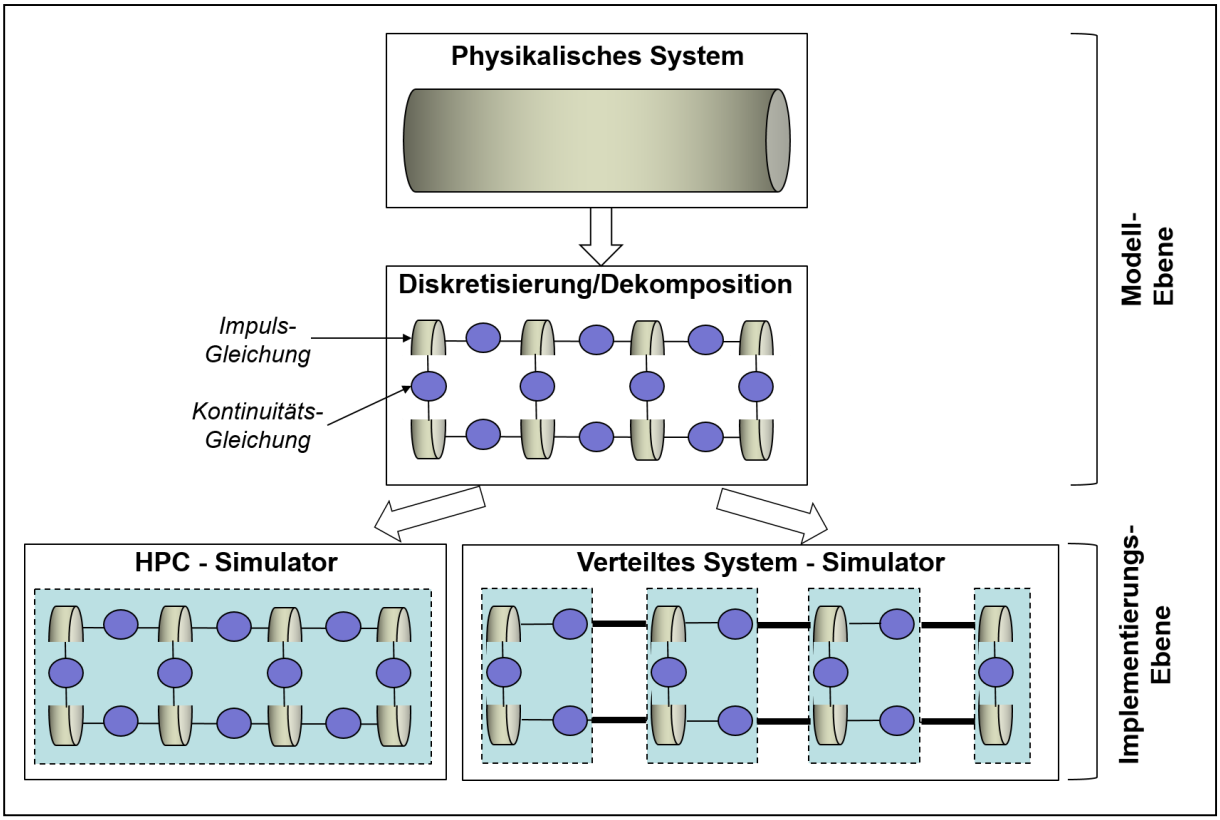


***Abbildung 12**. Implementierung der Modelle auf parallelen und verteilten Rechensystemen.*

Im Bereich des Höchstleistungsrechnens (oder einfach HPC – aus englischem " High Performance Computing") haben sich traditionell Parallelisierungsansätze angesiedelt, die entweder auf MPI (Message-Passing Interface) oder PGAS (Partitionable Global Address Space) Standarten basieren, wobei MPI [W27] eine dominantere Rolle spielt. Die überwiegende Mehrheit der HPC-fähigen Simulationstools sind zum großen Teil homogen und nach dem SPMD (Single Program Multiple Data) Ansatz aufgebaut. Die SPMD-Anwendungen werden z.B. mittels einer MPI-Bibliothek implementiert und können durch ein relativ einfaches Skalierungsverfahren von den verfügbaren Ressourcen Gebrauch machen. Die Dekomposition/Diskretisierung für Objekte mit komplexer Geometrie kann mittels spezieller Tools wie z.B. Parmetis erfolgen, wie etwa von Laflamme in [R70] für zahlreiche CFD-Anwendungsszenarien gezeigt wird. Für die Lösung der daraus resultierenden Gleichugnssysteme bieten sich zahlreiche kommerzielle (wie *Ansys-CFX*[27]) aber auch frei verfügbare (wie *OpenFOAM*[28]) CFD-

[27] https://www.ansys.com/products/fluids/ansys-cfx
[28] https://www.openfoam.com/

Tools an, durch die sich alle Vorteile der HPC-Infrastruktur (wie schneller Datenaustausch, große Speicherkapazität, usw.) effektiv nutzen lassen. Feroze und Genc zeigen in ihrer Veröffentlichung [R71] die Nutzung eines Turbulenz-strömungslösers des *ANSYS-CFX*-Pakets zum Aufbau einer Bewetterungsstudie für südafrikanische Bergwerke. Ein Simulationsbeispiel für australische Bergwerke geben Aminossadati und Hooman in [R72] – in ihrer Studie wurden die *ANSYS-FLUENT* und *ESI-OpenFOAM-ACE* Löser für die Analyse australischer Bergwerke verwendet. Allerdings lohnt sich der Einsatz von *ANSYS* und anderen CFD-Toolboxes eher für hochskalierbare parallele Anwendungsszenarien, die z.B. eine komplexe geometrische Struktur oder andere Besonderheiten aufweisen, da der Betreibungs-aufwand für diese Tools (die Größe auf der Disk, viele Abhängigkeiten von anderen HPC-Tools und Bibliotheken, usw.) auch relativ groß ist. Manche Toolboxes (wie etwa *OpenFOAM*) sind so komplex aufgebaut, dass ihre Distribution mittels eines Containers (z.B. Docker [W28]) erfolgt, der alle Abhängigkeiten beinhaltet, sodass eine Nutzung außerhalb einer HPC-Umgebung kaum möglich ist. Die Landschaft der CFD-Löser ist allerdings sehr breit und beinhaltet auch solche, die auf einer viel sparsamen Weise im Sinne der Ressourcennutzung gebaut sind. So präsentiert Lopez in [R73] die Nutzung des *EasyCFD*[29]-Lösers, der allerdings nur für akademische Nutzungszwecke gedacht ist. Darüber hinaus existieren zahlreiche problemspezifische Lösungen, wie sie z.B. in einer Übersicht von Xu et al. [R74] ausführlich beschrieben sind.

Im Gegenteil zu HPC, ist der Bereich des Verteilten und Cloud-Computing auf dezentralisierte und heterogene Anwendungen ausgerichtet, die im Wesentlichen von MPMD- (Multiple Programs Multiple Data) basierten Technologien geprägt sind. Der MIMD-Ansatz ermöglicht eine breitere Aufstellung der Funktionalität der parallel ausgeführten Softwareeinheiten und bietet mehr Flexibilität bei der Software-entwicklung an. Unter anderem finden verteilte Systeme großen Einsatz im industriellen Umfeld, wo eine dezentrale Aufstellung der Ressourcen (Software wie Hardware) als eine harte Anforderung gilt. Ein Beispiel dafür ist der so genannte "Software-in-the-Loop" (SIL)-Ansatz (wie in **Abbildung 13** dargestellt): Die Software

[29] http://www.easycfd.net/

wird bei der Projektierung sowie während der Nutzung der Automatisierungssysteme unterstützend eingesetzt. Im Falle einer Ausrichtung der Software in die Simulationsaufgaben redet man von einer so genannten "Co-Simulation": Die Simulationen werden lokal zu sämtlichen Kontrollobjekten ausgeführt und ihre Ergebnisse global unter allen beteiligten Objekten zum Optimierungszweck ausgetauscht und bewertet. So z.B. der Ansatz von Piper und Obermaisser, die in [R75] die Implementierung von einem verteilten Simulationsframework für die Analyse eines bahntechnischen Systems beschreiben. Hopkinson et al. präsentieren in der Veröffentlichung [R76] *EPOCHS* – eine Plattform, in welcher einzelne Simulationskomponenten als „Agenten" betrachtet werden, die in einer über das Netz verteilten Föderierungsumgebung funktionieren. Die Agenten-basierte Simulation zeichnet sich durch Eigenschaften wie Ereignisdiskretheit, Plattformunabhängigkeit und anderen aus, wie sie z.B. vom FIPA – Foundation for Intelligent Physical Agents[30] Standard vorgeschrieben sind. Die Anbindung externer Sensoren und Aktuatoren wird in solchen Systemen auch standardisiert gestaltet, z.B. mittels Standarten wie FMI (Functional Mockup Interface[31]) oder ähnlichen, die eine Spezifikation (z.B. in der Form eines XML-Schemas) des Datenaustausches zwischen dem Simulationssystem und den an sie gekoppelten „Instrumenten" vorgeben. Ochel et al. präsentieren in der Arbeit [R77] eine Integrationsschnittstelle für externe Sensoren für *Modelica* (eine verbreitete universelle Modellierungsumgebung).

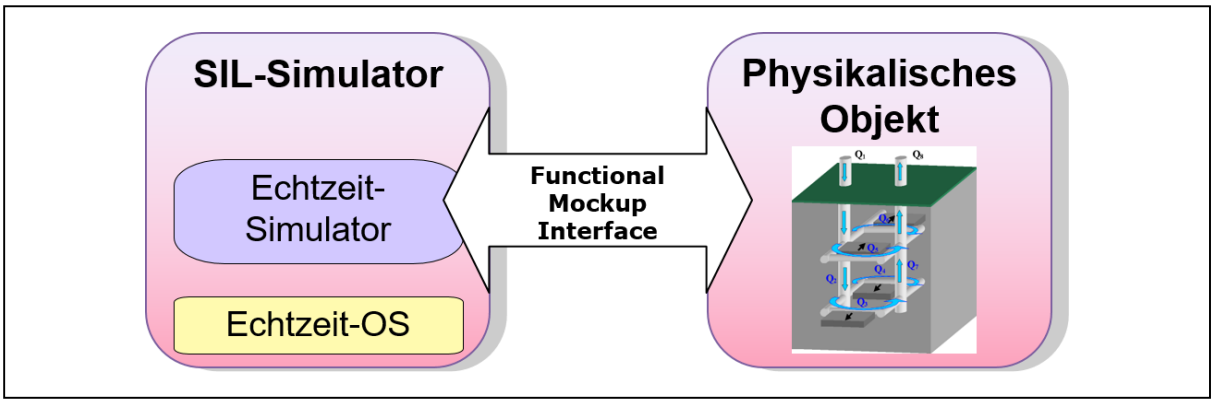


***Abbildung 13**. Simulation im "Software-in-the-Loop (SIL)"-Szenario.*

[30] http://www.fipa.org/
[31] https://fmi-standard.org/

Ähnlich wie im Falle der HPC-Anwendungen, findet die Synchronisation der parallelen Softwareeinheiten in verteilten Systemen mittels eines Datenaustausches über das Netz statt. Der Datenaustausch kann aber aufgrund schlechterer Charakteristiken der in verteilten Systemen eingesetzten nicht-spezialisierten Netzwerke (wie Ethernet) und auch wegen mangelhafter Unterstützung seitens der Programmiermodelle mit einer deutlich schlechteren Effizienz als in HPC-Systemen erfolgen. Auf der Softwareentwicklungsebene wird die Kommunikation in verteilten Systeme entweder mittels universeller Implementierungsmechanismen wie Sockets oder auch mit Hilfe von speziellen, zum industriellen Einsatz geeigneten Streaming-Bibliotheken realisiert, wie z.B. RabbitMQ (wie im Beitrag von Madhu und Dixit [R78] beschrieben) oder Apache Kafka (wie Sharma in [W29] darstellt). Die Leistung dieser Bibliotheken kann aber je nach dem konkreten Einsatzszenario schwanken und muss deswegen gründlich untersucht werden, wie z.B. Dobbelaere und Esmaili in [R79] gezeigt haben. Trotz der bekannten leistungsrelevanten Schwäche, überzeugen die Streaming-Kommunikationstools mit einer weiten Breite der unterstützten heterogenen Hardware- und Software-Typen, was ihren Einsatz in verteilten Systemen aber auch in der Cloud besonders attraktiv macht. Die Letzteren (die mittelgroßen Cloud- und Cluster-Infrastrukturen) werden übrigens immer häufiger mit einem hocheffizienten Netzwerk wie Infiniband ausgerüstet, für welche die Fragestellung der Nutzungs-effizienz der dezentral ausgeführten Softwarekomponente aufgrund hoher Betreibungskosten der darunterliegenden heterogenen Infrastruktur von einer besonderen Bedeutung ist.

Vor diesem Hintergrund ist eine Entwicklung der Programmiermodelle erforderlich, die die Vorteile beider Infrastruktur-Bereiche – HPC und Cloud/verteilten Systemen, im Sinne der Leistungsstärke, Effizienz aber auch Heterogenität der Ressourcen und Nutzungsfreundlichkeit vereinbaren lassen. Mit dieser Fragestellung beschäftigt sich das folgende Kapitel 2.4.

2.4 Programmiermodelle für Skalierbare Simulationssoftware

Wie die Analyse im vorigen Kapitel 2.3 zeigt, stellt die Nutzung der vorhandenen Simulationspakete/Löserbibliotheken wie *ANSYS-CFX* oder *OpenFOAM* zwar eine hinreichende Option für die Implementierung eines problem-spezifischen Simulationsalgorithmus dar, erfordert aber in vielen praktischen Anwendungsfällen eine umfangreiche Anpassung oder sie ist sogar, aufgrund von Faktoren wie z.B. der Verfügbarkeit nichtlinearer Komponenten und starkverzweigter Topologien oder komplexer Aufstellung der initialen Gleichungssysteme unmöglich. Auch im Fall der Echtzeitsimulation für Bewetterungsprobleme (Kapitel 1.4) ist die Nutzung der universellen Tools meistens unmöglich und fordert eher spezielle Modelle mit einem hohen Approximationswert aber gleichzeitig mit ausreichender Qualität der Ergebnisse. Auch die Umsetzung der bereits approximierten Modelle in die vorhandenen Löser-Frameworks ist aufgrund besonderer Anforderungen der Industrieumgebungen (Kapitel 1.5) bzw. der Komplexität der Letzteren nicht immer realistisch. Daher folgen viele Entwicklungen ihrem eigenen Weg. Die „von Scratch“-Entwicklung der qualitativ hochwertigen „custom“ Software mit dem Skalierungspotenzial von einem PC bis einem Cluster-, Cloud- oder HPC-System ist aber eine für das Lösen schwierige und nicht-triviale Aufgabe, insbesondere wenn es um die Implementierung komplexer Modellierungsalgorithmen geht, wie etwa der Strömungsanalyse. Die Erreichung einer hohen Anwendungseffizienz im Sinne der Leistung und Skalierbarkeit ist neben der Erfüllung der funktionellen Anforderungen die wichtigste Zielsetzung der Softwareentwicklung für Simulationszwecke. Voraussetzung dafür ist die Verfügbarkeit eines Programmiermodells, welches die Erstellung der parallel aufgebauten Softwarequellcodes unterstützt (mit dazugehörigen Aufgaben wie der Dekomposition, Synchronisierung, verteilter Ausführung usw.), indem dem Nutzer lösungsorientierte Schnittstellen für die Aufteilung der algorithmischen Funktionalität der Software in die parallel ausgeführten Programmfragmente angeboten und mittels effektiver Frameworks (wie z.B. im Kapitel 2.3 beschrieben) umgesetzt werden.

Am weitesten verbreitete Programmiersprachen (wie C, C++, Java, oder Skript-basierten wie Python) haben leider keine native Unterstützung der Parallelisierungs-

und Verteilungsfunktionalität – sie orientieren sich an seriellen CPUs bzw. an herkömmlichen Symmetrischen Multiprozessoren (SMP) mit einem kohärenten Speicherblock. Dabei verfügen sie aber weder über die Möglichkeit, die parallel auszuführenden Funktionalitäten im Softwarecode mit Hilfe einer deklarativen Syntax zu identifizieren, noch können sie die Ausführung der parallelen Softwarebereiche auf den verfügbaren verteilten Ressourcen unterstützen. Darüber hinaus sind die allgemeinen Programmiersprachen nicht sonderlich nutzerfreundlich für Modellierungsexperten – so wird viel fortgeschrittene Softwareengineering-Kenntnis zur Umsetzung der Modelle als Simulationssoftware benötigt (unter anderem das Wissen über MPI- und OpenMP-Standards der parallelen Kommunikation), welche von Modellierungsexperten nicht immer besetzt sind.

Eine gewisse Vereinfachung des Modellbildungsprozesses kann mit Hilfe spezieller Modellierungssprachen erreicht werden. Diese Sprachen erweitern oder ersetzen gar die native Sprachsyntax der Basisprogrammiersprache (*C* und *C++*) und bieten eine Modellentwicklung auf einem hohen Abstraktionsniveau an. Ein Beispiel hierfür ist *MATLAB* – eine kommerzielle Modellierungssoftwareentwicklungsumgebung von Mathworks bzw. *Octave*[32] – ein mit *MATLAB* kompatibler, Linux-basierter und frei verfügbares Modellierungspaket von GNU. *MATLAB/Octave* bietet eine spezielle, plattform- und programmiersprachenunabhängige Sprachenerweiterung an, die dem Entwickler die Beschreibung der problembezogenen Modellierungsaufgabestellungen (inkl. dynamischer Systeme) mittels vereinfachter Syntax ermöglicht, wie sie im Buch von Haußer und Luchko [R80] ausführlich beschrieben ist. Um die zeitlichen Anforderungen komplexer Simulationsaufgabestellungen (wie z.B. CFD) zu erfüllen, bieten sich problemspezifische MATLAB-Erweiterungen an, wie z.B. QuickerSim – eine auf die Lösung der strömungsdynamischen Problemstellungen ausgerichtete Toolbox (wie sie z.B. in dem Beitrag von Regulski [W30] für typische CFD-Aufgabestellungen präsentiert wird).

[32] https://www.gnu.org/software/octave/

Eine weitere Abstraktion des Modellerstellungsprozesses bietet *Simulink* an – eine Erweiterung der *MATLAB*-Programmierungsumgebung, welche die Erstellung der Modelle anhand von Funktionsblöcken auf visuelle Art ermöglicht, also mittels eines sogenannten blockorientierten Verfahrens. Die textbasierte *MATLAB*- und die grafik-basierten *Simulink*-Pakete werden bei der Modellentwicklung oft kombiniert, indem der fertige *MATLAB*-Code in die *Simulink*-Funktionsblöcke eingefügt wird, was ein perfekter Ansatz zur Entwicklung der vertikal-skalierbaren Algorithmen darstellt (wie z.B. von Kolassa et al. in [R81] beschrieben wurde). So präsentieren in [R82] Röck et al. einen *MATLAB/Simulink*-basierten Ansatz zur Echtzeitsimulation für eine dynamische Materialflussanwendung im Bereich der Automatisierungstechnik. Ein ähnlicher Ansatz wird auch von Spiteller und Trenkel in [R83] oder von Plummer in [R84] verfolgt, etwa für den Aufbau eines HSIL (Hardware/Software-in-the-Loop)-Simulators für sicherheitskritische Anwendungen im Automobilbereich. In **Abbildung 14** wird ein Beispiel zur Erstellung eines blockartigen Modells anhand des mathematischen Modells der Aerogasdynamik in einer Ventilationsstrecke gezeigt.

Die Methodologie des blockorientierten Ansatzes von *MATLAB/Simulink* hat einen besonderen Wert für die intuitive Entwicklung der Modelle – die Funktionalität der Basisblöcke beschreibt die zugrundeliegenden physikalischen Modelle und die Zusammenhänge zwischen den Blöcken bilden die Struktur des modellierten physikalischen Objekts ab. Das Modell wird mit (z.B. durch den Einsatz des Systemanalyseverfahrens) identifizierten elementaren Funktionen (z.B. Modelle der approximativen Elemente) aufgebaut, die nachfolgend auf eine angeordnete Weise zusammengesetzt werden, um die Funktionalität des gesamten Objekts zu simulieren. Die Stärke dieses Ansatzes ist aber gleichzeitig seine Schwäche – die blockartigen Modelle verlieren ihre Qualität (im Sinne der Verständlichkeit für den Nutzer) mit der rasant steigenden Dimensionalität des modellierten Systems (wie die Bewetterungs-netzte mit ihren komplexen, ständig erweiternden und wachsenden Topologien). Folglich lohnt sich der Einsatz des blockorientierten Ansatzes nur für weniger große Modelle mit vorwiegend statischer Struktur.

$$K_y \cdot \frac{dQ_y}{dt} + R_y \cdot Q_y^{\,2} + RR \cdot Q_y^{\,2} = H_y$$

$$A \cdot \frac{dQ_м}{dt} + Q_м = Q_{0м} + BR_ф \cdot \frac{dQ_y^{\,2}}{dt}$$

$$V_{Пу} \cdot \frac{dC_y}{dt} = Q_м - (Q_м + Q_y) \cdot C_y$$

$$V_л \cdot \frac{dC_л}{dt} = Q_{мл} - (Q_{мл} + Q_л) \cdot C_л$$

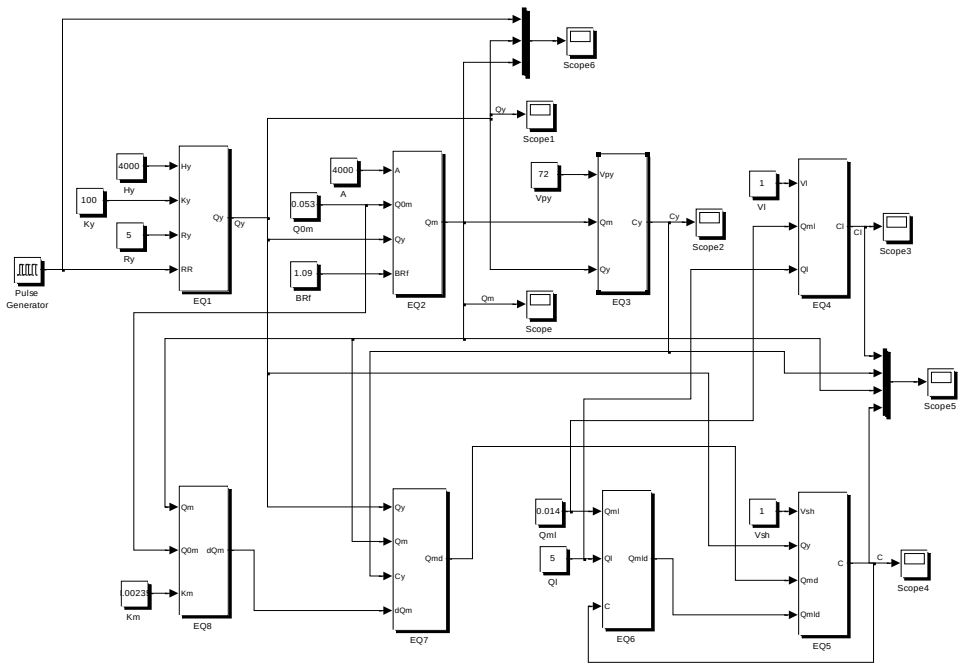
$$V_ш \cdot \frac{dC}{dt} = Q_{мд} + Q_{млд} - (Q_y + Q_{мд} + Q_{млд}) \cdot C$$

$$Q_{млд} = (Q_{мл} + Q_л) \cdot C$$

$$Q_{мд} = (Q_y + Q_м) \cdot C_y + \Delta Q_м$$

$$\Delta Q_м = K_м \cdot \Delta Q = K_м \cdot (Q_м - Q_{0м})$$

a) Gleichungen des Basis-mathematischen Modells



b) Blockorientierte Realisation des Modells

Abbildung 14*. Beispiel eines hierarchisch-aufgebauten blockartigen MATLAB/Simulink aerogasdynamischen Modells einer Ventilationsstrecke.*

Ein Verfahren zur automatischen Generierung des gesamten Modells anhand der Topologie des verzweigten Basisobjekts (z.B. angegeben in Form eines DAG oder aus CAD-Dateien) und einer Spezifikation der Parameter für alle Submodelle existiert aber zurzeit nicht und wird von vielen Forschungsgruppen untersucht, wie z.B. von Smagin

mit Teilergebnissen im Beitrag [R85]. Darüber hinaus sollen bei der Verwendung des MATLABs die Aspekte einer geringen Leistungsstärke (aufgrund eines Interpretationsverfahrens), begrenzter Unterstützung der Betriebssysteme und Systemsoftware sowie anderer, wie z.B. von Waleed in [R86] analysiert wird, beachtet werden.

Ein alternatives Verfahren für den modularen Aufbau der Simulationssoftware anhand gekoppelter, hierarchisch organisierter Funktionsblöcke bieten moderne softwaretechnische Konzepte wie **objektorientierte Modellierung** und **service-orientierte Plattformen** an. Dabei nähert sich der Softwareentwicklungsprozess im Grunde an den strukturierten und modularen Ansatz von MATLAB/Simulink, bietet aber mehr Flexibilität bei der Darstellung der inneren Struktur, Funktionen und Datenabhängigkeiten der Module an – der Softwarekomponente bzw. einzelner Modelle (im Falle einer Modellierungsanwendung) also, die im Sinne der Funktionalität als Softwaredienste (Service) bezeichnet werden. Der aktuelle Trend in der serviceorientierten Programmierung geht in die Richtung von **Mikroservicearchitekturen** – die Module werden als unabhängige (und voneinander isolierte) Dienste dargestellt, die jeweils einen Teil der Funktionalität des gesamten Systems implementieren und durch äußere Kommunikationsschnittstellen mit den anderen Diensten zur Realisierung der Anwendungslogik mittels Datenaustausches verbunden sind, wie z.B. von Fowler in [W31] erläutert wird. Die weiteste Verbreitung fanden die Mikroservicearchitekturen in den Cloud-Anwendungsszenarien, da sie besonders gut zur Implementierung der Container-Funktionalitäten (wie z.B. *Docker, Singularity*, u.a.) der prävalent homogenen Web- und Datenanalytics-Anwendungen geeignet sind. Die Kommunikation zwischen den Services erfolgt mithilfe einer einfachen (z.B. REST-basierten) Schnittstelle und über ein HTTP-Protokoll. Es sind aber auch Kommunikationsansätze für Hochgeschwindigkeitsnetzwerke einer Cluster-Infrastruktur bekannt, wie z.B. von Saha et al. in der Veröffentlichung [R87] geschildert. Die Funktionierung der Services erfolgt im Wesentlichen dezentral, mit einem geringeren Synchronisierungsaufwand der Komponenten des (globalen) Steuerungsframeworks. Das Letztere übernimmt außerdem die Aufgaben der

Lastbilanzierung, Parallelisierung und Ausführung der Services, wie beispielsweise im Beitrag von Meissen et al. [R88] formuliert wird.

In der Simulationstechnik liegt die Mikroserviceideologie am nächsten zu der **agentenbasierten Modellierung** – einem Konzept für die Analyse komplexer Systeme mittels expliziter Beschreibung der Verbindungen zwischen der Mikro- und der Makro-Funktionalebenen des modellierten Systems. Ähnlich zu Mikroservices sind Modellierungsagenten vorwiegend heterogener Natur, besitzen eine individuelle Verhaltungsweise und können Kommunikationsverbindungen innerhalb des Systems (z.B. entsprechend der Lagebeziehung) herstellen, wie beispielsweise von Weyer und Roos in der Veröffentlichung [R89] analysiert. Außerdem heben die Autoren die Rolle der Systemorganisation der Agenten entsprechend der Topologie des realen Objekts (oder eines Netzes von ihnen) hervor: Im Gegenteil zu den „klassischen“ Mikroservices erfolgt die Funktionalitätsunterteilung nach einem komplexen strukturellen bzw. hierarchischem Prinzip aufgrund einer funktionalen, örtlichen oder sonstigen Dekomposition des Zielsystems. Eine Multiagenten-simulation begleitet oftmals die Anwendungsentwicklung in Bereichen wie Logistikprozesse, Autonomverhaltenssysteme, vereinzelt auch für CFD-nahe Aufgabenstellungen (wie z.B. im Beitrag von Grahn et al. [R90]). Während die agentenbasierte Modellierung eine konzeptuelle Plattform zur Erstellung der hierarchisch aufgebauten, komponenten-basierten Modellierungssoftware bereit-stellt, bieten die Mikroservicearchitekturen eine perfekte Grundlage zur Implementierung der (Simulations-) Softwarekomponente für verteilte, heterogene und parallele Recheninfrastrukturen an. Besonders vorteilhaft sind die Mikroservicearchitekturen für Anwendungsfälle aus der Industrie mit ihren strengen Infrastrukturanforderungen. Die Implementierung der Modellierungsagenten als Services mit anschließender Umsetzung der Services in verteilten, plattform-unabhängigen Simulationsanwendungen mittels einer Mikroservicearchitektur bietet ein vielversprechendes Konzept zur Umsetzung dynamischer Modellierungsszenarien, wie beispielsweise im Paper von Collier et al. [R91] zusammengefasst. Praktische Anwendungen von diesem Konzept sind aber weitgehend unbekannt.

2.5 Zielsetzung und Aufgabenstellung der Arbeit

Aufgrund der Anforderungsanalyse (Kapitel 1.5) sowie des aktuellen Standes der Technik und Forschung (Kapitel 2.1-2.4) wurden für dieses Forschungsvorhaben folgende Ziele gesetzt:

(i) Entwicklung eines Programmiermodells sowie einer Ausführungsplattform (Frameworks) zur Durchführung von dynamischen CFD-Simulationsstudien auf den Ressourcen einer verteilten cyber-physischen Infrastruktur.

(ii) Anwendung von mikroservice-orientierten Programmierkonzepten zur Umsetzung der wichtigsten Modelle der Bergbau-Aero- und Gasdynamik in Form von dynamischen Echtzeit-Services sowie Entwicklung von praxis-orientierten Anwendungsszenarien auf ihrer Basis.

(iii) Einsatz von entwickelten Simulationsanwendungen zur Lösung von akuten Aufgabestellung im Basisforschungsbereich der Bergbaubewetterung sowie von anderen Anwendungsdomänen.

Die von der Plattform ausgeführten Simulationsstudien werden auf Modellen basieren, die als Services implementiert werden sollen und von denen in Echtzeit abrufbare Sensordateninformationen aus den vorhandenen cyber-physischen Komponenten der Betriebsinfrastruktur profitieren. Darüber hinaus sollen die Services nicht nur auf der betriebseigenen Infrastruktur (also – auf der Kommando-Zentrale), sondern auch auf verteilten Ressourcen einer externen Cloud- bzw. HPC-Infrastruktur (also **Cloud-Szenario**) sowie den verfügbaren Teilen des eigenen Automatisierungs-systems (also – **Fog-Szenario,** ggf. erweitert durch **Eingebettete Systeme**) ausgeführt werden können, wie in **Abbildung 15** gezeigt ist, um eine für die Durchführung der Echtzeitanwendungsszenarien ausreichende Leistung der Simulationsalgorithmen zu gewährleisten.

Die vorgeschlagene Plattform soll eine softwaretechnische Grundlage zum Einsatz der Simulation in den betriebstechnischen Bedingungen eines Bergwerks schaffen. Das

vorgeschlagene Konzept sollte einige Verbesserungen gegenüber den “state-of-the-art“ Technologien aufzeigen, wie in **Tabelle 2** zusammengefasst.

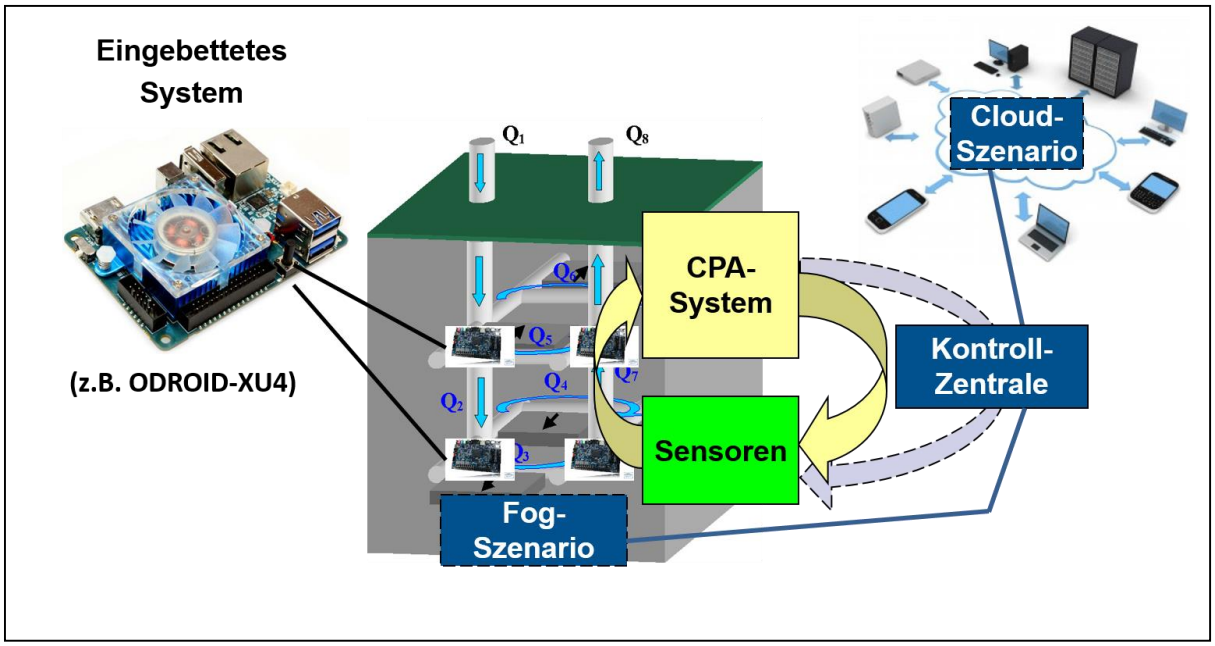


***Abbildung 15**. Verteilte Infrastruktur der dynamischen, mikroservice-basierten Simulationsplattform.*

***Tabelle 2**. Vergleich verschiedener Simulationsansätze.*

Technologie / *Eigenschaft*	**Digital Twin**	**Integrated Simulation**	**SW-in-the-loop (SIL)**	**Dynamische Simulation**
Echtzeitfähigkeit	Übereinstimmung der physikalischen und virtuellen Zeit durch **regelmäßige Synchronisierung**, wie z.B. in [R92]	Nutzung eines HW-SW **übergreifenden Monitoring-Systems**, wie z.B. in [R93]	**Anpassung der numerischen Schrittweite** an Zeitverlauf des physikalischen Prozesses, wie z.B. in [R83][R94]	Die dynamische Simulationsplattform sollte die Vorteile von **Synchronisierung**, **Monitoring**, sowie **numerischer Anpassung** nutzen
Unterstützung komplexer Topologien	-	-	-	Die dynamische Simulationsplattform sollte die **Methodik zur Abbildung komplexer Topologien durch vernetzte Simulations-komponente** anbieten
Dezentralisierte Datenverwaltung	-	-	-	Unterstützung komplexer

verteilter Komponenten				Topologien benötigt einen speziellen **Ansatz zur Speicherung und Analyse lokaler Daten** verteilter Objekte bzw. Simulationsdaten
Heterogenität der Hardware	Die Simulationskomponenten haben das Potenzial zu ihrer Ausführung auf heterogenen, teils mikrokontroller-basierten Plattformen. Allerdings verfügen die meisten Lösungen über eine geringfügige Portabilität auf andere Architekturtypen aufgrund ihrer Basierung auf plattform-abhängigen spezialisierten Frameworks ([R95], [R96])			Die dynamische Simulationsplattform sollte ein universelles **Framework zur Entwicklung der Simulations-Anwendungen** auf der Basis von gängigen Software-lösungen anbieten
Leistung und Skalierbarkeit	-	-	-	Das Design der dynamischen Simulationsplattform sollte die **Effizienz-Aspekte der Simulations-Komponenten, wie z.B. Leistung, Skalier-barkeit**, u.a. nicht vernachlässigen
Energieeffizienz der Simulations-software	-	-	-	Energieverbrauch der Simulations-komponenten von der dynamischen Plattform sollte **messbar** und, im besten Fall, **optimierbar sein**

Im Weiterem wird die Zielsetzung dieses Forschungsvorhabens durch die Forschungsziele (Z), die zur Erreichung dieser Ziele führenden Aufgaben (A) und ihre wesentlichsten Ergebnisse (E) erläutert.

Z1. Methodologische Prinzipien zur Entwicklung von CFD-Anwendungsszenarien mittels der Mikroservicearchitektur

Wie bereits die Erfahrungen aufgrund von Forschungsarbeiten zur agentenbasierten Modellierung gezeigt hat (wie z.B. von Deckert [R97]), profitieren viele Simulations-aufgabestellungen und insbesondere die, die sich durch zahlreiche und verzweigte

hierarchisch-organisierte physikalische bzw. informationstechnische Zusammensetzungen innerhalb des Zielobjektes auszeichnen, von einer komponenten-basierten Realisation der Software. Dabei könnten für service-basierte Simulationsanwendungen Vorteile durch flexiblerer Portabilität, erhöhte Skalierbarkeit und anderen vorher gezeigten entstehen. Hierfür sollen die Grundlagen der Implementierung von CFD-Simulationsalgorithmen mittels der Mikroservicearchitektur erarbeitet werden.

Für die Erreichung dieses Ziels sollen die folgenden Aufgaben gelöst werden:

- **[A1.1]** Erarbeitung eines hierarchischen Ansatzes zur Erstellung einer vereinheitlichten Klassifikation der Modelle von zusammengesetzten, topologisch gegliederten dynamischen Systemen
    - **[E1.1.1]** Formalisierte Darstellung einer hierarchischen Struktur und technologischer Ebenen von Objekten der zusammengesetzten Systeme
    - **[E1.1.2]** Hierarchischer Ansatz zur Klassifikation der Modelle von komplexen dynamischen Systemen
- **[A1.2]** Entwicklung eines Programmiermodells zur Implementierung von servicebasierten Simulationsanwendungen mithilfe eines hierarchischen Ansatzes
    - **[E1.2.1]** Konzeptuelle Architektur eines Mikroservice
    - **[E1.2.2]** Programmiertechniken zur Erstellung der zusammengesetzten (Verbund-) Mikroservices entsprechend der Modellhierarchie
    - **[E1.2.3]** Konzeptuelle Architektur einer mikroservice-basierten Anwendung
    - **[E1.2.4]** Kommunikationsverfahren für den Datenaustausch innerhalb der Mikroserviceanwendung auf einer verteilten Rechenarchitektur
- **[A1.3]** Entwicklung eines Prototyps des Ausführungsframeworks für mikroservice-basierte Anwendungen
    - **[E1.3.1]** Konzeptuelles Design der Architektur eines Ausführungsframeworks für mikroservice-basiere Anwendungen und ihre Services

- **[E1.3.2]** Referenzimplementierung des Ausführungsframeworks mithilfe von modernen software-technischen Lösungen.

Die Erreichung der oben dargestellten Ziele wird zum größten Teil im Kapitel 3 dargelegt.

Z2. Service-basierte Realisation der Basismodelle der Grubenbewetterung

Die in der vorherigen Zielsetzung ausgearbeiteten methodologischen Prinzipien des mikroservicebasierten Entwicklungsansatzes sollen nun auf die Basismodelle des Forschungsgebiets „Grubenbewetterung“ angewandt und die entsprechenden Services entwickelt werden. Zum einen soll diese Zielsetzung die Nutzungsaspekte der Mikroservicearchitektur im praktischen Einsatz vorhandener Aufgabestellungen in der Industrie und Forschung demonstrieren. Zum anderen wird eine gründliche Analyse der bestehenden Modellierungsansätze durchgeführt sowie neue Modellierungskonzepte erarbeitet.

Die spezifischen Aufgaben, die für die Erreichung dieses Ziels nötigt sind, werden wie folgt definiert:

- **[A2.1]** Klassifikation der Basismodelle der Grubenbewetterung auf der Basis eines hierarchischen Ansatzes
  - **[E2.1.1]** Überblick über vorhandene CFD-Modelle in der Aerodynamik
  - **[E2.1.2]** Überblick über vorhandenen CFD-Modelle in der Gasdynamik
  - **[E2.1.3]** Spezifikation der Objektebenen und Modellebereiche in der Modellhierarchie des Basisgebietes
  - **[E2.1.4]** Modellhierarchie, die alle elaborierten Modelle umfassen soll
- **[A2.2]** Erarbeitung von Techniken und Lösungen zur Implementierung der hierarchisch organisierten Services der Modellhierarchie
  - **[E2.2.1]** Workflow-basierter Entwurf einer Mikroservice-Anwendung
  - **[E2.2.2]** Techniken zur formalen Beschreibung der zu entwickelnden Simulations-Services und Anwendungen

Die Erreichung der oben dargestellten Zielsetzung wird im Wesentlichen im Kapitel 4 dargelegt.

Z3. Umsetzung der Modelle mittels der Mikroservicebasierten-Simulationsplattform und Validierung der Ergebnisse

Die vorhandenen Modelle werden mit Hilfe des Mikroservicearchitekturansatzes als Softwarekomponente (Simulatoren) implementiert und in die dynamische Simulationsplattform integriert. Die Validierung wird sowohl auf der Grundlage der Benchmarks, die zur Verifizierung der ursprünglichen Basis-Modelle verwendet wurden, durchgeführt als auch für die Aufgabestellungen aktueller Objekte (z.B. dem Bergwerk „Süd-Donbass 3“ in der Ukraine). Es wird dabei validiert, ob die funktionellen (Korrektheit der Ergebnisse) und nichtfunktionellen (Zeit, Leistung, Energieverbrauch usw.) Anforderungen gerfüllt werden können. Insbesondere soll das Einsatzpotenzial der entwickelten Simulatoren in Echtzeitanwendungsszenarien evaluiert werden.

Für die Erreichung dieses Ziels sollen die folgenden Aufgaben gelöst werden:

- **[A3.1]** Umsetzung der wichtigsten Modelle als Mikroservices
    - **[E3.1.1]** Bibliothek der Simulationsservices
    - **[E3.1.2]** Beschreibung der Benchmark-Tests
    - **[E3.1.3]** Validierungsergebnisse

Die Erreichung der oben dargestellten Zielsetzung wird in den Kapiteln 4 und 5 dargelegt.

Z4. Erstellung der Anwendungsszenarien

Die entwickelten Simulationsmikroservices werden zur Realisation der dynamischen Simulationsanwendungsszenarien angewendet. Dafür werden Applikationen konzipiert und entwickelt, die möglichst umfangreich verschiedene Funktionalitäten und Nutzungsaspekte der Mikroservicearchitektur zur Lösung aktueller betriebs-technischen Aufgaben verwenden sollen. Anschließend wird eine Evaluierung unter Betriebsbedingungen industrieller Einsatzobjekte durchgeführt.

Für die Erreichung dieses Ziels sollen die folgenden Aufgaben gelöst werden:

- **[A4.1]** Erstellung der Anwendungsszenarien
    - **[E4.1.1]** Spezifikation der Anwendungen
- **[A4.2]** Entwicklung der mikroservicebasierten Anwendungen und Evaluierung der Ergebnisse durch die Endnutzer
    - **[E4.2.1]** Evaluierungsprotokolle für Anwendungsergebnisse und Nutzererfahrungen.

Die Erreichung der oben dargestellten Zielsetzung wird im Kapitel 5 dargelegt.

3 Entwicklung der Simulationssoftware mittels Mikroservicearchitekturen

Dieses Kapitel beschreibt den vorgeschlagenen Ansatz zur Entwicklung der Simulations-Anwendungen und Studien mittels Mikroservicearchitektur-basierten softwaretechnischen Lösungen. Es wird eine Grundlage zur Erstellung der Plattform erarbeitet und die Umsetzung der Basis-CFD-Anwendungsszenarien dargelegt.

Der weitere Inhalt bietet zuerst eine Übersicht des konzeptuellen Designs einer Softwarearchitektur zur Entwicklung der mikroservice-basierten Simulations-anwendungen (Kapitel 3.1). Eine Übersicht über die Architektur eines einzelnen Mikroservice folgt (Kapitel 3.2); anschließend wird das Mikroservice-Anwendungs-modell erläutert (Kapitel 3.3). Im Abschluss wird ein Design für ein Ausführungs-framework der Mikroservice-Anwendung präsentiert (Kapitel 3.4).

3.1 Vorgehensmodell zur Entwicklung der Modellierungsservices

Das Entwicklungsvorgehen für Modelle physikalischer Objekte und Prozesse sowie für ihre Software (was im Softwareengineering als Vorgehensmodell bezeichnet wird) erfolgt meistens nach dem hierarchischen Prinzip – die infolge der Systemanalyse- und Dekompositionsanwendung erzeugten Submodelle (z.B. Erhaltungs- und Transport-Gleichungen in approximativen finiten Elementen der Navier-Stokes- Gleichung) werden kombiniert, indem sie, zusammen mit ihren funktionellen Abhängigkeiten, in die resultierenden Gleichungen des Gesamtsystems eingebaut werden. Dies ist auch der Fall für die meisten CFD-Modelle, die beispielsweise mithilfe von Approximationsverfahren wie FEM, FDM oder FVM (wie in Kapitel 2.3 beschrieben) aufgebaut werden. Die daraus resultierenden Gleichungssysteme zeichnen sich durch eine hohe Komplexität (große Dimensionalität des Systems, viele hierarchische Zusammenhänge, usw.) aus und erfordern spezielle Parallelisierungsansätze zur Softwareimplementierung (wie beispielsweise der von *ParMETIS*, wie in Kapitel 2.3 dargestellt).

Stattdessen wird in dieser Arbeit ein Ansatz vorgeschlagen und erarbeitet, nach welchem eine Simulationsanwendung durch die funktionelle Komposition vieler hierarchisch-organisierter, wechselwirkender Simulations-Mikroservices (Services einer meist feingranularen Natur), die Modelle der betroffenen Prozesse/ Systemfunktionalitäten im Sinne der folgenden Definition[33,34] realisiert:

> **Definition 1**: Als **Simulations-Mikroservice** wird eine autonome, kompositionsbare Komponente einer SOA-Anwendung definiert, welche einen (z.B. durch die Domäne-Diskretisierung) bestimmten Teil der Modell-Logik realisiert.

Ist die Funktionierung der Modellalgorithmen an die Echtzeit gebunden, so kann ein Mikroservice (MS) als ein echtzeitfähiger MS bezeichnet werden. Ähnlich wie die klassischen Services im Softwareengineering, zeichnet sich ein MS durch folgende Eigenschaften aus:

1. **Lokalität** – jeder MS ist für die Modellierung der Prozesse/Objekte in dem ihm zugewiesenen Teilbereich der gesamten Anwendungslogik (also des Modells) entsprechend der Systemdekomposition zuständig.
2. **Ausführungsautonomie** – jeder MS funktioniert unabhängig von den anderen Services des Systems und kann die Ausführungsordnung selbst bestimmen.
3. **Konnektivität** – jeder MS kann mit den anderen Services die für seine Funktionierung notwendigen Daten (z.B. Randbedingungendaten der Simulation) austauschen, z.B. mithilfe eines Kommunikationsmodells (siehe Beispiel in **Abbildung 16**).
4. **Kommunikationsautonomie** – jeder MS kann selbst über die Kommunikation mit anderen Teilen der Mikroservicearchitektur entscheiden.
5. **Assoziationsautonomie** – jeder MS kann selber entscheiden, ob und wie er seine Funktionalität mit den anderen Services teilt.

[33] Ein wesentlicher Unterschied zu den „klassischen" Mikroservices liegt in der Zustandsberücksichtigung, was für Simulationszwecke als wichtige industrielle Anforderung gilt.
[34] SOA – Service-Oriented Architecture

6. **Kompositionsbarkeit** – die MS können zusammengesetzt werden, um größere Systeme mit übergreifender Funktionalität mit einem möglichst geringen Implementierungsaufwand zu modellieren. Dabei beruht die Erstellung der komponierten Services auf der Wiederverwendung bereits vorhandener Funktionalitäten, die vorher in anderen MS implementiert wurden.
7. **Orchestrierung** – die MS besitzen eine einheitliche Schnittstelle zur Implementierung ggf. Steuerung (z.B. seitens einer Clientanwendung) aller oben aufgeführten Funktionen/Eigenschaften.

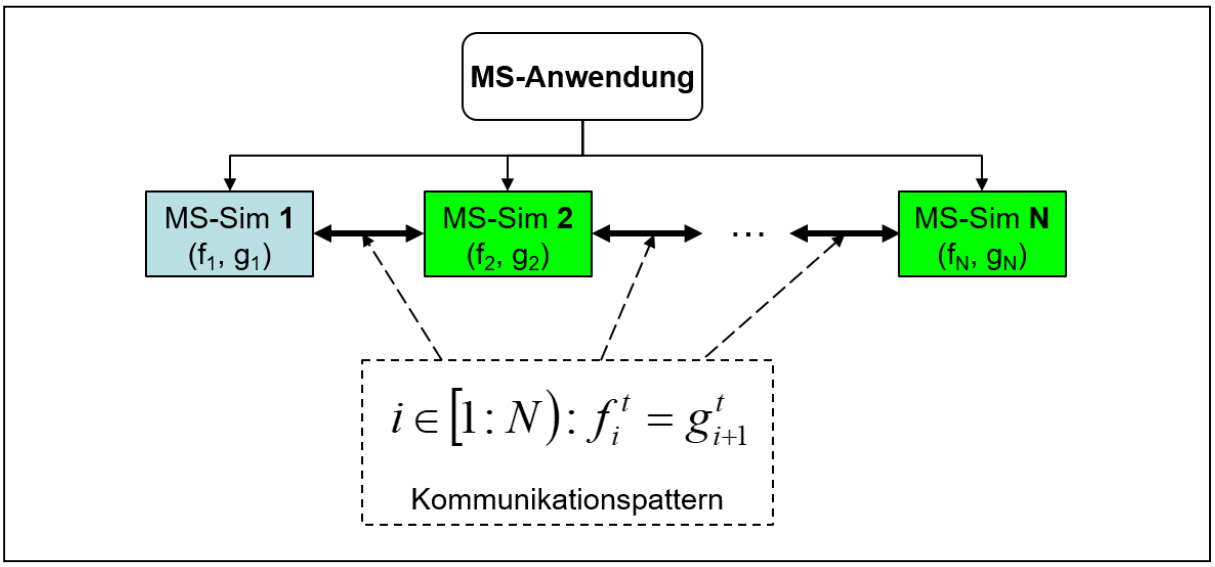


***Abbildung 16**. Kommunikationspattern einer Mikroservice-basierten Anwendung.*

Darüber hinaus soll die oben aufgeführte Liste durch eine besondere Eigenschaft der Modellierungsanwendungen für komplexe, hierarchisch organisierte dynamische Systeme erweitert werden:

8. **Hierarchische Angliederung** – die Erstellung der MS sowie ihre Komposition zu einer Anwendung wird durch ein Regelwerk gesteuert, welches auf der Klassifikation der hierarchischen Struktur der technologischen Objekte basiert.

Die Klassifikation der modellierten technologischen Objekte und die daraus folgende **Hierarchie der Services** entstehen meistens aus einer Anwendung der system-theoretischen Methodiken durch Domänexperten: Dabei werden die hierarchischen Ebenen aus Sicht der Kapselung einzelner Objekte bzw. Granularität der Prozesse in der Struktur des Systems geschaffen. Typischerweise entspricht die Hierarchie den

geschachtelten Strukturebenen und der Logik der Organisation der Betriebsprozesse. So fangen zum Beispiel für kontinuierliche physikalische Probleme die untersten Hierarchieebenen schon mit elementaren Zerlegungseinheiten des räumlichen Kontinuums (der Gitterelemente oder Punkte, je nach angewandten Diskretisierungsverfahren) an. Die Implementierungsalgorithmen für die in ihnen ablaufenden dynamischen Prozesse (z.B. Transport- und Erhaltungssgesetze für Fluide) werden in einzelne Mikroservices eingeteilt. Diese Services bilden dann die Grundlage zur Implementierung der Modelle höherer Funktionalebenen, bis hin zu den obersten Stufen der topologischen Organisation der Systeme.

Die Hierarchieebenen können weiterhin horizontal aufgespaltet werden, z.B. nach technologischen Funktionalitäten spezifischer Prozesse bzw. betriebsspezifischen Tätigkeitsrichtungen. Die Gesamtheit der Modelle $m(i,j) \in M$ aller hierarchischen Ebenen *(i=1..N)* und Tätigkeitsrichtungen *(j=1..K)* bilden die Menge der hierarchisch organisierten Modelle ab, wie in **Abbildung 17** dargestellt.

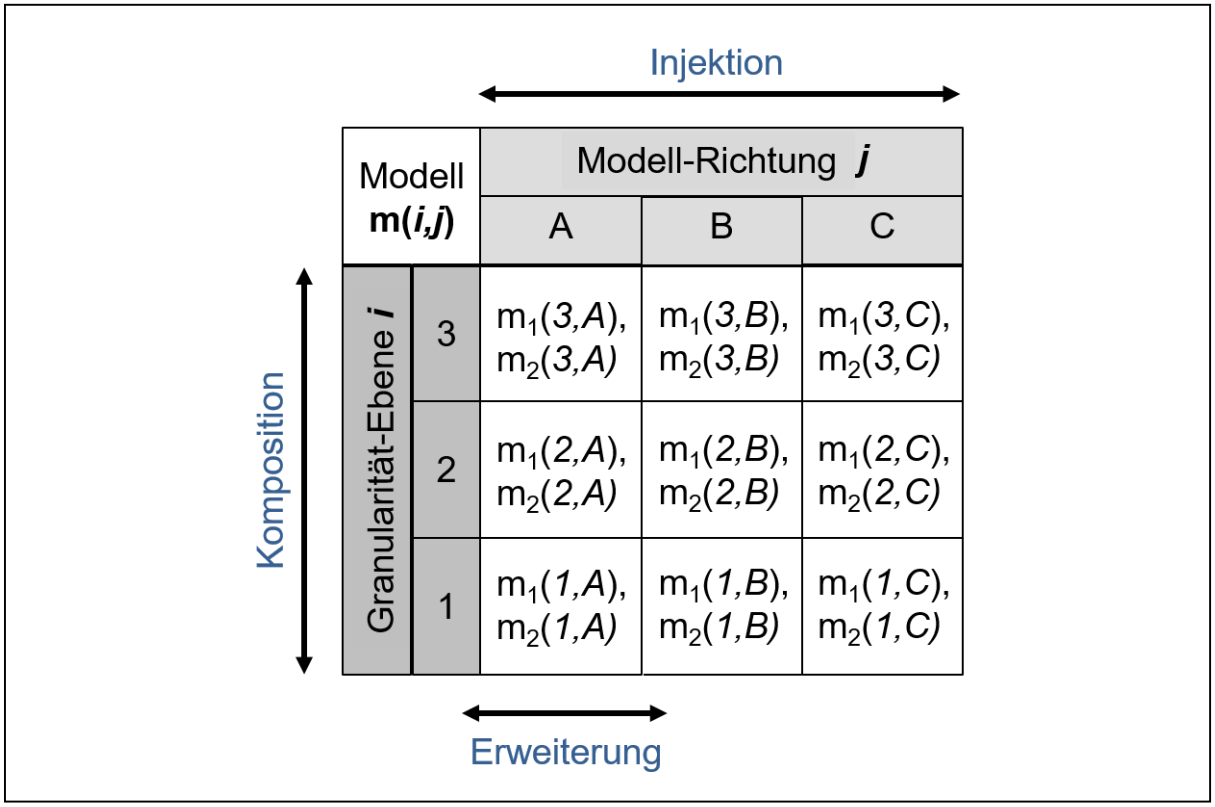


***Abbildung 17**. Klassifikation der Modelle anhand hierarchischer Struktur der Objekte.*

Die Entwicklung der Modellierungsanwendungen mit umfangreicher Funktionalität, also für komplexe, strukturiert aufgebaute und hierarchisch organisierte Objekte bzw. Prozesse, sollte mithilfe des MS-basierten Ansatzes deutlich einfacher zu gestalten sein als mittels klassischer Verfahren (wie in Kapitel 2.2 analysiert), indem der Implementierungsaufwand für die Menge aller hierarchisch zugeordneten Kompositionsservices durch softwaretechnische Formalisierung ihrer Zusammenhänge und Relationen stark reduziert werden kann. Dieses Ziel kann mit Hilfe der folgenden wichtigen Techniken erreicht werden:

- Objektorientierte Modellierung (z.B. Erweiterung/Polymorphismus, Injektion) – zur Implementierung der Modelle verschiedener Tätigkeitsbereiche.
- Servicekomposition – zur Implementierung der Modelle höherer Hierarchie-Ebenen.

Die Services *m(i,j)*, die zu unterschiedlichen Tätigkeitsbereichen ggf. Richtungen gehören (*i=const*), sollten möglichst mittels der Konzepte und Techniken einer **objektorientierten Modellierung** wie Polymorphismus implementiert werden. Dabei wird das umfangreichste Modell als Basis zur Realisierung der Modelle weiterer Tätigkeitsrichtungen gewählt, die während der Entwicklung um die benötigte Funktionalität für die anderen Modelle erweitert wird. Durch den Vorgang lässt sich sowohl der Implementierungsaufwand senken als auch die Größe und Komplexität des Softwarecodes durch die Wiederverwendung vorhandener Funktionalität reduzieren. Die für die Modellierung notwendigen funktionellen Zusammenhänge bzw. Abhängigkeiten zwischen den in Services implementierten Algorithmen (z.B. die Lösung des Cauchy-Problems im Fall der durch Differentialgleichungen definierten Modelle) werden mittels expliziter Kommunikation zwischen den betroffenen Mikroservices, im Sinne von Parallelcomputing-Ansätzen, wie z.B. MPI (wie in Kapitel 2.3 beschrieben) dargestellt. Der Mikroserviceansatz greift hierfür auf das block-orientierte Entwicklungsprinzip von MATLAB/Simulink zu, bietet aber die Vorteile der Software-basierten Entwicklungsmethoden, wie z.B. die Nutzung der zentralen Konzepte der objektorientierten Programmierung (wie Klassenverkapselung etc.), z.B. mittels UML[35].

[35] UML – Unified Modelling Language

Die wesentlichsten Vorteile verschafft jedoch der serviceorientierte Ansatz zur Erstellung der Services der oberen Hierarchieebenen. Dabei werden **Service-kompositionskonzepte** benutzt, die die vorhandenen (für die unteren Hierarchie-ebenen entwickelten) Services zu neuen Services der oberen Hierarchieebenen zusammenbauen lassen können (wie z.B. in **Abbildung 18** dargestellt). Dazu werden beispielsweise die Technologien der zentralisierten Koordination („Orchestration“) genutzt – der übergeordnete Service wird als „Orchestrator“ betrachtet, der die (Simulations-) Aufgaben unter den einzelnen untergeordneten Services verteilt. Die Verteilung kann mittels spezieller Ablaufspezifikationen (Kommunikationspattern) oder Protokollen wie *BPML*[36], *BPEL*[37] und anderen entsprechend der hierarchischen Struktur der Modellierungsobjekte erfolgen, wie es z.B. von Masak in [R98] gezeigt wird. So verschaffen komplexe Modelle die Möglichkeit, sofort auf dynamische Veränderungen in Teilsystemen zu reagieren. Die auf dem Serviceansatz basierten Clientanwendungen werden auf dezentrale Weise gebaut, beispielsweise mithilfe des Choreographie-Verfahrens (mittels Protokolle wie *WS-CDL*[38]).

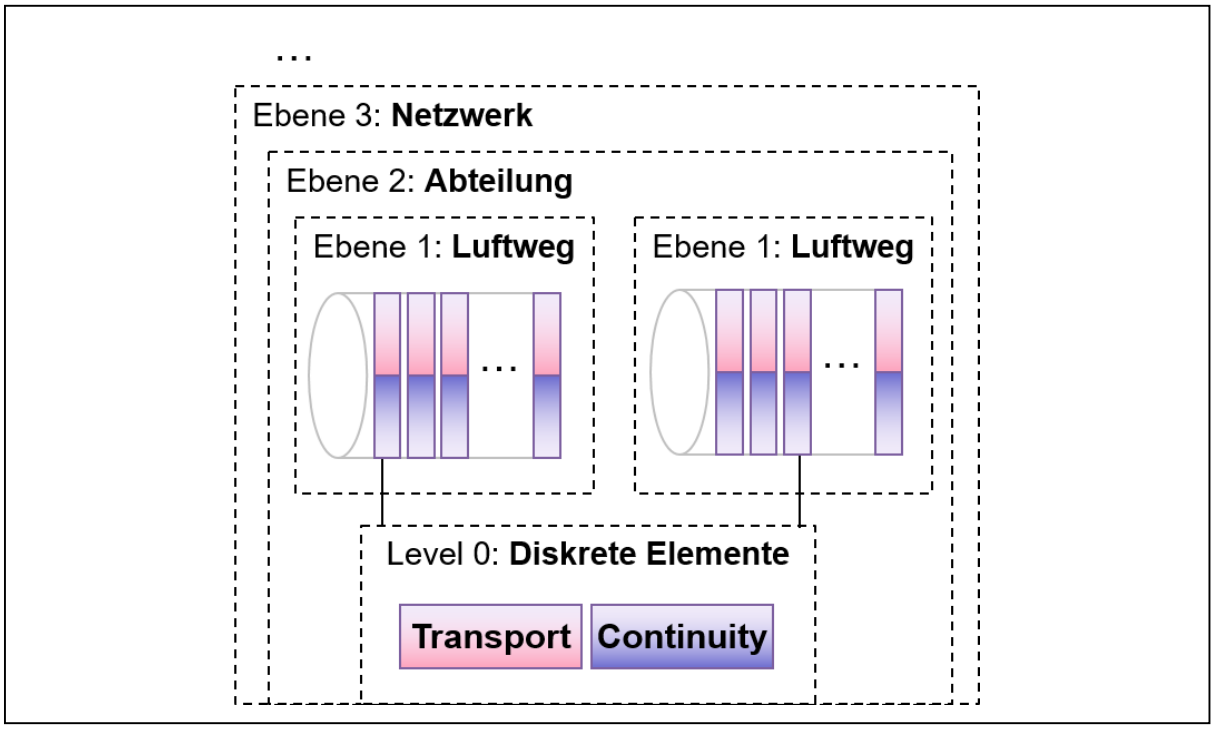


***Abbildung 18**. Hierarchische Komposition der Mikroservices.*

[36] BPML – Business Process Modelling Tool
[37] BPEL – Business Process Execution Language
[38] WSDL – WebService-Choreography Description Language

Die SOA-Architekturen ermöglichen einige wichtige Vorteile (schnellere Umsetzung, bessere Portabilität, usw.) gegenüber den monolithischen Softwareentwicklungsansätzen, benötigen dafür aber einen strukturierten, systematischen Vorgang zur Entwicklung der servicebasierten Anwendungen für komplexe, modulare, hierarchisch aufgebaute Systeme, insbesondere in Simulationsszenarien.

Der erarbeitete hierarchische Ansatz bietet sich als einheitliche methodologische Grundlage zur Systematisierung der Services für Modelle komplexer, hierarchisch aufgebauter dynamischer Systeme nach den Prinzipien der Zugehörigkeit zu vertikalen Hierarchieebenen der physikalischen Prozesse und horizontalen Tätigkeits- (bzw. Modell-) Bereichen der gesamten Modellhierarchie an.

Das auf diesem hierarchischen Ansatz aufgebaute Vorgehensmodell wird im Rahmen dieses Forschungsvorhabens zur Implementierung der Modelle des Basisgebietes „Grubenbewetterung“ umgesetzt (siehe Kapitel 4)

3.2 Architektur eines Mikroservice

Ein Mikroservice kann eine vollständige und eigenständige Funktion ausführen, unabhängig von anderen Diensten einer Anwendung. Jeder Mikroservice verfügt über seinen eigenen Code, eine Konfiguration sowie über Daten, die lokal an der ausführenden Plattform vorhanden sind. Dabei verfügt er über folgende grundlegende Eigenschaften:

- *Servicepartitionierung* – jeder Service ist nach dem MPMD[39]-Prinzip aufgebaut und in seiner Funktionalität auf die Aufgaben der zugewiesenen Domäne beschränkt, z.B. infolge der Anwendung einer Dekompositionsstrategie auf ein physikalisches Problem bzw. einen mathematischen Modellsatz, unabhängig von den anderen Services.
- *Plattformunabhängigkeit* – jeder Service kann auf eine beliebige unterstützte Hardwareplattform des verteilten Systems gekoppelt und ausgeführt werden.

[39] MPMD – Multiple Programs, Multiple Data

- *Lose Kopplung* – jeder Service kann sich mit jedem anderen Service mittels atomarer Transaktionen auf eine asynchrone Weise austauschen.
- *Koordinierte Ausführung der Simulationsalgorithmen* – der Service kann von einer zentralen (Master-) Komponente einer MS-basierten Anwendung gesteuert werden, um Simulationsstudie-spezifische Aufgaben zu erfüllen.
- *Umsetzbarkeit* – die Service-Implementierung soll mit einem minimalen Aufwand für Entwickler möglich sein.
- Darüber hinaus muss das MS-Programmiermodell (bzw. die verwendeten API) die folgenden spezifischen Anforderungen berücksichtigen:
- Leicht zu verwaltender Lebenszyklus, z.B. Präsenz eines klar definierten Einstiegspunktes und einer (Simulations-) Anwendungslogik.
- Konsistenz (Zustandsbehaftung) der relevanten Eigenschaften (falls von der Anwendungslogik her erforderlich).
- Ein von der Funktionalität unabhängiges und austauschbares Kommunikations-modell für die Synchronisation zwischen den Services.
- Eine konsistente und zuverlässige Speicherung der Ergebnisse, z.B. auf einem lokalen Datenträger oder einer entfernten Datenbank.

Um den Programmcode schnellstmöglich in einen Mikroservice einbinden und die Mikroservice-basierte Anwendung ausführen zu können, wird vorgeschlagen, den Lebenszyklus einer MS mit Hilfe der folgenden Methoden (z.B. zu implementieren innerhalb einer zugehörigen Klasse) zu implementieren (siehe auch **Abbildung 19**):

- **Initialisierung** – erfolgt mit Hilfe eines Konstruktors der Mikroservice-Klasse. Dabei wird der Service durch einen eindeutigen Identifikator gekennzeichnet, welcher entweder als Parameter zum Konstruktor hinzugefügt oder, andernfalls, von dem MS-Ausführungsframework automatisch zugewiesen wird. Auf die gleiche Weise werden dem Konstruktor alle spezifischen Parameter des Mikroservice übergeben.

- **Ausführung** – geschieht durch das Aufrufen der speziellen Methode „*run*“ der MS-Implementierungsklasse durch das MS-Ausführungsframework, die als Eintrittspunkt für die zugehörige Simulations-Geschäftslogik dient. So werden jegliche Hintergrundprozesse gestartet, die während der Dauer des Mikroservice ausgeführt werden sollen.
- **Finalisierung** – wird durch das Ausführen der Methode „*stop*“ durchgeführt. In dieser Phase werden Abschlußfunktionen durchgeführt, wie beispielsweise eine finale Datenspeicherung und sonstige Abbruchsfunktionen.

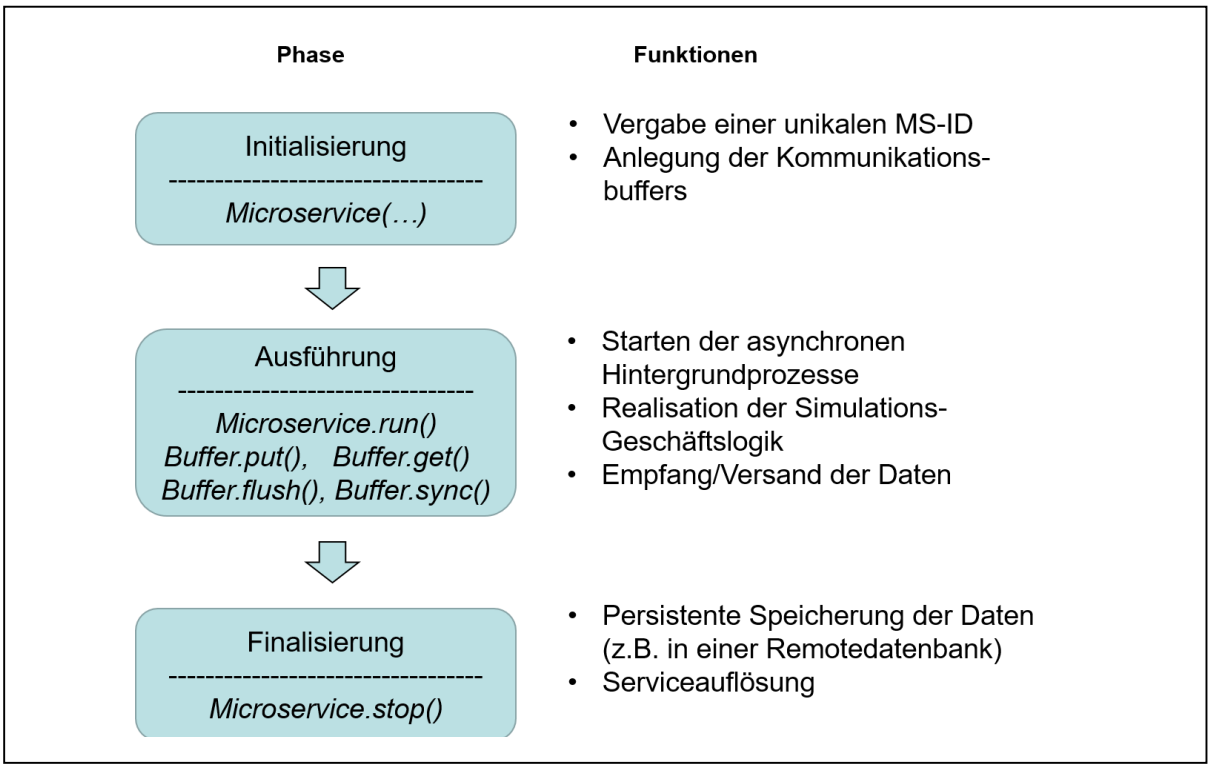


Abbildung 19. *Lebenszyklus eines Mikroservice.*

Falls von der Anwendungslogik benötigt, können die Mikroservices während der Ausführung mit den anderen Services Daten austauschen. Der Austausch erfolgt mithilfe von **Buffers** – geordneten Listen von Daten eines bestimmten Typs, die entweder vom Service selbst oder vom Ausführungsframework in der MS-Anwendung angelegt werden. Die Buffers dienen beispielsweise der Übermittlung der datenflußsteuernden Befehle zwischen den Services oder können zum Austausch der „Halo“-Daten zwischen den topologisch benachbarten Elementen während der Durchführung der Simulationsaufgaben oder sonstiger Austauschzwecke

entsprechend der Anwendungslogik benutzt werden. Alle Buffers des Service werden mittels einer fortlaufenden Nummerierung, die weiterhin als **Port** bezeichnet wird, eindeutig gekennzeichnet.

Die Daten können in bzw. aus einem Buffer mithilfe von „*Put“*- und „*Get“*-Methoden der Buffer-API kopiert bzw. geholt werden. Der Datenversand durch den Buffer erfolgt auf synchrone Weise durch den expliziten Aufruf der „*Flush*“-Methode (vom Absender-Service) bzw. der „*Sync*“-Methode (vom Empfänger-Service). Weitere Kommunikationsmöglichkeiten sind im nächsten Kapitel 3.3 beschrieben. Ein Listing des Codes zweier einfachen Beispielservices und die Realisation der Datenübermittlung zwischen ihnen sind in **Abbildung 20** gezeigt.

3.3 Mikroservice-Anwendungsmodell

Im Sinne des verfolgten serviceorientierten Ansatzes kann eine Anwendung als eine Service-Fabrik betrachtet werden – eine Zusammensetzung von einzelnen Mikroservices, die eine bzw. mehrere bestimmte Funktionen realisieren und mit den anderen Service durch die gemeinsame anwendungsübergreifende Geschäftslogik vereint sind, wie sie z.B. von einem Simulationsalgorithmus gefordert wird. Die praktischen Aspekte der Nutzung von Mikroservices, die nach dem im vorigen Kapitel 3.2 erarbeiteten Leitpfaden erstellt werden können, belaufen sich auf die Lösung von zwei wichtigen Fragestellungen, die in den anschließenden Unterkapiteln diskutiert werden:

- Datenaustausch zwischen den Mikroservices (bzw. ihren parallelen Instanzen) in dynamischen Anwendungsszenarien mithilfe von Kommunikatoren.
- Zusammensetzung der Mikroservices mithilfe von Proxy-Buffers.

Kommunikationsmodell

Für die Synchronisierung der Mikroservices und eine koordinierte Durchführung von Steuerungsalgorithmen der (Simulations-) Anwendungen bzw. dem Austausch der notwendigen Daten zwischen den Services ist ein Kommunikationsmodell erforderlich. Der Datenaustausch sollte zwischen Mikroservices mittels Buffers geschehen, wie im

```
class Commands {
public:

    enum Command_codes {
            init                = 0,
            simulation          = 1,
            report_results      = 2,
            stop                = 3
    };
};

class MS_Master: public Microservice {
public:

    LocalIntBuffer* snd_buf;

    MS_Master(int id_, string id_str_, Communicator* communicator_)
        : Microservice(id_, id_str_, communicator_) {

        // initialisation of buffer
        snd_buf = new LocalIntBuffer (
            id /*ms_id*/, id_str, 0 /*port*/, communicator);
    };

    void run() {

        // adding command code to sending buffer
        snd_buf->add_value (Commands::init /*value*/);
        // sending command
        snd_buf->flush_collective_replicate ();
        // clearing buffer
        snd_buf->clear ();

        cout << "Master: sent init command" << endl;
    };
};
```

(a) Absender

```
class Commands {
public:

    enum Command_codes {
            init                = 0,
            simulation          = 1,
            report_results      = 2,
            stop                = 3
    };
};

class MS_Slave: public Microservice {
public:

    LocalIntBuffer* rcv_buf;

    MS_Slave(int id_, string id_str_, Communicator* communicator_)
        : Microservice(id_, id_str_, communicator_) {

        // initialisation of buffer
        rcv_buf = new LocalIntBuffer (
            id /*ms_id*/, id_str, 0 /*port*/, communicator);
    };

    void run() {

        // receiving command
        rcv_buf->sync();
        int command = rcv_buf->get_int_value (0 /*index*/);

        cout << "Slave: received command: " << command << endl;
    }

};
```

(b) Empfänger

Abbildung 20. *Beispiele von einfachen Mikroservices.*

vorigen Kapitel 3.2 dargestellt ist. Allerdings verfügen die Mikroservices zum Zeitpunkt ihrer Entwicklung grundsätzlich über keine Kenntnis darüber, von welchen anderen Services die Daten geholt bzw. an welche Stellen sie verschickt werden müssen: Die Gewährleistung der Flexibilität der dynamisch zu gestaltenden Mikroservice-Anwendungen erfordert, dass die Verbindungspatterns zwischen den Services erst zur Laufzeit der Applikation, also dynamisch, bestimmt und konfiguriert werden sollen. Aus diesem Grund wurde für die Zwecke des Buffer-Austausches zwischen den Mikroservices ein **indirektes Kommunikationsverfahren** erarbeitet: Die Verbindungs-patterns werden im globalen Anwendungskontext definiert und vom Service mithilfe eines lokalen Buffer-API-Aufrufs für die Abwicklung des Datenaustauschvorgangs benutzt. Die Definition der inter-service Verbindungen erfolgt mithilfe des **Kommunikators** – einer Komponente der Mikroserevice-API, die jedem Mikroservice von der Mikroservice-Anwendung während der Initialisierungsphase zugewiesen wird und die Datenübermittlungsstrategie für den Mikroservice bestimmt, wie er in **Abbildung 21** skizziert ist.

Ein Kommunikator enthält alle Kommunikationsverbindungen als gerichtete „Links“ – die Listen mit paarweise eingetragenen Verbindungen zwischen einem Sender und einem Empfänger. Für jede neue inter-service Verbindung muss ein neuer Link angelegt werden, wie er für das Beispiel der Master-Slave Anwendung (von der **Abbildung 20**) im Listing in der **Abbildung 22** aufgezeigt wird.

Der Kommunikator wird in der internen Implementierung der Buffer-API auch benutzt, um den Datenaustausch mithilfe einer Kommunikationsbibliothek (siehe Kapitel 3.3) abzuwickeln, die an die gegebene Netzwerktransport-Infrastruktur (z.B. Ethernet, Infiniband) angepasst ist. Außerdem bietet der Kommunikator den Mikroservices auch die Möglichkeit an, die Einzelheiten über die ein- und ausgehenden Verbindungen aller Buffers abzufragen, was auch mittels einer spezieller API erfolgen kann.

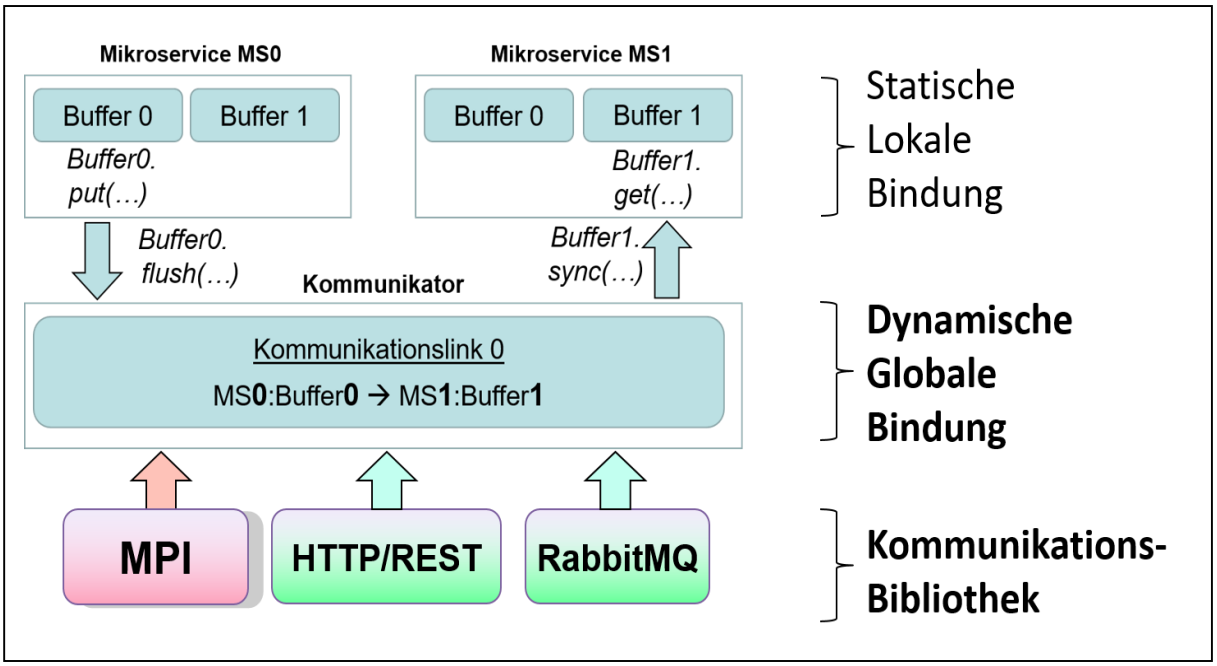


***Abbildung 21**. Asynchrones und indirektes Kommunikationsmodell für Mikroservices.*

```
1 Communicator* communicator = new NumaCommunicator();
2
3 // Link from Master-MS to Slave-MS
4 communicator->add_comm_link(
5     new CommLink(0 /*snd_id*/, 0 /*port*/,
6                  1 /*rcv_id*/, 0 /*port*/));
7
8 MS_Master* master = new MS_Master (0 /*id*/, "master", communicator);
9 MS_Slave*  slave  = new MS_Slave  (1       , "slave",  communicator);
```

***Abbildung 22**. Definition und Nutzung des Kommunikators.*

Die Verbindungen zwischen den Services können in vielen Anwendungsszenarien nach einer komplexeren Topologie erfolgen, als nach dem einfachen „point-to-point“-Versand zwischen einem Paar der Services – dem Sender und dem Empfänger. Daher muss der Kommunikator auch einen **kollektiven Datenaustausch** unterstützen, also „viele zu einem“ und „einer zu vielen“ Operationen, in Analogie zu den Kommunikationspatterns paralleler HPC-Anwendungen. Daher wurden zusätzliche Kommunikationsfunktionen für die Kommunikator-API eingeführt, die auch auf kollektiven Operationen, in Anlehnung an den MPI-Standard [W27], beruhen wie es in **Abbildung 23** zusammengestellt ist.

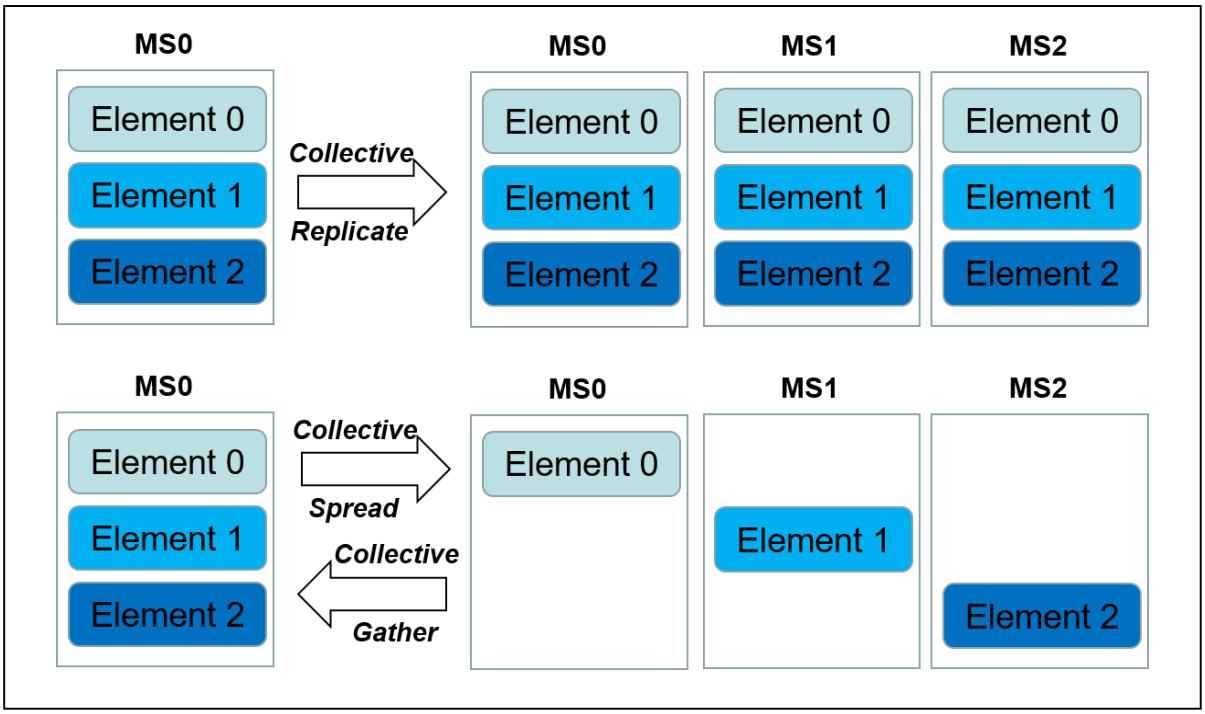


***Abbildung 23**. Kollektive Datenaustauschfunktionen des Kommunikators.*

Weitere Kommunikationspatterns können nach Bedarf zusätzlich eingeführt und mittels Kommunikator-API implementiert werden.

Komposition der Mikroservices

Die Mikroservices können nicht nur direkt in Mikroservice-Anwendungen eingebaut, sondern auch vielmehr zur Erstellung der übergeordneten Services (bspw. entsprechend der in **Abbildung 17** dargestellten Hierarchie) benutzt werden. Darüber hinaus folgen viele Mikroservicearchitekturen dem Ansatz, nach welchen die MS-Anwendung selbst als Service organisiert wird, welcher den Lebenszyklus der untergegliederten Services weiterhin steuert. Um die beiden Szenarien zu ermöglichen, wird in dieser Arbeit ein Verfahren vorgeschlagen, das die beiden Aktivitäten in einem universellen Vorgang vereinheitlichen kann. Und zwar sollen die Anwendungen als übergeordnete Services dargestellt werden, mit dem einzigen Unterschied, dass sie zusätzlich einen Eingangspunkt für das Ausführungsframework aufweisen (z.B. durch eine *main()*-Methode in der Service-Implementierungsklasse). Die Instanziierung der weiteren Services erfolgt dann vom übergeordneten Service. Die Anordnung der Services in einen übergeordneten Service bzw. Anwendung (weiterhin Master-Service genannt) erfolgt wie es bspw. in **Abbildung 24** dargestellt wird.

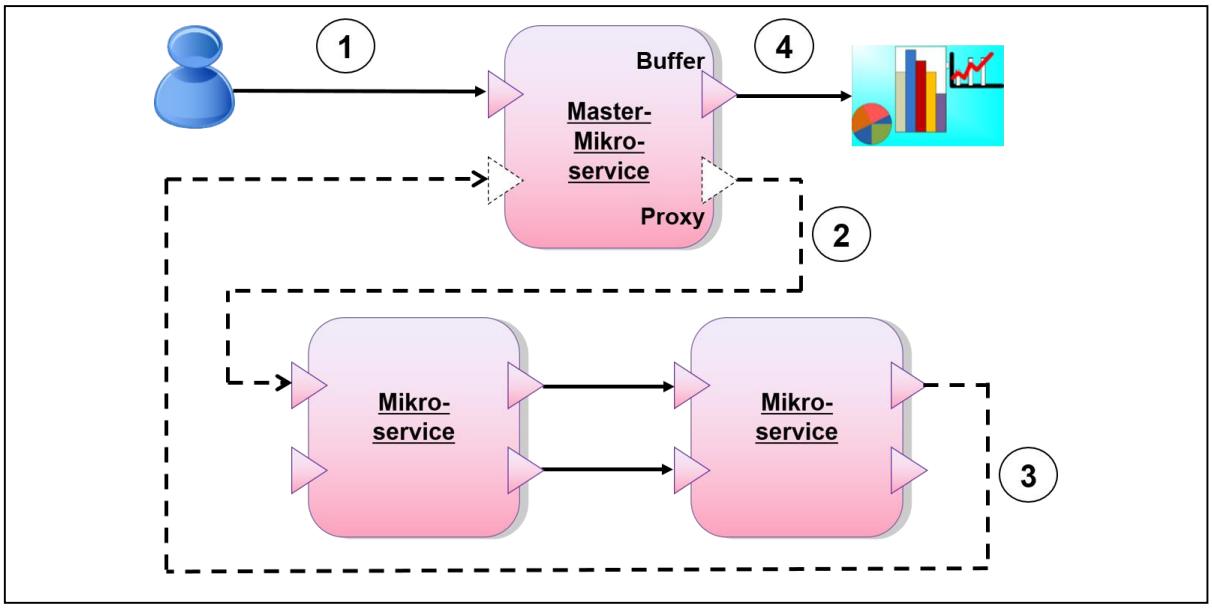


***Abbildung 24**. Organisation der Mikroservice-Anwendung.*

Zur beiderseitigen Versendung der Daten zwischen dem Master- und eingegliederten Services dienen eine spezielle Art der Buffers, die weiterhin als Proxy-Buffer (oder einfach – "**Proxy“**) bezeichnet werden. Einen Unterschied zwischen einem Proxy und einem Buffer gibt es im Sinne der technischen Umsetzung grundsätzlich nicht – das Proxy-Konzept dient lediglich der logischen Aufspaltung der Kommunikations-funktionalität in die horizontalen Richtungen (z.B. zwischen den Services einer Hierarchiestufe mithilfe von Buffers) und die vertikalen Ebenen (z.B. zwischen einem Master-Service und seinen gegliederten Services unterer Hierarchiestufe mithilfe von Proxys). Dies könnte für die Projektierung komplexer Anwendungen mit den Services, die über mehrere Hierarchieebenen hinweg organisiert sind, erforderlich sein. Die Proxys eines Service werden bei seiner Inkludierung in einen Master-Service nicht berücksichtigt: Wird ein zusammengesetzter Service selbst zum Teil einer Anwendung bzw. eines anderen Master-Services höherer Hierarchiestufe, bleiben die Details seiner Kommunikation mit gegliederten Services (durch die Proxies) von seinem Master-Service versteckt. So kann nicht nur der Entwicklungsprozess vereinfacht werden, sondern sich auch unnötige Fehler bei der Realisation komplexer Kommunikationspatterns zwischen mehreren ineinander geschachtelten Services vermeiden. Ist dieses Verhalten seitens des Entwicklers aber nicht gewünscht, so kann die benötigte Kommunikationsfunktionalität immer mithilfe von gewöhnlichen Buffers

implementiert werden, die immer global für die gesamte Anwendung bleiben – der übliche Fall in den meisten horizontal aufgebauten Mikroservice-Plattformen, wie bspw. Microsofts Azur. In allen sonstigen Fällen bietet das vorgeschlagene Verfahren eine vereinfachte Zusammensetzung der Services innerhalb von komplexen, hierarchisch aufgebauten Anwendungen an, insbesondere für vertikal organisierte Simulationsanwendungen.

Die in Kapitel 3.3 beschriebenen Kommunikationspatterns der Buffers können genauso für Proxys angewendet werden.

3.4 Ausführungsframework für Mikroservice-Anwendungen

Übersicht der Hauptkomponente

Zur Ausführung und Bereitstellung von Mikroservice-Anwendungen an die Nutzer ist eine einheitliche softwaretechnische Plattform (Ausführungsframework) erforderlich, die die Anwendungen durch ihren Lebenszyklus begleiten und die dazu gehörigen Steuerfunktionen übernehmen soll. Die praktikable Umsetzung dieser Funktionen erfordert vom Framework (wie in **Abbildung 25** skizziert) die Erfüllung folgender Aufgaben:

- Bereitstellung und Ausführung der Services auf verteilten Rechenressourcen mithilfe des *Deployment Manager*.
- Abwicklung der Kommunikationsfunktionen der Services in verteilten Systemen mithilfe der *Kommunikationsbibliothek*.
- Verwaltung der Infrastruktur mithilfe des *Ressource Manager*.

Darüber hinaus müssen vom Framework wichtige Begleitfunktionen erfüllen, wie beispielsweise:

- Monitoring des Status der auszuführenden Services mithilfe des *Monitoring-framework.*
- Datenspeicherung in einer entfernten Datenbank.

In den weiteren Kapiteln werden die Maßnahmen und Technologien diskutiert, die zur Erfüllung von diesen und sonstigen Aufgaben durch das Ausführungsframework notwendig sind.

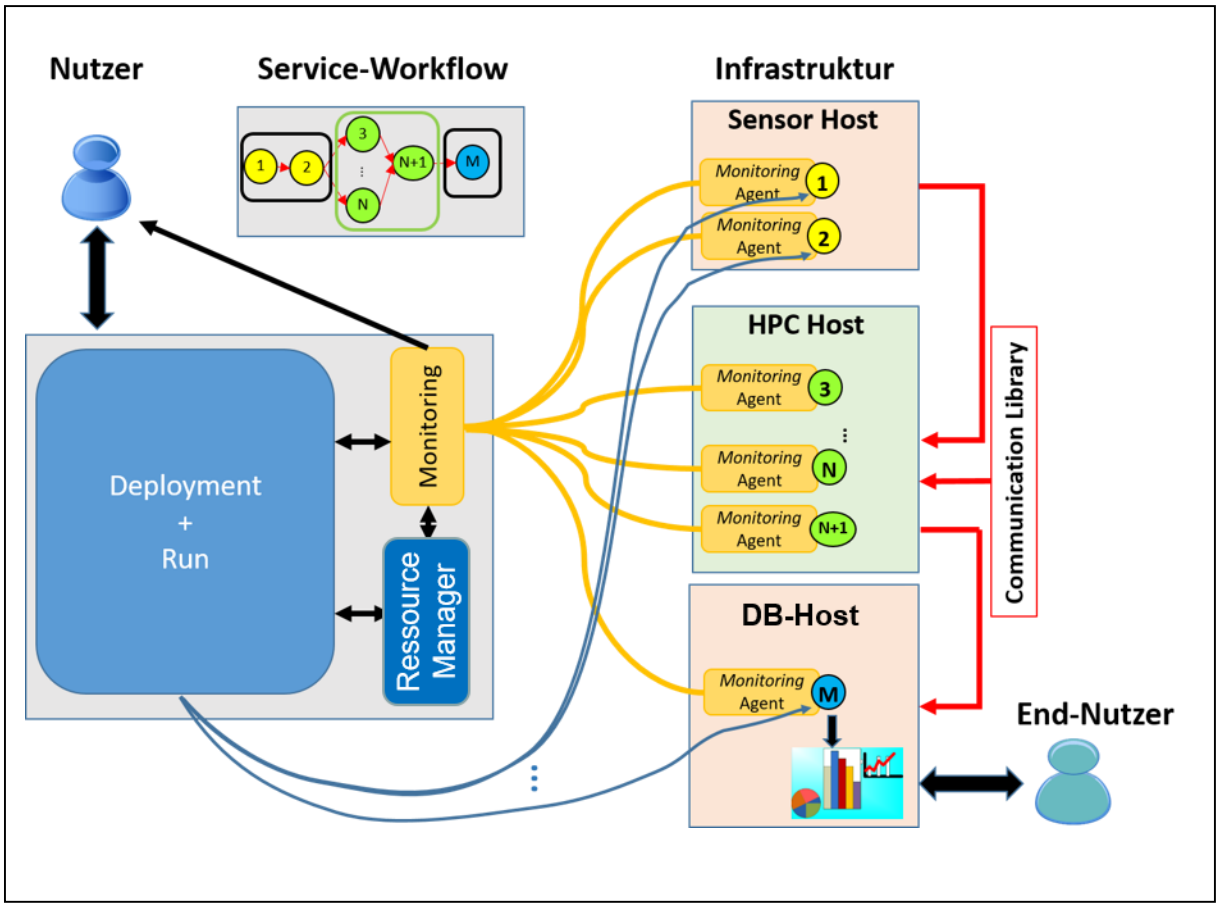


Abbildung 25. *Struktur und Komponente des Ausführungsframeworks.*

Deployment und Ausführung

Das Deployment (Bereitstellung und Ausführung auf der parallelen Infrastruktur) der Services hängt prinzipiell von der Strategie der Instanziierung der Software-anwendungen auf der vorhandenen Infrastruktur ab, die die Mikroservices implementieren. Das Deployment erfolgt mithilfe einer speziellen **Deployment-API**, die eine Instanziierung-Strategie für Services einer MS-Anwendung realisieren. Im Rahmen dieser Arbeit wurden zwei wichtige Instanziierungsstrategien ausgearbeitet: Multithreading und Multiprocessing. Weitere mögliche Strategien können die Ausführung auf Cloud-Infrastrukturen mittels Containers, HPC-Systemen mithilfe von Jobs und viele andere umfassen.

Die Multithreading-Deployment-Strategie wird mithilfe von Softwarethreads realisiert – einer Technologie zur gleichzeitigen Ausführung von mehreren Software-funktionen (Threads) auf parallel aufgebauten CPU-Kernen eines Prozessors. Threads bieten eine Technologie mit einem relativ geringen Aufwand für Kontextumschaltung auf der Seite

des Prozessors an. Ein weiterer Vorteil ist die Verfügbarkeit eines gemeinsamen Hauptspeicherbereiches für alle Threads, die innerhalb der ccNUMA[40]-Architektur ausgeführt werden. Für die Multithreading-basierte Ausführung bietet das Deployment-API zwei wichtige Komponente an: den **NumaCommunicator** und den **ThreadDeploymentPool**. Der *NumaCommunicator* ist eine spezielle Implementierung des Kommunikators für ccNUMA-Architekturen. Der Datenaustausch erfolgt durch einen direkten Kopiervorgang von den betroffenen Hauptspeicherbereichen der involvierten Buffers. Die Synchronisierung erfolgt dabei mittels der *muttex::lock*-Funktionen von einem C++ Compilerframework. Während des Deployments wird für jeden Mikroservice ein neuer asynchroner Thread angelegt, was im Deployment-API mithilfe eines *ThreadDeploymentPool* erfolgen kann, wie es z.B. im Listing der **Abbildung 26** aufgeführt wird.

```
1  Communicator* communicator = new NumaCommunicator();
2
3  // Link from Master-MS to Slave-MS
4  communicator->add_comm_link(
5      new CommLink(0 /*snd_id*/, 0 /*port*/,
6                   1 /*rcv_id*/, 0 /*port*/));
7
8  // Initialization of services
9  MS_Master* master = new MS_Master (0 /*id*/, "master", communicator);
10 MS_Slave*  slave  = new MS_Slave  (1       , "slave",  communicator);
11
12 // Deployment of services in multi-threading way
13 ThreadDeploymentPool deployment_pool;
14
15 deployment_pool.add_ms(master);
16 deployment_pool.add_ms(slave);
17
18 deployment_pool.deploy_all();
19 deployment_pool.join_all();
```

***Abbildung 26**. Ausführung der Services mittels Multithreading-Strategie.*

Alle Services, die nach dem Multithreading-Strategie-Verfahren aufgebaut sind, werden von einer gemeinsamen Binärdatei ausgeführt und erfordern keine zusätzliche Unterstützung seitens der Systemsoftware der Zielressource.

Im Gegenteil zu Multithreading, fällt die Ausführung auf verteilten Ressourcen, also – nach der Multiprocessing-Strategie, deutlich schwieriger aus: Erstens müssen alle

[40] ccNUMA – cache coherent Non-Uniform Memory Architecture

Datenaustauschvorgänge ausschließlich über das Netzwerk abgewickelt werden. Zweitens müssen die Services als eigenständige Softwareprozesse auf jeder Ressource gestartet werden. Zu den beiden Problemstellungen bietet die Systemsoftware eine ungenügende Unterstützung an, was den Einsatz externer Middleware erfordert. Im Rahmen dieser Arbeit wird mit der Ausführungs- (bzw. Runtime-) Umgebung von *OpenMPI* (wie von Gabriel et al. in [R99] präsentiert) experimentiert und gute Ergebnisse erzielt.

Zur Unterstützung der Nutzer bei der Ausführung der Mikroservices nach der Multiprocessing-Strategie (mit *OpenMPI* im Hintergrund) werden vom Ausführungsframework die folgenden Komponenten des Deployment-APIs angeboten: der **MpiCommunicator** und der **MpiDeploymentPool**. Die Spezifikation des MpiCommunikators erfolgt ähnlich wie im Fall des ThreadCommunicator, benötigt aber zusätzlich ein *Map* – eine Deploymentkonfiguration, die die Zuordnung der Mikroservices zu den parallelen Einheiten (MPI-Prozessen) bestimmt, wie im Listing der **Abbildung 27** aufgeführt ist. Die *OpenMPI*-Runtimeumgebung instanziiert dann die MPI-Prozesse auf den benötigten Ressourcen des parallelen Systems, z.B. mithilfe von Affinitätsparametern des *mpirun*-Kommandos, was zu der Aufgabestellung des Ressource Manager gehört. Darüber hinaus unterstützt *OpenMPI* die typischen HPC- (wie *Torque/PBS*) sowie Cloud- (wie *Amazon-AWS*) Schedulers, was den Deploymentvorgang vereinfacht.

Datenaustausch

Die Implementierung des Datenaustausches zwischen Mikroservices (wie im Kapitel 3.3 beschrieben) mittels des vorhandenen Netzwerks wird im Kommunikationsinterfacestapel einer externen Kommunikationsbibliothek eingesetzt, wie in **Abbildung 21** dargestellt. Die Bibliothek soll verschiedene physikalische Netzwerkinterfaces unterstützen, möglichst ohne Anpassung seitens des Nutzers. Unter den vorhandenen Kommunikationsbibliotheken (wie im Kapitel 2.3 beschrieben) wurde *OpenMPI* ausgewählt, weil sie die folgenden besonderen Eigenschaften aufweist:

- Modulare Organisation der Kommunikationsinterfaces mit Unterstützung verschiedener Netzwerkinterfaces, unterdessen TCP-basierten (z.B. Ethernet), OpenFabrics (z.B. Infiniband) und vieler anderen.
- Umfassender API-Stapel zur Implementierung der Kommunikationen mittels des MPI-Standards.

```
1  // Deployment options
2  MpiProcessMap mpi_map;
3  mpi_map.add({"master" /*id*/, 0 /*rank*/});
4  mpi_map.add({"slave",         1         });
5
6  // Communication options
7  Communicator* communicator = new MpiCommunicator(mpi_map);
8
9  // Link from Master-MS to Slave-MS
10 communicator->add_comm_link(
11     new CommLink("master" /*snd_id*/, 0 /*port*/,
12                  "slave"  /*rcv_id*/, 0 /*port*/));
13
14 // Initialization of services
15 MS_Master* master = new MS_Master (0 /*id*/, "master", communicator);
16 MS_Slave*  slave  = new MS_Slave  (1       , "slave",  communicator);
17
18 // Deployment of services in multi-processor way
19 MpiDeploymentPool deployment_pool(mpi_map);
20
21 deployment_pool.add_ms(master);
22 deployment_pool.add_ms(slave);
23
24 deployment_pool.deploy_all();
```

***Abbildung 27**. Ausführung der Services mittels Multiprocessing-Strategie.*

Die OpenMPI-Kommunikationsbibliothek wurde bei der Erstellung von **MpiCommunicator** zur Implementierung von *Flush*- und *Sync*-Funktionen der Buffers (siehe Kapitel 3.3) verwendet. Je nach der Zusammenstellung der Kommunikations-links zwischen den Ports verschiedener Mikroservices, kann der MpiCommunicator verschiedene Nachrichtenaustauschstrategien anwenden, wie z.B. Punkt-zu-Punkt- (*MPI_Send*, *MPI_Recv* u.a.) oder kollektive (*MPI_Reduce*, *MPI_Gather* u.a.) Kommunikationen.

Monitoring und Datenspreicherung

Das Monitoring ist eine wichtige Begleitfunktion der Ausführung von zusammen-gesetzten Mikroservice-Anwendungen, insbesondere in funktionell geordneten,

workflow-basierten Szenarien – es bietet die Möglichkeit an, die Werte verschiedener anwendungsbezogener Metriken zur Anwendungslaufzeit abzufragen, zu speichern und zu analysieren, mit dem Ziel, die aktuellen (laufenden) sowie die zukünftigen Ausführungen zu optimieren (im Sinne der Leistung, des Energieverbrauchs usw.). Im Rahmen dieser Arbeit wurde ein innovatives Verfahren ausgearbeitet, welches drei der wichtigsten Aspekte des Monitorings in einer Technologie vereint:

- Hardwareplattform- und Systemumgebung-Monitoring
- Anwendungssoftware-Monitoring
- Speicherung der anwendungsspezifischen Daten.

So basiert das Verfahren auf einer verteilten Monitoring-Architektur, die aus einer Client- und einer Server-Anwendung besteht (siehe **Abbildung 28**). Der Client läuft als asynchroner Softwareprozess im Hintergrund des Systemsoftwarestapels auf jeder verfügbaren Ressource des Ziel- verteilten Systems und sammelt wichtige Informationen von Hardwarezählern (wie *PAPI*, *RAPL* usw.) über die Leistungsaktivitäten sowie den Energieverbrauch (wie z.B. von Montanana in [R100] gezeigt.). Die Daten werden in einem vom Nutzer konfigurierten Zeitabstand abgefragt und lokal gespeichert (z.B. alle volle Millisekunde). Genau so kann das Intervall der Datenübermittlung an den Monitoring-Server konfiguriert werden. Der Monitoring-Server wird von einer verteilten Datenbank gehostet, die als zentrale Anlaufstelle für alle Monitoring-Daten dient. Die Datenbank wird nach dem NoSQL-Prinzip aufgebaut und verfügt über ein flexibles, frei zu wählendes Datenschema zur Speicherung und Abfragung der Daten.

Im Rahmen der vorliegenden Arbeit wurde die Funktionalität vom Montananas Monitoring-Clienten auf die Sammlung der anwendungsspezifischen Daten der Mikroservices erweitert (wie in **Abbildung 29** skizziert). Zu diesem Zweck wurde eine spezielle Monitoring-API entwickelt, die es einem Mikroservice ermöglicht, seine wichtigen Daten (z.B. den aktuellen Stand der Modellparameter für Simulationsalgorithmen) an den Server zur Speicherung und weiteren Analyse zu übermitteln. Das API bietet die Möglichkeit an, die Daten in Echtzeit zu erfassen – jeder

ermittelte Parameterwert wird mit einem aktuellen Zeitstempel versehen, der gemeinsam mit dem zu speichernden Parameter an den Server übermittelt wird.

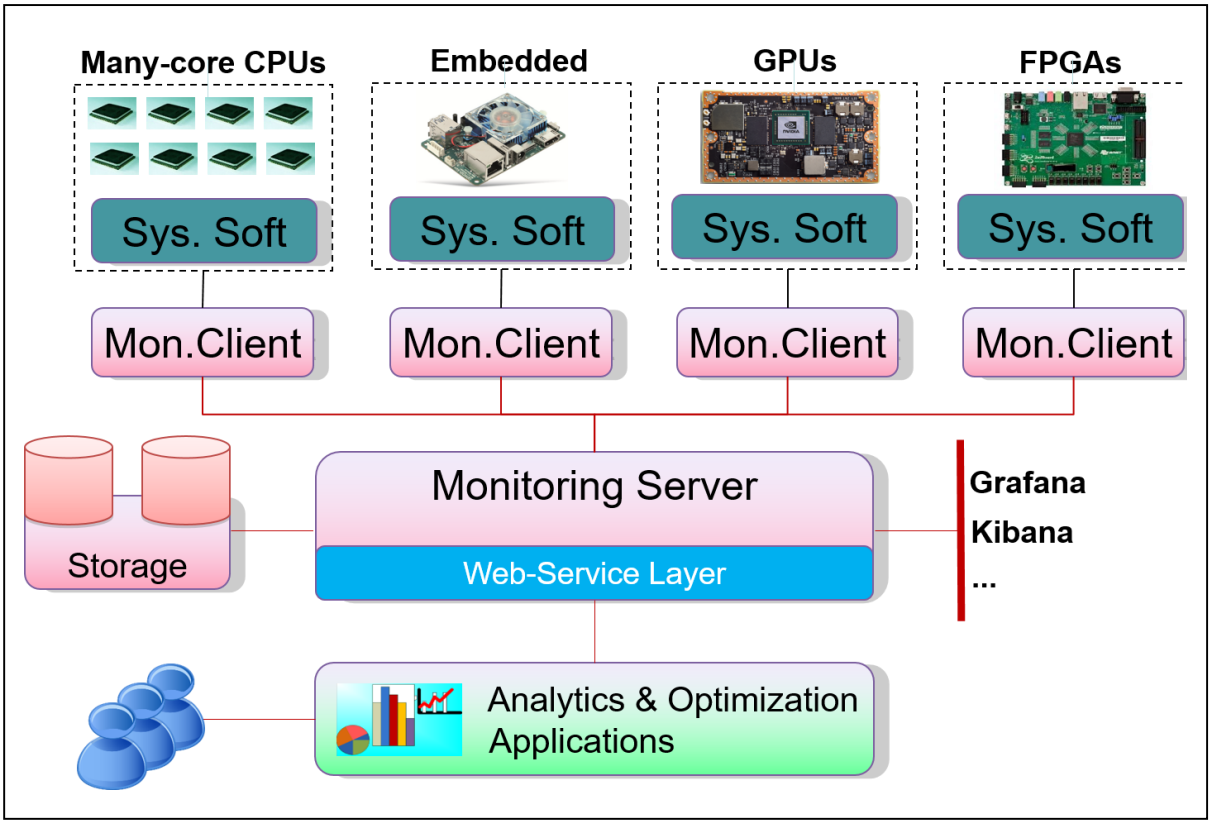


***Abbildung 28**. Architektur der verteilten Monitoring-Plattform.*

Um die Entwicklung der Mikroservices noch mehr zu vereinfachen, bietet die Monitoring-API eine mikroserviceübergreifende Funktionalität zur Speicherung der Ergebnisse an, was entweder auf dem Host-System (z.B. in einer Datei auf dem lokalen Filesystem) oder auf dem Fern-Monitoring-Server bzw. auch auf beiden Wegen erfolgen kann. Im Fall, dass die Speicherung auf dem Monitoring-Server ausgewählt ist, werden zur Übermittlung der Daten lokale Cache-Mechanismen verwendet, um die Netzwerkbelastung durch das Versenden der Monitoring-Daten zu minimieren.

Das vorgeschlagene kombinierte Verfahren für Monitoring und Datenerfassung wurde u.a. im EU-Projekt PHANTOM angewendet (siehe hierzu die relevante Veröffentlichung von Li et al. [R101]), um die Ausführung paralleler CFD-Anwendungen zu unterstützen.

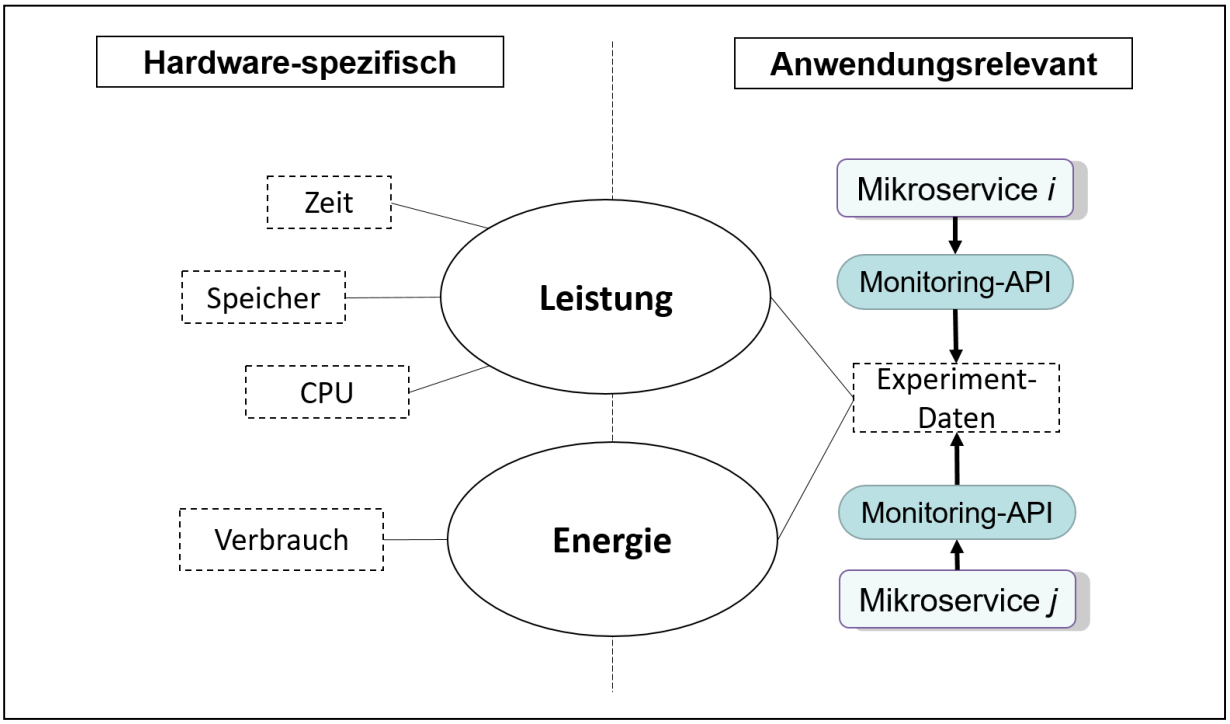


***Abbildung 29**. Erfassung der Mikroservice-Daten durch Monitoring-Plattform.*

3.5 Zusammenfassung der Ergebnisse

Das Ziel dieses Kapitelinhalts war die Schaffung einer methodologischen Grundlage zur Entwicklung der Mikroservices sowie der auf ihnen basierten Anwendungen. Es sollten dabei insbesondere die Aspekte der hierarchischen Organisation und der Umsetzung der zusammengesetzten Services berücksichtigt werden, wie sie von den Modellen des Basisgegenstandsgebiets gefordert ist.

Die wichtigsten erzielten Ergebnisse sind wie folgt:

- Es wurde ein **hierarchischer Ansatz** zur Erstellung einer Klassifikation der Modelle für komplexe, hierarchisch aufgebaute dynamische Systeme entwickelt. Der Ansatz wurde zu Grunde eines Vorgehensmodells gelegt, welches das Ziel hat, die systematische Erstellung einzelner sowie zusammengesetzter Simulationsservices mithilfe von Methoden der objektorientierten Modellierung und Servicekomposition zu ermöglichen.
- Zur Umsetzung der hierarchisch aufgebauten Modelle als abgekoppelter Mikroservices sowie zur Entwicklung kompletter Softwareanwendungen auf ihrer Basis wurde ein **service-orientiertes Programmiermodell** elaboriert,

welches auf dem MIMD-Ansatz beruht und nützliche Schnittstellen zur Erstellung der Softwarekomponente bietet, die die Vorteile einer Service-Interoperabilität und einer Plattformunabhängigkeit optimal ausschöpfen.

- Zur Konsolidierung der Services innerhalb der dynamisch aufgebauten Anwendungen wurde ein **indirektes Kommunikationsverfahren** ausgearbeitet, das die Vorteile des vereinfachten Erstellens paralleler Software für verschiedene parallele Architekturen (von NUMA-Prozessoren bis hin zu verteilten cluster-basierten Systemen) anbietet.
- Das indirekte Kommunikationsverfahren wurde prototypenmäßig in einer **Kommunikationsbibliothek** implementiert, die auf *OpenMPI* basiert und sowohl einzelnen als auch kollektiven Datenaustausch mit genügender Effizienz im Vergleich zu spezialisierten Lösungen (die viel Aufwand und spezialisiertes Fachpersonal benötigen) ermöglicht.
- Das Konzept des **Monitorings in verteilten Systemen** wurde auf Mikroservices-Architekturen ausgedehnt und eine Erweiterung zwecks Speicherung der anwendungsspezifischen Daten vorgeschlagen, mit dem Zweck, den Implementierungsaufwand der Datenverwaltung in verteilten Mikroservices zu reduzieren.
- Das **Referenzframework**, das die grundlegenden Konzepte der ausgearbeiteten Schnittstellen für mikroservice-basierte Anwendungen in der C++-Programmiersprache implementiert, steht frei zugänglich auf dem Github-Repository [W32] zur Verfügung. Es soll in der industriellen Praxis zur Reduzierung der Entwicklungszeit sowie der Kosten beitragen.

4 Erstellung und Umsetzung der Bewetterung-Modelle

In diesem Kapitel werden die Grundlagen, Vorgänge und Ergebnisse der Realisierung von CFD-Modellen der Grubenbewetterung nach dem erarbeiteten (siehe Kapitel 3) Mikroservice-Ansatz präsentiert.

Der weitere Inhalt bietet zuerst eine detaillierte Analyse der grundlegenden CFD-Modelle der Grubenbewetterung (siehe Kapitel 4.1). Eine Klassifikation der Modellierungsobjekte wird mithilfe des hierarchischen Ansatzes im Kapitel 4.2 erarbeitet. Die theoretischen und praktischen Aspekte der Entwicklung der Simulationsservices für die Hauptobjekte der Modellhierarchie werden im Kapitel 4.3 erläutert. Die wichtigsten Aspekte der Simulationseinsetzung in Echtzeit-anwendungsszenarien werden im Kapitel 4.4 diskutiert. Das abschließende Kapitel 4.5 befasst sich mit der Problematik der Simulationsdatenspeicherung und der Analyse.

4.1 Grundlegende mathematische Modelle

Die Analyse von Aero- und Gasdynamikprozessen in Elementen eines Gruben-bewetterungsnetzes erfolgt mittels der Modelle der numerischen Strömungs-mechanik, welche die wichtigsten dynamischen Eigenschaften eines Strömungs-prozesses (wie z.B. die Erhaltungs- und Transportgesetze für Energie, Masse und Impuls) mit einer hohen Präzision abbilden lassen. Im Gegensatz zu vielen anderen Bereichen der Technik, wo die numerische Strömungsmechanik „nur“ als eine kostengünstige Hilfsalternative zur Naturmodellierung (z.B. Versuche in Wind- oder Wasserkanälen) verwendet wird, ist die Modellierung (natürlich neben sensor-gesteuerter Überwachung) die einzige Erkenntnisquelle über den Verlauf der dynamischen Prozesse im Untertagebergbau und damit ein unverzichtbares Werkzeug für die Kohleindustrie.

In den folgenden Unterkapiteln werden die grundlegenden aero- und gasdynamischen Modelle der Grubenbewetterung erläutert.

CFD-Modelle der Aerodynamik

Die klassische und gleichzeitig als umfassendste (im Sinne aller relevanten physikalischen Effekte) geltende Modellierungsmethode der Strömungsmechanik ist das Navier-Stockes-Verfahren (wie z.B. von Chorin in [R102] beschrieben). Hierbei handelt es sich um eine Beschreibung der Charakteristiken einer Strömung, die durch Erhaltungsgesetze von Masse, Impuls und Energie (für kompressible Fluide) an einem makroskopisch-kleinen Volumenelement des Strömungsgebiets abgebildet wird. Die Navier-Stokes-Gleichungen ermöglichen die Bestimmung der Strömungs-Geschwindigkeit, der Dichte und des physikalischen Drucks an jedem Punkt des analysierten Strömungsraums. Unter Annahme der Abwesenheit einer Turbulenz (aufgrund einer geringen Geschwindigkeit des Stromflusses), kann das Basismodell der GBN-Aero-dynamik in Form eines Systems nichtlinearer partiellen Differentialgleichungen für die Erhaltung der Masse (auch als Kontinuitätsgleichung bekannt):

$$\nabla \cdot \vec{\upsilon} = 0, \tag{IV.1}$$

und für Impuls:

$$\frac{\partial \overline{\upsilon}}{\partial t} + \overline{\upsilon} \cdot (\nabla \overline{\upsilon}) = \overline{F} - \frac{1}{\rho} \nabla p + \nu \Delta \overline{\upsilon} \tag{IV.2}$$

dargestellt werden, wobei p – der physikalische Druck, $\overline{\upsilon}$ – der Vektor der Strömungsgeschwindigkeit, ρ – die Dichte, $\overline{F}$ – der Vektor der Massenkräfte, ν – die kinematische Viskosität, $\nabla = \bar{i}\frac{\partial}{\partial x} + \bar{j}\frac{\partial}{\partial y} + \bar{k}\frac{\partial}{\partial z}$ (Nabla-Operator), $\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}$ (Laplace-Operator) sind.

Unter Einbeziehung der Identitätsgleichung

$$\nabla \cdot (\overline{\upsilon}\overline{\upsilon}) = (\nabla \cdot \overline{\upsilon}\upsilon)\overline{\upsilon} + \overline{\upsilon} \times (\nabla \cdot \overline{\upsilon}), \tag{IV.4}$$

und Berücksichtigung der Kontinuitätsgleichung (IV.1), wird die Impulsgleichung (IV.2) in Form der folgenden Transportgleichung dargestellt werden:

$$\frac{\partial \overline{\upsilon}}{\partial t} + \nabla \cdot \left(\overline{\upsilon}\,\overline{\upsilon}\right) = \overline{F} - \frac{1}{\rho}\nabla p + \nu \Delta \overline{\upsilon} \,, \qquad \text{(IV.5)}$$

Eine Methodik zur praktischen Anwendung der Transportgleichung (IV.5) auf die Modellierung von komplexen Strömungsverhältnissen in der Untertagbewetterung ohne Nachbildung von komplexen geometrischen Strukturen wurde von Feldmann (zusammen mit Svjatnjy und Abramov, wie in [R15], [R103] berichtet), mit Anwendung auf die Problematik von Kohlebergwerken in der Region Donbass[41] erarbeitet. Nach dieser Methodik, wird der klassische Ansatz, wie z.B. die Projektion des Strömungsvektors $\overline{\upsilon}$ auf die entlang der Strecke verlaufende Koordinatenachse ψ und Bestimmung seines Mittelwertes $\upsilon_S(\psi, t)$ in einem Querschnitt S des Strömungsgebietes (also – einer Strecke), der Anwendung des Gauss-Ostrogradski-Divergenzsatzes (Zusammenhang zwischen der Divergenz des Vektorfeldes in einem Volumen und der Strömung durch eine geschlossene Oberfläche dieses Volumens), die Approximation der räumliche Dimensionalität und anderen Techniken, mit den grundlegenden Gesetzen der Kontinuums-, Fluid- und Gasmechanik sowie mit praktischen Beobachtungen und Validierungen in Betriebsanlagen kombiniert, um die dynamischen Prozesse möglichst genau und nah an den genauesten Navier-Stokes-Satz abzubilden.

Nach der Anwendung der Feldmann-Methodik zur Darstellung der Viskositätskräfte:

$$\nu \int_V \Delta \upsilon_\psi \, d\tau = \frac{\lambda S \upsilon_S^2}{r_h} d\psi \,, \qquad \text{(IV.6)}$$

wobei υ_ψ – Strömungsvektor-Komponente längs der Strecke-Achse ψ, υ_S – Strömungsmittelwert im Strecke-Querschnitt *S*, ρ – die Dichte, λ – Widerstandswert, r_h – hydraulischer Radius des Strömungskanals (Strecke), ξ – Strömungsgeschwindigkeitsverteilungsfaktor durch Querschnitt S sind,

transformiert sich die Gleichung (IV.5) in die folgende Form:

[41] Der Kohlebecken in der Ost-Ukraine mit dem größten Steinkohlevorkommen

$$\frac{\partial \upsilon_S}{\partial t} + (1+\xi)\frac{\partial(\upsilon_S^2)}{\partial \psi} = F_\psi - \frac{1}{\rho}\frac{\partial p}{\partial \psi} - \frac{\lambda \upsilon_S^2}{r_h}. \qquad \text{(IV.7)}$$

Hier sind p – der Druck, ρ – die Dichte, ξ – Strömungsgeschwindigkeitsverteilungsfaktor entlang des Querschnitts S, F_ψ – Projektion des Massenkraftvektors *F* auf die Strecke-Achse.

Die Einführung des Luftmitteldurchsatzes $Q = S\upsilon_S$ und des spezifischen aerodynamischen Widerstands $r = \frac{\lambda\rho}{S^2 r_h}$ ergibt das folgende daraus resultierende Modell der inkompressiblen Aerodynamik im Element des Bewetterungsnetzes:

$$\frac{\partial Q}{\partial t} + \frac{(1+\xi)}{S}\frac{\partial(Q^2)}{\partial \psi} = F_\psi S - \frac{S}{\rho}\frac{\partial p}{\partial \psi} - \frac{S}{\rho} r Q^2. \qquad \text{(IV.8)}$$

Die entsprechende Kontinuitätsgleichung wird unter Definition des Druck-Dichteverhältnisses in der Luft als $a^2 = \frac{dp}{d\rho}$ (wobei a – die Schallgeschwindigkeit ist) in der folgenden Form dargestellt:

$$-\frac{\partial p}{\partial t} = \frac{pa^2}{S}\frac{\partial Q}{\partial \psi}. \qquad \text{(IV.9)}$$

Die Gleichungen (IV.8), (IV.9) bieten das Basismodell der inkompressiblen Strömung in einer allgemeinen GBN-Strecke. Sollte im Element eine Luftleckstrom durch den Filtrationsraum berücksichtigt werden (siehe die Gasdynamik-Beschreibung im Unterkapitel 4.1.2), wird dies durch die folgende Anpassung der Kontinuitätsgleichung erreicht:

$$-\frac{\partial p}{\partial t} = \frac{pa^2}{S}\frac{\partial Q}{\partial \psi} + \frac{pa^2}{S} q, \qquad \text{(IV.10)}$$

wobei q – der Durchsatz des Luftleckstroms (bzw. ihres angrenzenden approximierenden Elements) im Filtrationsraum ist.

Die lokalen Regulatoren in einem Luftführungsweg werden durch Einführen der Regelungswiderstände r' in der Transportgleichung (IV.8) berücksichtigt:

$$\frac{\partial Q}{\partial t} + \frac{(1+\xi)}{S}\frac{\partial (Q^2)}{\partial \psi} = F_{\psi} - \frac{S}{\rho}\frac{\partial p}{\partial \psi} - rQ^2 - r'Q^2 . \qquad \text{(IV.11)}$$

Im Falle einer schartigen Änderung im Ventilatoren-Betrieb oder beim Auftreten verschiedener Störfallsituationen in Elementen des gesamten Netzes, wobei die Luftverteilung maßgeblich gestört werden könnte (z.B. in einer Notfallsituation), kann die Strömung in einen inkompressiblen Zustand wechseln. Wie die Untersuchung von Svjatnyj in [R15] aber gezeigt hatte, ändert sich das Basismodell der Luftwegströmung (IV.8), (IV.9) bzw. (IV.8), (IV.10) nicht maßgeblich und kann weiterhin angewandt werden.

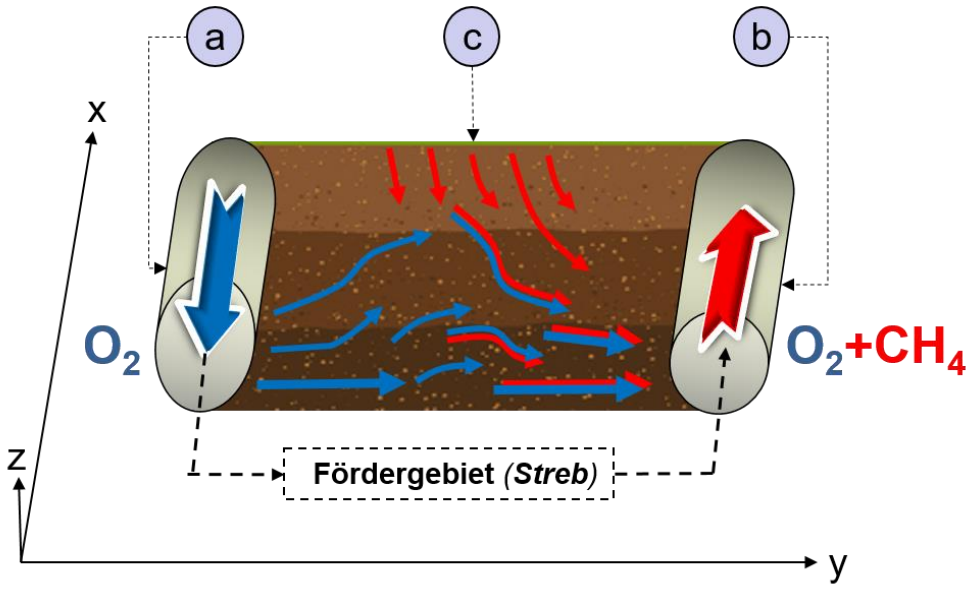
Als Randbedingungen für die Basis- aerodynamischer Modelle gelten die nichtlinearen Eigenschaften der Ventilatoren und der durch sie erzeugten Drücken sowie die Regelungswiderstände. Die Anfangsbedingungen werden von den initialen Luftstromwerten geformt.

CFD-Modelle der Gasdynamik

Die gasdynamischen Modelle umfassen die folgenden dynamischen Prozesse:

- Freilassung von Methan aus dynamischen Austrittsquellen,
- Vermischung von Methan mit dem Luftstrom und
- Verteilung (Transport) der Gas-Luft-Gemische in Strecken und Luftwegen eines Bewetterungsnetzes.

Die wichtigsten Methanergiebigkeitsquellen im Grubenbewetterungsnetz sind Streb (Kohlefördergebiet) und Hohlraum (Filtrationsraum zwischen den an Streb angrenzenden Luftführungsstrecken), wie in **Abbildung 30** gezeigt wird.



(a) Luft-Zufuhrstrecke, (b) Luft-Abfuhrstrecke, (c) Filtrations- (Hohl-) Raum (im Bergbaulexikon auch als „Alter Mann“ bekannt)

***Abbildung 30**. Schema der Methanbildung in einer Ventilationsabteilung.*

Die Methanbildung im Streb ist ein äußerst komplexer dynamischer Prozess, der durch die Beeinflussung der tiefliegenden Kohleschichten von einer zunehmend komplexer werdenden Maschinerie, die bei Kohleabbauaktivitäten eingesetzt wird (siehe Stewart et al. [R104]), zustande kommt und von mehreren Faktoren, meistens geologischer Natur, abhängt. Durch eine Unbeständigkeit der Gasauftritte während der Abbauprozesse ist grundsätzlich keine umfangreiche prädiktive Analyse durch Modellierung möglich. Deswegen sind die Modelle der Gasbildung im Streb auf den Integrationsansatz mit Sensordaten angewiesen, wie z.B. von Stewart et al. in [R34] auf der Basis von *VENTSIM*-Software oder von Agasty in [R61] für einen allgemeinen Löser (*Ansys-CFX*) geschildert. Dabei werden die Methankonzentrationen von Sensoren erfasst und als Randbedingungen an die Transport-Modelle übergeben. Die Sensoren werden z.B. am Anfang der an den Streb grenzenden Luftabfuhrstrecke installiert und liefern die Informationen über die Gasaustrittmenge an die Simulatoren (z.B. durch eine CPS-Schnittstelle).

Die zweitbedeutendste Quelle des Methanaustritts in einem Bewetterungsnetz ist der Hohlraum – der mit Bruchstein und Kohlelagerungsresten gefüllte Ort, welcher nach dem fortschreitenden Kohleabbauprozess, also hinter dem Streb, entsteht und eine große Menge an hochkonzentriertem Methan beinhaltet. Die Methanergiebigkeit aus dem Hohlraum entsteht durch Diffusionsprozesse zwischen den Luftzufuhr- und Abfuhrstrecken (siehe **Abbildung 30**), die durch den Hohlraum (der aus diesem Grund in der Bergbauaerologie auch als Filtrationsraum genannt wird) verlaufen. Dabei bleibt der Methanergiebigkeitswert durchaus konstant bei einem unveränderten Bewetterungsbetrieb (also konstanten Luftstrom in angrenzenden Strecken). Ein konstantes Verhalten der Methanergiebigkeit aus dem Filtrationsraum in die Abfuhrstrecke wird aber bei jeglicher Änderung im Bewetterungsbetrieb unterbrochen – bei sinkender Geschwindigkeit des Luftstroms durch den Filtrationsraum wird auch die Aufschlämmungsmasse des Methans geringer, was eine zusätzliche Ausschüttung des Methans in der gleichen Menge wie die Aufschlämmungsmassendifferenz zur Folge hat. Die Höhe des Methanausschlages hängt grundsätzlich vom Basisergiebigkeitswert des Methans und der Sprunggröße der Geschwindigkeit von Filtrationsflüssen durch den Hohlraum ab. Die Transport-gleichung für die Methanmassenbilanz kann in der folgenden allgemeinen Form dargestellt werden:

$$G = \int_{\sigma} \rho C \upsilon_N d\sigma \,, \tag{IV.12}$$

wobei G – der mit der Leckströmung (durch Filtrationsraum) herausgetragene Methanmassendurchsatz, σ – die Oberfläche von dem Element des Filtrationsraums, durch welches Methan in die Abflussströmung gelangt, υ_N – die Projektion der Methanfiltrationsgeschwindigkeit auf die Oberflächennormale von σ, ρ – die Dichte des Luft-Gas-Gemisches im Element der Leckströmung, C – die Methankonzentration ist.

Die entsprechende Kontinuitätsgleichung nimmt dann die folgende Form an:

$$\int_{V} m \frac{\partial(\rho C)}{\partial t} dV = G_0 - G \,, \tag{IV.12}$$

wobei G_0 – der Methanmassenstrom im Volumen des Filtrationsraums, m – der Porositätsfaktor der oberen Schicht des Filtrationsraums, V – der Fassungsraum des Filtrationsmediums ist.

Unter der Annahme von i) der Indifferenz der Leck- und Methanstrom-geschwindigkeit und ii) der Indifferenz der Methandichte im Übergangs- und Endzustand, kann die Abhängigkeit der Methanausschüttungsmenge Q_m (der Mittelwert über die Länge des Hohlraums) von ihrem Basiswert Q_{m0} und der Luftleck-Strommenge q (wie von Svjatnyj in [R8] erarbeitet) in einem Element des Filtrationsraums in der folgenden Form definiert werden:

$$\frac{mV}{q}\frac{dQ_m}{dt} + Q_m = Q_{m0} + \frac{mV}{q^2}Q_m\frac{dq}{dt}\,. \qquad \text{(IV.13)}$$

Das Modell (IV.13) wird weiterhin als Basismodell der Gasdynamik im Hohlraum genutzt. Die Randbedingungen sind die Basiswerte der Methanergiebigkeit und der Luftleckstrom im Hohlraum.

Zur Analyse der Ausbreitung von aus den oben genannten Quellen einströmenden Methanmengen durch die Elemente des Ventilationsnetzes wird in dieser Arbeit ein Ansatz erarbeitet, nach welchem die Transportgleichung in der allgemeinen Form (IV.8) auf die Mehrphasenströmung (Luft-Gas-Gemisch) angewandt werden soll. Im Gegenteil zum Svajtnjys Ansatz von [R8], der auf der Berechnung der Mittelkonzentrationswerten entlang der Elemente basiert, ermöglicht die vorgeschlagene Alternative eine bessere Berücksichtigung der physikalischen Bedingungen und erfordert einen potenziell geringeren Synchronisierungsaufwand zwischen den komponentenbasierten Implementierungen der Modelle der involvierten Elemente (wie Streb, Hohlraum, Abfuhrstrecke). Dabei kann auf die Transportgleichung (IV.8) das Euler-Verfahren angewendet werden, um die einzelnen Bestandteile des Gemisches (also – der Luft und des Methans) anteilig pro Volumenelement zu berechnen. Dabei wird jede Phase als ein separates Kontinuum betrachtet. Dem Ansatz wird eine Gleichheit der Strömungsgeschwindigkeit von Luft und Gas vorausgesetzt.

Numerische Lösungsverfahren

Die Basismodelle der Aero- bzw. Gasdynamikprozesse in verschiedenen Elementen des Bewetterungsnetzes sind von nichtlinearen (parabolischen) Differentialgleichungen (IV.8), (IV.9), (IV.10) und (IV.13) bestimmt, für welche leider noch keine analytische Lösung bekannt ist. Um diese Gleichungssysteme rechnerisch zu lösen, ist die Anwendung eines numerischen Verfahrens erforderlich. Aufgrund der Spezifik von Bewetterungsobjekten (wie z.B. starke horizontale Ausdehnung der Strecken, dominierende Rolle der Länge über die Höhe und Breite und weitere) werden in der Praxis die Basisgleichungen einer Approximation der Ortsableitungen mithilfe der Finiten Differenzen Methode (**FDM**) und, anschließend, eine Diskretisierung der Zeitvariable (z.B. mittels der Linienmethode **NMOL**) angewandt.

Durch den Einsatz der oben genannten numerischen Methoden und der Anwendung weiterer Vereinfachungen (wie z.B. Vernachlässigung der Massenkräfte, wie z.B. von Abramov et al. in [R15] gezeigt) entsteht ein System gewöhnlicher Differentialgleichungen (wie in **Abbildung 31** illustriert) für ein approximatives Element (Transportgleichung) für Luft:

$$\frac{\partial Q_j}{\partial t} = -\frac{S_j}{\rho}\frac{(P_i - P_{i-1})}{\Delta\psi} - rQ_j^2 - r'Q_j^2, \qquad \text{(IV.14)}$$

Gas:

$$\frac{mV}{q}\frac{dQ_{m_j}}{dt} + Q_{m_j} = Q_{m0_j} + \frac{mV}{Q_j^2}Q_m \cdot \left(-\frac{S_j}{\rho}\frac{(P_i - P_{i-1})}{\Delta\psi} - rQ_j^2\right), \qquad \text{(IV.15)}$$

sowie einen approximativen Knoten (Kontinuitätsgleichung):

$$\frac{\partial P_i}{\partial t} = -\frac{pa^2}{S_i}\frac{(Q_{j+1} - Q_j)}{\Delta\psi}. \qquad \text{(IV.16)}$$

Hierbei ist $\Delta\psi$ – die Länge des Approximationselements.

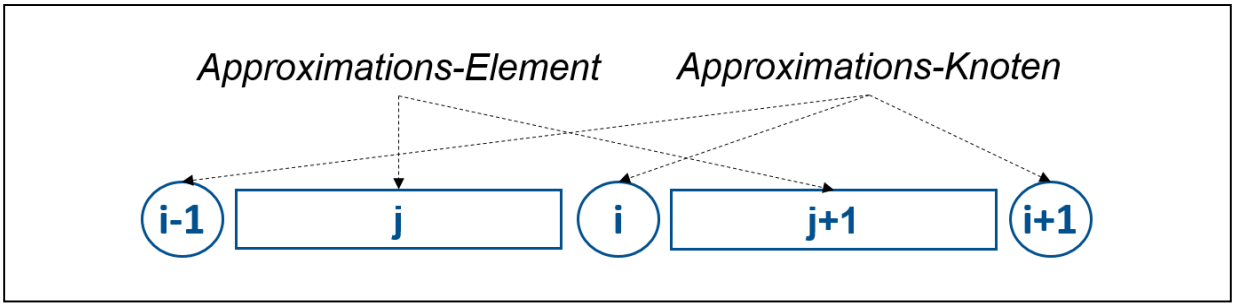


***Abbildung 31**. Schema der Diskretisierung einer Strecke hinsichtlich der räumlichen Längsausdehnung.*

Das System (IV.14)-(IV.16) bildet die Grundlage zu der in dieser Arbeit vorgenommenen Implementierung der Modellhierarchie für Bewetterungsprobleme, wie sie im nächsten Kapitel 4.2 beschrieben ist.

Die Lösung der daraus resultierenden Gleichungssysteme erfolgt durch die Anwendung eines Ein- oder Mehrschnittverfahren (z.B. Euler- oder RK4-Verfahren). Dabei ist eine Umformung der Systeme in die Form eines Anfangswertproblems erster Ordnung erforderlich.

Die Randbedingungen sind dabei die Drücke an äußeren approximativen Knoten (die durch Ventilator-Depressionen bzw. Oberflächenluftdruck definiert werden), die Widerstände der Regelelemente in approximativen Elementen und die Methankonzentration-Sensorangaben (für Gas-Modelle).

4.2 Klassifikation der Objekte und Hierarchie der Modelle

Zur Erstellung der Modelle wird der hierarchische Ansatz verwendet, wie er im Kapitel 3.1 ausgearbeitet wurde. Dabei werden die Hauptelemente der Bewetterung (siehe Erklärung der Hauptbegrifflichkeit in Kapitel 1.4, insb. **Abbildung 6a**) in drei Hierarchieebenen eingeteilt:

- Basiselemente – umfasst einen **Streb** (den Förderort), Wetterwege zum und vom Streb (horizontale zu- und abführende **Strecken** und vertikale **Schächte**) sowie einen **Hohlraum** (Filtrationsraum zwischen den Strecken).
- Zusammengesetze Elemente, inkl.:

- **Bewetterungsabteilungen** – eine Bewetterungseinheit, die aus einem Streb, den dazu gehörigen Strecken und einem Hohlraum besteht.

  Je nach Spezifik der bergbautechnischen Bedingungen des Förderbetriebs sowie in Abhängigkeit von der verwendeten Abbaustrategie (z.B. schwebender Abbau, abwärts geführter Abbau usw.), umfasst die Klassifikation von Bewetterungsabteilungen ein breites Spektrum verschiedenen Aufbauvarianten. Svjatnyj z.B. definiert in seinem Werk [R8] insgesamt 144 verschiedene Typen der Bewetterungsabteilungen für die charakteristischen Bergbauwerke des Donbass-Kohlebeckens.
- **Bewetterungsnetz** – kann eine oder mehrere Abteilungen, Hilfs-Strecken sowie Schächte beinhalten.

Die Menge aller Bewetterungselemente *O(i,j)* (wobei *i* – die Hierarchieebene, *j* – ein Objekt-Identifikator) kann wie in **Abbildung 32** dargestellt werden. Dabei werden die funktionellen Zusammenhänge durch gewichtete Verbindungen zwischen den Objekten aufgeführt.

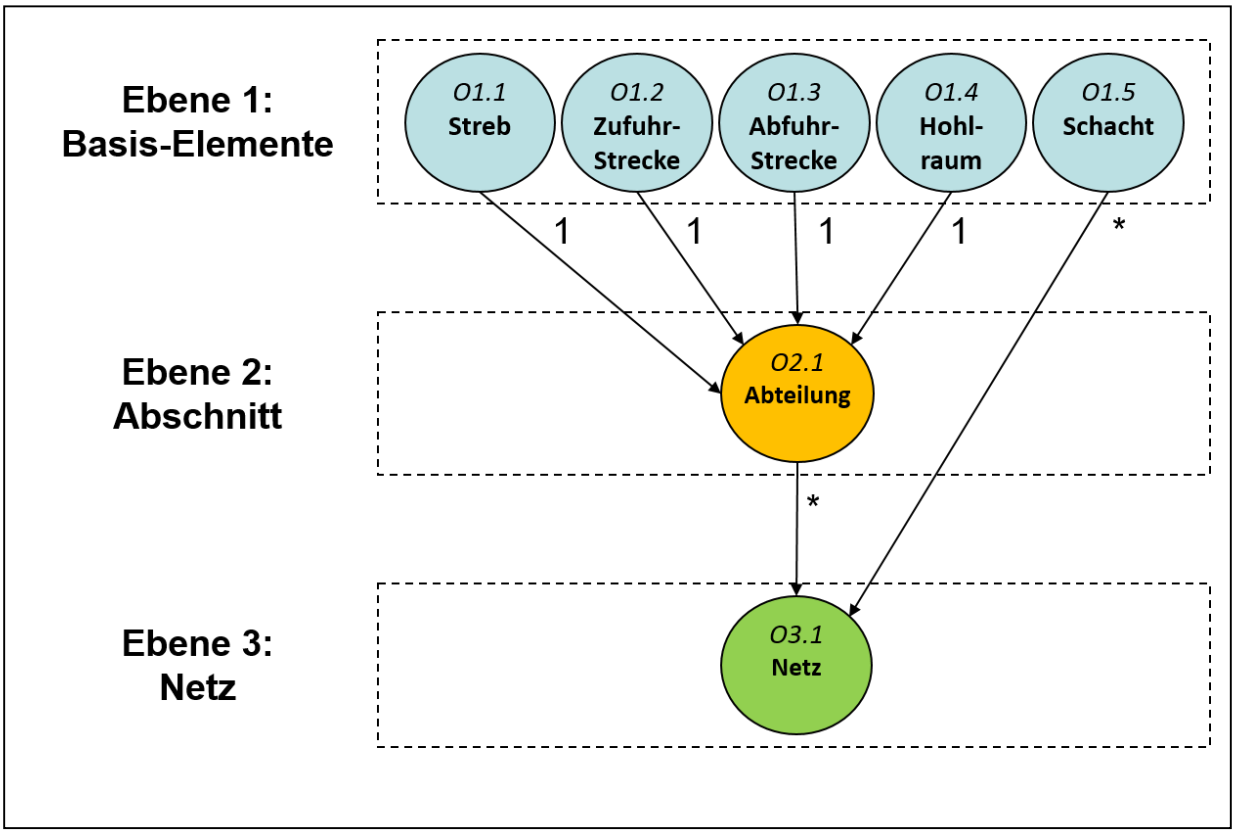


***Abbildung 32**. Klassifikation der Bewetterungsobjekte.*

Die Anwendung der Methodik der Servicekomposition innerhalb des erarbeiteten hierarchischen Ansatzes (siehe Kapitel 3.1) ermöglicht eine flexible Erstellung von zusammensetzten Modellen (z.B. der Bewetterungsabteilungen und des Netzes) durch Kombinieren und Verknüpfung von elementaren Modellen (z.B. von Streb, Strecken, Schächte). Dabei können in zusammengesetzten Modellen alle betriebsspezifischen Aspekte wie z.B. bergbautechnische Bedingungen berücksichtigt werden.

Die Menge aller charakteristischen Modelle *M(i,j)* der entsprechenden Objekte *O(i,j)* wird aus Modellen aufgebaut, die in zwei grundlegende hierarchische Modellbereiche gegliedert sind: Aerodynamik (mit $A(i, j) \subseteq M$ Modellen) und Gasdynamik (mit $G(i, j) \subset M$ Modellen). Diese Aufteilung ist insbesondere bei der Implementierung von *G*-Modellen sinnvoll, da diese meistens durch die Erweiterung entsprechender *A*-Modellen aufgebaut werden können: Die Letzteren fließen meistens in Form der Randbedingungen oder funktioneller Abhängigkeiten in die *G*-Modelle hinein.

Die Hierarchiestruktur der Modelle entspricht der der Klassifikation der Objekte (siehe **Abbildung 32**), wird aber durch das Hinzufügen zweier zusätzlichen Hilfsebenen erweitert: Diskrete Elemente (beschreiben Prozesse feinster Granularität der Diskretisierung) und Regelungselemente (berücksichtigen regelungstechnische Einheiten in Elementen). Die daraus resultierende Modellhierarchie für Bewetterungsobjekte ist in **Abbildung 33** dargestellt.

Die Erstellung der Services, die die hierarchisch zugeordneten Modelle implementieren, sollte womöglich mithilfe von folgenden Konzepten erfolgen:

- **Erweiterung** – Implementierung neuer Services mittels Vererbung der Service-Funktionalität. Diese Technik erfordert die volle Kompatibilität der Interfaces von Basis- und erbenden Services und ist aus diesem Grund nur auf homogene (mit einheitlicher Funktionalität, Ausführungslogik und Kommunikationsstruktur) Services anwendbar. Die Technik sollte insbesondere zur Erstellung der Services innerhalb einer Hierarchieebene und eines Tätigkeitsbereiches benutzt werden.

- **Injektion** – Implementierung neuer Services mittels Einbettung der Funktionalität der bereits vorhandenen Services. Diese Technik ist ähnlich zur Erweiterung, erzwingt aber keine Kompatibilität zwischen dem einbettenden und den eingebetteten Services, weil bei der Einbettung nur die Funktionalität und Kommunikationsstruktur aber keine Ausführungslogik (die Befehlsstruktur) der untergegliederten Services übernommen werden. Der einbettende Service muss seine eigene Logik implementieren, die teilweise auf der Funktionalität der eingebetteten Services basieren darf. Die Datenübermittlung zwischen dem übergeordnetem und den eingebetteten Services erfolgt mittels des direkten Zugriffs auf die Instanzen der Letzteren. Die Injektionstechnik bietet besondere Vorteile bei der Implementierung der Services auf einer funktionellen Ebene aber über unterschiedliche Tätigkeitsbereiche, also in heterogenen Szenarien. Soweit dem Autor bekannt ist, wurde die Injektionstechnik bei der Entwicklung der Mikroservices bisher weder von einem Softwareprodukt eingesetzt noch wurde sie in der Literatur beschrieben.
- **Komposition** – Erstellung der zusammengesetzten (Master-) Services aus mehreren untergeordneten (Komposita-) Services. Vom Master-Service wird weder die Ausführungslogik noch die Kommunikationsstruktur der darunterliegenden Services übernommen. Der Datenaustausch erfolgt ausschließlich mittels zusätzlicher Kommunikation zwischen den Proxies des Master-Service und den Ports der Komposita-Services.

Eine Zusammenfassung der oben beschriebenen Servicekompositionskonzepte ist in **Abbildung 34** in Bezug auf Aero- („*A..*“), Gas- („*G..*“) und kombinierte („*AG..*“) Modelle gegeben.

In folgendem Kapitel 4.3 werden die Implementierungsmöglichkeiten der Modellhierarchie mittels der ausgearbeiteten service-orientierten Implementierungsstrategien erläutert.

Objekte und Modelle	Bereich 1: (**A**)erodynamik	Bereich 2: (**G**)asdynamik
	Ebene 1: Diskrete Elemente	
	A1.1 Approximationselement (Transport) **A1.2** Approximationsknoten (Kontinuität)	**G1.1.1** Approximationselement mit Methan-Bildung **G1.1.2** Approximationselement mit Methan-Transport
	Ebene 2: Basis-Elemente	
	A2.1 Streb **A2.2** Zufuhrstrecke **A2.3** Abfuhrstrecke **A2.4** Schacht **A2.5** Hohl- (Filtrations-) Raum	**G2.3** Abfuhrstrecke mit Methan-Transport **G2.5** Hohl- (Filtrations-) Raum mit Methan-Bildung
	Ebene 3: Regelungs-Elemente	
	A3.1 Zufuhrstrecke mit lokalem Regler	
	Ebene 4: Bewetterungsabteilung	
	A4.1 Abteilung ohne Hohlraum **A4.2** Abteilung mit Hohlraum	**G4.2** Abteilung mit Hohlraum
	Ebene 5: Netz	
	A5.1 Netz	**G5.1** Netz

***Abbildung 33**. Hierarchie der Grubenbewetterungsmodelle.*

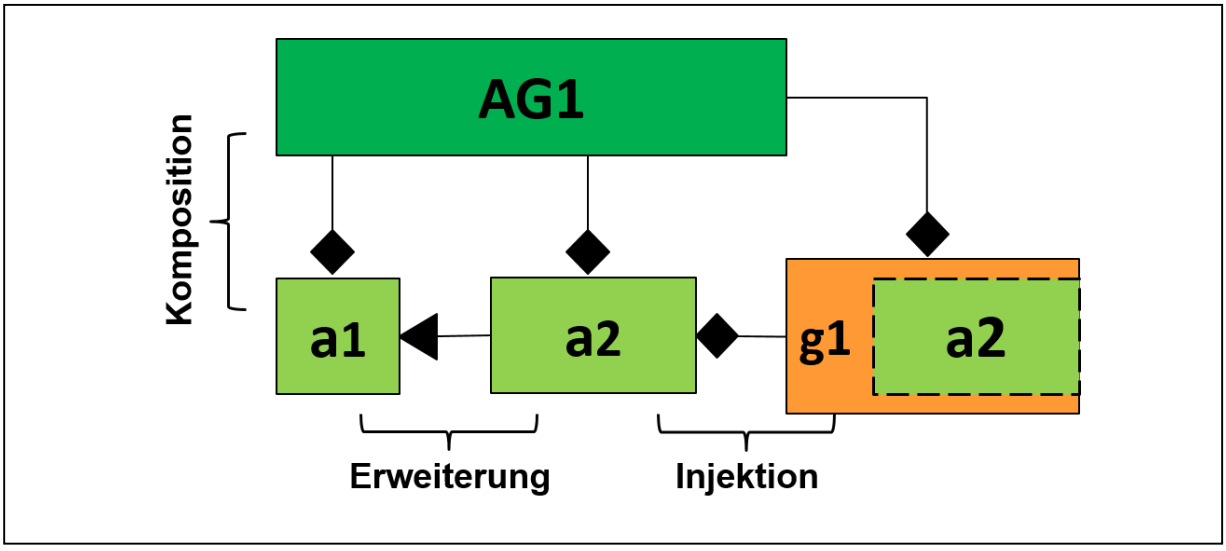


***Abbildung 34**. Erstellungsstrategien für hierarchisch zugeordnete Services.*

4.3 Umsetzung der Modelle mittels Mikroservices

Der Inhalt dieses Kapitels befasst sich mit der Implementierung der Basismodelle der Modellhierarchie in der Form von Mikroservices. Die dafür notwendigen Design- und Gestaltungstrategien, Kommunikationsverfahren und sonstige Aspekte der Entwicklung service-orientierter Software werden erläutert. Die Erstellung- und Nutzungsstrategien der Services werden anhand von praktischen Beispielen aus dem Referenz-Implementierungsframework MS-Sim [W32] erläutert.

Methodologie der Simulationsservice-Beschreibung

Die Service-Entwicklung richtet sich nach dem in Kapitel 3.1 erläuterten Vorgehens-modell aus. Dabei sind die Modellierungsmikroservices meistens als Teilkomponente eines Workflows organisiert, die einen spezifischen Teil bzw. eine Funktion der gesamten Anwendungslogik implementieren, geordnet und gesteuert von einer dedizierten Servicekomponente bzw. einer Anwendung (die weiterhin als *Master* bezeichnet werden). Diese Steuerung erfolgt mithilfe eines Protokolls, das bei der Übermittlung von Instruktionen (Befehlen) vom Master an die untergeordneten Services (also - *Slaves*) einheitlich von allen Services benutzt wird, wie es bspw. in **Abbildung 35** dargestellt ist.

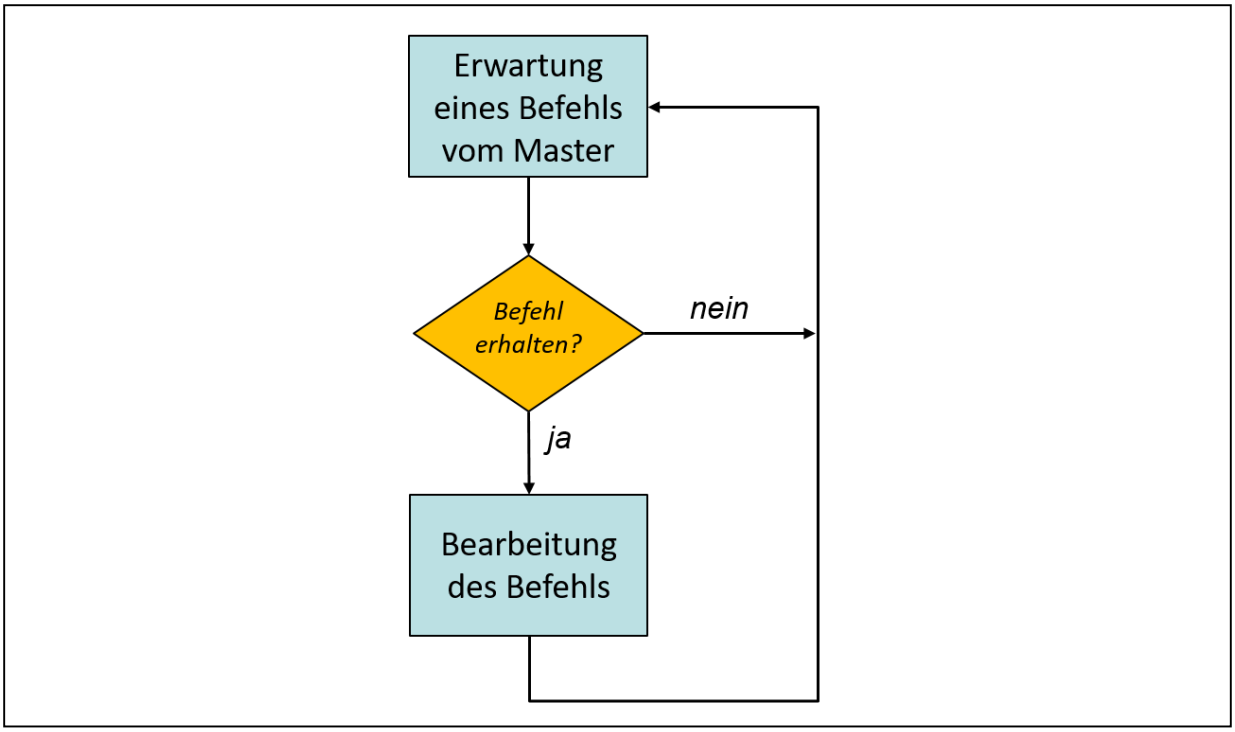


***Abbildung 35**. Koordination der Mikroservice-Funktionen in Workflows.*

Vor diesem Hintergrund sollte die formale Beschreibung eines Mikroservice die folgenden Aspekte seiner Funktionierung abdecken:

- Simulationsfunktionalität:
    - Beschreibung der Parameter und Eigenschaften der zu implementierenden Modelle.
    - Definition der Prozeduren zur Durchführung der Simulationsalgorithmen.
- Kommunikationsmodell:
    - Spezifikation des Instruktionsprotokolls für den Austausch mit dem Master und Erläuterung der einzuleitenden Funktionen. Das Protokoll sollte Befehle zum Starten und Stoppen des Service, zur Ausführung der Modellierungsaufgaben und sonstige wichtige Funktionen beinhalten.
    - Spezifikation der Kommunikationsports (siehe Kapitel 3.2) der Mikroservices für den Datenaustausch mit dem Master sowie anderen Mikroservices im Workflow, wie sie vom Kommunikator (siehe Kapitel 3.3) vorgegeben ist.

Bei der Kommunikationsspezifikation ist die Nutzung spezieller Tools wie die von UML-Sequenzdiagrammen vorteilhaft. Leider ist die vom aktuellen UML-Standard angebotene Syntax aber eher auf generische Problemstellungen ausgerichtet und kann daher nicht alle von Mikroservices angebotenen Funktionalitäten abdecken. Aus diesem Grund wird im Rahmen dieser Arbeit eine Erweiterung des allgemeinen Sequenzdiagramms (wie sie z.B. von Brücher und Endl in [R105] oder von Rumpe in [R106] beschrieben ist) für eine mikroservicespezifische Aufgabestellung vorgeschlagen, und zwar:

- Bei der Übermittlung der Kommandobefehle vom Master werden auf den Verbindungen zwischen den Mikroservices die entsprechenden Identifikatoren angegeben.
- Bei den Kommunikationen werden die Service-Endpoints- (*Ports* bzw. *Proxys*) Bezeichnungen mit Angabe der entsprechenden ID eingeführt.

- Die Kommunikationen werden in einzelne (*Punkt-zu-Punkt*) und kollektive (z.B. *Reduce*) eingeteilt.

Abbildung 36 skizziert die wichtigsten vorgeschlagenen Erweiterungen im UML-Sequenzdiagramm. Die UML-Diagramme werden im Übrigen auch zur Unterstützung weiterer Entwicklungsaufgaben benutzt, z.B. für die Spezifikation der Komposita-Services.

Modelle diskreter Elemente der Aerodynamik

Approximationseinheiten bilden die dynamischen Prozesse feinster Granularität in Elementen des Luft- bzw. Gas-Strömungskontinuums ab, wie sie durch Basis-gleichungen (IV.14)-(IV.16) definiert werden kann, entsprechend der Struktur der **Abbildung 31**. Die mit diesen Prozessen verbundene Modellierungsfunktionalität wird in die Mikroservices ***A1.1*** und ***A1.2*** entsprechend für ein Diskretisierungs-Element und einen Diskretisierungs-Knoten implementiert, sowie für Elemente mit Methanbildung (***G1.1.1***) und Transport (***G1.1.2***).

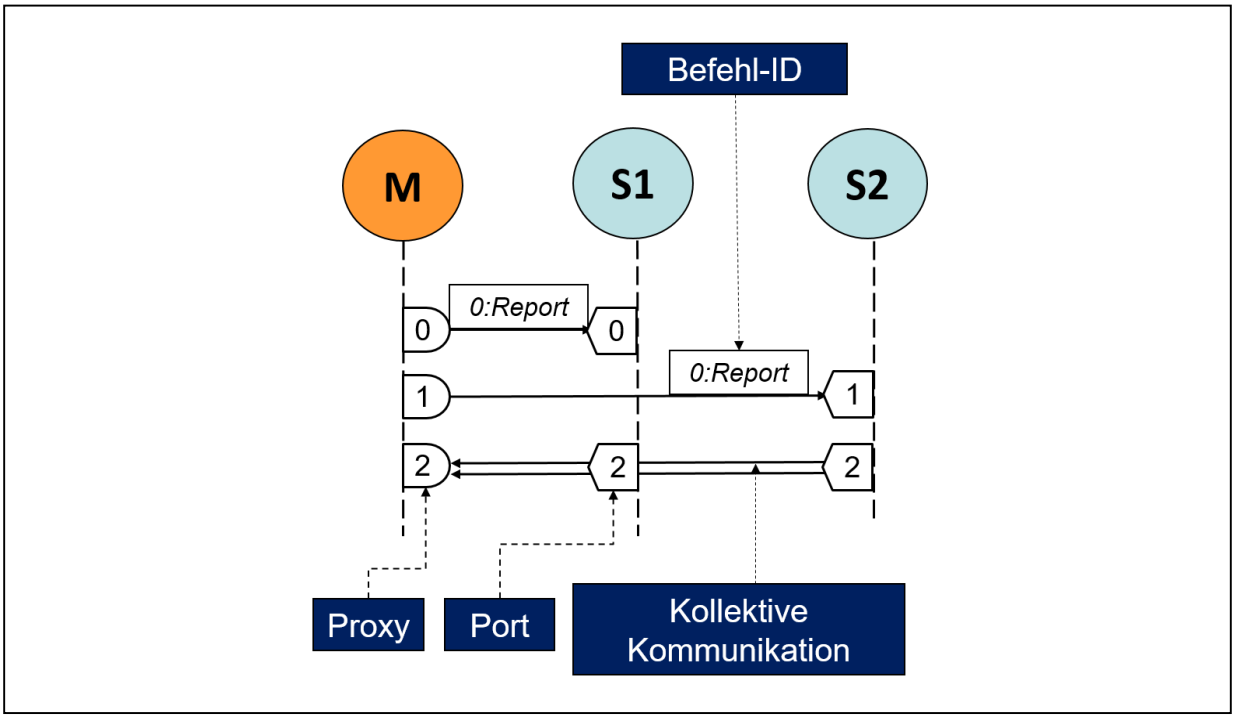


Abbildung 36*. Erweitertes UML-Sequenzdiagramm für ein Mikroservices-Kommunikationsmodell.*

Die bedeutendsten Eigenschaften und Parameter der beiden *A1.x*–Services der Aero-dynamik-Modelle sind in der **Tabelle 3** zusammengefasst. Dabei werden die aero-

dynamischen und topologischen Eigenschaften meistens statisch initialisiert, also einmalig zum Start der Simulation in der Initialisierungsphase, da sich ihre Werte während der Simulation nicht maßgeblich verändern. Dieses sollte mittels entsprechender Konstruktoren erfolgen, wie z.B. im Listing in **Abbildung 37** aufgeführt.

```
1  class A1.1: public Microservice {
2      public:
3
4      A1.1(int id, float S, float r, float l) : Microservice(id_)
5  }
6
7  class A1.2: public Microservice {
8      public:
9
10     A1.2(int id, float S, float l) : Microservice(id_)
```

***Abbildung 37**. Initialisierung von Services für diskrete Element und Knoten.*

***Tabelle 3**. Parameter von A1.x-Services diskreter Elemente.*

	A1.1 – Diskretes Element (q)	**A1.2 – Diskreter Knoten (p)**
Statische	*Aerodynamische Eigenschaften*	
	$s\ [m^2]$ – die Querschnittfläche	
	l *[m]* – die Länge des Diskretisierungselements	
	r $[Ns^2/m^9]$ – der spezifische aerodynamische Widerstand	
	Topologische Anordnung	
	ID – die Identifikation, die die Position des Elements in der Netzstruktur bezeichnet (freies Format mit Angaben der Netz-, Abteilungs- und Element-Bezeichnung)	
		Is_boundary – ein Flag, das bestimmt, ob es für Knoten eine Rand-bedingung erfüllt ist (z.B. Anschluss eines Ventilators).
Dynamische	*Aerodynamische Parameter*	
	r´ $[Ns^2/m^8]$ – der Widerstand des Regelelements (z.B. Schieber, Ventilationstür)	P $[N/m^2]$ – der Druck
	q $[m^3/s]$ – der Modellwert des Luftdurchsatzes	
	q_s $[m^3/s]$ – der Sensorwert des Luftdurchsatzes (optional)	

Die Parameter von Modellen (z.B. Luftdurchsatz im Diskretisierungselement oder Druck im Diskretisierungsknoten) werden dagegen auf dynamische Weise bestimmt, also während der Simulation von einem iterativen Modellierungsalgorithmus errechnet oder von einem Sensor erfasst.

Um die Setzung bzw. die Abrufung von Parametern und Eigenschaften der Modelle seitens der benachbarten sowie übergeordneten Service zu ermöglichen, sollten für *A1.X*-Services die entsprechenden Ports angelegt werden, wie z.B. in der **Tabelle 4** gezeigt.

Dabei werden die Ein- und Ausgabefunktionen durch unterschiedliche Ports (*Set* und *Get*) realisiert, damit ggf. die Service-Synchronisierung nicht gestört wird. Die Nutzung von *Port 0* wird in der Regel zur Realisation der Anwendungslogik (also der Übermittlung der Instruktionsbefehle vom Master an Simulatoren) reserviert, wie es die **Abbildung 38** aufzeigt.

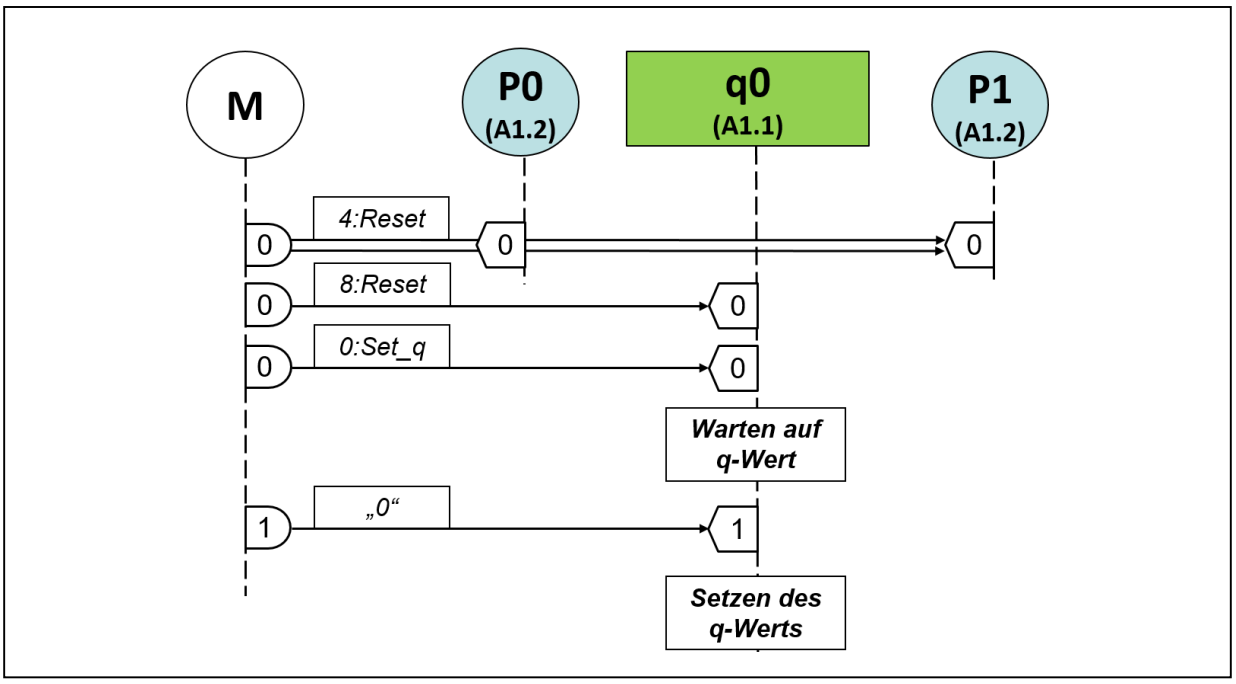


***Abbildung 38**. Steuerung der Mikroservices der Master-Anwendung.*

Um den Umgang mit dynamischen Parametern zu ermöglichen, sollte die Kommunikationsschnittstelle der *A1.x*-Mikroservices die Erfüllung folgender Funktionen ermöglichen:

- Setzung und Abrufung der aerodynamische Parameter (*Q, Q_s, P*) von einem übergeordneten Service oder einer Anwendung.
- Berechnung der Werte der aerodynamischen Parameter im nächsten Zeitschritt der numerischen Lösung.
- Speicherung der aktuellen Werte der aerodynamischen Parameter.
- Steuerung im Rahmen der Workflow-Logik (Stoppen von Service, Identifikation, Zurücksetzung der Parameter usw.).

Die wichtigsten Befehle sowie die mit ihnen assoziierten Ein- und Ausgabe-Ports vom Kommunikationsprotokoll der *A1.x*-Services sind in **Tabelle 5** erfasst[42].

***Tabelle 4**. Kommunikationsports der Diskrete-Elemente-Services.*
(a) Diskretes Element (A1.1)

ID	Bezeichnung	Typ	Größe	Funktion
0	Command	Eingabe	1	Empfang eines Befehls
1	**Set_q**	Eingabe	1	Setzen des Wertes von Q-Parameter
2	**Get_q**	Ausgabe	1	Ausgabe des Wertes von Q-Parameter
3	**Set_qs**	Eingabe	1	Setzen des Wertes von Q_s-Parameter
4	**Get_qs**	Ausgabe	1	Ausgabe des Wertes von Q_s-Parameter
5	Num_set_pstart	Eingabe	1	Austausch mit A1.2-Services während eines numerischen Lösungsvorgangs
6	Num_set_pend	Eingabe	1	
7	Num_get_q	Ausgabe	1	

(b) Diskreter Knoten (A1.2)

ID	Bezeichnung	Typ	Größe	Funktion
0	Command	Eingabe	1	Empfang eines Befehls
1	**Set_p**	Eingabe	1	Setzen des Wertes von P-Parameter
2	**Get_p**	Ausgabe	1	Ausgabe des Wertes von P-Parameter
3	Num_set_qin	Eingabe	*[43]	Austausch mit A1.1-Services während eines numerischen Lösung-Vorgangs
4	Num_set_qout	Eingabe	*	
5	Num_get_p	Ausgabe	1	

[42] Es soll bemerkt werden, dass das Kommunikationsprotokoll nach Bedarf durch Hinzufügen der weiteren bzw. Änderung der bestehenden Instruktionen zu jeder Zeit erweitert werden kann.
[43] Falls mit einem Knoten mehrere Elemente verbunden sind, sollten alle Verbindungen berücksichtigt werden

***Tabelle 5**. Steuerbefehle der Diskrete-Elemente-Services.*

(a)Diskretes Element (A1.1)

ID	Bezeichnung	Funktion	Assoziierte Ports
0	**Set_q**	Setzen des Wertes von q-Parameter	1
1	**Set_qs**	Setzen des Wertes von q_s-Parameter	2
2	**Get_q**	Ausgabe des Wertes von q-Parameter	3
3	**Get_qs**	Ausgabe des Wertes von q_s-Parameter	4
4	**Simulation**	Berechnung des q-Parameters im nächsten numerischen Zeitschritt	5-7
5	Save	Speicherung der Daten	
6	Stop	Beendigung des Services	
7	Id	Ausgabe der ID-Information	
8	Reset	Setzen aller Parameter in Initialzustand	

(a) Diskreter Knoten (A1.2)

0	**Set_p**	Setzen des Wertes von p-Parameter	1
1	**Get_p**	Ausgabe des Wertes von p-Parameter	2
2	**Simulation**	Berechnung des p-Parameters im nächsten numerischen Zeitschritt	3-5
3	Save	Speicherung der Daten	
4	Stop	Beendigung des Services	
5	Id	Ausgabe der ID-Information	
6	Reset	Setzen aller Parameter in Initialzustand	

Die Modellierungsfunktionalität beider Services ist relativ unkompliziert und besteht aus der Anwendung eines numerischen Verfahrens auf das Basisgleichungssystem des entsprechenden Modells zur Berechnung des approximativen Wertes von aerodynamischen Parametern in einem folgenden Zeitschritt der numerischen Lösung (entsprechend des Zeitdiskretisierungsschemas). Dafür wurden in Services zwei numerischen Verfahren implementiert – das Einschritt-Euler-Verfahren:

$$y_i = y_{i-1} + h \cdot f(x_{i-1}, y_{i-1}), \qquad \text{(IV.17)}$$

und das vierstufige Runge-Kutta-Verfahren (RK4):

$$
\begin{aligned}
y_i &= y_{i-1} + \frac{h}{6} \cdot (k_1 + 2k_2 + 2k_3 + k_4), \\
k_1 &= f(x_{i-1}, y_{i-1}), \\
k_2 &= f(x_{i-1} + \frac{h}{2}, y_{i-1} + \frac{h}{2} k_1), \\
k_3 &= f(x_{i-1} + \frac{h}{2}, y_{i-1} + \frac{h}{2} k_2), \\
k_4 &= f(x_{i-1} + h, y_{i-1} + h \cdot k_3),
\end{aligned}
\tag{IV.18}
$$

wobei *i* – der Zeitschritt, *h* – die Schrittweite, *y=f(x,y)* – der Basisparameter des Modells sind. Die beiden Methoden realisieren ein explizites Lösungsschema.

Die Definition des zu benutzenden numerischen Lösers sowie der dazu notwendigen Parameter erfolgt in der Initialisierungsphase der Mikroservices. Der Austausch der Daten, die während eines Simulationsschritts die mutualen funktionalen Abhängigkeiten zwischen den *Q*- und *P*-Services entsprechend der Modelle (IV.14), (IV.16) abbilden, erfolgt mittels eines Kommunikationsschemas, wie in **Abbildung 39** für das Euler-Verfahren gezeigt.

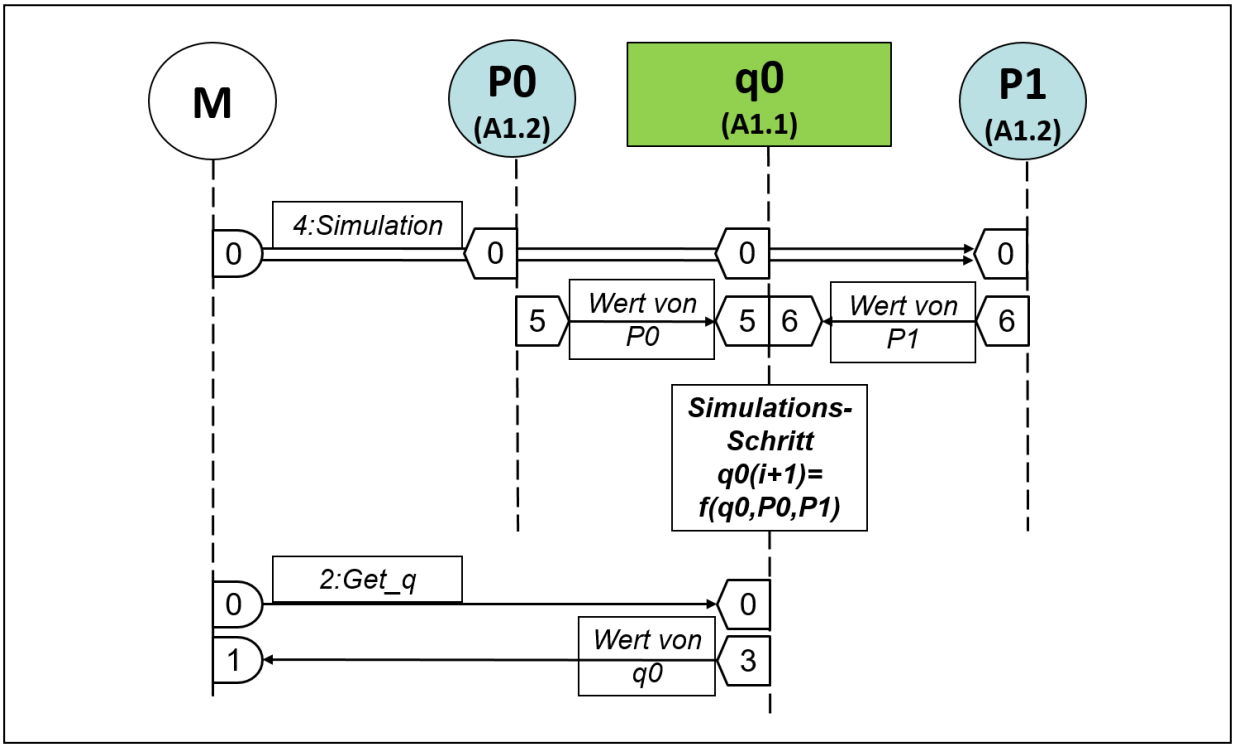


***Abbildung 39**. Kommunikation zwischen Services während der Berechnung eines Simulationsschritts mit dem Euler-Verfahren.*

Zur Realisation des Runge-Kutta-Verfahrens ist jeweils eine Austauschoperation für jeden der $K_{1\text{-}4}$-Koeffizienten (IV.18) auf jedem Schritt der numerischen Lösung nötig.

Der Ablauf jeder Simulation wird vom Master-Service (bzw. der Anwendung) zentral überwacht, indem die Konvergenz der Lösung der modellierten Parameter in regelmäßigen Zeitabständen überprüft werden (z.B. mithilfe von entsprechenden *Set*- und *Get*-Befehlen und anschließender Evaluierung der Werte). Der Master kann so koordinieren, wann die iterative numerische Lösung beendet werden soll, ob und wie die Ergebnisse ausgegeben werden, kann den nächsten Simulationsvorgang planen sowie viele weitere Aktivitäten einleiten und kontrollieren.

Modelle diskreter Elemente der Gasdynamik

Die gasdynamischen Prozesse der Methanbildung und Transportierung werden maßgeblich von der Aerodynamik unter Tage beeinflusst, wie sie auch bei der Entwicklung ihrer Modelle berücksichtigt wurde (siehe Kapitel 4.1.2). Insbesondere verdeutlicht sich die Problematik der funktionalen Abhängigkeiten zwischen den aero- und gasdynamischen Modellen auf der Ebene der Simulationssoftware, die vor der entsprechenden Herausforderung einer möglichst effizienten und nahtlosen Integration mit den vorher erstellten Services steht.

Die Einbindung der aerodynamischen Modelle in die gasdynamischen Simulationsservices kann auf zwei verschiedene Wege erfolgen, die über die unterschiedlichen Architekturlösungen für die Letzteren erreicht werden sollen:

- Die gasdynamischen Services implementieren nur gasspezifische Modelle und beziehen die notwendigen Daten von den aerodynamischen Services über die üblichen Kommunikationsschnittstellen.
- Die gasdynamischen Services werden durch **Injektionstechnik** (siehe Details im Kapitel 4.2) der aerodynamischen Services mit gasdynamischen Modellen implementiert. In diesem Fall würden die Services beide Funktionalitäten „unter einem Dach“ vereinbaren – die aero- sowie die gasdynamische Analyse in einem Servicepaket.

In der Praxis ist die Umsetzung des Trennungskonzepts zwischen dem Service beider Modelltypen sehr aufwendig (Notwendigkeit vieler zusätzlicher Ports, Kommunikator-Einträge usw.) und sollte womöglich vermieden werden. Aufgrund der Tatsache, dass es keine speziellen Anforderungen gibt, die für die aero- und nicht für die gasdynamischen Modelle gelten würden, wurde im Rahmen der vorliegenden Arbeit sowie für die MS-Sim-Referenzimplementierung die Entscheidung getroffen, die Entwicklung der Services für gasdynamischen Modelle – der Methan-Ergiebigkeit im Filtrationsraum (***G1.1.1***) und dem Transport im Luftweg (***G1.1.2***) – auf der Basis von ***A1.1***-Services mithilfe der Injektionstechnik durchzuführen, entsprechend der Struktur in **Abbildung 40**.

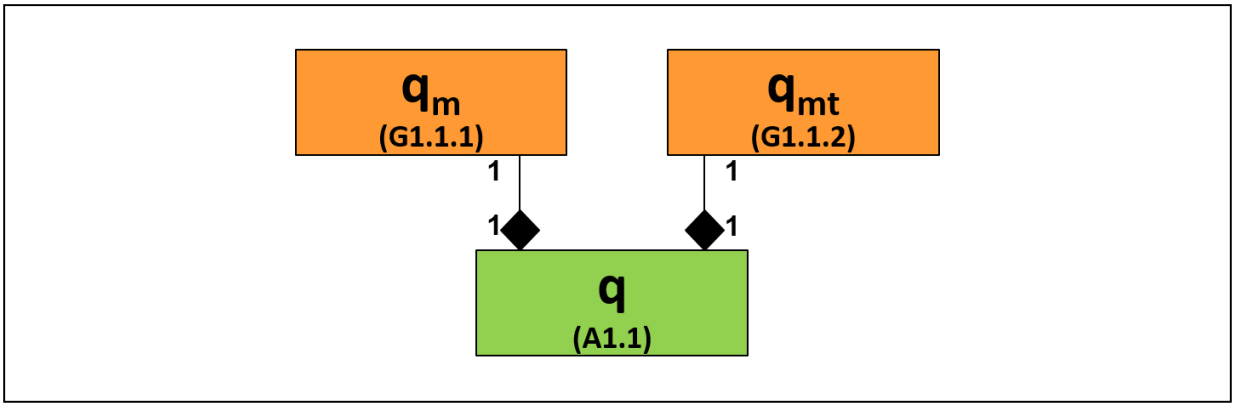


Abbildung 40. *Aufbau der Modelle von Gasergiebigkeit und Transport (G1.1.x).*

Bei jedem Simulationsschritt *i* löst das *G1.1.1*-Modell die Basisgleichung (IV.15), in ihrer allgemeinen Form aus:

$$q_m^i = f(q_m^i, q^i, dq^i)\,, \tag{IV.19}$$

wobei dq^i dem folgenden Term der Gleichung (IV.15) entspricht:

$$-\frac{S_j}{\rho}\frac{(P_i - P_{i-1})}{\Delta\psi} - rQ_j^2\,. \tag{IV.20}$$

Der Simulationsvorgang wird im *G1.1.1*-Modell also in drei Stufen durchgeführt:

- Der Berechnung von *q* und *dq* durch das eingebundene *A2.1*-Aerodynamik-modell.
- Der Berechnung von q_m

- und der Weiterleitung des q_m -Wertes an die angeschlossenen g_{mt}-Elemente (die von *G1.1.2*-Modellen realisiert werden).

Das *G1.1.2*-Modell berücksichtigt die Übertragung (Transport) des Methans, das aus i) den Ergiebigkeitsquellen im Filtrationsraum entsteht (wie vom *G1.1.1*-Modell definiert) oder ii) es am Ausgang des Kohlefördergebiets (Streb, siehe **Abbildung 6a**) vom Sensor erfasst wird. Unter der Prämisse, dass die Dauer der gasdynamischen Übergangsprozesse im Filtrationsraum deutlich länger ist als die der aerodynamischen Prozesse in einem diskreten Element, sowie auf Grund der unveränderlichen Sensorangaben während der Ausführung eines Simulationsvorgangs, lässt sich die Dynamik der Methanübertragung in der folgenden Form definieren:

$$q_{mt}^{i} = q_{m}^{i} + q_{ms}\,. \tag{IV.21}$$

Die Parameter der beiden *G1.1.x*-Modelle sind in **Tabelle 6** zusammengefasst.

***Tabelle 6**. Zusatzparameter von G1.1.x-Services diskreter Elemente mit Methanbildung.*

	G1.1.1 – Methanergiebigkeit im Filtrationsraum (qm)	**G1.1.2 – Methantransport im Luftweg (qmt)**
Statische	*Gasdynamische Eigenschaften*	
	v $[m^3]$ – das Volumen des Filtrationsraums *m* – das Koeffizient der Porosität	
Dynamische	*Gasdynamische Parameter*	
	q_m $[m^3/s]$ – der Modellwert der Methanergiebigkeit	q_{mt} $[m^3/s]$ – der Modellwert der transportierten Methanmenge
	q_{m0} $[m^3/s]$ – der Messwert der Methanergiebigkeit in den Filtrationsraum	q_{ms} $[m^3/s]$ – der Sensorwert der transportierten Methanmenge (optional)

Die Steuerbefehle und Kommunikationsports von *G1.1.x*-Services sind in der Struktur ähnlich zum *A1.1*-Service aufgebaut, berücksichtigen aber noch zusätzlich die in **Tabelle 6** aufgeführten spezifischen gasdynamischen Parameter und Eigenschaften.

Modelle der Hauptelemente

Die Modelle der ersten Hierarchieebene bilden die Grundlage zur Erstellung der Modelle der wichtigsten Elemente eines Bewetterungsnetzes – des Strebs (***A2.1***), der Zufuhr- (***A2.2***) und Abfuhrstrecken (***A2.3***), des Schachts (***A2.4***) und des Hohlraums (***A2.5***), die z.B. entsprechend der Struktur in der **Abbildung 32** zur Realisation des Abteilungsmodells (mit und ohne Leckströmung im Filtrationsraum sowie Gasbildung) zusammengesetzt werden können.

A2.1 – Aerodynamik im Streb

Das ***A2.1*-Modell** befasst sich mit der Analyse aerodynamischer Prozesse in einem **Streb** – einem Kohleförderungsraum des Bewetterungsnetzes. Das Modell kann aus *N* Modellen (wobei *N* – die Anzahl der Diskretelemente, die von der angewandten örtlichen Dekomposition abhängig ist), Modellen *q* (*A1.1*-Service) und Knoten *p* (*A1.2*-Service) mithilfe der Kompositionstechnik, wie im Kapitel 4.2 beschrieben, umgesetzt werden. Die Zusammensetzung der Services kann in der Form dargestellt werden, wie sie in **Abbildung 41** skizziert ist.

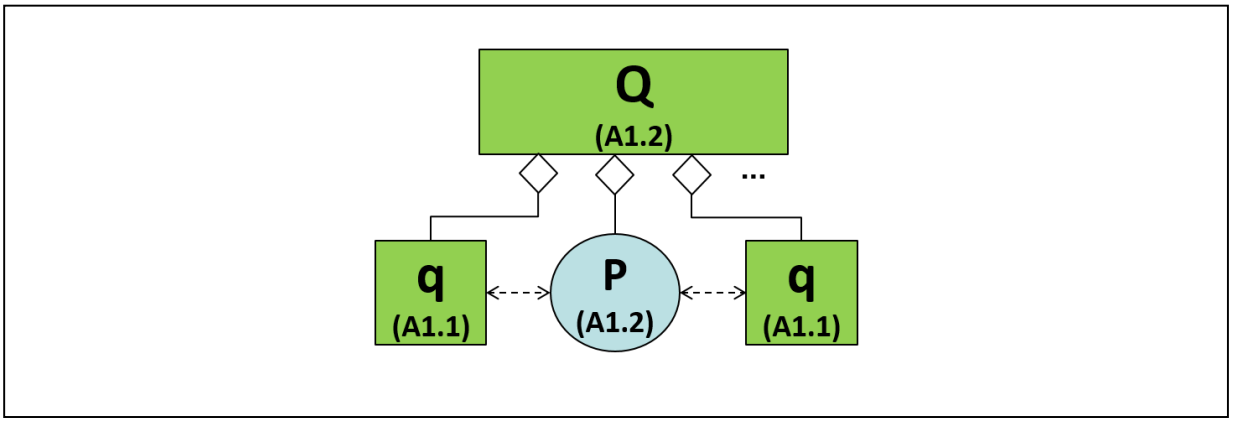


***Abbildung 41**. Kompositionsstruktur des Modells der Aerodynamik des Strebs.*

Die Parameter des Streb-Modells sind an die von den eingegliederten elementaren Services angelehnt, wie es in **Tabelle 7** zusammengefasst ist.

Die eingegliederten *A1.x*-Modelle werden von dem *A2.1*-Modell mit Hilfe des *MS-Deployment-API* angelegt und mit Parametern initialisiert, die aus den Parametern des Strebs abgeleitet werden sollten, wie es im Teil (b) der **Tabelle 7** aufgezeigt ist.

***Tabelle 7**. Parameter des Q-Modells des Strebs (A2.1) und Ableitung der Parameter von eingegliederten Services.*

|——————— (a) ———————|——————— (b) ———————|

	A2.1 – Streb **Q**	**A1.1 – Diskretes Element** $\mathbf{q_i}$ (i=1..*N*)	**A1.2 – Diskreter Knoten** $\mathbf{p_j}$ (j=1..*N*-1)
Statische	*Aerodynamische Eigenschaften*		
	dx – die Länge der örtlichen Diskretisierung		
	N=L/dx – Anzahl diskreter Elemente		
	S $[m^2]$ – die Querschnittfläche	$s_i = S$	
	L *[m]* – die Länge des Elements	l_i = dx	
	R $[Ns^2/m^8]$ – der aerodynamische Widerstand	r_i = R/L	
	Topologische Anordnung		
	ID_Element (z.B. „Q0“)	*ID_Element _„***q**i“	*ID_Element _„***p**j“
Dynamische	*Aerodynamische Parameter*		
	$Q=\{q_1,...,q_n\}$ $[m^3/s]$ – der Modellwert des Luftdurchsatzes in allen diskreten Elementen vom Streb	$q_i=Q$[i]	
	$P=\{p_1,...,p_{n-1}\}$ $[N/m^2]$ – der Druck in allen diskreten Knoten vom Streb		p_j=P[i]

Zu den Aufgaben des übergeordneten *Q*-Modells gehört auch die Initialisierung des Kommunikationsmodells für die eingegliederten q_i- und p_j-Services, welches die beiden folgenden Ebenen umspannt:

- Die Kommunikation zwischen den *q*- und *p*-Services beim Austausch von Randbedingungswerten während der numerischen Lösung (zu implementieren mithilfe von Punkt-zu-Punkt-Kommunikationsoperationen, wie im Sequence-Diagramm in **Abbildung 39** beschrieben):

  q_i:**get_num_q** → p_i:**set_q_in** (i=1..N-1)

  q_i:**get_num_q** → p_{i-1}:**set_q_out** (i=2..N)

  p_j:**get_num_p** → q_{j+1}:**set_p_start** (j=1..N-1)

  p_j:**get_num_p** → q_{j+1}:**set_p_start** (j=1..N-1)

- Die Kommunikation zwischen dem Q-Service und seinen eingegliederten q- und p-Services (zu implementieren mithilfe der kollektiven Kommunikations-operationen “Replicate“, “Spread“ und “Gather“):

  Q:proxy_**Command** (Replicate) →

  q_i:**Command**, p_j:**Command** (i=1..N, j=1..N-1)

  Q:proxy_**set_q** (Spread) → q_i:**set_q**

  Q:proxy_**get_q** (Gather) ← q_i:**get_q**

  Q:proxy_**set_qs** (Spread)→ q_i:**set_qs**

  Q:proxy_**get_qs** (Gather)← q_i:**get_qs**

  Q:proxy_**set_p** (Spread)→ p_j:**set_p**

  Q:proxy_**get_p** (Gather)← p_j:**get_p**

Eine Zusammenfassung aller Steuerbefehle und Ports für den *Q*-Service (A2.1) ist in **Tabelle 8** dargestellt.

Tabelle 8*. Steuerbefehle und Ports des Q-Services (A2.1).*

ID	Bezeichnung	Funktion	Assoziierte Ports (E-Eingabe, A-Ausgabe)	
			Port	Größe
0	**Set_Q**	Setzen des Wertes von allen q-Parametern	E: Set_Q	N
1	**Set_P**	Setzen des Wertes von allen p-Parametern	E: Set_P	N-1
2	**Get_Q**	Abrufen der Werte von allen q-Parametern	A: Get_Q	N
3	**Get_P**	Abrufen der Werte von allen p-Parametern	A: Get_P	N-1
4	**Simulation**	Berechnung aller q- und p-Parameter der eingegliederten A1.x-Services im nächsten numerischen Zeitschritt		
5	Save	Speicherung aller q- und p-Parameter der eingegliederten A1.x-Services		
6	Stop	Beendigung des Services sowie aller eingegliederten A1.x-Services		
7	Id	Ausgabe der ID-Information aller eingegliederten A1.x-Services		
8	Reset	Setzen der Parameter aller eingegliederten A1.x-Services in Initialzustand		

A2.2 – Aerodynamik in der Zufuhrstrecke

Das ***A2.2*-Modell** befasst sich mit der Analyse von aerodynamischen Prozessen in einer **Zufuhrstrecke** – einem Luftweg, durch den die Frischluft in den Förderbereich (Streb) hineingelangt. Das Zufuhrstrecken-Modell ist mit dem oben beschriebenen *A2.1*-Modell des Strebs identisch, kann aber in zwei verschieden Szenarien ange-wandt werden:

- Ohne Berücksichtigung des Filtrationsraum
- Unter Berücksichtigung des Filtrationsraums.

Im letzteren Fall sollten die Luft-Abflüsse (Leckflüsse) in den Filtrationsraum berücksichtigt werden, der an die betroffene Zufuhrstrecke angeschlossen ist. Dies bedingt aber keinerlei Änderung an dem *A2.1*-Modell aufgrund vieler möglicher zugelassener ausgehenden *q_out*-Verbindungen bei den eingegliederten *p*-Modellen (*A1.2).* Diese Verbindungen werden aber erst von dem übergeordneten Service (z.B. von dem *A4.2*-Modell) erstellt (im entsprechenden Kommunikator), sodass das *A2.1*-Modell gleichzeitig auch als Modell der zuführenden Strecke (*A2.2*) gilt.

A2.3 – Aerodynamik in der Abfuhrstrecke

Das ***A2.3*-Modell** befasst sich mit der Analyse von aerodynamischen Prozessen in einer **Abfuhrstrecke** – einem Luftweg, durch den die Luft aus dem Förderbereich (Streb) abgeführt wird. Ähnlich wie das oben beschriebene *A2.2*-Modell, ist das *A2.3*-Modell mit dem Streb-Modell (*A2.1*) identisch. Der mögliche Zufluss der Leckströmung aus dem Filtrationsraum kann durch *q_in*-Verbindungen bei den eingegliederten *p*-Modellen (*A1.2*) in der Phase der Erstellung eines Kommunikators von einem übergeordneten Service berücksichtigt werden.

A2.4 – Aerodynamik im Schacht

Das ***A2.4*-Modell** befasst sich mit der Analyse von aerodynamischen Prozessen in einem Schacht – einem vertikalen Luftweg, durch den sämtliche Elemente des Bewetterungsnetzes mit der Grundfläche des Grubenbetriebs verbunden sind. Aufgrund der Möglichkeit zur Vernachlässigung der Massenkräfte in dem Transport-

satz der Basisgleichung (IV.8), kann das *A2.4*-Modell identisch zu dem oben beschriebenen Streb-Modell (*A2.1*) implementiert werden, wie von Svjatnyj [R8] für eine Reihe praktischer Anwendungen gezeigt wurde.

A2.5 – Aerodynamik im Hohl- (Filtrations-) Raum

Das ***A2.5*-Modell** befasst sich mit der Analyse von aerodynamischen Prozessen im Hohlraum – einem an den Streb angrenzenden Raum zwischen der Zu- und der Abfuhrstrecke, der nach dem Kohleabbauprozess im fortlaufenden Kohlegewinnbetrieb entsteht. Aufgrund seiner porösen Struktur tritt im Hohlraum eine Leckströmung durch die Zufuhrt in die Abfuhrstrecken auf. Svjatnyj schlug in [R8] eine Methodik der Approximation der Leckströmung durch einen konzentrierten Strom vor, der zwischen zwei approximativen Knoten in jeweiligen Strecken parallel zum Streb verläuft. In diesem Fall ist das *A2.1*-Modell auch auf die aerodynamischen Prozesse im Hohlraum unverändert anwendbar.

G2.5 – Methanergiebigkeit im Hohl- (Filtrations-) Raum

Das ***G2.5*-Modell** erweitert die Funktionalität des aerodynamischen *Q*-Modells des Filtrationsraums (*A2.5*) zur Analyse der Methanbildungsprozesse (*Qm*), wie von *G1.1.1*-Modellen befasst wird. Das *G2.5*-Modell wird mit Hilfe des Kompositionsansatzes auf der Basis der *G1.1.1*- und *A1.2*-Modelle auf eine ähnliche Weise wie das vorher beschriebene *A2.5*-Modell aufgebaut (siehe **Abbildung 42**).

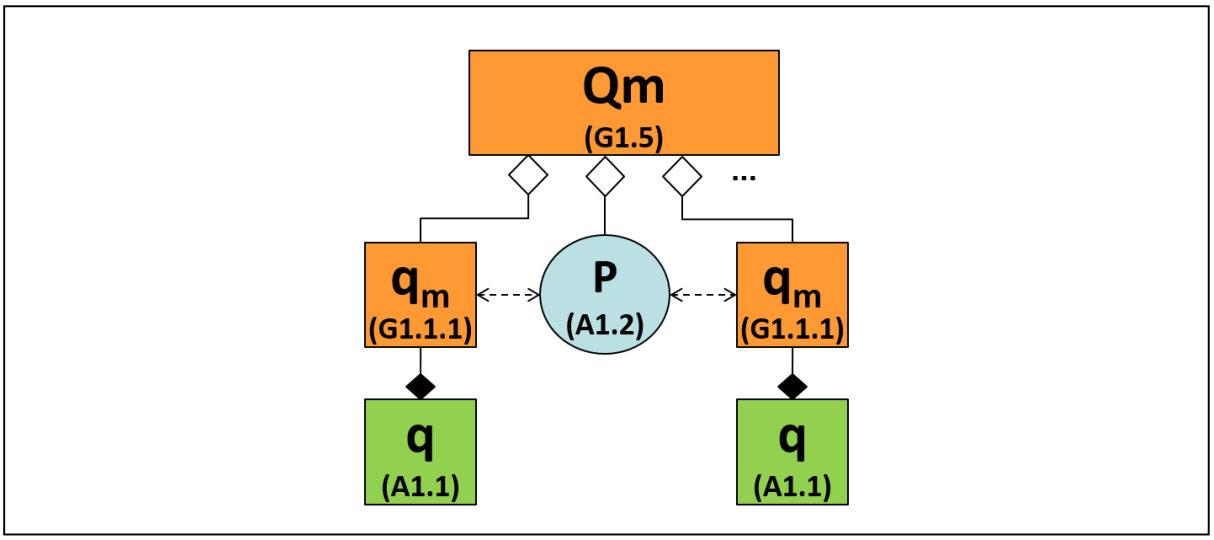


Abbildung 42. *Kompositionsstruktur des Modells der Gasdynamik im Filtrationsraum.*

Dabei basieren die Parameter des *Qm*-Modells (*G2.5*) auf denen des *Q*-Modells (*A2.5*), ergänzen diese aber mit den für die Methanbildung spezifischen Komponenten, wie in **Tabelle 9** zusammengefasst ist.

***Tabelle 9**. Zusatzparameter des Qm-Modells der Methanergiebigkeit im Filtrationsraum (G2.5) und Ableitung der Parameter von eingegliederten Services.*

|——————— (a) ———————|——— (b) ———|

	G2.5 – Filtrationsraum Qm	**G1.1.1 – Diskretes Element qm_i** (i=1..*N*)
Statische	*Gasdynamische Eigenschaften*	
	V $[m^3]$ – das Volumen des Filtrationsraums *M* – das Koeffizient der Porosität	$v_i = V/N$ $m_i = M$
	Topologische Anordnung	
	ID_Element (z.B. „Qm0“)	*ID_Element_„***q**i“ *ID_Element_„***qm**i“
Dynamisch	*Gasdynamische Parameter*	
	$Qm=\{qm_1,...,qm_n\}$ $[m^3/s]$ – der Modellwert der Methanergiebigkeit in allen diskreten Elementen des Strebs	$qm_i=Qm$[i]

Die Struktur der Steuerbefehle und Ports des *Qm*-Service bleiben auch ähnlich wie bei dem *Q*-Service und werden um zur Übermittlung des Methandurchsatzes notwendige Einträgen erweitert, wie in **Tabelle 10** zusammengefasst wird.

***Tabelle 10**. Zusatzsteuerbefehle und Ports des Qm-Services (G2.5).*

ID	**Bezeichnung**	**Funktion**	**Assoziierte Ports (E-Eingabe, A-Ausgabe)**	
			Port	**Größe**
0	**Set_Qm**	Setzen der Werte von allen qm-Parametern	E: Set_Qm	N
1	**Set_Qm0**	Setzen der Werte von allen qm0-Parametern	E: Set_Qm0	N
2	**Get_Qm**	Abrufen der Werte von allen qm-Parametern	A: Get_Qm	N
3	**Get_Qm0**	Abrufen der Werte von allen qm0-Parametern	A: Get_Qm0	N

G2.3 – Methanübertragung im Hohl- (Filtrations-) Raum

Das ***G2.3*-Modell** erweitert die Funktionalität des aerodynamischen *Q*-Modells der Abfuhrstrecke (*A2.3*) zur Analyse der Methantransportprozesse (*Qmt*), wie von *G1.1.2*-Modellen befasst wird. Das *G2.3*-Modell und der Service werden nach einem ähnlichen Kompositionsprinzip wie das vorher erläuterte *G2.5*-Modell aufgebaut, allerdings auf der Basis von *qmt*-Modellen (*G1.1.2*) anstatt den in *G2.5*-Modell verwendeten *qm*-Modellen.

Die Parameter des *Qmt*-Modells sind mit denen des *Q*-Modells (**Tabelle 7**) identisch. Das Kommunikationsmodell des *Qmt*-Service wird durch die Hintereinanderschaltung der *qmt*-Services entsprechend der Strömungsrichtung, wie folgt definiert:

$$qmt_i: gas\_get\_qm \rightarrow qmt_{i+1}: gas\_set\_qm \ (i=1..N-1)$$

Die *Qmt*-Erweiterungen der Steuerbefehle des Basis-*Q*-Service (*A1.1*) beinhalten lediglich "*Set_Qm*"- und "*Get_Qm*"-Operationen.

A3.1 – Aerodynamik in der Zufuhrstrecke mit lokalem Regler

Das *A3.1*-Modell sollte, zusätzlich zu den im *A2.2*-Modell bereits inbegriffenen Faktoren, die Regelwirkung eines lokalen Reglers (im einfachsten Fall – einer quer zur Strömungsrichtung platzierten Schiebetür) betrachten. Diese Regelwirkung wird durch den *r´*-Parameter des Basis-Modells (IV.11) berücksichtigt, der dem dynamischen Widerstand des lokalen Reglers entspricht. Da sich die physikalische Lage des Reglers in verschiedenen Betrieben unterscheiden kann, sollte bei dem *A3.1*-Modell davon ausgegangen werden, dass jedes *q*-Diskretelement (*A1.1*-Modell) über einen eigenen lokalen Regler verfügen kann.

Das *A3.1*-Modell sollte die Setzung des Wertes der lokalen Regelwirkung in entsprechenden *q*-Modellen der Diskretelemente (*A1.1*) ermöglichen. Dies wird durch die Erweiterung der Ports und des Kommunikationsmodells des *A2.2*-Service mit entsprechenden Funktionen für den „*r_reg*" Parameter des *q*-Modells (*A1.1*) erreicht, wie in **Tabelle 11** und **Tabelle 12** unten skizziert ist.

***Tabelle 11**. Zusatzparameter des Q-Modells der Aerodynamik in der Strecke mit lokalem Regler (A3.1).*

|——————— (a) ———————————|——————— (b) ———————|

	A3.1 – Zufuhrstrecke mit lokalem Regler Q	**A1.1 – Diskretes Element q_i** (i=1..*N*)
Dynamische	*Dynamische Parameter*	
	R_reg={r_reg₁,...,r_regₙ} [m³/s] – der Modellwert der Methanergiebigkeit in allen diskreten Elementen von der Strecke	*r_regᵢ=R_reg*[i]

***Tabelle 12**. Zusatzsteuerbefehle und Ports des Qm-Services (G2.5).*

ID	**Bezeichnung**	**Funktion**	**Assoziierte Ports (E-Eingabe, A-Ausgabe)**	
			Port	**Größe**
4	**Set_R_reg**	Setzen des Wertes von allen r_reg-Parametern	E: Set_R_reg	N
5	**Get_R_reg**	Abrufen der Werte von allen r_reg-Parametern	A: Get_R_reg	N

Modelle der Bewetterungsabteilung

Die Modelle der zweiten (*A/G-2.x*) und dritten (*A3.1*) Hierarchieebenen bilden die Grundlage zur Erstellung der Modelle einer Bewetterungsabteilung – dem wichtigsten Bauwerk eines Grubenbewetterungsnetzes, das aus einem Streb, einer Zufuhr- und einer Abfuhr-Strecke sowie ggf. dem Filtrationsraum besteht. Die Modelle umfassen die Aerodynamik in einer vereinfachten Variante der Abteilung – ohne Leckströmung und dementsprechend ohne Filtrationsraum (***A4.1***), mit Filtrationsraum und Leckströmung, (***A4.2***) sowie mit Gasdynamik im Filtrationsraum und Abfuhrstrecke (***G4.2***).

A4.1 – Aerodynamik in der einfachen Abteilung

Das Modell einer einfachen Abteilung (ohne Filtrationsraum) wird entsprechend seiner physikalischen Struktur mithilfe des Kompositionsansatzes aufgebaut, indem die Modelle der Q_{ZS}–Zufuhrstrecke, in der Version ohne (*A2.2*) bzw. mit einem lokalen Regler (*A3.1*), des Q_{Streb}–Strebs (*A2.1*) und der Q_{AS}–Abfuhrstrecke (*A2.3*) durch zwei *P*-

Diskretknoten (*A1.2*) entsprechend der Struktur[44] in **Abbildung 43** hintereinander geschaltet werden.

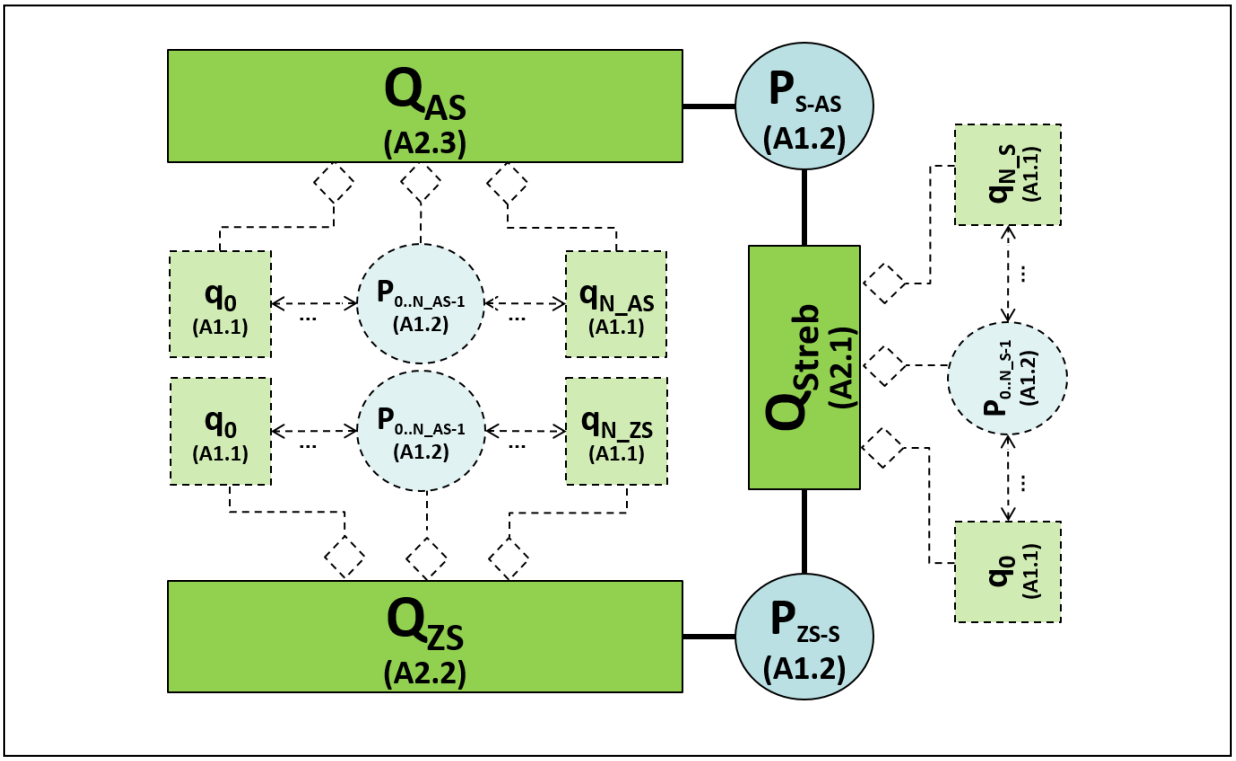


***Abbildung 43**. Kompositionsstruktur des Modells der Aerodynamik in einer einfachen Bewetterungsabteilung.*

Das *A4.1*-Modell ist ein reines Kompositumsmodell – sein Parametersatz wird aus den Parametern entsprechend eingegliederter Services aufgebaut wobei die Portstruktur im Wesentlichen auf die Kommunikation mit den eingefügten Services ausgerichtet ist, mit dem Ziel, die Werte dynamischer Parameter (siehe Beispiel in **Tabelle 7**) der Letzteren abzufragen bzw. zu setzen (mithilfe von Proxies, wie im Kapitel 3.4.3 erläutert), wie in **Tabelle 13** unten zusammengefasst ist.

Die Anlegung, Initialisierung und Steuerung der eingegliederten Modelle während der Durchführung der Simulation erfolgt im *A4.1*-Modell nach der gleichen Strategie, wie im Kapitel 4.3.4 für *A2.1*-Service beschrieben.

[44] Alle Q-Elemente sind der Vollständigkeit halber mit den beinhalteten Komponenten unterer Hierarchieebenen angegeben.

***Tabelle 13**. Steuerbefehle und Ports des A4.1-Service.*

ID	Bezeichnung	Funktion	Assoziierte Ports eingegliederter Services (E-Eingabe, A-Ausgabe)	
			Port	Größe
0	**Set_Q_ZS**	Setzen der Werte von Q-Parametern	E: Q_ZS ::Set_Q	N_{ZS}
1	**_Streb**		E: Q_Streb ::Set_Q	N_{Streb}
2	**_VS**		E: Q_VS ::Set_Q	N_{VS}
3	**Get_Q_ZS**	Abrufen der Werte von Q-Parametern	A: Q_ZS ::Get_Q	N_{ZS}
4	**_Streb**		A: Q_Streb ::Get_Q	N_{Streb}
5	**_VS**		A: Q_VS ::Get_Q	N_{VS}
6	**Set_R_Reg**	Abrufen der Werte von R_Reg-Parametern	A: Q_ZS::Get_R_reg	N_{ZS}
7	**Get_R_Reg**	Abrufung der Werte von R_Reg-Parametern	A: Q_ZS::Get_R_reg	N_{ZS}
8	**Simulation**	Berechnung aller Parameter der eingegliederten Services im nächsten numerischen Zeitschritt		
9	Save	Speicherung aller Parameter der eingegliederten A1.x-Services		
10	Stop	Beendigung des Services sowie aller eingegliederten Services		
11	Id	Ausgabe der ID-Information aller eingegliederten Services		
12	Reset	Setzen der Parameter aller eingegliederten Services in Initialzustand		

Das Kommunikationsmodell des A4.1-Service ist entsprechend der Verbindungen in **Abbildung 43** aufgebaut, wie folgt:

$Q\_ZS\_q_{N\_ZS}$:***get_num_q*** → P_{ZS_S}:***set_q_in***

$Q\_Streb\_q_{N\_S}$:***get_num_q*** → P_{S_AS}:***set_q_in***

$Q\_Streb\_q_0$:***get_num_q*** → P_{ZS_S}:***set_q_out***

$Q\_AS\_q_0$:***get_num_q*** → P_{S_AS}:***set_q_out***

P_{ZS_S}:***get_num_p*** → $Q\_S\_q_0$:***set_p_start***

P_{ZS_S}:***get_num_p*** → $Q\_ZS\_q_{N\_ZS}$:***set_p_end***

P_{S_AS}:***get_num_p*** → $Q\_AS\_q_0$:***set_p_start***

P_{S_AS}:***get_num_p*** → $Q\_Streb\_q_{N\_S}$:***set_p_end***

A4.2 – Aerodynamik in einer Abteilung mit Filtrationsraum

Das *A4.2*-Modell erweitert das *A4.1*-Basismodell der einfachen Bewetterungsabteilung um ein Q_{FR}-Modell des Filtrationsraums (*A2.5*), wie in **Abbildung 44** gezeigt.

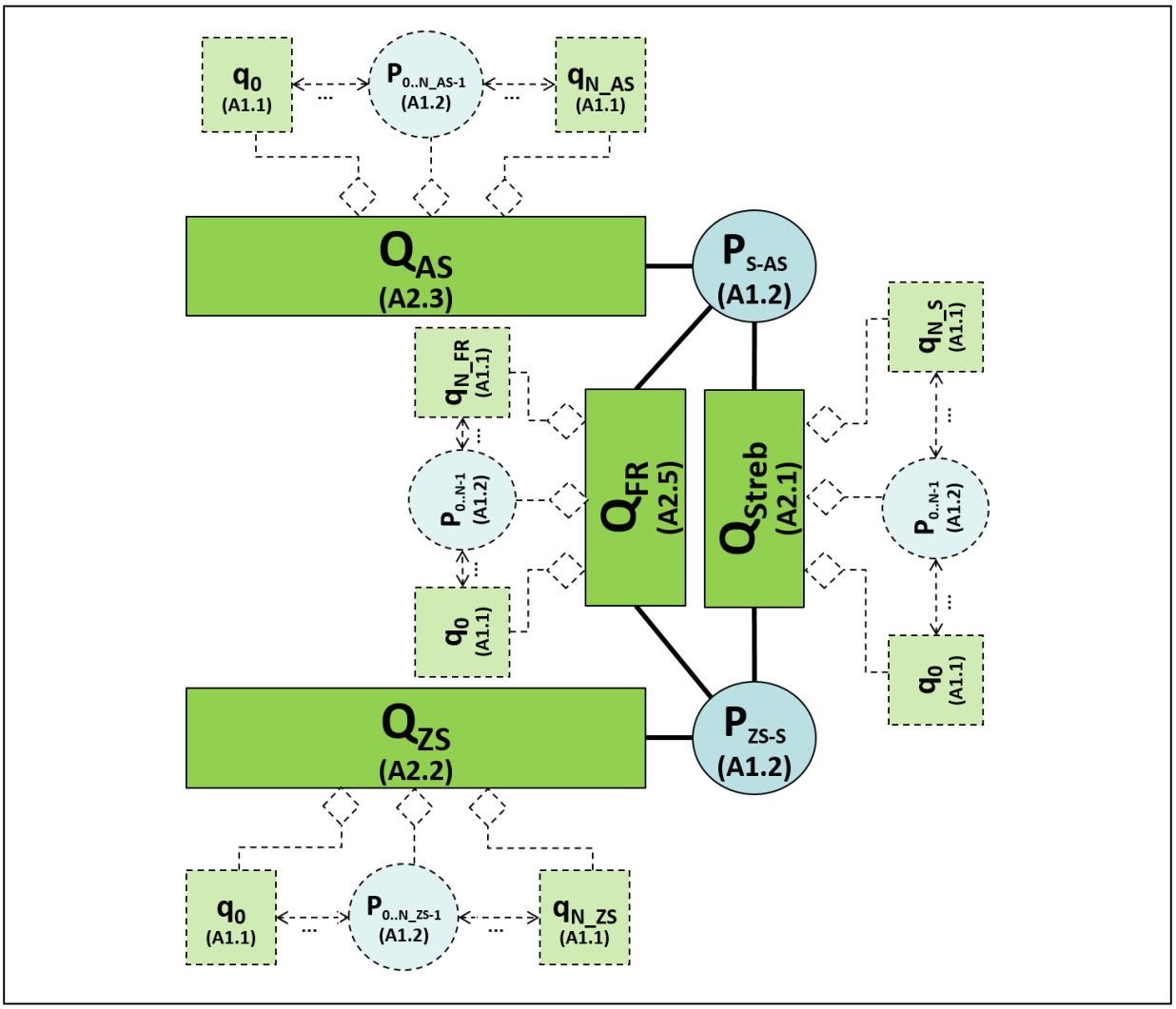


***Abbildung 44**. Kompositionsstruktur des Modells der Aerodynamik in einer Bewetterungsabteilung mit Filtrationsraum.*

Das Q_{FR}-Modell wird dabei parallel zum Streb an die Zufuhr- und Abfuhrstrecken angeschlossen, welches die folgende Erweiterung des Kommunikationsmodells des *A4.2*-Service benötigt:

Q_FR_q_{N_FR}:**get_num_q** → P_{S_AS}:**set_q_in**

Q_FR_q_0:**get_num_q** → P_{ZS_S}:**set_q_out**

P_{ZS_S}:**get_num_p** → Q_FR_q_0:**set_p_start**

P_{S_AS}:**get_num_p** → Q_FR_q_{N_FR}:**set_p_end**

G4.3 – Gasdynamik in einer Abteilung mit Filtrationsraum

Das *G4.3*-Modell ist nach einem ähnlichen Prinzip wie das A4.2-Modell aufgebaut, wobei die Modelle der Q_{AS}-Abfuhrstrecke und des Q_{FR}-Filtrationsraums durch ihre entsprechende Analogen ersetzt werden, die zusätzlich die Gasdynamik berücksichtigen, wie in **Abbildung 45** gezeigt ist. Also werden für die Abfuhrstrecke das Qmt_{AS}-Modell des Methantransports und für den Filtrationsraum das Qm_{FR}-Modell der Methanergiebigkeit im abgebauten Raum benutzt.

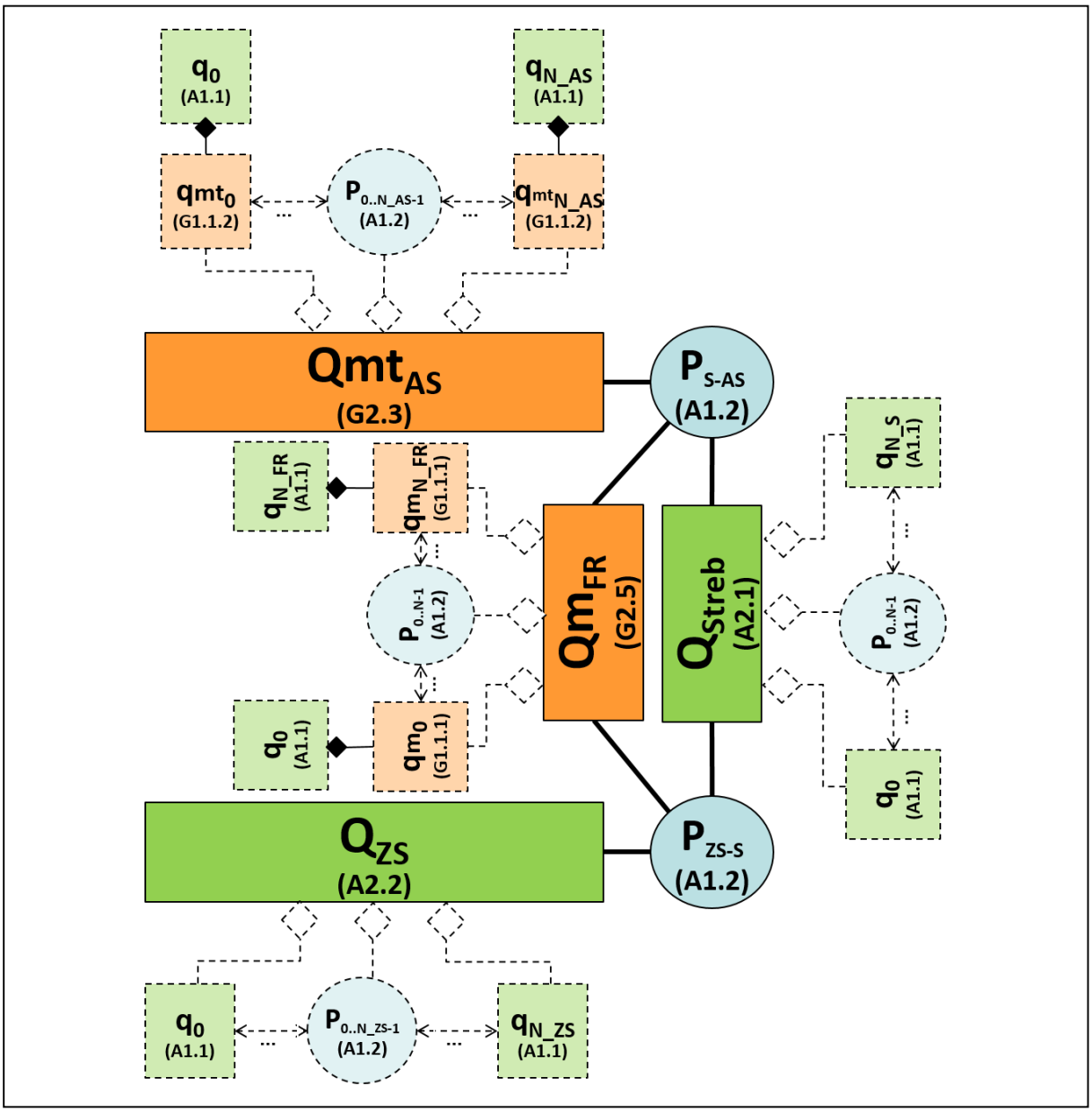


***Abbildung 45**. Kompositionsstruktur des Modells der Gasdynamik in einer Bewetterungsabteilung mit Filtrationsraum.*

Die Methantransport-Verbindungen zwischen den *Qm*- und *Qmt*-Modellen sollten im Kommunikationsmodell des *G4.2*-Service wie folgt abgebildet werden:

$$Qm\_FR\_qm_{N\_FR}:\mathbf{gas\_get\_qm} \rightarrow Qmt\_AS\_qmt_0:\mathbf{gas\_set\_qm}$$

Modelle des Netzes

Die Modelle der fünften Hierarchieebene (Netze) beschreiben die dynamischen Prozesse, die im gesamten Bewetterungsnetz ablaufen, und zwar die Aerodynamik (mithilfe des *A5.1*-Modells) und zusätzlich auch die Gas- (Methan-) Dynamik (mithilfe des *G5.1*-Modells). Die Netze können aus allen vorher beschriebenen Elementen (Strecken, Streben, Schächten usw.) bestehen. Die Struktur der Verbindungen zwischen den Elementen eines Bewetterungsnetzes ist im Wesentlichen irregulär und unterscheidet sich von einem Objekt zum anderen. Deswegen sollte ab dieser Hierarchieebene ein dynamisches Mittel zur Topologiedarstellung aufgegriffen werden.

Die bekannten Ansätze zur Topologiedarstellung der Bewetterungsnetze basieren auf gerichteten Graphen, in welchen die Zweige den Elementen entsprechen und die Knoten ihren Verbindungen. Die Parameter der Elemente (also ihrer Modelle) werden mitsamt ihrer Verbindungen über die Knoten (z.B. Ansatz von Smagin und Masjuk [R85]) eingetragen (ein Prozess, der auch als „Topologiekodierung“ bekannt ist), bspw. in tabellarischer Form, wie sie im Ausschnitt in der **Tabelle 14** aufgeführt ist.

Tabelle 14*. Kodierung der Netztopologie.*

Q	P_Start	P_End	H	R	L	S
1	1	2	242,6	0.24	500	7,0
2	2	3	0	81	130	2,3
…	…	…	…	…	…	…

Der Nachteil dieses Ansatzes ist, dass die Struktur der Parameter in der Kodierungstabelle immer konstant bleibt, sodass der Ansatz nur auf homogene Modelle angewandt werden kann. Sollten die Modelle z.B. eine lokale Regulierung benötigen (wie im Falle des *A3.1*-Modells) oder sogar Gasdynamik analysieren (wie im Falle des *G4.2*-Modells), muss die Parameterstruktur (also das Kodierungsschema) komplett umgeformt werden.

Vor diesem Hintergrund wird in der vorliegenden Arbeit ein Verfahren zur Parametererfassung und zur topologischen Kodierung der Zusammenhänge zwischen den Elementen ausgearbeitet, welches auf einem dynamischen Schema basiert, wie z.B. unterstützt von JSON[45] – einem nicht-auszeichnungssprachigen, nicht-relationalen Datenformat. Das JSON-Format wurde ursprünglich zum Zweck des Austausches von serialisierten Daten zwischen (JavaScript-) Webservices entwickelt, findet aber heutzutage einen verbreiteten Einsatz in vielen IT-Bereichen, insb. in Datenbanken. Die wesentliche und gleichzeitig attraktivste Eigenschaft (im Sinne des von dieser Arbeit angestrebten Ansatzes) von JSON ist die flexible Gestaltung des Datenschemas (Klassifikation der Attribute und ihrer Datentypen), was aber eine vorausgehende Festlegung des Datenschemas nicht erforderlich macht – die Datenstrukturdefinition kann flexibel und freiwählbar direkt in der JSON-Spezifikation erfolgen, sofern sie mit der Anwendungslogik übereinstimmt.

Die JSON-Spezifikation eines Basiselements des Bewetterungsnetzes (z.B. eines Strebs) kann in Form eines Listings wie in **Abbildung 46a** gestaltet werden. Auf ähnliche Weise können auch die Methaneigenschaften der gasdynamischen Modelle (wie z.B. eines Filtrationsraums) erfasst werden, wie **Abbildung 46b** zeigt.

Die Topologiekodierung kann dann nach demselben Prinzip wie bei Smagin und Masjuk [R85] erfolgen – mittels Eintragung der Inzidenzverbindungen zwischen den Elementen und Knoten, aber in Übereinstimmung mit der JSON-Spezifikation, wie es bspw. in **Abbildung 47** aufgezeigt ist.

Die Aufgaben der *A(G)5.x*-Services bestehen im Wesentlichen aus Folgendem:

- Der Analysierung der JSON-Spezifikation der Elemente.
- Der Instanziierung der entsprechenden Services und Initialisierung der Modelle mit den Parametern, die aus der JSON-Spezifikation entnommen werden.
- Der Erstellung des Kommunikationsmodells entsprechend der Topologiekodierung der Elemente.

[45] JSON – JavaScript Object Notation

```json
{
    "id": "Streb_1",
    "type": "A2.1",
    "descr": "Ost-Streb PL-S10",

    "properties": [{
        "aerodynamik": [{
            "S": 7.0,
            "R": 0.24,
            "L": 500.0
        }]
    }]
}
```

(a) Aerodynamik

```json
{
    "id": "FR_1",
    "type": "G2.5",
    "descr": "Filtrationsraum des Ost-Strebs PL-S10",

    "properties": [{
        "aerodynamik": [{
            "S": 2.3,
            "R": 81,
            "L": 130.0
        }],

        "methan": [{
            "V": 2800.0,
            "Qm0": 0.0175
        }]
    }]
}
```

(b) Gasdynamik

***Abbildung 46**. JSON-Spezifikation der Modellparameter.*

```json
    "topology": [{
        "start": "P0",
        "end": "P1"
    }]
```

***Abbildung 47**. JSON-Spezifikation topologischer Verbindungen.*

4.4 Echtzeitaspekte der Simulationsdurchführung

Die flexible Architektur der im vorigen Kapitel 4.3 erläuterten Services ermöglicht ihren Einsatz den vielfältigen dynamischen Anwendungsszenarien, die anhand des aktuellen Zustands der Modellierungsobjekte (erfasst durch Sensoren oder vorhergesagt durch vorige Simulationen) geplant, organisiert und eingeleitet werden können. Die Simulationen werden dadurch eine wichtige dynamische Begleitfunktion erfüllen können, indem sie gleichzeitig mit den zu analysierenden Prozessen gestartet werden und quasi parallel zu ihnen laufen können (was sich dem Konzept des Digitalen Zwillings nähert).

Unter der Voraussetzung einer korrekten Aufstellung (wird durch Faktoren wie einem umfangreichen physikalischen Modell, der Genauigkeit der numerischen Lösung usw. bestimmt) kann eine Simulation genaue Vorhersagen über den zeitlichen Ablauf des analysierten dynamischen Prozesses treffen. Die Anwendung eines Approximations-verfahrens und die Anpassung der variablen Solver-Eigenschaften einerseits sowie die hohe Rechenleistung moderner paralleler Hardware (auch im betrieblichen Umfeld) andererseits ermöglichen in vielen Fällen (wie z.B. von Menghal et al. in [R107] gezeigt) eine schnellere Durchführung der Simulation als sie die tatsächliche Dauer des physikalischen Übergangsprozesses benötigt. Dabei wird in der Simulation eine eigene Zeitdefinition verwendet, die sich von der tatsächlichen Zeit des physika-lischen Objekts unterscheidet.

Darüber hinaus, wird weder die Berechnungsdauer noch die aktuelle (zum Moment des Startes der Simulation) Zeit bei den bekannten Simulationstechnologien berücksichtigt. Vor diesem Hintergrund lässt sich die Tatsache unterstreichen, dass die “offline“-Simulation zur Begleitung der laufenden industriellen Objekte eher ungeeignet ist. Eine “online“-Simulation sollte mit einem realen Zeitlauf verbunden werden, also sollten die Simulationsergebnisse in einer Relation zur Echtzeit aufgefasst und präsentiert werden.

Vor diesem Hintergrund sollte grundsätzlich zwischen den folgenden Zeitkategorien unterschieden werden (siehe Illustration in **Abbildung 48**):

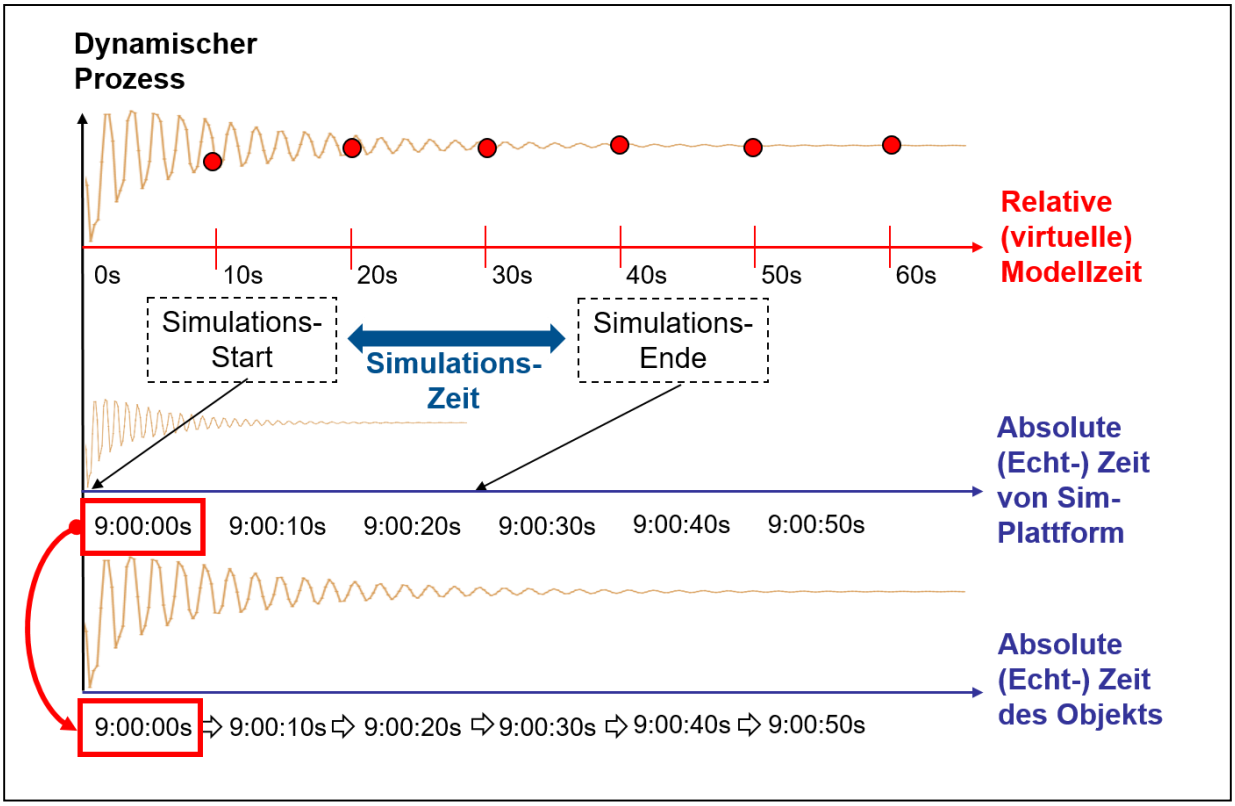


***Abbildung 48**. Zeitablauf in Simulationsexperimenten.*

- Der kontinuierlichen absoluten physikalischen Zeit des dynamischen Prozesses.
- Der diskreten und relativen (zum Diskretisierungsverfahren) virtuellen Modellzeit.
- Der Zeit, die für die Ausführung des Simulationsexperiments notwendig ist.

Als „Echtzeit"-Simulation definiert Wikipedia [W33] einen Simulationsvorgang, dessen Ausführung auf einer Hardwareplattform genauso so viel Zeit benötigt wie der simulierte tatsächliche Prozess. Im Zusammenhang mit dem Forschungsvorhaben der vorgelegten Arbeit (dynamische Simulation für industrielle Objekte) wird diese Basisdefinition auf folgende Weise erweitert:

Definition 2: Als **Echtzeit-** kann ein solcher **Simulationsvorgang** bezeichnet werden, bei welchem die Bereitstellung der Simulationsergebnisse **vor** dem Moment der Erreichung vom analysierten dynamischen System des nächsten vordefinierten oder messbaren Zustands bzw. Operationsmodus unter Einhaltung der Funktionalanforderungen garantiert werden kann.

beschrieben) gehen mit den beiden Aspekten durch Anpassung der Simulationsdauer an die tatsächliche Dauer des physikalischen Prozesses um, z.B. durch Variierung der numerischen Schrittgröße (z.B. Euler-Verfahren), wie von der Spezifik ihrer Anwendung (z.B. Projektierung und Testen der Kontrollsysteme) notwendig ist (siehe einen Vergleich verschiedener Ansätze in **Tabelle 15**).

***Tabelle 15**. Eigenschaften der Simulationsvorgänge.*

Typ / *Eigenschaft*	**Konventioneller**	**Echtzeit**	**Im Rahmen dieser Arbeit**
Typ der math. Modelle	PDG höherer Ordnungen	GDG erster Ordnung	PDG erster Ordnung
Dauer des dynamischen Prozesses	Minuten	Sekunden bis Minuten	Minuten bis Stunden
Dauer der Simulation	Minuten bis Stunden	Sekunden bis Minuten	Sekunden bis Minuten
Schrittart des num. Verfahrens	Adaptiver	Fixierter	Fixierter und eventuell adaptiver
Algorithmische Komplexität	Hoch (Tera- bis Giga-FLOPs)	Niedrig (GigaFLOPs)	Mittel (GigaFLOPs)
Objektgröße	Groß	Klein	Mittel
Zeitliche Vorhersehbarkeit	Gesamtsimulation	Jeder numerische Lösungsschritt	Gesamtsimulation
Typische numerische Löser	OpenFOAM (z.B. pisoFoam simpleFoam), Matlab (ode45), …	Euler	Runge-Kutta, Adams-Baschfort

Dabei muss eine adäquate zeitliche Übereinstimmung der Simulationsergebnisse mit den Referenzwerten der Ausführungsplattform gewährleistet werden.

Im Gegensatz zu den vorhandenen Echtzeitsimulationsplattformen, ist für die Aufgabenstellung der vorliegenden Arbeit eine schnellstmögliche Ausführung der Simulationsalgorithmen notwendig, z.B. bei sicherheitskritischen Anwendungs-szenarien oder auch zur Senkung des Energieverbrauchs der Simulationshardware. Die Simulationsergebnisse sollten jedoch eindeutig in der absoluten Objektzeit identifizierbar sein. Zu diesem Zweck wird das folgende Verfahren vorgeschlagen:

1. Zum Start der Simulation wird die Objektzeit (z.B. durch eine Schnittstelle des Echtzeitbetriebssystems) erfasst.

2. Alle zu speichernden Simulationsergebnisse werden zusätzlich durch die Einführung eines Zeitstempels identifiziert, dessen Wert wie folgt berechnet wird:

$$T = T_{start} + \tau', \tag{IV.22}$$

wobei T_{start} – die absolute Anfangszeit des Simulationsexperiments, τ' – *die* aktuelle virtuelle Modellzeit ist.

3. Am Ende der Simulation wird die Objektzeit erneut erfasst und, zusammen mit der Startzeit des Experiments, an die Experimentplanungskomponente des Ausführungsframeworks übermittelt. Im Falle der kontinuierlichen Ausführung (Start des neuen Experiments woraufhin die Daten des vorigen Experiments ihre Aktualität verloren haben) kann das Framework den Zeitpunkt der Initiierung des nachfolgenden Experiments ermitteln, z.B. nach der folgenden Formel:

$$T = \tau_{end} - (T_{end} - T_{start} - \Delta T), \tag{IV.22}$$

wobei τ_{end} – der Zeitstempel des letzten gültigen ErgebnisWertes, T_{end} – die absolute Zeit des Simulationsendes, ($T_{end} - T_{start}$) – die Dauer der Simulation in absoluter Zeit, ΔT – der Planungspuffer ist.

Das vorgeschlagene Verfahren sollte die Datenerfassung und die Planung der Simulationsexperimente in Echtzeit (bzw. Near-Echtzeit) ermöglichen, mit den Vorteilen einer erhöhten Leistung und Energieeffizienz gegenüber den bestehenden Echtzeit-Ansätzen. Der Overhead der Umrechnung der virtuellen Modellzeit in die absolute Objektzeit sollte noch zusätzlich evaluiert werden.

4.5 Datenspeicherung und Analyse

Die Speicherung der Simulationsergebnisse ist eine wichtige Anforderung an CFD-Simulationssoftware. Diese Daten können dann von den Steuerungskomponenten der CPS-Systeme oder weiterhin von anderen Services analysiert bzw. verarbeitet werden. In typischen CFD-Simulationen werden die Ergebnisse in regelmäßigen Zeitabständen gespeichert, die von der Anwendungslogik bestimmt werden können.

Die Simulatoren verfügen über eine gewisse Autonomie, was die Ergebnisspeicherung angeht. Es können dafür verbreitete Datenformate wie *HDF5*, *NetCDF* aber auch klassisches *CSV* (siehe **Abbildung 49**) verwendet werden. Die Zeitreferenzen sollten für alle zu speichernden Werte in der Regel mitgespeichert werden.

t	Q0	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9
0.01	0.271712	0	0	0	0	0	0	0	0	0
0.02	0.542238	0.00118358	0	0	0	0	0	0	0	0
0.03	0.810402	0.00471883	5.15566e-06	0	0	0	0	0	0	0
0.04	1.07505	0.0117431	3.08217e-05	2.24581e-08	0	0	0	0	0	0
0.05	1.33505	0.0233481	0.000107372	1.7898e-07	9.78273e-11	0	0	0	0	0
0.06	1.58931	0.0405656	0.000284693	8.01656e-07	9.74437e-10	4.26136e-13	0	0	0	0
0.07	1.83682	0.064354	0.00063624	2.65747e-06	5.33457e-09	5.09321e-12	1.85625e-15	0	0	0
0.08	2.07658	0.0955856	0.00126258	7.26162e-06	2.12242e-08	3.29533e-11	2.58823e-14	8.08581e-18	0	0
0.09	2.30768	0.135035	0.00229433	1.73024e-05	6.85607e-08	1.52979e-10	1.93228e-13	1.28845e-16	3.52218e-20	0
0.1	2.52929	0.183371	0.00389437	3.71868e-05	1.9067e-07	5.70323e-10	1.02527e-12	1.09018e-15	6.31384e-19	1.53426e-22
0.11	2.74066	0.241143	0.00625942	7.3712e-05	4.73106e-07	1.81326e-09	4.33271e-12	6.50808e-15	5.97087e-18	3.05498e-21
0.12	2.94111	0.308779	0.00962067	0.000136863	1.07252e-06	5.10127e-09	1.5501e-11	3.07426e-14	3.96076e-17	3.19223e-20
0.13	3.13006	0.386577	0.0142437	0.000240734	2.25878e-06	1.30168e-08	4.87555e-11	1.22232e-13	2.06814e-16	2.32868e-19
0.14	3.30702	0.474703	0.0204277	0.000404563	4.47407e-06	3.06584e-08	1.38287e-10	4.25036e-13	9.04663e-16	1.3314e-18
0.15	3.47163	0.573185	0.0285032	0.000653871	8.41278e-06	6.75225e-08	3.60163e-10	1.32652e-12	3.4461e-15	6.35182e-18
0.16	3.62357	0.681918	0.0388301	0.00102168	1.51268e-05	1.40446e-07	8.73036e-10	3.78533e-12	1.17359e-14	2.62938e-17
0.17	3.76267	0.800659	0.0517935	0.00154979	2.616e-05	2.78042e-07	1.9901e-09	1.00142e-11	3.64126e-14	9.69869e-17
0.18	3.88885	0.929036	0.0678001	0.00229014	4.37174e-05	5.27177e-07	4.30103e-09	2.48248e-11	1.04395e-13	3.24929e-16
0.19	4.0021	1.06655	0.0872729	0.00330605	7.08721e-05	9.62172e-07	8.87097e-09	5.81549e-11	2.79607e-13	1.00305e-15
0.2	4.10254	1.21258	0.110645	0.00467364	0.000111815	1.69754e-06	1.75551e-08	1.29621e-10	7.05716e-13	2.8851e-15
0.21	4.19037	1.36638	0.138356	0.00648296	0.000172149	2.90526e-06	3.34813e-08	2.76432e-10	1.69034e-12	7.80092e-15

***Abbildung 49**. Speicherung der Simulationsergebnisse (Luftdurchsatz) in einer CSV-Datei.*

Die Autonomie der Services, die auf verteilten Rechenressourcen ausgeführt werden können, stellt gleichzeitig die größte Herausforderung zur Konsolidierung der Daten dar. Falls eine dezentralisierte (also lokale oder „vor-Ort“) Verarbeitung aus irgend-einem Grund unmöglich/unzweckmäßig ist, können die Daten auf folgenden Wegen konsolidiert werden:

(1) Mittels zentraler Datenverarbeitung durch einen Master-Service. Die Daten aller eingegliederten Services werden vom Master-Service mithilfe von Kommunikations-Schnittstellen der Service-API oder alternativ mithilfe spezieller Bibliotheken während der Service-Laufzeit (im Fall, dass Online-Datenverarbeitung erforderlich ist) oder auch im Anschluss an das Simulationsende gesammelt (falls die Verarbeitung bis zum Ende der Simulation warten kann).

(2) Im Falle der Verfügbarkeit eines parallelen Netzwerk-Filesystems (wie z.B. *NFS*, *Lustre*) können spezielle Bibliotheken für paralleles I/O (wie z.B. *MPI-IO*) verwendet werden.

Zusätzlich zu den oben aufgeführten Optionen wird im Rahmen dieser Arbeit eine weitere Datenspeicherungsmöglichkeit evaluiert:

(3) **Die Nutzung verteilter Datenbankmanagementsysteme**. Datenbanken bzw. Datenbankmanagementsysteme (DBMS) sind ein effektives Instrument zur Verwaltung großer Datenmengen. Aufgrund ihrer Spezifik, wie komplexer relationaler Abhängigkeit, hohem Grad der Homogenität usw., fand die erste Generation der Datenbanken traditionell keinen breiten Nutzen in der Simulationstechnik. Im Gegensatz zu ihnen, weisen die modernen Datenbanken praktisch alle Eigenschaften der Simulationsservices auf – Autonomie, Heterogenität und Verteilung (siehe Kapitel 3.1). Die neue Generation verteilter Datenbanken wird nach dem nicht-relationalen Ansatz aufgebaut, zeichnet sich durch einen hohen Datenspeicherung- und Verarbeitungsdurchsatz aus und hat durchaus ein Anwendungspotenzial bei den Mikroservicearchitekturen.

Unter industriellen Bedingungen können die Voraussetzungen für Datenverarbeitung (Konfiguration der Hardware, Größe des Speichers usw.) stark abweichen, sodass bei der Realisation der Simulationsservices sowohl dezentrale als auch zentralisierte Datenspeicherung in Anspruch genommen werden kann. Damit die Simulationsservices jedoch ihre Universalität beibehalten und gleichzeitig eine hohe Effizienz der Datenspeicherung gewährleisten können, wurde im Rahmen der vorliegenden Arbeit eine Lösung erarbeitet, nach welcher jeder Service für die Ausgabe seiner Daten selbst verantwortlich ist. Der bevorzugte Weg zur Datenspeicherung, inkl. eines spezifischen End-Points (Speicherungsorts), Eigenschaften der Datenübermittlung an die Speicherinfrastruktur usw., wird jedoch vom Master (einem übergeordneten Service oder einer End-Anwendung) bestimmt, worüber die Services über ein spezielles API informiert werden (siehe **Abbildung 50**).

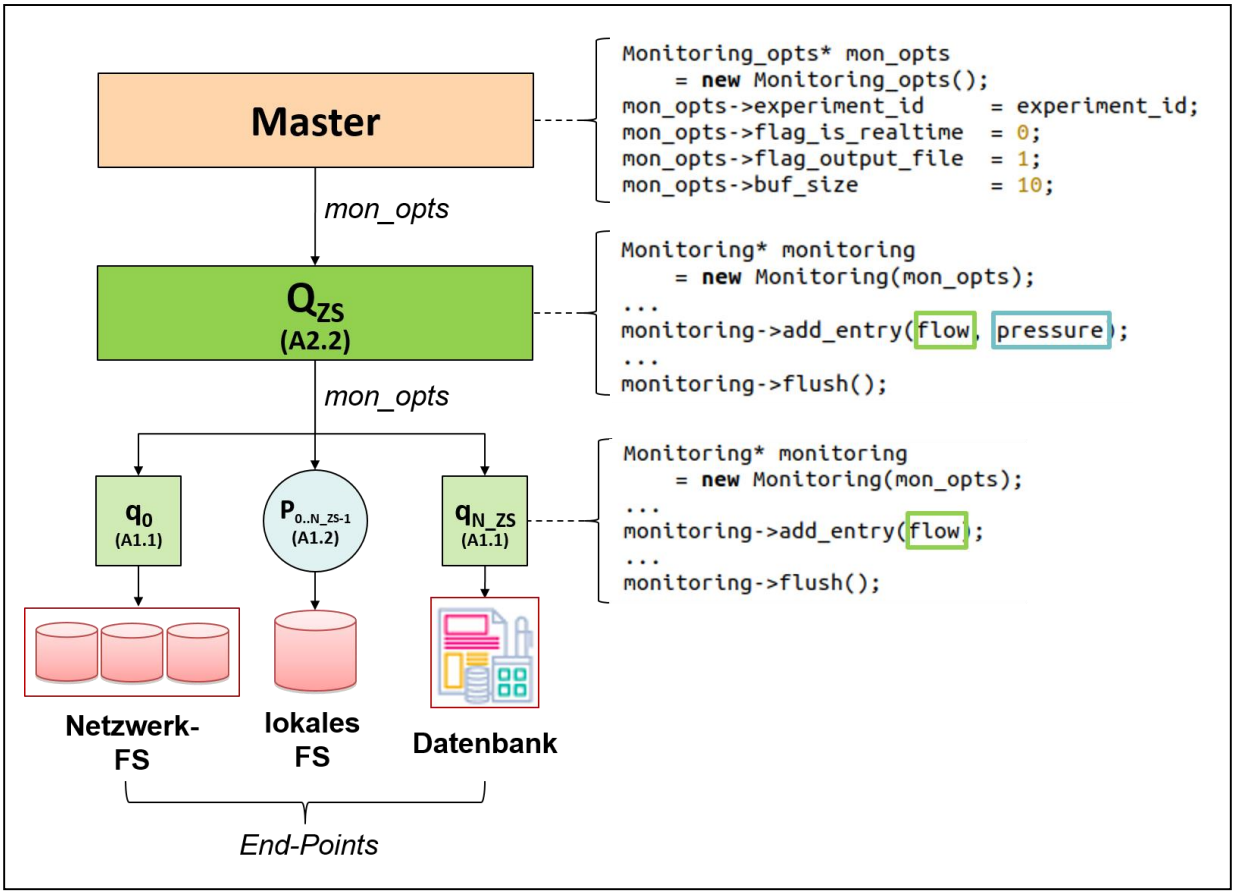


***Abbildung 50**. Datenspeicherung in hierarchischen Servicearchitekturen.*

Die anwendungs- (service-) spezifische Datenspeicherung kommt in ihrer Funktionalität dem des Monitoring-Frameworks sehr nahe, mit dem einzigen Unterschied, dass das Letztere die infrastruktur-spezifische Daten speichert (die vom Monitoring-Client auf der System-Ebene ermittelt werden, wie im Kapitel 3.4 erläutert). Alle sonstigen Funktionen und Technologien bleiben aber im Wesentlichen unverändert. Aus diesem Grund wurde in der vorliegenden Arbeit ein Verfahren entwickelt, welches die beiden Funktionalitäten innerhalb des einheitlichen Monitoring-Frameworks vereinbaren lässt.

Mithilfe des Monitoring-APIs können wichtige Eigenschaften der Datenspeicherung spezifiziert werden, wie bspw. ob die Ausgabe in eine Datei auf einem lokalen oder netzwerk-basiertem Filesystem oder in eine Datenbank erfolgen soll, ob die Ergebnisse mit ihren entsprechenden absoluten (Echt-) oder relativen (Modell-) Zeit-stempeln versehen werden sollen, ob die Speicherung in Blöcken (Buffer) von einer bestimmten Größe erfolgen soll u.v.m. Die *MS-Monitoring-API* bietet den Mikro-services eine

transparente, unabhängige von der darunterliegenden Infrastruktur Funktionalität zur Speicherung der service-spezifischen Daten an.

Damit die verschiedenen Simulationsabläufe in der Analysephase voneinander unterschieden werden können, wird vor dem Start jeder Simulation ein unikaler Identifikator (*Experiment-ID*) generiert und bei allen zu speichernden Daten dem Monitoring-API mitgegeben. Somit können die zusammengehörenden Ergebnisse im Nachhinein unter der Angabe der IDs herausgefiltert werden.

Die "*add_entry*"-Funktion des *MS-Monitoring-API* wird zur Speicherung der Datenwerte (z.B. des Luftdurchsatzes für die Services diskreter Elemente oder des Drucks für die Knoten-Services) verwendet. Je nachdem wie der Buffer angelegt ist, werden die Ergebnisse entweder sofort oder erst bei der Erreichung des Buffer der bestimmten Größe an die Speicher-Infrastruktur übergeben. In dem Fall dass ein Buffer angewendet wird und die Simulation vor dem Erreichen seiner maximalen Größe beendet wird, werden die restlichen Daten mithilfe der "*Flush"*-Funktion endgültig gespeichert und die vorübergehenden Buffer geleert.

Als DBMS-Lösung wurde die heutzutage weit verbreitete *ElasticSearch*[46] (ES)-Lösung evaluiert, wie z.B. von Gheorghe in [R108] beschrieben. ES ist eine nicht-relationale Datenbank, die durch ihre Eigenschaften wie Plattformunabhängigkeit, Verteilung, Hochverfügbarkeit sowie aufgrund ihres geringeren Overheads der Software-komponente für Systemsoftware einen breiten Nutzen in verschiedenen Bereichen gefunden hat, sogar für eingebettete Systeme (wie z.B. aus dem Paper [R109] von Li et. al mit einem Erfahrungsbericht über das PHANTOM [47] -Projekt folgt). Die Kommunikation zwischen einem Client (in unserem Fall – dem datenspeichernden Mikroservice) und dem ES-Server (einem dedizierten Knoten der verteilten Infra-struktur) erfolgt über REST (einem http-basierten Datenübertragungsprotokoll der Web-Anwendungen) mithilfe des JSON-Formats (welches zuvor im Kapitel 4.3.6 im Kontext der Darstellung der Netz-Topologie diskutiert wurde). Die Aufgabe des Monitoring-API im Fall des Nutzens von ES als Datenspeicherung-Endpoints besteht in

[46] https://www.elastic.co/elasticsearch
[47] https://www.phantom-project.org/

der Erzeugung einer JSON-Spezifikation für den zu speichernden Datenwert (siehe Listing in**Abbildung 51**) und deren Übermittlung an den ES-Server (mithilfe von Eingaben in die Monitoring-Konfigurationseigenschaften). Ein großer Vorteil der Nutzung des JSON-Datenformats ist seine Flexibilität im Umgang mit beliebigen (Meta-) Daten, die vom Mikroservice zusätzlich zu dem tatsächlich zu speichernden Wert hinzugefügt werden können, um z.B. die nachfolgende Datenverarbeitung zu vereinfache bzw. zu optimieren.

```
{
 "ExperimentID":"2020-02-18T15:08:26.890",
 "Network":"Network1","Section":"Section1","Element":"ZS",
 "@timestamp":"2020-02-18T15:09:51.005",
 "q":1.60045
}
```

***Abbildung 51**. JSON-Spezifikation von zu speichernden Daten.*

Die Daten, die auf dem *Elasticsearch*-Server gespeichert wurden, stehen zur sofortigen Verfügbarkeit (also – in „Near-Echtzeit“) für jegliche Analyse- und Nachverarbeitungs-services, die z.B. Visualisierung der Daten durchführen. Die Daten werden dabei mittels desselben JSON-Formats abgefragt, das auch bei der Speicherung verwendet wird. Das Ökosystem der ES-Tools bietet zudem viele Visualisierungsclienten mit üppiger Funktionalität an, wie z.B. das populäre *Kibana*[48] (**Abbildung 52**).

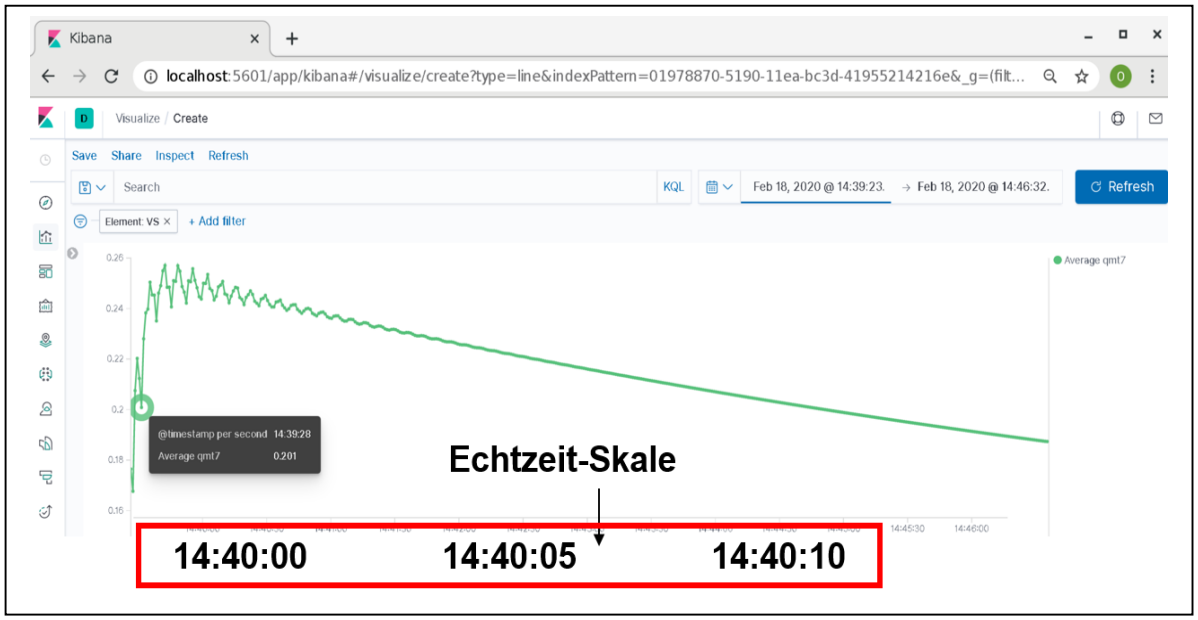


Abbildung 52. Echtzeit-Datenvisualisierung mit Kibana-Tool.

[48] https://www.elastic.co/kibana

4.6 Zusammenfassung der Ergebnisse

Das primäre Ziel dieses Kapitelinhalts war die Elaboration des Anwendungspotenzials eines Mikroservice-Ansatzes zur Lösung der Simulationsaufgaben des Basis-Forschungsgebiets „Grubenbewetterung“. Die wichtigsten erzielten Ergebnisse sind:

- Die wichtigsten theoretischen Aspekte zur Erstellung der mathematischen Modelle der Aero- und Gasdynamik wurden gründlich analysiert und ausführlich beschrieben, mit dem Ziel, eine **einheitliche Methodologie zur Implementierung der Modellierungssoftware** zu entwickeln.
- Die technologischen Objekte des Basisgebiets wurden mithilfe des hierarchischen Ansatzes klassifiziert und **anhand der Klassifikation wurde eine Modellhierarchie erstellt**. Die Hierarchie umfasst zwei horizontale Modellbereiche (Aero- und Gas-Dynamik) sowie fünf vertikale Zusammensetzungs- (Schachtelungs-) Ebenen von Modellen (von Diskretisierungselementen bis hin zum Netzwerk).
- Anhand der Modellhierarchie wurden die beiden wichtigen **Implementierungsstrategien für die Simulationsservices der hierarchisch-zugeordneten Modelle** erörtert – Erweiterung (beruht auf Vererbung) und Komposition (basiert auf Eingliederung). Zusätzlich wurde zum ersten Mal eine weitere Strategie der Servicezusammensetzung ausgearbeitet – **die Injektion**, die die Einbettung eines Service in einen anderen Service ohne Verlust der Eigenschaft der funktionellen Autonomie von dem Ersteren ermöglicht. Die Injektionstechnik hat sich im Nachhinein als besonders wichtig zur Implementierung der Modelle über verschiedene Bereiche hinweg erwiesen.
- Zur **Entwicklung der Simulationsservices** wurde eine **umfangreiche Methodologie** erarbeitet, welche die verschiedenen Aspekte des Entwicklungsprozesses vom Entwurf der Anwendungslogik über den Aufbau der Simulationsfunktionalität einzelner Services und die Projektierung des Kommunikationsmodells zur Kopplung mit den anderen Services innerhalb der

hierarchischen Organisation bis hin zur Entwicklung der komplexen Anwendungen beinhaltet, die aus mehreren zusammengesetzten Services aufgebaut sind. Dabei wurden **Vorschläge zur Erweiterung herkömmlicher Software-Entwicklungsmethoden** (wie z.B. eine Erweiterung der UML-Sequenzdiagramme oder die Nutzung des JSON-Formats zur Topologiebeschreibung) gegeben, welche die spezifischen Anforderungen des service-orientierten Entwicklungsansatzes besser und präziser erfüllen können.

- Die ausgearbeitete Methodologie wurde erfolgreich zum **Entwerfen der Simulationsservices** für die wichtigsten Modelle der Modellhierarchie des Forschungsgebiets „Grubenbewetterung“ eingesetzt: Modelle des Strebs, der Strecken, der Schächte, der Filtrationsräume, der Abteilungen und letztendlich des gesamten Bewetterungsnetzes. Die Services wurden als Komponente des freizugänglichen Simulationsframeworks MS-Sim entwickelt und stehen anderen Forscherinnen und Forschern zur Nutzung und ggf. Erweiterung auf der *MS-Github-Plattform* (https://github.com/alexey-cheptsov/MS-Sim).
- Es wurde ein innovativer Ansatz zur **persistenten Simulationsdatenspeicherung mittels verteilten nicht-relationalen Datenbankmanagementsystemen** erarbeitet und die notwendigen Grundlagen zu seiner Implementierung mittels der populären Elasticsearch-Lösung ausgearbeitet.

5 Validierung und Anwendungen

In diesem Kapitel werden funktionale und nichtfunktionale Eigenschaften der implementierten Modelle anhand vorhandener Datensätze realer Objekte bewertet und validiert sowie die wichtigsten Anwendungsszenarien für die entwickelten Simulationsservices präsentiert.

Der weitere Inhalt dieses Kapitels beschreibt zuerst die Test-Szenarien und präsentiert die Ergebnisse der Validierung der entwickelten Modellen und Simulationsservices (Kapitel 5.1). Danach werden die wichtigsten Anwendungsfälle der realen Objekte des Basisforschungsgebiets erarbeitet und eine Nutzungsstrategien des entwickelten Simulationsframeworks für sie dargelegt (Kapitel 5.2). Weitere potenzielle Ansatzgebiete und ihre Anwendungsszenarien für den Microservice-basierten Simulationsansatz werden in Kapitel 5.3 erläutert.

5.1 Validierung der Modelle und Simulationsservices

Die Strategie der Validierung der Modelle, die in Form der Simulationsservices nach der im Kapitel 4 erläuterten Strategie entwickelt wurden, umfasst die folgenden drei Etappen:

- In der ersten Etappe werden die service-basierten Modellimplementierungen anhand der Test-Anwendungsszenarien verifiziert, die für die Validierung der Basismodelle verwendet wurden. Dabei müssen insbesondere die funktionalen Anforderungen beachtet werden, wie z.B. die Genauigkeit der Simulationsergebnisse im Vergleich mit den bereits vorhandenen Muster-Experimenten von Bewetterungsabteilungen der Bergwerke.
- In der zweiten Etappe wird eine Möglichkeit zur Nutzung der Simulationsservices für die Unterstützung der realen Objekte validiert, in dem die Simulationsergebnisse mit den aktuellen Vermessungsdaten verglichen werden. Dieser Schritt sieht eine aktive Involvierung der Experte des Basisgebiets vor.
- In der letzten Etappe werden die nichtfunktionalen Eigenschaften evaluiert, wie z.B. die Echtzeitfähigkeit der Simulationsalgorithmen, was maßgeblich von der

Simulations-, Speicherungs- und Reaktions-Dauer auf der vorhandenen Rechen- und Datenverwaltungs-Infrastruktur sowie von Implementierungs-aspekten der Simulationsservices abhängig ist. Die Korrektheit der Darstellung des zeitlichen Ablaufes der Übergangsprozesse in Visualisierungsinterfaces wird durch die Gegenstandsgebiet-Experte validiert. Insbesondere wird auf die praktischen Aspekte der Nutzung der verteilten Datenbanken für die persistente Ergebnisspeicherung geachtet.

Validierung funktionaler Anforderungen

Die funktionalen Anforderungen (Korrektheit und Genauigkeit der Simulations-ergebnisse) wurden für die folgenden Testobjekte (beide mit Standort in der Ukraine) validiert:

(1) **TO-1**: Bergwerk „Vostochnaja“ (Abteilung des 3. öst. Streb)

(2) **TO-2**: Bergwerk „Süd-Donbass 3“ (Abteilung des 1. Oststreb „*plS11*“ und des 7. Weststreb „*plS10-2*“).

Das erste Testobjekt (Bergwerk „Vostochnaja“) wurde, unter anderem, von Svjatnjy in [R8] als Referenzobjekt für die Validierung seiner und Feldmanns ausgearbeiteter Modelle benutzt. Die durch diese Autoren durchgeführte Validierung ergab eine sehr hohe Qualität ihrer Modelle – die maximale Abweichung von den Messwerten betrug nicht mehr als 6,5%. Damit haben sich die Modelle als praktisch anwendbar auf die gegebene Problematik der Analyse von aerogasdynamischen Prozessen im Bergbau erwiesen. Das Hauptziel der in dieser Arbeit durchgeführten Verifizierung war es festzustellen, ob die Implementierung der Modelle mittels Services (mit ggf. von den Originalexperimenten abweichenden Diskretisierungs- und numerischen Eigenschaften) identische Ergebnisse liefern kann.

Der Service der Testabteilung (*Abteilung des 3. öst. Strebs*) des ersten Testobjekts ist nach der klassischen Struktur des A4.2-Modells (Zufuhrstrecke mit Leckströmung→Streb→Filtrationsmedium→Abfuhrstrecke) aufgebaut, siehe Kapitel 4.3.5. Die Parameter aller Bestandteile der Testabteilung sind in **Tabelle 16** erfasst.

***Tabelle 16**. Aerodynamische Parameter des Testobjekts TO-1.*

Parameter	Zufuhr-Strecke (ZS)	Streb	Filtr. Medium (FR)	Abfuhr-Strecke (AS)
Länge **L** *[m]*	500	130	130	500
• *Anzahl diskreter Elemente*	*5*	*3*	*3*	*5*
• *Anzahl diskreter Knoten*	*4*	*2*	*2*	*4*
Querschnittfläche **S** $[m^2]$	7,0	2,3	2,3	5,5
Widerstand **R** $[Ns^2/m^8]$	0,24	4	49	0,44

Das Modellexperiment wurde für die drei folgenden Szenarien durchgeführt (siehe **Tabelle 17**): ***(A)*** Anstieg der Ventilator-Depression, ***(B)*** Anwendung der Regelwirkung des lokalen Regulators in der Abfuhrstrecke, ***(C)*** Senkung der Regelwirkung des Reglers.

***Tabelle 17**. Ablauf und Ergebnisse der Aerodynamik-Modellexperimente für Testobjekt TO-1.*

Parameter	Experiment A		Experiment B		Experiment C	
	Anfang	Ende	Anfang	Ende	Anfang	Ende
Regulierte Parameter						
Regler-Widerstand **r´** $[Ns^2/m^8]$	0	0	0	**25**	25	**5**
Depression **ΔP** $[N/m^2]$	0	**160.69**	160.69	160.69	160.69	160.69
Basiswerte						
Luftdurchsatz **Q** $[m^3/s]$						
- ZS	0	7.2	7.2	2.38	2.38	4.45
- Streb	0	5.6	5.6	1.86	1.86	3.45
- FR	0	1.6	1.6	0.52	0.52	1.00
- AS	0	7.2	7.2	2.38	2.38	4.45
Modellergebnisse						
Luftdurchsatz **Q** $[m^3/s]$						
- ZS	0	7.199	7.199	2.39	2.39	4.45
- Streb	0	5.599	5.599	1.86	1.86	3.46
- FR	0	1.599	1.599	0.53	0.53	0.99
- AS	0	7.199	7.199	2.39	2.39	4.45

Der zeitliche Verlauf entsprechender aerodynamischer Prozesse ist in **Abbildung 53** aufgezeigt. Alle Tests wurden mit der Zeitschrittweite *h=0.15 s.* und der örtlichen Diskretisierung von *ΔL=100m.* durchgeführt.

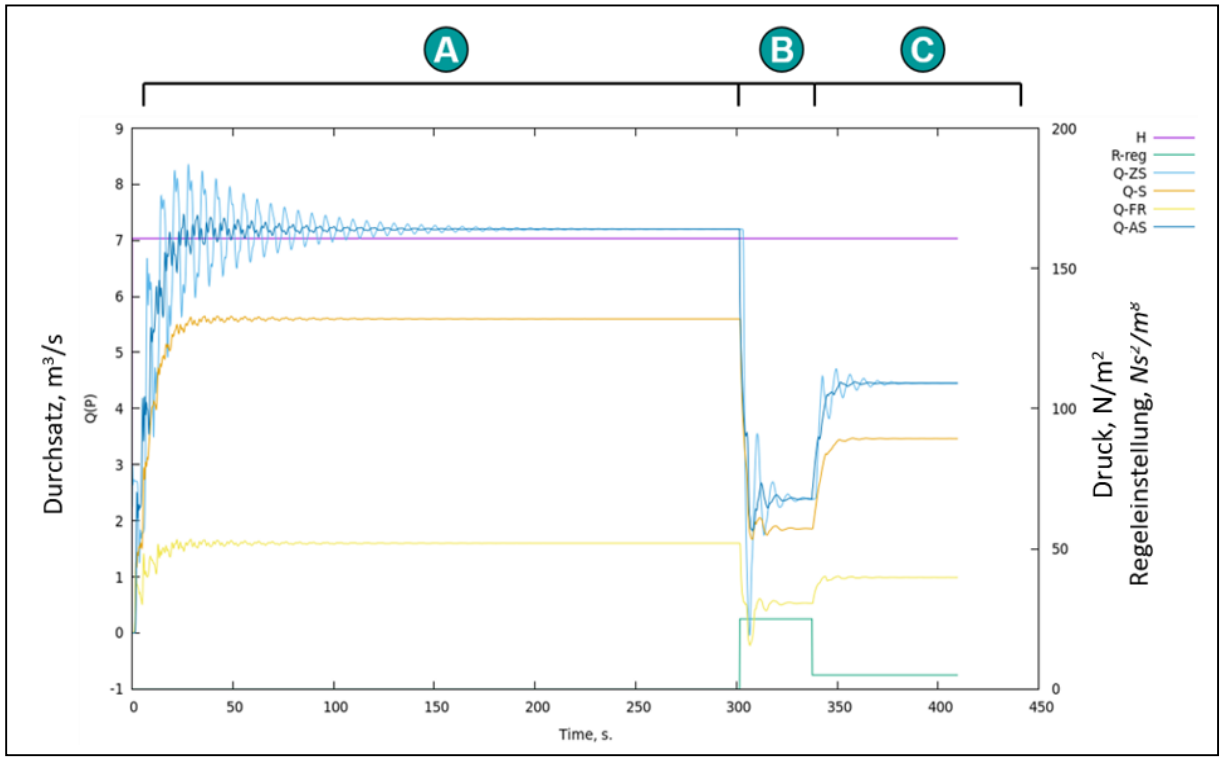


***Abbildung 53**. Modellergebnisse für aerodynamische Prozesse im Testobjekt TO-1.*

Die Steuerung der Simulationsexperimente erfolgte mittels einer Master-Slave-Architektur, wobei die eingesetzten *A/G[i.j]*–Simulationsservices (siehe Kapitel 4.2) die Slave-Rolle erfüllt haben, wie in **Abbildung 54** mit Hilfe eines vereinfachten UML-Diagramms gezeigt ist. Die Validierung der Gasdynamik-Modelle wurde nach demselben Prinzip durchgeführt, wie für die oben beschriebenen Aerodynamik-experimente, aber mithilfe des G4.2-Modells Die gasdynamischen Parameter des Testobjekts 1 sind in **Tabelle 18** aufgeführt.

***Tabelle 18**. Gasdynamische Parameter des Testobjekts TO-1.*

Parameter	Filtr. Medium (FR)	Streb
Volumen der Filtration **V** *[m³]*	8000	-
Methanstrom **Qm** *[m³/s]*	0.0175	0.0025

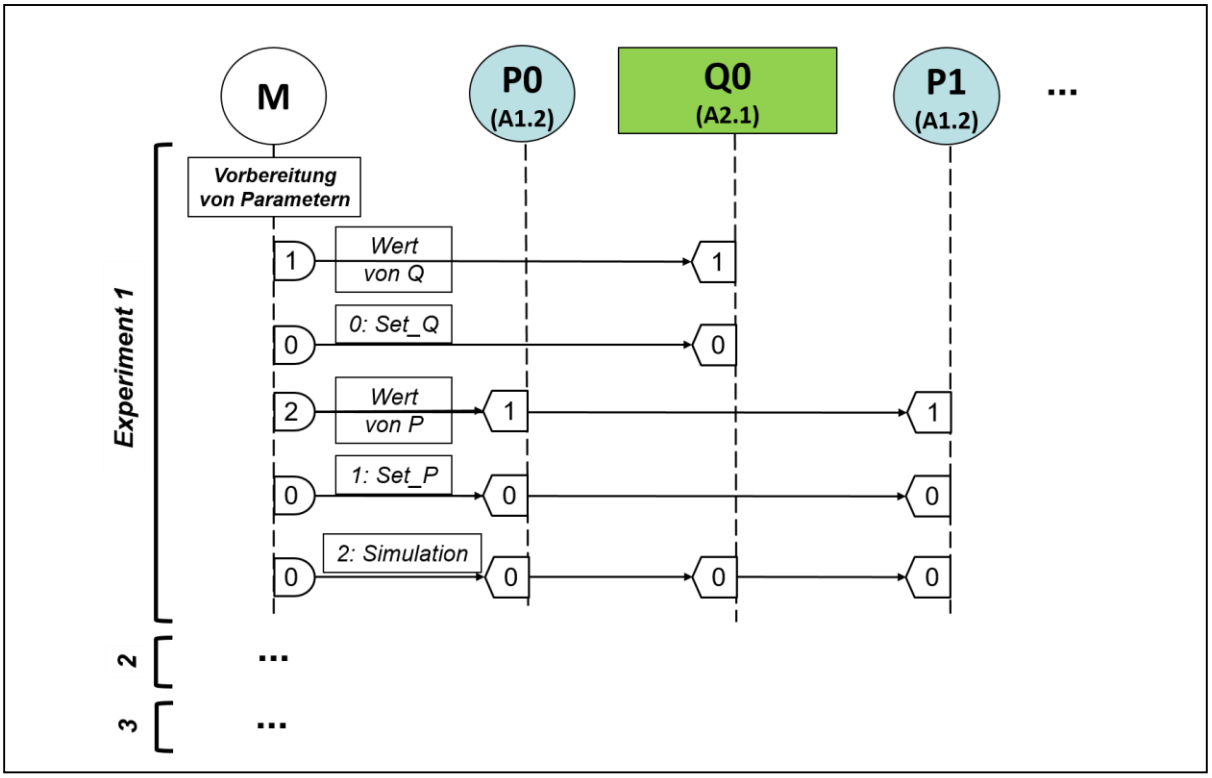


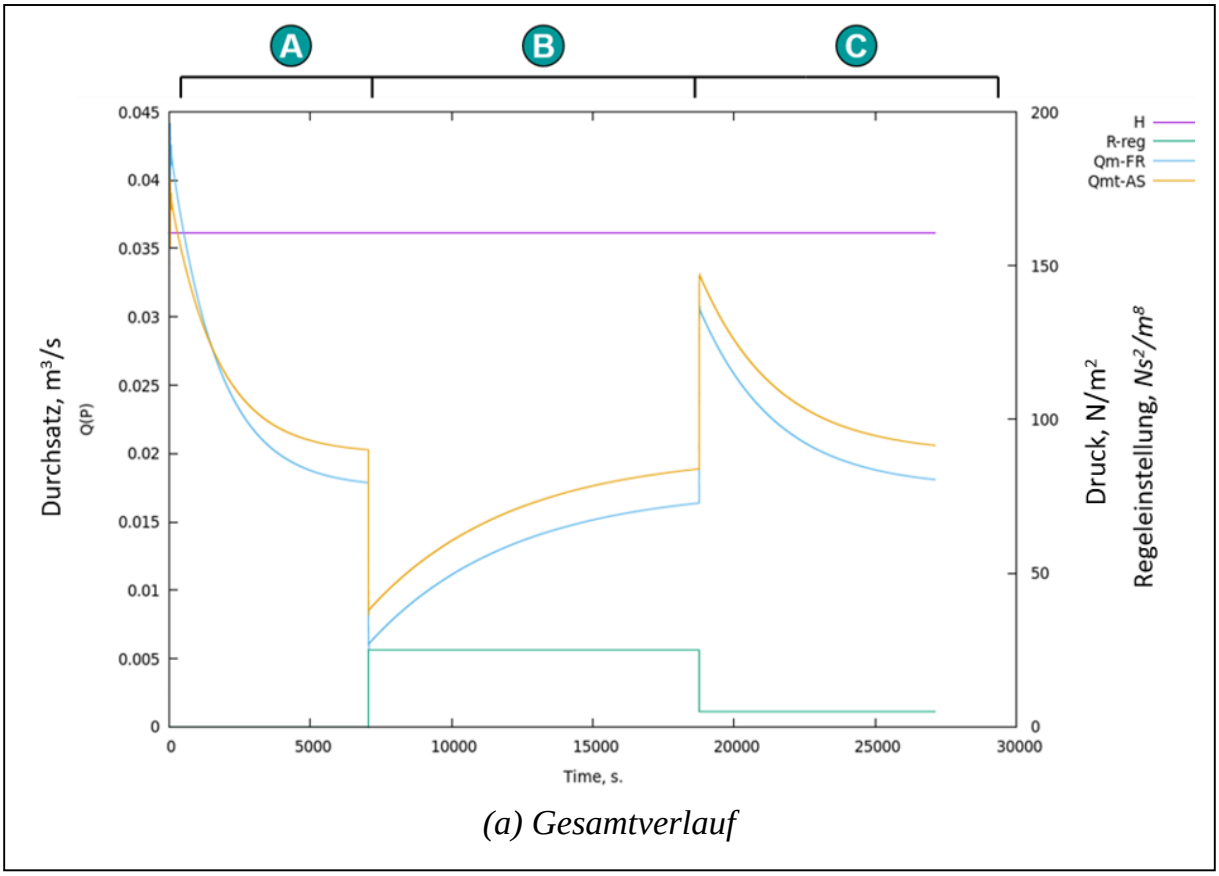
***Abbildung 54**. Ablauf von Simulationsexperimenten.*

Der Verlauf der Gas-Übergangsprozesse ist in **Abbildung 55** aufgeführt.

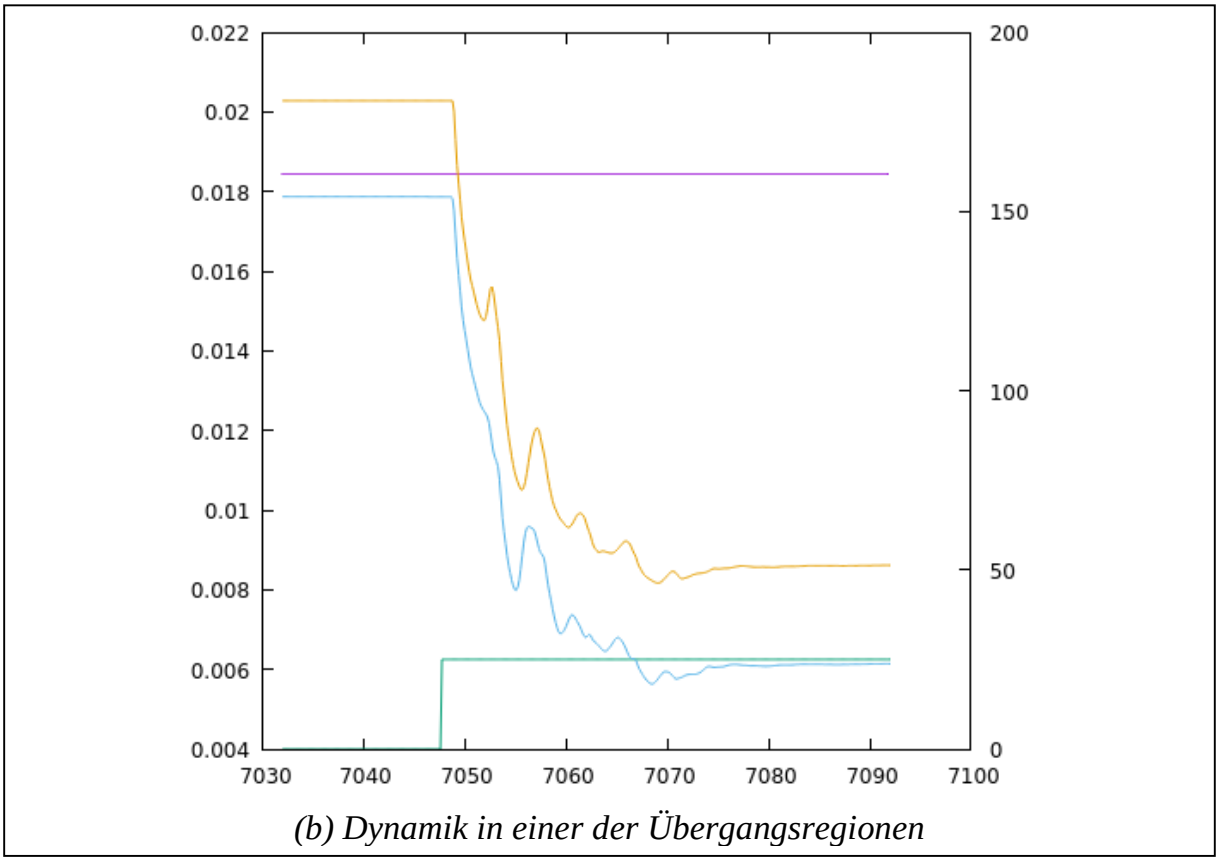
Die oben aufgeführten Verifizierungsergebnisse stehen in einer genügenden Übereinstimmung (Abweichung geringer als 2%) mit den physischen Eigenschaften des ersten Testobjekts sowie mit den vorher geschilderten Modellierungsergebnissen (Svjatnjy [R8], Hanf [R14], Moldovanova [R58], Solonin [R59], Cheptsov [R110]).

Auf Grund der Validierungsergebnisse des Testobjekts TO-1 konnte eine korrekte Erfüllung der funktionalen Anforderungen zu den entwickelten Modellen und Services bestätigt werden so dass die Experimente mit dem 2. Testobjekt durchgeführt werden können.

Das Testobjekt TO-2 ist die Kohlegrube „Süd-Donbass“ – eines der größten Bergbauwerke in der Ukraine und insgesamt in Osteuropa. Im Rahmen der vorliegenden Arbeit wurden die Validierung zweier Ventilationsabteilungen durchgeführt – der Abteilung des 1. Oststrebs „*plS11*“ (bestehend aus insgesamt 6 Wetterwegen) und der Abteilung des 7. Weststreb „*plS10-2*“ (bestehend aus insgesamt 7 Wetterwegen), mit Topologien wie in **Abbildung 56** skizziert (Originalbezeichnungen beibehalten, Erklärung in der **Tabelle 19** gegeben).



(a) Gesamtverlauf

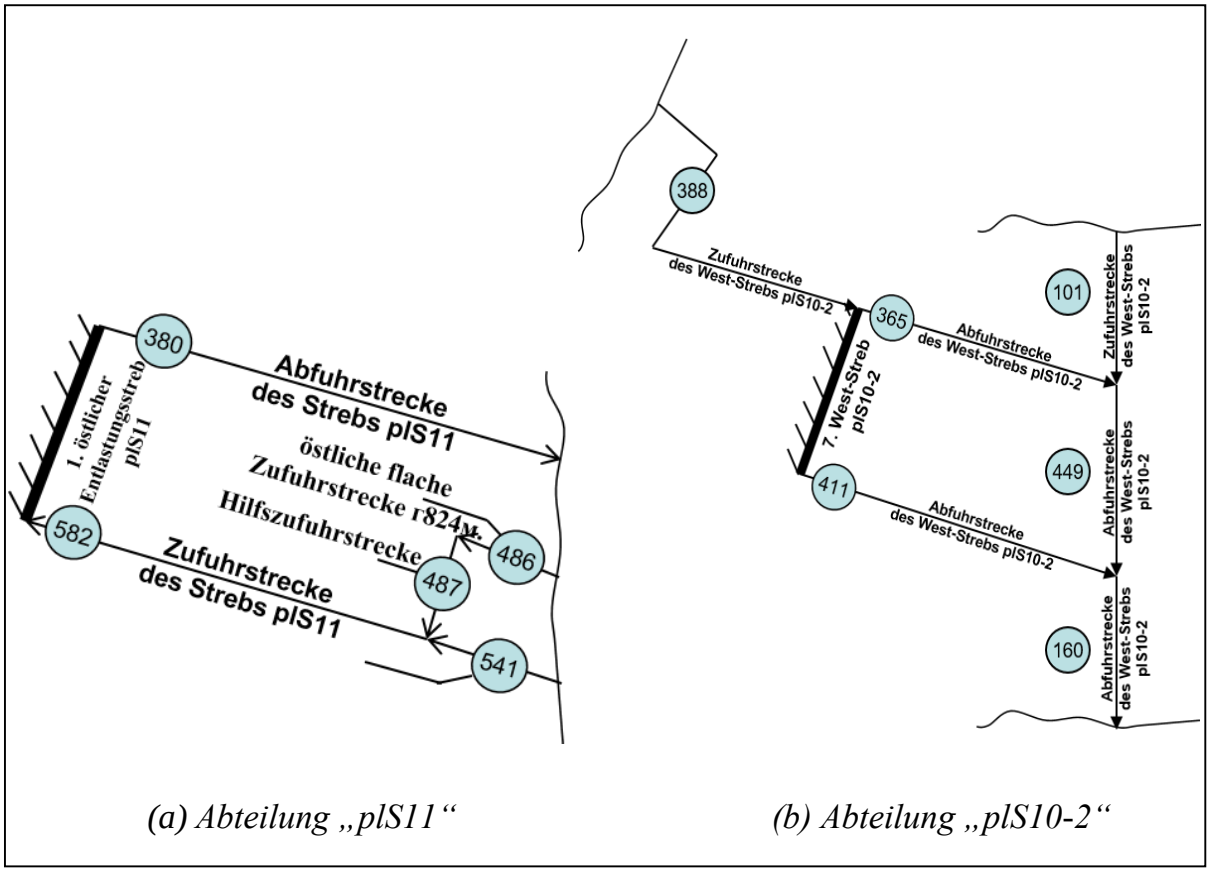


(b) Dynamik in einer der Übergangsregionen

***Abbildung 55**. Modellergebnisse für gasdynamische Prozesse im Testobjekt TO-1.*

Die Parameter des Testobjekts werden regelmäßig (halbjährlich) durch Bewetterungsspezialisten erfasst – einer Maßnahme, die im Bergbau als Depressionsaufnahme bekannt ist. In der vorliegenden Arbeit wurde bei der Validierung der (zum Zeitpunkt der Texterfassung) aktuellste Parametersatz benutzt – die Depressionsaufnahme der ersten Hälfte 2020, wie sie in **Tabelle 19** dargestellt ist.

Die Prozessdynamik entsteht in der ersten Abteilung durch die Depression des Hilfs-Ventilators, welcher an die Strecke *AS-380* angeschlossen ist und im Absaugmodus funktionierte (also durch einen negativen Depressionswert bestimmt war). In der zweiten Abteilung – *plS10-2* – werden die Luftströmungen durch den Hauptgruben-Ventilator erzeugt.



(a) Abteilung „plS11“ *(b) Abteilung „plS10-2“*

***Abbildung 56**. Topologie der Testabteilungen des Testobjekts TO-2.*

***Tabelle 19**. Aerodynamische Parameter der Testabteilungen des Testobjekts TO-2.*

(a) Abteilung „plS11“

Parameter	Streb S-340	Zufuhr-Strecke ZS-486	Zufuhr-Strecke ZS-487	Zufuhr-Strecke ZS-541	Zufuhr-Strecke ZS-582	Abfuhr-Strecke AS-380
Länge **L** *[m]*	250	390	100	220	450	750
Querschnittfläche **S** $[m^2]$	4.5	16	11	9.4	8.5	10
Widerstand **R** $[Ns^2/m^8]$	0.03	0.00291	0.00195	0.00885	0.054	0.054

(b) Abteilung „plS10-2“

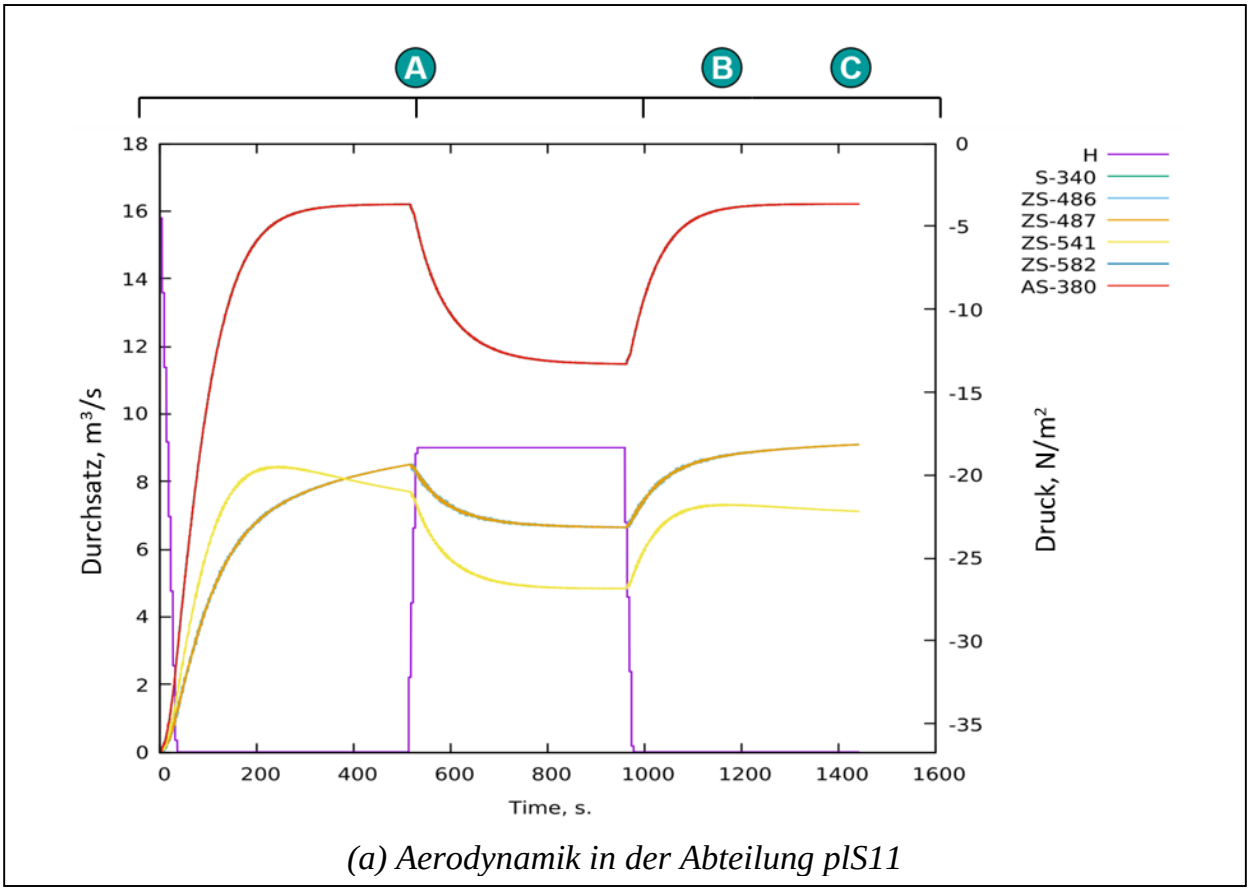
Parameter	Streb S-404	Zufuhr-Strecke ZS-388	Zufuhr-Strecke AS-101	Abfuhr-Strecke AS-160	Abfuhr-Strecke AS-365	Abfuhr-Strecke AS-388	Abfuhr-Strecke AS-411
Länge **L** *[m]*	220	500	750	135	650	500	870
Querschnittfläche **S** $[m^2]$	3.5	11	4	8	4.7	11	11
Widerstand **R** $[Ns^2/m^8]$	0.035	0.028	10.5	0.009	9999	0.0028	0.028
Leckstrom **q** *[%]*	3	-	-	-	-	-	-
Basiswert der Methanergieb. $\mathbf{Q_{m0}}$ $[m^3/s]$	0.005	-	-	-	-	-	-
FR-Volumen **V** $[m^3]$	5000	-	-	-	-	-	-

Die Simulationsexperimente wurden nach einem ähnlichen Prinzip wie beim Testobjekt TO-1 aufgebaut und folgten dem Schema der **Tabelle 20**. Im Gegensatz zu TO1-Experimenten, wurde die Depression stufenweise mit der Geschwindigkeit 1 $[N/m^2s]$ geändert, was dem tatsächlichen Ablauf der zugrundeliegenden physikalischen Prozessen eher entspricht und, als Folge, zu einer gewissen Glättung der Kompressionsprozesse in Strecken geführt hat (siehe graphische Darstellung der daraus resultierenden dynamischen Übergangsprozesse in **Abbildung 57a**). In der Simulationsservice-Implementierung wurden die Depressionswerte an den Service, der für den Druck im entsprechenden Abteilungsknoten zuständig ist, durch eine direkte p2p-Kommunikation vom Master übermittelt.

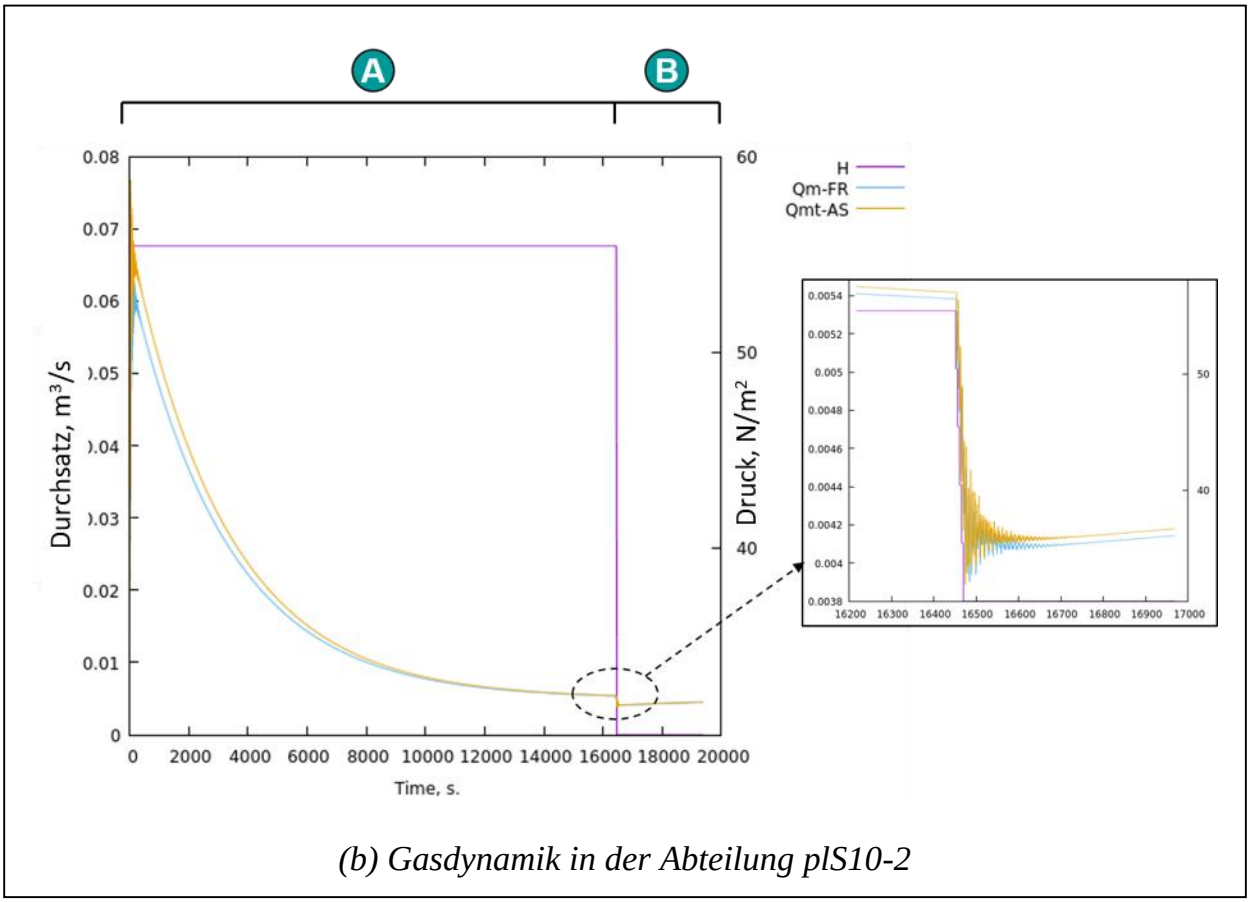
Die gasdynamischen Prozesse in der Abteilung plS10-2 sind in **Abbildung 57b** gezeigt. Es wurden zwei Experimente durchgeführt: (A) schlagartiger Depressionsanstieg von 0 auf 55 *[N/m²s]* und (B) Ablassen (schrittweise) des Drucks auf den halbierten Wert. Die Ergebnisse entsprechen den Messwerten mit einer Genauigkeit von mehr als 99% (der Q-Messwert von 25 m^3/s gegenüber dem Modellwert von 24.97 m^3/s) für aerodynamische Modelle. Auch bei gasdynamischen Modellen wurde eine hohe Genauigkeit (der Q_{m0}-Messwert von 0.005 m^3/s gegenüber dem Modellwert von 0.0045 m^3/s) von ca. 90% erreicht.

***Tabelle 20**. Ablauf und Ergebnisse der Aerodynamik-Modellexperimente für plS11-Abteilung des Testobjekts 2.*

Parameter	**Experiment A**		**Experiment B**		**Experiment C**	
	Anfang	Ende	Anfang	Ende	Anfang	Ende
Regulierte Parameter						
Depression **ΔP** *[N/m²]*	0	**-36.7**	-36.7	**-18.35**	-18.35	**-36.7**
Messwerte						
Luftdurchsatz **Q** *[m³/s]*						
- ZS-486	0	**9.2**	9.2	**6.5**	6.5	**9.2**
- ZS-487	0	**9.2**	9.2	**6.5**	6.5	**9.2**
- ZS-541	0	**7.1**	7.1	**5**	5	**7.1**
- ZS-582	0	**16.3**	16.3	**11.5**	11.5	**16.3**
- S-340	0	**16.3**	16.3	**11.5**	11.5	**16.3**
- AS-380	0	**16.3**	16.3	**11.5**	11.5	**16.3**
Simulationsergebnisse						
Luftdurchsatz **Q** *[m³/s]*						
- ZS-486	0	**9.09**	9.09	**6.65**	6.65	9.09
- ZS-487	0	**9.09**	9.09	**6.65**	6.65	9.09
- ZS-541	0	**7.12**	7.12	**4.83**	4.83	7.12
- ZS-582	0	**16.21**	16.21	**11.48**	11.48	16.21
- S-340	0	**16.21**	16.21	**11.48**	11.48	16.21
- AS-380	0	**16.21**	16.21	**11.48**	11.48	16.21



(a) Aerodynamik in der Abteilung plS11



(b) Gasdynamik in der Abteilung plS10-2

***Abbildung 57**. Dynamische Prozesse im Testobjekt TO-2.*

Die Validierungsergebnisse bestätigen eine sehr gute Übereinstimmung der Simulationsergebnisse mit den experimentellen Messwerten – die maximale Abweichung lag für die beiden Abteilungen (pls11 und pls10-2) unter 1% für die aerodynamischen und 10% für die Gasdynamik-Modelle.

Evaluierung nichtfunktionaler Eigenschaften

Unter die Kategorie der nichtfunktionalen fallen die folgenden Eigenschaften:

- Die Zeit/Leistungs-Eigenschaften (Performance) von Simulationsservices bei der Durchführung von Simulationsexperimenten. Insbesondere in den Echtzeitszenarien darf die Simulationsdauer nicht die physikalische Zeit des modellierten Prozesses übersteigen.
- Die Effektivität der Nutzung von Hardware-Ressourcen, unter anderem von parallelen und verteilten Systemen. Der Kommunikationsoverhead zwischen den Mikroservices sollte möglichst gering und die parallele Effizienz möglichst hoch bleiben.

Der Evaluierungsprozess lief nach dem folgenden Schema ab:

(1) Die Leistung eines Einzelservice wurde auf unterschiedlichen Architekturen[49] gemessen:

- Handelsüblicher PC (ein Intel x86 CPU mit 8 Kernen)
- Eingebettetes System ODROID-XE4 [W35] (ein ARM Cortex A-15 und ein ARM Cortex-A7 mit jeweils 4 Kernen)
- Ein Knoten des HPC-Systems Vulcan des HLRS [W36] (mehrere *Xeon Haswell* CPUs mit jeweils 24 Kernen).

Das Ziel dieser Evaluierung war es einen Vergleich der Leistung von unterschiedlichen Architekturtypen (unter maximaler Rechenauslastung) bei der Lösung einer spezifischen Simulationsaufgabe zu schaffen. Es sollte

[49] Unabhängig vom System wurde Open-MPI in der Version 4.0.1 verwendet, gebaut auf der Basis des GNU-Compiler in der Version 7.4.0

insbesondere evaluiert werden, ob die eingebetteten Systeme die Echtzeitanforderungen gewährleisten können.

(2) Die Skalierbarkeit einer Simulationsanwendung wurde auf einem parallelen System evaluiert. Das Ziel war dabei die Untersuchung der Effizienz des implementierten Kommunikationsmodells zwischen den parallel laufenden Services.

(3) Die Nutzung von *NoSQL*-Datenbanken zur Speicherung der Simulationsdaten wurde evaluiert. Das Ziel der Evaluierung ist die Untersuchung der Overheads gegenüber der lokalen Speicherung auf einer Disk und Erkenntnisse über die Brauchbarkeit der Datenbankennutzung in Echtzeitszenarien.

Die Leistung der Simulationsservices auf unterschiedlichen Hardware-Architekturtypen wurde anhand des gasdynamischen Modells des FR-Mediums des TO1-Objekts (mit den Parametern von **Tabelle 18**) evaluiert. Die Depression an Grenzpunkten zwischen dem FR-Medium und den Strecken wurde aus den Ergebnissen funktionaler Validierungstests entnommen und orientierte sich an dem Ablauf des Experiments von **Tabelle 17**: (A) Steigung vom Initialwert *0* auf den Basiswert *157 [N/m^{2}s]* am Anfang der Simulation, (B) Senkung um 25% nach 5 Stunden physikalischer Zeit, (C) Rückkehr zum Basiswert nach weiteren 2 Stunden und weiterer Verlauf dynamischer Prozesse innerhalb von 3 Stunden. Die virtuelle Modellzeit betrug damit für das Evaluierungsexperiment insgesamt ca. 10 Stunden. Die gemessenen Simulationsdauerwerte auf den unterschiedlichen Plattformen sind in **Abbildung 58** zusammengefasst. Die Modellgröße (Anzahl der Diskretelemente) wurde so ausgewählt, um alle vorhandenen Recheneinheiten des leistungsschwächsten der getesteten Systeme (Odroid) zu saturieren: Bei einer Länge *L* eines Diskretelements (siehe Spezifikation des *G1.1.1*-Modells im Kapitel 4.3.3) von ca. 50 m wurden genau 8 Recheneinheiten (MPI-Prozesse) benötigt – so viele wie die Anzahl verfügbarer CPU-Kerne im Odroid. Die Tests haben eine insgesamt gute Leistung der eingebetteten Systeme bei der Simulationsdurchführung bewiesen: Das Odroid-System war zwar deutlich (um das knapp 11-fache) langsamer als der vorhandene HPC-Knoten, konnte aber trotzdem einen deutlichen Vorsprung der virtuellen Modellierungszeit gegenüber der

tatsächlichen physikalischen Zeit gewährleisten. Um das Leistungsbild der Rechensysteme zu vervollständigen, wurde die Anzahl der Elemente verdoppelt (der Wert von *L* auf 25 m gesetzt) und damit der Rechenaufwand auf das System verdoppelt. Wie die Ergebnisse zeigen, schneidet das „überladene" Odroid-System mit den größten Leistungseinbußen ab (mit einem fast 3-fachen Anstieg der Simulationsdauer), ist aber immer noch in der Lage, die Echtzeitanforderungen mit einem sicheren zeitlichen Vorsprung gegenüber der physikalischen Zeit zu gewährleisten.

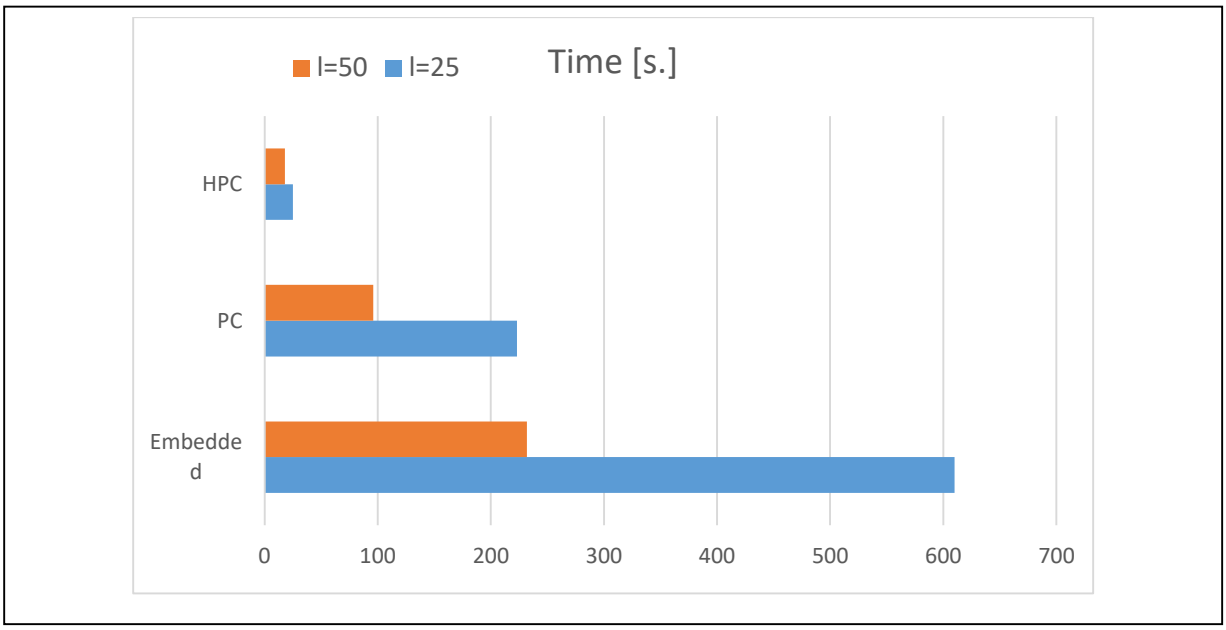


***Abbildung 58**. Ausführungszeiten der Gasdynamiksimulation auf unterschiedlichen Architekturen.*

Im nächsten Evaluierungsschritt wurde das Skalierungspotenzial der Simulations-services evaluiert. Dafür wurde der größte von den im Kapitel 4 entwickelten Simulationsservices als Basis genommen – die Implementierung des *G4.2*-Modells der Gasdynamik in der Bewetterungsabteilung des Testobjekts TO-1, welches aus insgesamt *n*=58 Services besteht. Die Evaluierung erfolgte auf dem Vulcan-Cluster[50] [W36] mit Haswell-Knoten (mit jeweils 24 CPU-Kernen). Die Zeitmessungen (siehe **Abbildung 59**) ergaben, dass die service-basierte Simulationsanwendung sehr gut skaliert – die parallele Effizienz (die Verkürzung der Rechenzeit proportional zu der Steigerung der Rechenkapazitäten) liegt deutlich im Bereich von über 100%, soweit alle

[50] Spezifisch wurden die Simulationsservices auf den folgenden Knoten ausgeführt: n081501, n081502, n081602, n081601 und n080701.

CPU-Cores genügend ausgelastet sind (also $n \geq m$, wobei n – die Anzahl der Services, m – die aggregierte Anzahl der CPU-Kerne ist). Wie die Messungen zeigen, skaliert die Anwendung gut (die parallele Effizienz verläuft konstant), bis die Anzahl der Services ausreichend ist, um die vorhandenen Rechenressourcen zu saturieren (bis 3 Knoten mit insgesamt 72 CPU-Cores). Mit der Senkung des SaturationsWertes der Hardware (im getesteten Fall – ab 4 Knoten mit insgesamt 104 CPU-Kernen), fängt die parallele Effizienz an erwartungsgemäß zu fallen (in den Bereich von deutlich unter 100% zu rutschen). Die Evaluierung zeigt aber insgesamt die gute Effizienz der Parallelisierung der service-orientierten Implementierung der Modelle.

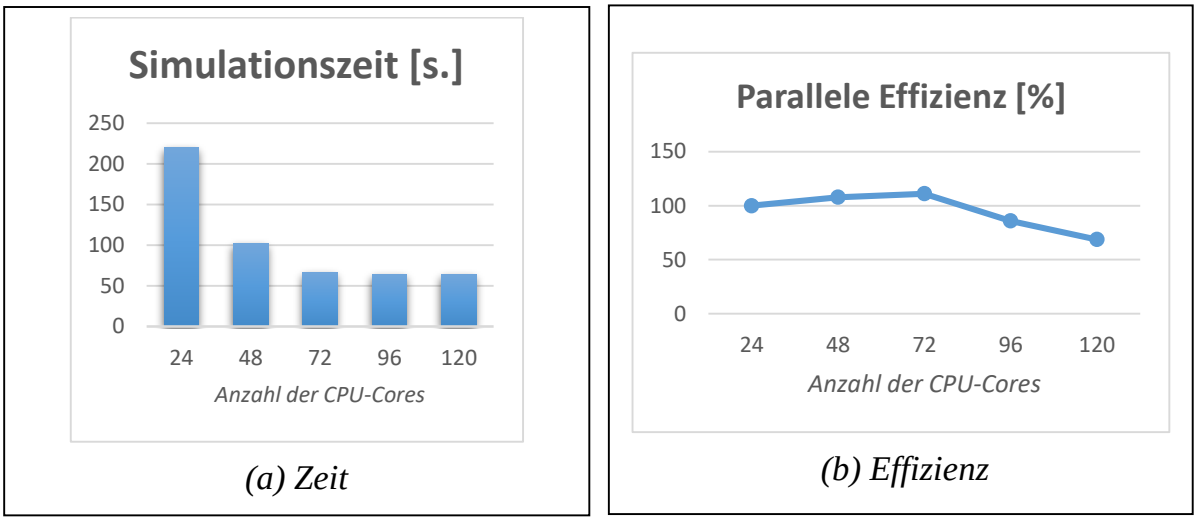


***Abbildung 59**. Skalierbarkeit und Effizienz der parallelen Realisierung des gasdynamischen G4.2-Modells der Bewetterungsabteilung.*

In der abschließenden Evaluierungsphase wurden die praktischen Aspekte der Nutzung von verteilten Datenbanken zur Speicherung der Simulationsexperiments-Daten evaluiert. Hierfür wurden die Ausführungszeiten der Simulationsanwendungen für das Testobjekt TO-1 (Gassimulationsanwendung bestehend aus 58 Services, 180K Iterationen werden benötigt) und das Testobjekt TO-2 (Luftsimulation, 91 Services, 9.6K Iterationen) mit folgenden Ausgabeoptionen gemessen:

- Keine Datenausgabe
- Ausgabe in eine Datei auf einem lokalen Filesystem
- Ausgabe in die *Elasticsearch*-Datenbank.

Bei jedem Simulationsschritt wurde von jedem Service der untersten Hierarchieebene (diskrete Elemente) der Wert des Strömungsdurchsatzes bzw. des Drucks ausgegeben. Alle Tests wurden auf einem Intel-x86-PC durchgeführt. Die gemessenen Ergebnisse sind in **Tabelle 21** zusammengefasst.

***Tabelle 21**. Performanceeigenschaften der Simulationsexperimente auf einem System.*

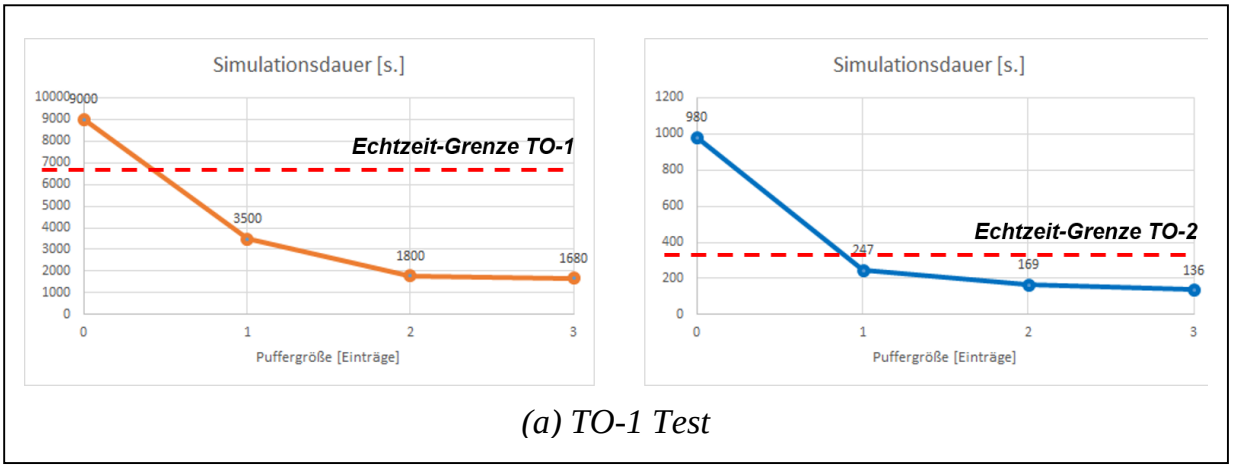
TO-1 (Methan)			**TO-2** (plS11, Luft)		
Keine Ausgabe	Ausgabe in Datei	Ausgabe in DB	Keine Ausgabe	Ausgabe in Datei	Ausgabe in DB
Virtuelle Experimentzeit, s.					
6778			360		
Simulationsdauer, s.					
600	620	9000	65	68	980

Die Ergebnisse zeigen, dass die Datenspeicherung ins *Elasticsearch*-DBMS sehr ineffizient ist – es wird die 10-fache Zeit benötigt wie im Falle der Speicherung in die Dateien auf einer Disk. Deswegen könnte diese Speicherungsoption nicht bei den gegebenen Echtzeitanforderungen (die Simulationszeit darf nicht die virtuelle Experimentzeit übersteigen) zum Einsatz kommen.

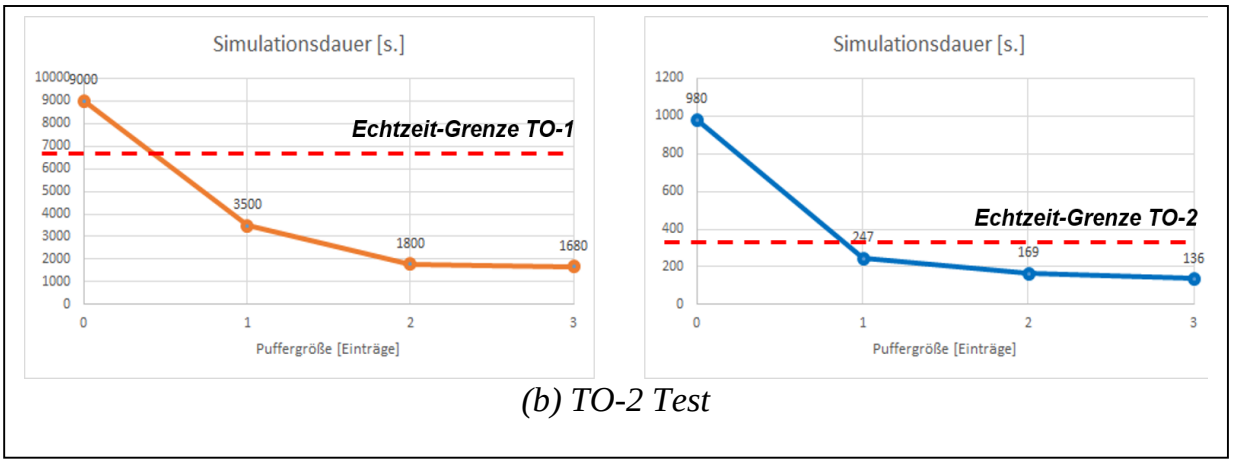
Die Gründe für diese dramatische Ineffizienz der Datenbankspeicherung könnten an zwei möglichen Ursachen liegen. Zum einen ist der Durchsatz von TCP/IP – dem Netzwerk-Transportmechanismus, der von *Elasticsearch*-Realisation verwendeten wird, deutlich geringer als der der Disk-Schnittstelle. Zum anderen wurde ein erheblicher Zeitverlust beim Aufbau von sicheren Übertragungskanälen verzeichnet, was grundsätzlich bei jeder Datentransaktion (also zur Speicherung nach jedem numerischen Schritt) notwendig war. In diesem Fall wäre die Echtzeitfähigkeits-anforderung nicht erfüllt.

Während die bekannten Forschungsansätze in Richtung Integration der Hochge-schwindigkeitsnetzwerke mit NoSQL-Datenbanken fortschreiten, wie z.B. von Andre et al. [R111], bleibt die Minimierung der Transaktionenanzahl durch die Eingruppierung von einzelnen zu versendeten Nachrichten (also Pufferung) als einzige effektive Methode zur Reduzierung des Zeitverlustes bei der Datenübertragung an eine externe Datenbank. Um den Einfluss der Pufferung auf die Speicherungseffizienz zu evaluieren,

wurde im *MS-Monitoring-API*[51] eine Funktion eingebaut, die die Pufferung durch die Definition des Parameters "*buf_size*" transparent für den Nutzer steuern lässt. Der Einsatz von Pufferung reduziert den Zeitoverhead der *Elasticsearch*-Speicherung um ein Vielfaches, wie **Abbildung 60** zeigt. Somit sollte der Einsatz der Datenbank-Technologien zur Speicherung der Simulationsergebnisse auch in Echtzeit-szenarien (wie z.B. im nächsten Kapitel 5.2 gezeigt wird) unbedenklich möglich sein.



(a) TO-1 Test



(b) TO-2 Test

***Abbildung 60**. Performance mit variierter Speicherpuffergröße.*

Zusätzlich wurde noch eine weitere Datenspeicherungsmöglichkeit evaluiert – ein zweistufiges Verfahren, nach welchem die Daten zuerst in die Dateien auf einer lokalen Disk gespeichert und erst am Ende der Simulation an *Elasticsearch* übertragen werden. Es wurden allerdings keine Vorteile bei der Nutzung dieses Verfahrens festgestellt – die Speicherungszeit verdoppelt sich im Vergleich zur sofortigen Übertragung an die Datenbank.

[51] https://github.com/alexey-cheptsov/MS-Sim/tree/master/Monitoring

5.2 Anwendungsszenarien des Basisforschungsgebiets

In diesem Kapitel wird ein Überblick über die wichtigsten Anwendungsszenarien für die entwickelten Simulationsservices (siehe Kapitel 4) des Forschungsgebiets „Untertagebewetterung“ gegeben, die in Zusammenarbeit mit den Fachgebietsexperten ausgearbeitet wurden.

Planung der VOD-Maßnahmen

VOD (engl. = Ventilation-on-Demand – bedarfsgerechte Bewetterung) ist ein Komplex von Maßnahmen zur Gewährleistung des notwendigen Bewetterungsbetriebs in Untertagebauten. Ein Ziel der VOD-Analyse ist die Maximierung der Leistungseffizienz von Grubenventilatoren, die zur Erreichung des notwendigen Luftdurchsatzes in allen Elementen des Bewetterungsnetzes ausreichend sein sollte. Dabei sollte der aktuellste Stand der Bewetterungselemente berücksichtigt werden.

Die Effektivität der VOD-Berechnung beeinflusst die Energieeffizienz des ganzen Bergbaubetriebs maßgeblich, aufgrund eines sehr hohen Energieverbrauchs von Bewetterungsventilatoren (bis zu 50% des gesamten Energieverbrauchs des Bergbaubetriebs, wie von Clausen in [R12] evaluiert wurde). Die VOD-Maßnahmen werden in der Regel anhand statischer (z.B. Methode elektrischer Analogien oder Hardy-Cross-Verfahren) Modelle berechnet (wie z.B. Ansatz von Babu [R112], Pritchard [R113] u.v.a.), sollten aber innerhalb dieses Anwendungsszenarios anhand der dynamischen Modelle evaluiert werden, wie im Diagramm in **Abbildung 61** gezeigt ist. Die dynamischen Modelle ermöglichen eine präzise Einschätzung der vom Bewetterungsbetrieb abhängigen Luft- und Methanverteilungsprozesse, die die Korrektheit der von statischen Ansätzen erstellten Bewetterungspläne erhöhen sollen. Die Evaluierung erfolgt anhand typischer Szenarien des konkreten Bergbauwerks, die von Domäneexperten definiert werden können.

Der Anwendungsfall kann durch eine Erweiterung der Basisservices (mithilfe der Injektiontechnik) implementiert werden, wobei die übergeordneten Service die für die statischen Modelle notwendige Parameter (wie z.B. die Durchsatzbreite) sammeln.

Kontinuierliche Modellvorhersage der Bewetterungsverhältnisse

In diesem Anwendungsszenario sollten die Simulationsservices eine “ahead-of-time“-Prognose über den weiteren Verlauf der Wetterdynamik anhand der aktuellen Funktionierungsbedingungen geben. Die Prognose kann einen Zeitraum von bis zu 4-5 Stunden umspannen, abhängig von der Dauer der physikalischen Dynamikprozesse. Dieses Anwendungsszenario soll mithilfe von der Echtzeit-unterstützenden Funktionalität des Monitoring-API realisiert werden (siehe Kapitel 4.4), wie in **Abbildung 62** gezeigt ist.

Ein kontinuierliches Experiment läuft nach dem folgenden Schema ab:

- Zu Beginn der Simulation werden die Randbedingungen (z.B. Sensorwerte für gasdynamische Modelle) erfasst, Modelle initialisiert und der initiale Simulationsvorgang durchgeführt.
- Dann wird vom Planer anhand des Zeitstempels, mit dem das letzte Simulationsergebnis markiert ist, der Zeitpunkt der Durchführung des nachfolgenden Simulationsexperiments bestimmt (nach der Methodik von Kapitel 4.4).
- Anschließend wird das nächste Simulationsexperiment vom Planer gestartet. Dies geschieht spätestens zum ermittelten Zeitpunkt oder auch früher, wenn es Änderungen an den Modellparametern (erfasst von Sensoren oder vorgegeben von Nutzern) geben soll (siehe Diagramm in **Abbildung 63**). Im letzteren Fall wird die Simulation unmittelbar nach der Änderungsübermittlung an den Planer durchgeführt.

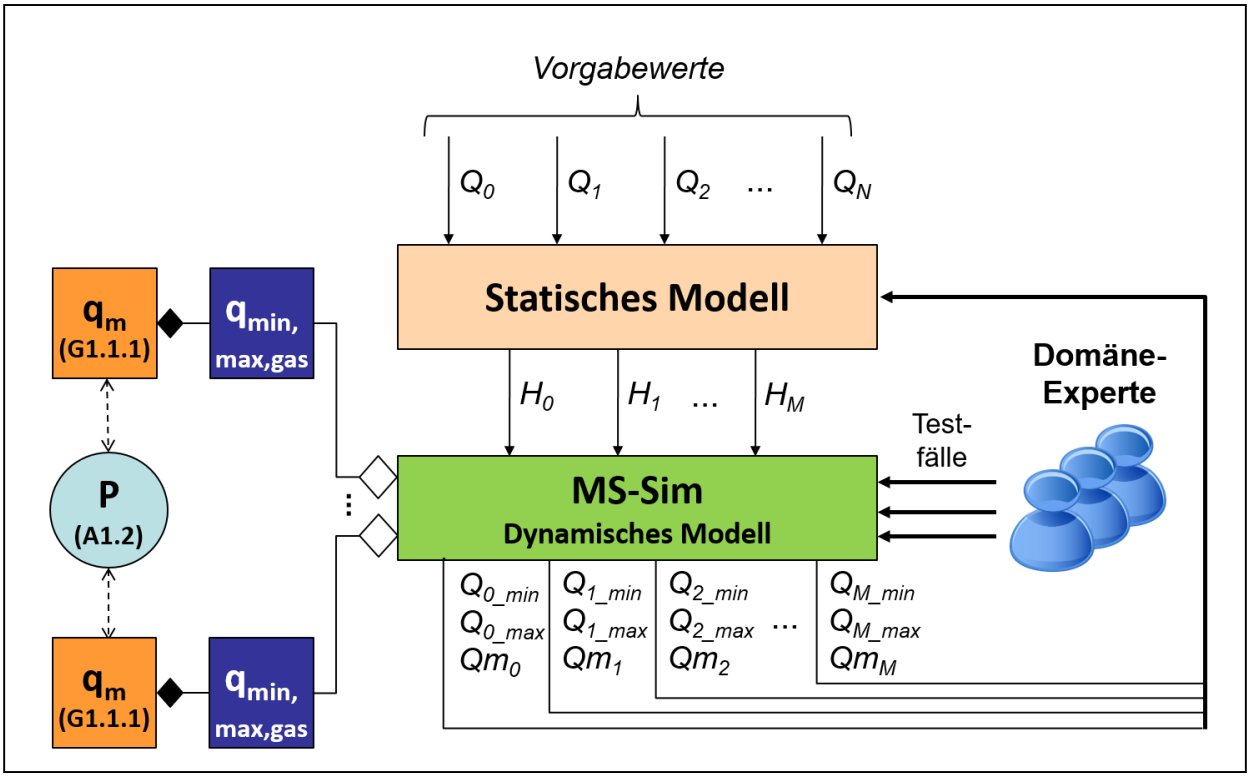


***Abbildung 61**. Nutzung der Simulationsservices bei der VOD-Planung.*

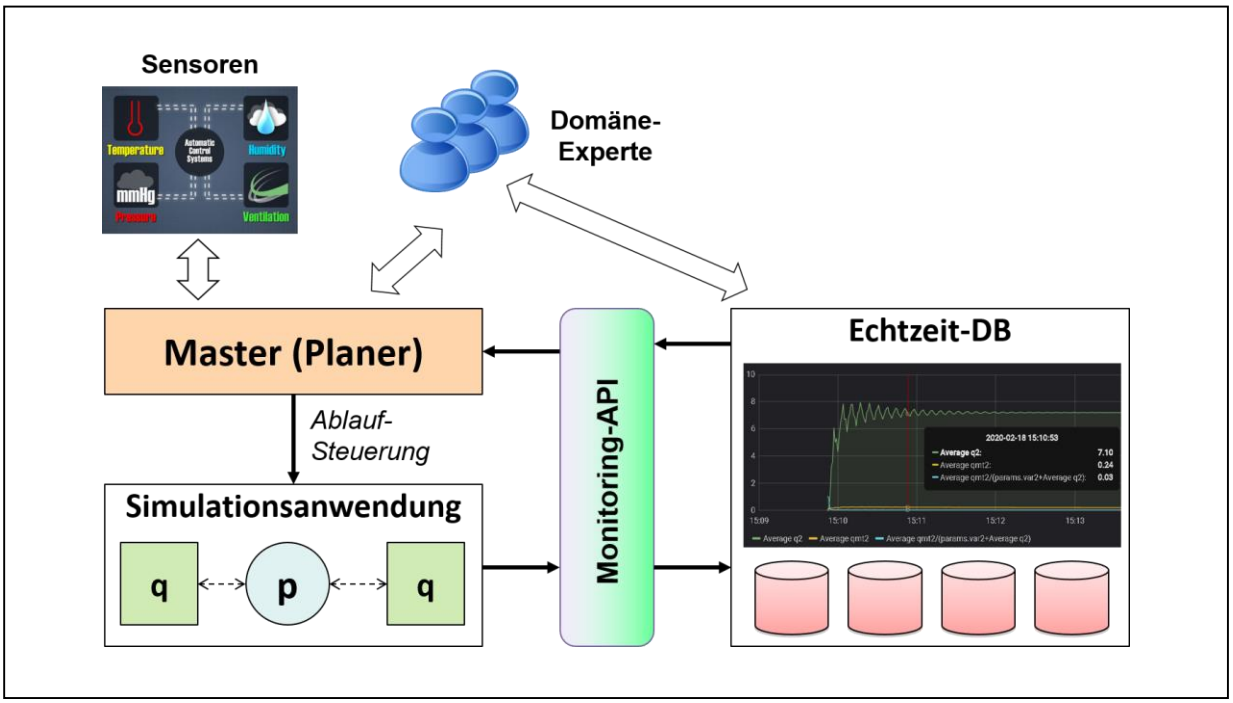


***Abbildung 62**. Aufbau eines kontinuierlichen Simulationsexperiments.*

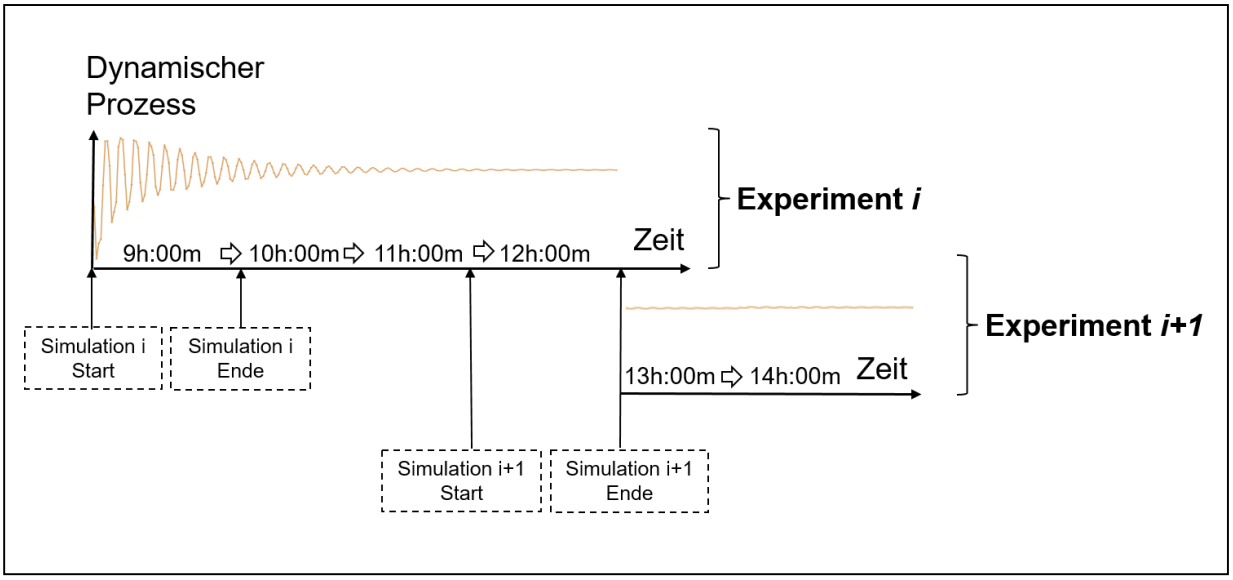


***Abbildung 63**. Ablauf und Planung von kontinuierlichen Simulationsexperimenten.*

Die Abfrage des Zeitstempels des letzten Ergebnisses der Echtzeitdatenbank sowie die Berechnung des Zeitpunkts der Ausführung des nächsten Experiments erfolgt mittels des Monitoring-API (Kapitel 4.5).

Die Master-Anwendung (der Planer) kann auch entsprechend auf die Änderungen von Modellparametern (z.B. Streckenlängen) reagieren – die Modelle können entweder mit den neuen Parameterwerten wiederinitialisiert werden oder die neuen Parameter werden durch entsprechende Kommunikationsschnittstellen übermittelt und von Modellen übernommen. Die Anwendung kann in diesem Fall ununterbrochen verlaufen.

Erstellung der Depressionspläne

Als Depressionsplan wird im Bergbau die Druck- und Luftdurchsatzverteilung entlang aller Strecken des Wetternetzes bezeichnet. Die Druck- und Luftdurchsatz-Auf-nahmen erfolgen regelmäßig (alle sechs Monate oder öfters[52]) mithilfe portabler Sensoren und verfolgen zwei wichtige Ziele: Evaluierung der Bewetterungseffizienz und Erfassung der Daten zur Berechnung des optimalen Ventilatorenbetriebs.

[52] Erfolgt nach gesetzlich-vorgeschriebenen Sicherheitsrichtlinien

Die Depressionsaufnahme ist ein aufwändiger Prozess, sowohl im Sinne des Zeitaufwands als auch aufgrund hoher anfallender Kosten. Die Idee dieses Anwendungsszenarios ist eine Reduzierung der Anzahl der physikalischen Messungen, indem die Messungen nur bei den strategisch wichtigen Punkten des Wetternetzes (falls nicht bereits von den Sensoren erfasst) durchgeführt werden. Bei den restlichen Punkten des Bewetterungsnetzes sollten die Ergebnisse aus den Simulationen entnommen werden, wie in **Abbildung 64** illustriert.

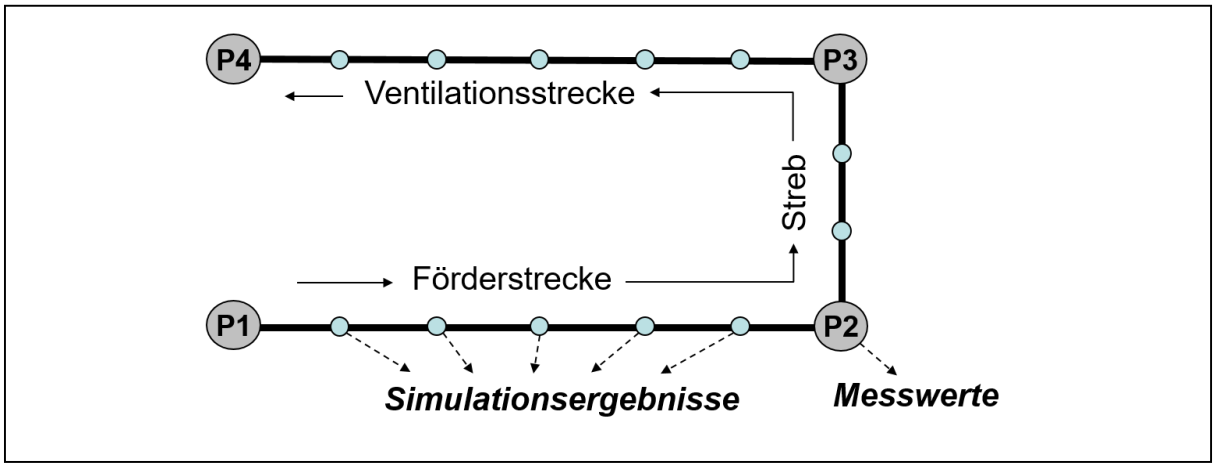


***Abbildung 64**. Simulationsbasierte Erstellung der Depressionspläne für eine Bewetterungsabteilung.*

Die Länge der Approximationselemente wird in diesem Szenario an die notwendige Granularität der Depressionsmessungen angepasst, sodass die Ergebnisse der Modelle ohne weitere Anpassungen im Depressionsaufnahmebericht aufgenommen werden können.

Identifikation der Modellparameter anhand der Sensordaten

Bei der Identifikation handelt es sich um eine Lösung der umgekehrten zu der ursprünglichen Aufgabestellung: Die Findung der Modellparameter, die den vorhandenen (gemessenen) Ergebnissen entsprechen. Dies ist besonders wichtig bei Parametern, die nicht eindeutig bestimmt werden können. So ist dies zum Beispiel der Fall bei gasdynamischen Prozessen in Bewetterungsnetzen, die maßgeblich von der Fassung und der Durchlässigkeit des abgebauten Raums (Filtrationsmediums) beeinflusst werden. Eine genaue Berechnung der realen Werte ist aber ein äußerst komplexer Prozess, der einen großen Raum für die Ungenauigkeit der Ergebnisse

zulässt. Daher wird in der Praxis üblicherweise ein Abschätzungswert verwendet, der wiederum zu einer Ungenauigkeit bei den Modellergebnissen führen kann.

Die Methodik der Parameteridentifikation kann am Beispiel der gasdynamischen *G1.1.1*-Modelle (IV.19) erläutert werden. Dabei sollte die folgende Identifikations-funktion gelöst werden:

$$mV = f(q_m^{\max}, t^{\max}), \tag{V.1}$$

wobei $m \in [0,1] \in Q, V \in [V^{\min}, V^{\max}]$ – die Parameter des Models (IV.12), $q_m^{\max}$ – der Maximalwert der Methanströmung, $t^{\max}$ – die Dauer des Übergangsprozesses ist.

Durch einen Parameterstudie-ähnlichen Simulationsvorgang (siehe **Abbildung 65**) wird iterativ die optimale Konfiguration der gesuchten Parameter innerhalb ihrer zugelassenen Wertebereiche ermittelt.

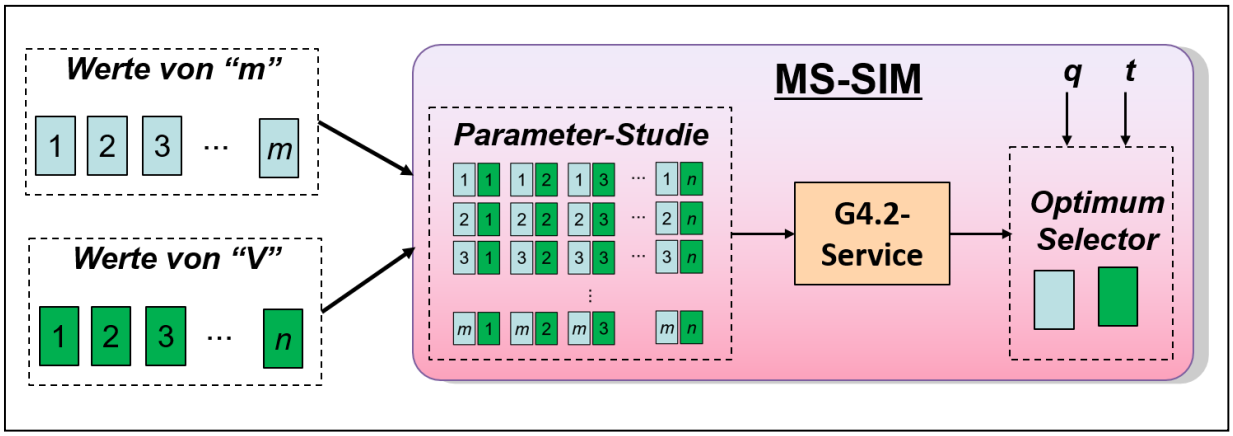


***Abbildung 65**. Ablauf der Parameteridentifikationsstudie.*

Ein ähnliches Verfahren könnte auch bei den aerodynamischen Modellen eingesetzt werden, wie etwa zur Findung der Querschnittfläche von Verbindungsknoten (der "*S*" Parameter in der Gleichung (IV.9)). Die Problematik der Wertfindung besteht im Falle der Verbindung von Strecken mit unterschiedlichen Durchmessern durch einen Knoten, wie in **Abbildung 66** illustriert ist.

In den Modellen, die im Rahmen der vorliegenden Arbeit entwickelt wurden (siehe Kapitel 4.3), wurde ein Ansatz implementiert, nach welchem der Knotenquerschnitt als Durchschnittswert aller verbundenen Strecken berechnet wurde:

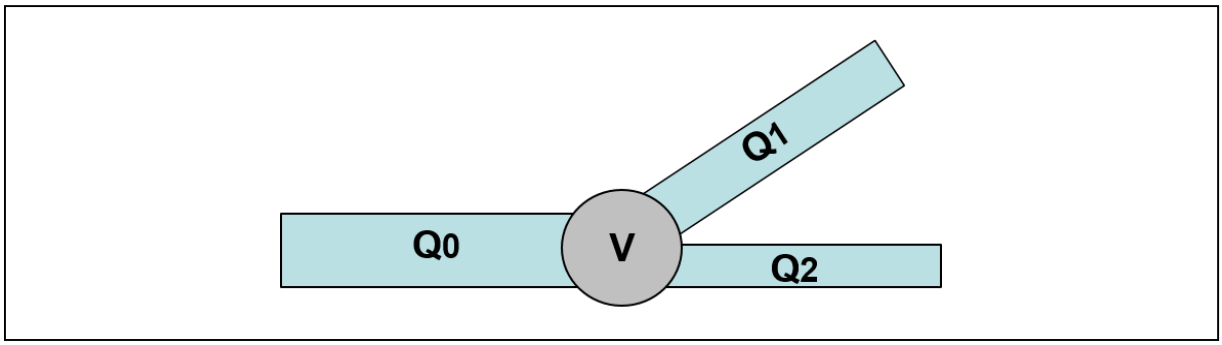


***Abbildung 66**. Verbindung von Strecken mit unterschiedlichen Durchmessern.*

$$S_V = \frac{\sum S_{Qi}}{n}, i = 1..n\,. \tag{V.3}$$

Diese Methodik konnte die Genauigkeit der Ergebnisse (allerdings nur unwesentlich) verbessern. In der Realität kann der Querschnittflächenwert zwischen dem von der Strecke mit dem kleinsten Durchmesser und dem von der Strecke mit dem größten Durchmesser liegen. Die Identifikationsaufgabe besteht in dem Fall in der Findung der Querschnittfläche, die den gemessenen Werten (von einem Sensor oder als Ergebnis einer Depressionsaufnahme) am ehesten entspricht.

5.3 Anwendungsbeispiele aus weiteren Forschungsgebieten

In diesem Kapitel werden exemplarische Beispiele der Anwendungen aus weiteren herausfordernden Forschungsbereichen (Stadtplanung und Umweltschutz) erarbeitet, die vom Einsetzen der entwickelten Mikroservicearchitektur stark profitieren können.

Dezentralisierte Infrastruktur zur intelligenten Verkehrssteuerung

Die Verkehrssteuerung ist eine herausfordernde Aufgabe für jede Stadtplanung. Das Hauptmittel der Steuerung ist ein verteiltes Ampelsystem, das eine bedarfsgerechte Schaltung anwendet, die mittels spezieller Algorithmen gesteuert wird. Die meist-verwendeten Algorithmen sind aber von ihrer Natur her statisch – die vorprogrammierten Regulierungsabläufe der Ampelanlagen können weder die aktuelle Verkehrssituation berücksichtigen noch auf dringende kurzfristige Änderungen (z.B. aufgrund eines Notfalleinsatzes im kontrollierten Gebiet) entsprechend reagieren.

Das Fraunhofer-IPA[53] erwägt in Kooperation mit der Firma Evia[54] ein Projektvorhaben, welches das Ziel hat, ein innovatives KI-System zu entwickeln, das die vorhandenen statischen Algorithmen durch die Einbindung dynamischer Verkehrsinformationen erweitern soll. Die Informationen sollen aus Smart-Sensoren einfließen, die in das IT-System der Ampelanlagen installiert werden und ein Bild über die aktuelle Verkehrslage liefern sollen. Die Auswertung aller anfallenden Daten sollte dann auf einer verteilten Cloud-Infrastruktur aus portablen Rechnern erfolgen, die mit sehr geringem Energieverbrauch und Platzbedarf in die Ampelanlageinfrastruktur integriert werden. So werden die Kurzfristprognosen aufgrund der aktuellen Situation (Anzahl der Fahrzeuge, Auftreten eines Notfalls) von allen Ampelanlagen im kontrollierten Bereich (von einem Bezirk bis über die ganze Stadt) gemacht und die Funktionierung des gesamten Systems optimiert (z.B. der Verkehrsdurchsatz maximiert, die Luftverschmutzung minimiert usw.).

Zu einer koordinierten Ausführung der Komponenten vom zu entwickelnden Simulationsalgorithmus, die über alle vorhandenen Cloud-Ressourcen verteilt werden sollen, ist eine Softwareplattform benötigt. Das *MS-Sim*-Framework wird von seinen Entwicklern als Hauptkandidat zur Implementierung des Systems betrachtet. Die Implementierung des verteilten Algorithmus mittels Mikroservices bietet alle Vorteile einer lokalen Ausführung im global-definierten Anwendungskontext an, inkl. der Erfüllung der wesentlichen Anforderungen an Portabilität, Vernetzung, Koordination, Energieeffizienz und sonstige wichtige Eigenschaften des umgesetzten Simulations-systems.

Entwicklung Digitaler Zwillinge von umwelttechnischen Systemen

Viele Ökosysteme benötigen umfassende Modelle, die ihre Dynamik darstellen können. Mithilfe der Modelle lassen sich die komplexen Zusammenhänge zwischen den Bestandteilen des Systems analysieren, verschiedene Einflüsse von äußeren Faktoren (inkl. der menschlichen) betrachten, Steuerungsmaßnahmen planen, u.v.m. Die für Ökosysteme bestehenden Modelle werden oft auf der Basis weniger Messreihen

[53] Institut für Produktionstechnik und Automatisierung, https://www.ipa.fraunhofer.de/
[54] https://www.evia.de/

entwickelt, die ihre Komplexität nur ungenügend abbilden können. Viele Modelle sind deswegen veraltet und unter aktuellen Bedingungen gar nicht mehr brauchbar. Die neueren Ansätze bauen dagegen auf flexible Modelle, die anhand von Daten, die aus den Objekten mittels Smart-Sensoren bezogen werden, in Echtzeit korrigiert und validiert werden können.

Das ISWA-Institut[55] betreibt eine experimentelle Kläranlage (siehe Beitrag von Otto [R114]), mithilfe welcher die Anforderungen an industrielle Systeme präziser definiert und ihre Einhaltung validiert werden können. Diese Anlage verfügt über ein Modell (Digitaler Zwilling), in welchem die einzuführenden Innovationen und Know-hows in komplexen Simulationsvorgängen virtuell abgespielt werden können, bevor sie im realen System implementiert werden. Um die Qualität der Modellvorhersagen zu erhöhen, indem eine präzise Anpassung an die schwankenden Volumenströme, Konzentrationen von biochemischen Indikatorparametern und weiterer wichtigen Eigenschaften realer Systeme ermöglicht wird, ist eine Kopplung mit Sensoren im Testobjekt notwendig.

Für die praktische Umsetzung des Projektvorhabens wird in jedem Fall eine Softwareplattform benötigt. Mit dem Einsatz von *MS-Sim-Framework* wird es erlaubt, die einfachsten Simulationen an portablen Rechnern unmittelbar am Testsystem durchzuführen und mit dem Hauptmodell des Digitalen Zwillings im Labor zu verkoppeln. Damit sollten Entscheidungswege verkürzt, Verlässlichkeit am Standort geschaffen und ein wirkungsorientierter Steuerungsablauf gesichert werden.

5.4 Zusammenfassung der Ergebnisse

Dieses Kapitel befasst sich mit der Evaluierung der Modelle, die mithilfe des erarbeiteten service-orientierten Ansatzes implementiert wurden. Das primäre Ziel war dabei die Eignungsbewertung von entwickelten Modellen und implementierten Services zum Einsatz unter industriellen Bedingungen realer technologischer Objekte.

Die wichtigsten Erkenntnisse sind wie folgt:

[55] Institut für Siedungswasserbau, Wassergüte- und Abfallwirtschaft der Universität Stuttgart - https://www.iswa.uni-stuttgart.de

- Die Software für komplexe Modelle von topologisch organisierten Objekten lässt sich mit **relativ geringem Entwicklungsaufwand** auf der Basis der erarbeiteten Methodologiegrundlagen der hierarchisch aufgebauten Mikro-service-Architektur **als Services implementieren und in zahlreichen Anwendungsszenarien einsetzen**.

- Die Validierung der Modelle anhand der Musterbeispiele des Referenzobjekts (Bergwerk „Vostochnaja“) sowie auf der Basis der aktuellen Aufgabestellung realer industrieller Objekte (Bergwerk „Süd-Donbass 3“) hat gezeigt, dass die **funktionalen Eigenschaften und Anforderungen (Qualität und Genauigkeit der Ergebnisse) durch die entwickelten Modelle erfüllt sind**.

- Die Evaluierung der Modellimplementierungen in Form von Services einer Mikroservice-Architektur hat ergeben, dass die service-basierte Implementierung auf unterschiedlichen Hardware-Plattformen effektiv ausgeführt werden kann. Unter anderem kann bei der Ausführung auf einem eingebetteten System die **Erfüllung der Echtzeitanforderungen gewährleistet** und bei der Nutzung eines HPC-Systems eine **hohe parallele Effizienz unter maximaler Ressourcenauslastung** erreicht werden.

- Die potenziellen Nutzer der MS-Sim Simulationsplattform (Fachgebiets-experten) haben insbesondere die **einfache Portabilität auf unterschiedliche Hardwarearchitekturen** mit der Möglichkeit zur **Erreichung einer akzeptablen Leistung** von Simulationsanwendungen verzeichnet. Dabei wurden mehrere Anwendungsszenarien für die entwickelten Simulationsservices ausgearbeitet, die bei der Lösung aktueller betrieblicher Aufgabestellungen (unter Vorbehalt der Zustimmung der Sicherheitsaufsichtbehörde) zum Einsatz kommen sollen.

- Abgesehen von den Ziel-Aufgabestellungen des Basisforschungsgebiets (Bewetterungssysteme) zeichnen sich für die entwickelte MS-Sim-Simulationsplattform weitere vielversprechende Einsatzszenarien in aktuellen Forschungs- und Entwicklungsgebieten ab, wie z.B. im Umweltschutz oder der Stadtplanung.

6 Zusammenfassung und Ausblick

Die moderne Simulationstechnik wird mit den Anforderungen der Flexibilisierung und Anpassungsfähigkeit zu den immer dynamischer werdenden industriellen Problemstellungen konfrontiert, mit der Erwartung der Aufweisung von Eigenschaften der erhöhten Portabilität, Verteilungs- und Echtzeitfähigkeit, Energieeffizienz und Leistungsstärke der eingesetzten Simulationssoftware. Hinzu kommen noch zusätzliche softwaretechnische Anforderungen wie die Wiederverwendbarkeit eines Codes bei der Entwicklung der Simulationsanwendungen für komplexe, hierarchisch organisierte Produktionsobjekte oder die Unterstützung während der Ausführung seitens einer speziellen Plattform, welche die Aufgaben des Deployments, der Ausführung sowie der Datenverwaltung unterstützen soll. Die bisher etablierten Programmieransätze halten diesen Herausforderungen nur sehr beschränkt stand. Vielmehr werden Konzepte einer eventbasierten Softwarearchitektur, service-orientierten Entwicklung und andere fortgeschrittene Technologien gefordert, die eine durchgehende Adaption der Simulationscodes an die Dynamik des Industrie-umfelds unterstützen sollen.

Im Rahmen der vorliegenden Arbeit wurde der Versuch unternommen, einen service-orientierten Ansatz zur Simulationssoftwareentwicklung auf der Basis der Mikro-servicearchitektur zu erarbeiten. Dabei wurde der Entwicklung einer Simulations-anwendung das Vorgehensmodell zu Grunde gelegt, welches auf der funktionellen Komposition vieler hierarchisch strukturierter, loskoppelter und über die heterogene Infrastruktur verteilter (Simulations-) Mikroservices basiert. Jeder Mikroservice implementiert einen autonomen Teil des Gesamtmodells von physikalischen Prozessen oder Systemfunktionalitäten. Durch die Implementierung der funktionalen und informationstechnischen Abhängigkeiten zwischen den Services können innerhalb einer Mikroservice-Anwendung komplexe Simulationsszenarien realisiert werden.

Die Hauptinnovationen (erläutert in den Kapiteln 3 und 4 der Arbeit) des angestrebten Ansatzes können wie folgt zusammengefasst werden:

- Der entwickelte **hierarchische Ansatz** ermöglicht eine einheitliche Systematisierung der Modelle für komplexe, hierarchisch aufgebaute dynamische Systeme nach den Prinzipien der Zugehörigkeit zu vertikalen Hierarchieebenen und horizontalen Tätigkeitsbereichen der gesamten Modellhierarchie.

- Mithilfe eines elaborierten **service-orientierten Programmiermodells** können mit geringen Programmieraufwand loggekoppelte, plattformunabhängige und skalierbare Simulationsservices als Komponenten der Modellhierarchie für die Modelle entwickelt werden.

- Die Anwendung der ausgearbeiteten Konzepte einer (i) **funktionalen Erweiterung** und (ii) einer **Servicekomposition** zusammen mit der neuentwickelten und vorher noch nicht bekannten (iii) **Injektionstechnik** erlaubt die Entwicklung von Services für komplexe, zusammengesetzte Modelle verschiedener hierarchischer Ebenen.

- Das erarbeitete MIMD-basierte **Anwendungsmodell** bietet ein effektives Instrument zur Erstellung der Simulationsanwendungen auf der Basis der entwickelten Services an.

- Das auf der Basis von *OpenMPI* entwickelte **Kommunikationsmodell** erlaubt (aufgrund eines indirekten Verfahrens) einen einfach realisierbaren aber gleichzeitigen effizienten Datenaustausch zwischen den Services der Mikro-service-Anwendungsimplementierung.

- Das entwickelte Konzept und die Implementierung des **dezentralisierten Monitorings** ermöglicht eine effektive Speicherung und Verwaltung der Mikro-service-Daten auf den Ressourcen einer verteilten *Elasticsearch*-Datenbank.

- Der Prototyp einer Plattform zur Implementierung der service-orientierten Simulationsanwendungen wurde der Community als freizugängliches Framework (*MS-Sim*) zur unbegrenzten Nutzung zur Verfügung gestellt.

Die erarbeiteten Konzepte wurden zur Entwicklung der wichtigsten Modelle des Basisforschungsgebiets verwendet – der Bewetterungsnetze des Bergbaus, die im Kapitel 4 der Arbeit präsentiert sind. Diese Modelle zeichnen sich durch eine hohe Komplexität der mathematischen Darstellung (z.B. Verteilung der Parameter, Nichtlinearität usw.) sowie durch vieldimensionale Zusammenhänge zwischen den Objekten der Hierarchie aus. Die durchgeführte Klassifizierungsarbeit sowie der Umfang der durchgeführten aufwändigen Implementierungsarbeiten sind für den Basisbereich vorher nicht bekannt gewesen.

Die entwickelten Services und Anwendungen wurden anhand der Modellierungs-Aufgabestellungen von Test- und realen Objekten validiert (Kapitel 5). Die Validierungsergebnisse bestätigten eine ausreichend hohe funktionale Genauigkeit der Vorhersagen durch die implementierten Services. In der darauffolgenden Evaluierung der nichtfunktionalen Leistungs- (Performance, Skalierbarkeit) sowie Nutzungs- (Bedienbarkeit) Eigenschaften wurden für die service-basierten Modellimplementierungen ebenfalls eine gute Ergebnisqualität nachgewiesen. Unter anderem wurde die Brauchbarkeit der Systeme mit leistungsschwachen Prozessoren für die Erfüllung einer Echtzeitanforderungen festgestellt. Darüber hinaus wurde auf einem HPC-System eine hohe parallele Effizienz der service-basierten Simulations-anwendung unter maximaler Ressourcenauslastung erreicht. Die potenziellen Nutzer der MS-Sim-Simulationsplattform (Fachgebietexperten) haben insbesondere die einfache Portabilität auf unterschiedliche Hardwarearchitekturen mit der Möglichkeit der Erreichung einer akzeptablen Leistung von Simulationsanwendungen verzeichnet.

Zu den erwünschten Nachbesserungen der aktuellen MS-Sim-Implementierung gehört die Reduzierung der Aufbaukomplexität des Deployment-Modells für alle Services einer Mikroservice-Anwendung. Diese Operation verfügt allerdings über ein großes Automatisierungspotenzial und wird als wichtiger Teil in die weiteren Arbeiten einfließen. Eine weitere große Erweiterung des MS-Sim-Frameworks sollte in Richtung der Fehlertoleranz-Erhöhung erfolgen, da das Basis-Kommunikations-framework (OpenMPI) aktuell nur eine sehr geringere Unterstützung diesbezüglich anbietet.

Die insgesamt sehr positive Bilanz der gesammelten Erfahrungen beim Einsatz des Konzepts der service-orientierten Simulation auf die Problemstellungen der Industrieobjekte des Basisforschungsgebiets sollte um weitere Anwendungsgebiete erweitert werden. Einige Beispiele der potenziellen Anwendungsszenarien wurden bereits für Bereiche wie den Umweltschutz und die Stadtplanung ausgearbeitet.

Alle am Anfang der Arbeit gesetzten Zielsetzungen (Kapitel 2.5) wurden erfolgreich erreicht.

Referenzverzeichnis

Alle Literaturverweise und Referenzen, die in den obigen Arbeitsinhalten verwendet wurden, sind in zwei Kategorien eingeteilt: Referenzen, die aus gedruckten Literaturquellen stammten – [**R***x*], und diejenige, die aus den öffentlichen Web-quellen entnommen wurden – [**W***y*].

Gedruckte Literaturquellen

[R1] L. Feldmann et al. *Forschungsgebiet: Parallele Simulationstechnik*. Tagungsband der 3. Internationalen Konferenz „Simulation und Computergraphik – 2009“. Donezk, 7-9 Oktober 2009. S. 139–168.

[R2] T. Junker. *Die Evolution des Menschen*. C.H.Beck, Science, 2016.

[R3] P. James and N. Thorpe. *Ancient Inventions*. Ballantine Books, New York, 1994.

[R4] G. Agricola. *Vom Bergwerk*. Froben, Basel, 1557. Nachdruck Essen, Verlag Glückauf, 1985.

[R5] I. Stewart. *Welt-Formeln. 17 mathematische Gleichungen, die Geschickte machten*. Rowohlt Taschenbuch Verlag. 6. Auflage. 2014.

[R6] R. Lauber und P. Göhner. *Prozessautomatisierung 1*. Springer, 3. Auflage. 1999.

[R7] W. Halang et al. *Perspektiven der Informatik in der Echtzeitverarbeitung*. Informatik-Spektrum 16 (1993), S. 357–362.

[R8] V. Svjatnyj. *Modellierung der aerodynamischen Prozesse und Entwicklung der Steuerungssysteme von Grubenbewetterungsnetzen*. Dissertationarbeit zur Erlangung der Würde eines Doktor-Ingenieurs. Donezker Politechnisches Institut, 1985 (*auf Russisch*).

[R9] A. Schappei. *Angebotsperspektiven des Kohleweltmarkts unsicher. Die deutsche Steinkohle. Fakten, Analysen, Argumente*. EU-JRC-Studie (41), Juni 2007.

[R10] F. Gründger et al. *Microbial methane formation in deep aquifers of a coal-bearing sedimentary basin*. Front. Microbiol.6, 2015. S. 1–14.

[R11] H.-M. Schulz et al. *From shale oil to biogenic shale gas: Retracing organic–inorganic interactions in the Alum Shale (Furongian–Lower Ordovician) in southern Sweden*. AAPG Bulletin 99, 2015. S. 927–956.

[R12] E. Clausen. *Wettertechnik im 21.Jahrhundert – Entwicklung adaptiver wettertechnischer Systeme*. Mining Report Glückauf, 153 (4). 2017. S. 326–332.

[R13] B. Widanarko et al. *Risk Factors Associated with Work-Related Fatigue Among Indonesian Mining Workers*. Proceedings of the 20th Congress of the International Ergonomics Association (IEA 2018). 2018. S. 1029.

[R14] G. Hanf. *Modellierung und Simulation instantionärer Grubenbewetterung auf verteilten Rechnerarchitekturen*. VDI Verlag, Düsseldorf, 2002.

[R15] Abramov et al. *Modellierung dynamischer Prozesse in der Bergbauerologie*. Kiew, Naukowa Dumka. 1981 (*auf Russisch*).

[R16] M. Denker. *Einführung in die Analysis dynamischer Systeme*. Springer, Berlin u. a. 2005.

[R17] H. Busche. *Betriebliche Maßnahmen gegen die Gefahren des Grubengases durch Überwachung der Bewetterung*. Glückauf 99 (1963), Nr. 10. S. 504–512.

[R18] E. Cayirci. *Modeling and simulation as a cloud service: a survey*. In IEEE Simulation Conference (WSC), 2013 Winter. S. 389–400.

[R19] A. Cheptsov et al. *Towards Deployment of the Remote Instrumentation e-Infrastructure (in the Frame of the DORII Project)*. Computational Methods in Science and Technology, 15(1), 2009. S. 65–74.

[R20] M. Broy. *Cyber-Physical Systems. Innovation durch Software-Intensive Eingebettete Systeme*. Springer Verlag, 2010.

[R21] B. Vogel-Heuser. *Herausforderungen und Anforderungen aus Sicht der IT und der Automatisierungstechnik*. In: Bauernhansl T. et al. (eds), Industrie 4.0 in Produktion, Automatisierung und Logistik. Springer Vieweg, Wiesbaden. 2014.

[R22] M. Wollschlaeger et al. *The Future of Industrial Communication: Automation Networks in the Era of the Internet of Things and Industry 4.0*. In IEEE Industrial Electronics Magazine, vol. 11, no. 1, March 2017. S. 17–27.

[R23] J. Kay et al. *Industrial Ethernet - overview and best practices*. Conference Record of 2014 Annual Pulp and Paper Industry Technical Conference, Atlanta, GA, 2014. S. 18–27.

[R24] M. Metter und R. Bucher. *Industrial Ethernet in der Automatisierungstechnik*. 2. wesentlich überarb. u. erw. Auflage, 2007.

[R25] T. Leyrer und M. Adzan. *Selecting the right industrial communications standard for sensors*. Texas Instrumental, 2018.

[R26] A.D.S. Gillies. *Real Time Integrated Mine Ventilation Monitoring.* Proc. Queensland Mining Industry Health & Safety Conference 2003. S. 133–140.

[R27] Q. Liu und Y. Li. *Modbus/TCP based Network Control System for Water Process in the Firepower Plant*. Proc. 6th World Congress on Intelligent Control and Automation, Dalian, 2006. Ss. 432 – 435.

[R28] T. Wienbruch et al. *Evolution of SMEs towards Industrie 4.0 through a scenario based learning factory training*. Proc. Manufacturing, Volume 23, 2018. Ss. 141–146.

[R29] B. Schneider et al. *Evaluating Software-defined Networking for Deterministic Communication in Distributed Industrial Automation Systems*. IEEE International Conference on Emerging Technologies and Factory Automation (ETFA). 2017.

[R30] M. Paavola und K. Leiviska. *Wireless sensor networks in industrial automation*. In Factory Automation, Silvestre-Blanes, ed. intech, 2010.

[R31] S. Basu et al. *Fire monitoring in coal mines using wireless underground sensor network and interval type-2 fuzzy logic controller*. In International Journal of Coal Science Technology (6). 2019. S. 274–285.

[R32] J. Song et al. *Automatic monitoring system for coal mine safety based on wireless sensor network.* Proc. 2011 Cross Strait Quad-Regional Radio Science and Wireless Technology Conference, Harbin. 2011. S. 933–936.

[R33] Y Wei et al. *Constructing of wireless emergency communication system for underground coalmine based on WMN technology*. Journal of Coal Science & Engineering, Vol. 16, No. 4. Dec. 2010. S. 441–448.

[R34] C. Stewart et al. *Use of live sensor data in transient simulations of mine ventilation models*. Mining Report 153 (4/2017). S. 356–363.

[R35] T. Müller. *Aufbau eines Strömungsversuchsstandes und Durchführung von Versuchen zur Untersuchung von auftretenden Effekten bei der vertikalen Förderung von Feststoff-Wasser-Gemischen in einem Großversuchsstand über 136 m Höhe im Rahmen des europäischen Forschungsprojektes „Blue Mining“*. Kolloquium Fördertechnik im Bergbau, Technische Universität Clausthal, 2018.

[R36] D. Brady. *The role of gas monitoring in the prevention and treatment of mine fires.* In Coal 2008 – Coal Operators' Conference, University of Wollongong, Australia, 2012. Ss. 340–344.

[R37] K.-D. Walter. *Sensordaten für die ML-Trainingsphase*. Design & Elektronik, 12/2019, S. 34–37.

[R38] N. Ignatovych. *Perspectives of implementation of automated contol system UTAS in coal-mines*. Problems of work safety in Ukraine, 17(2009). S. 3–16 *(auf Ukrainisch).*

[R39] W. Zdanovskyi. *Use of modern diagnostic systems to increase the coal mining safety*. Problems of work safety in Ukraine, 19(2010). S. 44–56 *(auf Ukrainisch).*

[R40] R. Zhong et al. *An IoT-enabled Real-time Machine Status Monitoring Approach for Cloud Manufacturing*. Proceedings CIRP, Volume 63, 2017. S. 709–714.

[R41] J. Byungwan et al. *An Internet of Things System for Underground Mine Air Quality Pollutant Prediction Based on Azure Machine Learning*. Sensors 18(4), 2018. S. 930.

[R42] C. Stergiou. *Secure integration of IoT and Cloud Computing.* Future Generation Computer Systems, Volume 78, Part 3, 2018. S. 964–975.

[R43] S. Sathavara. *A Survey Paper on Secure Integration of IoT and Cloud Computing using Agent Mechanism*. GRD Journals- Global Research and Development Journal for Engineering, Volume 4, Issue 4. March 2019. S. 2455–5703.

[R44] D. Koelemeijer et al. *A Model-based Approach to Certification of Adaptive MILS*. International Workshop on MILS: Architecture and Assurance for Secure Systems, MILS@DSN 2018, Jun 2018, Luxembourg, Luxembourg.

[R45] W. Zdanovskyi. *Some aspect of coal mining safety.* Problems of work safety in Ukraine (34). 2018. S. 21–30 *(auf Ukrainisch).*

[R46] M. Kluge. *PC und SPS kombiniert*. Computer & Automation. Fachmedium der Automatisierungstechnik, 7/8-2019. S. 38–41.

[R47] T. Böhler. *MES – Enabler einer digitalen Fertigung*. Computer & Automation. 11/2019. S. 28–30.

[R48] F. Riemenschneider. *Erster GHz-Mikrocontroller der Welt von NXP*. Elektronik. 23/2019. S. 48–51.

[R49] A. Bergbauer. *100-Watt-Ökosystem für Edge-Server*. Elektronik, 22/2019. S. 20–24.

[R50] T. Fernández-Caramés et al. *A Fog Computing Based Cyber-Physical System for the Automation of Pipe-Related Tasks in the Industry 4.0 Shipyard*. Sensors (18). 2018.

[R51] P. Fraga-Lamas et al. *Towards the Internet of Smart Trains: A Review on Industrial IoT-Connected Railways. Sensors* (17). 2017.

[R52] K. Zeman et al. *Modelbildung und Simulation – eine permanente Herausforderung auf dem Weg zur cyber-physischen Produktion*. BHM Berg- Hüttenmann. Monatshefte, Bd. 161, Nr. 11, 2016. S. 532–538.

[R53] M. McPherson. *Subsurface ventilation and environmental engineering*. Springer Science & Business Media, 2012.

[R54] D. Brake. *Ventilation Challenges Facing the Metalliferous Sector*. In: The Austalian mine ventilation conference. Carlton, Victoria, The Australian Institute of Mining and Metallurgy, 2013. S. 3–12.

[R55] M. Grieves and J. Vickers. *Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems*. In: F. Kahlen et al. (eds), Transdisciplinary Perspectives on Complex Systems. Springer, Cham, 2017. S. 85–113.

[R56] W. Yan et al. *Integrated simulation and emulation platform for cyber-physical system security experimentation*. Proceedings of the 1st international conference on High Confidence Networked Systems, 2012. S. 81–88.

[R57] E. Lee. *The Past, Present and Future of Cyber-Physical Systems: A Focus on Models*. Sensors 15(3). 2015. S. 4837–4869.

[R58] O. Moldovanova. *Ways of organizing parallel computations in the problems of modeling mine venting networks*. Dissertationsarbeit zur Erlangung der Würde eines Doktor-Ingenieurs. DonNTU, Donezk, 2008 *(auf Ukrainisch).*

[R59] O. Solonin. *Optimal organization of modelling processes in parallel modelling environment*. Dissertationsarbeit zur Erlangung der Würde eines Doktor-Ingenieurs. IPME, Kiew, 2007 *(auf Ukrainisch).*

[R60] E. Schmeyer. *Numerische Verfahren zur Simulation von Mehrphasenströmungen mittels Populationsbilanzen*. Dissertation zur Erlangung des Gradeseines Doktors der Naturwissenschaften (Dr. rer. nat.) am Fachbereich Mathematik und Informatik der Freien Universität Berlin. 2012

[R61] E. Clausen und A. Agasty. *Anwendung eines Hierarchischen Ansatzes für die gezielte Verdünnung von unter Tage auftretenden Gasen*. Mining Report Glückauf (152/2). 2016. S. 150–160.

[R62] A. Gärtner. *Die Berechnung der Wetterströmung in verzweigten Grubengebäuden.* Glückauf 63 (1927), Nr. 48, S. 1741–1747 und Nr. 49, S. 1777–1787.

[R63] W. Hu and I.Longson. *The optimization of airflow distribution in ventilation networks using a nonlinear programming method*. Mining Science and Tehcnology (10), 1990. S. 209–219.

[R64] S. Shcundin, A. Ivannikov. *The interventilation net nodes method elaboration for nets modelling in regular and emergency situations*. MIAB – Mining Informational and Analytical Bulletin (1), 2010. S. 448–459 *(auf Russisch).*

[R65] J. Jingwei et al. *Computer simulation of evacuation in underground coal mines*. Mining Science and Technology (China), Volume 20, Issue 5, 2010. S. 677–681.

[R66] P. Tantsov. *Actuality of dynamic calculation of mine ventilation networks*. Coal of the world (3), 2012. S. 345–349 *(auf Russisch).*

[R67] C. Feuchter. *Lattice-Boltzmann Methods and their Application*. CFD Conference 2013, Behr GmbH, Stuttgart.

[R68] D. Brake. *Fire Modelling in Underground Mines using Ventsim Visual VentFIRE Software.* Australian Mine Ventilation Conference/Adelaide, SA, Australia, 2013. S. 265–276.

[R69] R. Fujimoto. *Parallel and distributed Simulation*. Proc. Winter Simulation Conference 1999. S. 122-13.

[R70] S. Laflamme. *Applying Parmetis to Structured Remeshing For Industrial CFD Applications.* International Journal of High Performance Computing Applications, 17(1). 2003. S. 63–76.

[R71] T. Feroze und B. Gene. *Ventilation of underground coal mines - A Computational Fluid Dynamics study*. MVSSA Annual Conference 2016: Increasing the relevance of the Mine Ventilation Profession. S. 89–95.

[R72] S. Aminossadati und K. Hooman. *Numerical simulation of ventilation air flow in underground mine workings*. 12th U.S./North American Mine Ventilation Symposium. 2008. S. 1–7.

[R73] A. Gameiro Lopes. *A Versatile Software Tool for the Numerical Simulation of Fluid Flow and Heat Transfer in Simple Geometries*. Computer Applications in Engineering Education, 18/2009. S. 14–27.

[R74] G. Xu et al. *Computational fluid dynamics applied to mining engineering: a review*. International Journal of Mining, Reclamation and Environment. 31(4), 2016. S. 251–275.

[R75] T. Pieper and R. Obermaisser. *Distributed co-simulation for software-in-the-loop testing of networked railway systems.* Proc. 7th Mediterranean Conference on Embedded Computing (MECO 2018), Budva, Montenegro, 2018. S. 1–5.

[R76] K. Hopkinson et al. *Epochs: a platform for agent-based electric power and communication simulation built from commercial off-the-shelf components*. Proc. IEEE Trans-actions on Power Systems, 21(2), 2006. S. 548–558.

[R77] Ochel et al. *OMSimulator – Integrated FMI and TLM-based Co-simulation with Composite Model Editing and SSP*. Proceedings of the 13th International Modelica Conference. 2019. S. 69–78.

[R78] M. Madhu und D. Sunanda. *Distributing Messages Using RabbitMQ with Advance Message Exchanges*. International Journal of Research Studies in Computer Science and Engineering (IJRSCSE), 6(2), 2019. S. 24–28.

[R79] P. Dobbelaere und K. Esmaili. *Industry Paper: Kafka versus RabbitMQ*. Proc. DEBS ’17, Barcelona, Spain, 2017. S. 227–238.

[R80] F. Haußer, Y. Luchko. *Mathematische Modellierung mit MATLAB und Octave. Eine praxisorientierte Einführung.* Springer, 2019.

[R81] C. Kolassa et al. *Objektorientierte Graphendarstellung von Simulink-Modellen zur einfachen Analyse und Transformation*. In: Tagungsband AALE 2013, 10. Fachkonferenz, Das Forum für Fachleute der Automatisierungstechnik aus Hochschulen und Wirtschaft, München, 2013. S. 277–286.

[R82] S. Röck et al. *Simulationsbasierte Methoden für die Automatisierungstechnik in der Produktion.* SRC SimTech, Issue No. 2010-48.

[R83] Trenkel K. und Spiteller F. *Sensorsimulation in Hardware-in-the-Loop-Anwendungen*. In: Halang W., Unger H. (eds) Industrie 4.0 und Echtzeit. Informatik aktuell. Springer Vieweg, Berlin, Heidelberg, 2014. S. 51–60.

[R84] A. Plummer: *Model-in-the-Loop Testing*. In: Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering, 220(3), 2006 (4). S. 183–199.

[R85] A. Smagin et al. *Simulation und Prozessführung in einer parallelen problemorientierten Simulationsumgebung*. 18. Symposium ASIM, Erlangen, September 12-15, 2005. S. 742–747.

[R86] K. Waleed. *Advantages and Disadvantages of Using MATLAB/ode45 for Solving Differential Equations in Engineering Applications*. International Journal of Engineering (IJE), Volume (7), Issue (1), 2013. S. 25–31.

[R87] P. Saha et al. *Evaluation of Docker Containers for Scientific Workloads in the Cloud*. In Proceedings of the Practice and Experience on Advanced Research Computing (PEARC '18). ACM, New York, NY, USA, Article 11, 2018.

[R88] U. Meissen et al. *Eine Mikroservice-basierte Referenzarchitektur für interoperable, flexible und robuste Warnsysteme*. In: Tagungsband des 25. Workshops "Umweltinformationssysteme 2018 - Umweltdaten - in allen Dimensionen und zu jeder Zeit? (UIS 2018)" des Arbeitskreises "Umweltinformationssysteme" der Fachgruppe "Informatik im Umweltschutz" der Gesellschaft für Informatik (GI), 2018. S. 209–224.

[R89] J. Weyer und M. Roos. *Agentenbasierte Modellierung und Simulation*. TATuP Zeitschrift für Technikfolgenabschätzung in Theorie und Praxis, 26(3), 2017. S. 11–16.

[R90] A. Grahn et al. *Implementation of a Pressure Drop Model for the CFD Simulation of Clogged Containment Sump Strainers*. Journal of Engineering for Gas Turbines and Power-transactions 132(8), 2010.

[R91] R. Collier et al. *MAMS: Multi-Agent MicroServices*. In Companion Proceedings of The 2019 World Wide Web Conference (WWW '19), Ling Liu and Ryen White (Eds.). ACM, New York, NY, USA, 2019. S. 655–662.

[R92] T. Uhlemann et al. *The Digital Twin: Demonstrating the Potential of Real Time Data Acquisition in Production Systems*. Procedia Manufacturing, Volume 9, 2017. S. 113–120.

[R93] H. Khoo et al. *Integrated Simulation and Modelling Approach to Decision Making and Environmental Protection*. Environment, Development and Sustainability 3, 2001. S. 93–108.

[R94] J. Frochte. *Evaluation and Adaptation of Techniques for Higher Index DAE with Respect to Real-Time Simulation*. SNE Simulation Notes Europe, 2012. S. 45–52.

[R95] P. Aivaliotis et al. *Methodology for enabling Digital Twin using advanced physics-based modelling in predictive maintenance*. Procedia CIRP, Volume 81, 2019, S. 417–422.

[R96] M. Hirsch. *Systematic Design of Distributed Industrial Manufacturing Control Systems*. Dissertation zur Erlangung des akademischen Grades Doktor-Ingenieur. Hallenser Schriften zur Automatisierungstechnik (6), Halle (Saale), 2010.

[R97] A. Deckert und A. Klein. *Agentenbasierte Simulation zur Analyse und Lösung betriebswirtschaftlicher Entscheidungsprobleme*. Journal Betriebswirtsch 60, 2010. S. 89–125.

[R98] D Masak. *Digitale Ökosysteme. Serviceorientierung bei dynamisch vernetzten Unternehmen*. Springer-Verlag Berlin Heidelberg, 2009.

[R99] E. Gabriel et al. *Open MPI: Goals, Concept, and Design of a Next Generation MPI Implementation*. In Proceedings, 11th European PVM/MPI Users' Group Meeting. 2004. S. 97–104.

[R100] J. Montañana und A. Hervas. *Towards energy efficient computing based on the estimation of energy consumption by monitoring the computing resources used*. In: Proc. 30th Workshop on Sustained Simulation Performance (WSSP), 09-10th of October 2019, Stuttgart.

[R101] W. Li et al. *Quality assurance for component based systems in embedded environments*. In IEEE International Conference on Internet of Things, Embedded Systems and Communications (IINTEC2018), December 2018

[R102] A. Chorin, J.-E. Marsden. *A Mathematical Introduction to Fluid Mechanics.* Springer Verlag, 2000

[R103] L. Feldmann. *Untersuchung der Dynamik und Synthese der Automatisierungssysteme für Grubenbewetterung*. Dissertationarbeit zur Erlangung der Würde eines Doktor-Ingenieurs, Donezk, 1974 *(auf Russisch).*

[R104] C. Stewart. *Underground fire rollback simulation in large scale ventilation models.* (2015).

[R105] H. Brücher und R. Endl. *Erweiterung von UML zur geschäftsregelorientierten Prozessmodellierung*. In: Becker J., Knackstedt R. (eds) Wissensmanagement mit Referenzmodellen. Physica, Heidelberg, 2002. S. 145–161.

[R106] B. Rumple. *Agile Modeling with UML: Code Generation, Testing, Refactoring.* Springer. 2017.

[R107] P. Menghal and A. Laxmi. *Real time simulation: Recent progress & challenges*. Proc. Int. Conf. Power, Signals, Controls Comput. (EPSCICON), Thrissur, India, Jan. 2012, S. 1–6.

[R108] R. Gheorghe et al. *Elasticsearch in Action.* Springer. Taschenbuch – 3. 2015.

[R109] W. Li et al. *A Survey on Model-Based Testing Tools for Test Case Generation*. Proc. 4th International Conference on Tools and Methods of Program Analysis. 2018.

[R110] A. Cheptsov. *The system organization and basic algorithms of the simulation and servicing center for the coal industry*. IEEE Proceeding International Conference "Modern Problems of Radio Engineering, Telecommunications and Computer Science TCSET'2007", 2007. S. 205–207.

[R111] J.-M. Andre et al. *A Scalable Online Monitoring System Based on Elasticsearch for Distributed Data Acquisition*. In The European Physical Journal Conferences 214(3), 2019.

[R112] V. Babu et al. *Energy Saving Techniques for Ventilation Fans Used in Underground Coal Mines - A Survey*. Journal of Mining Science, Vol. 51, No. 5, 2015. S. 1001–1008.

[R113] C. Pritchard. *Methods to improve efficiency of mine ventilation systems*. Trans Soc Min Metall Explor (326), 2010. S. 34 –38.

[R114] N. Otto. Spectrometric detection and photo-oxidative removal of formaldehyde in aqueous solutions. Proceedings of the 11th IWA Micropol & Ecohazard Conference 2019. S. 718–719.

Web-Quellen

[W1] *TOP500 – die Liste von 500 leistungsstärksten Rechner der Welt*. Stand vom November 2019. Verfügbar unter: https://www.top500.org/. Zuletzt geprüft am: 1.04.2021.

[W2] T. Bauernhansl et al. *Industrie4.0*. Whitepaper FuE-Themen, 2014. Verfügbar unter: https://www.zvei.org/fileadmin/user_upload/Presse_und_Medien/Publikationen/2014/april/Industrie_4.0__Whitepaper_zu_Forschungs-_und_Entwicklungsthemen/Industrie-40-Whitepaper-Forschung-20140403.pdf. Zuletzt geprüft am: 1.04.2021.

[W3] A.-W. Scheer. *Industrie 4.0: From vision to implementation.* Whitepaper, 2016. Verfügbar unter: https://www.researchgate.net/profile/August_Wilhelm_Scheer/publication/281447305_Whitepaper_-_Industry_40_From_vision_to_implementation/links/55e803ae08aeb6516262f188/Whitepaper-Industry-40-From-vision-to-implementation.pdf. Zuletzt geprüft am: 1.04.2021.

[W4] *Die nächste Phase der Energiewende kann beginnen*. Bundesministerium für Wirtschaft und Innovationen (BMWI), 2019. Verfügbar unter: http://www.bmwi.de/Redaktion/DE/Dossier/energiewende.html. Zuletzt geprüft am: 1.04.2021.

[W5] *Daten und Entwicklungen der deutschen und globalen Energieversorgung. BGR Energiestudie 2018*. Bundesanstalt für Geowissenschaften und Rohstoffe (BGR), 2019. Verfügbar unter: https://www.bgr.bund.de/DE/Themen/Energie/Downloads/energiestudie_2018.pdf?__blob=publicationFile&v=10. Zuletzt geprüft am: 1.04.2021.

[W6] *Kennzahlen des deutschen Steinkohlenbergbaus 1957 bis 2018*. Statistik der Kohlenwirtschaft e. V. Verfügbar unter: https://gvst.de/wp-content/uploads/2019/07/GVST_Steinkohlenbergbau_Kennzahlen_1957-2018.pdf. Zuletzt geprüft am: 1.04.2021.

[W7] *World Energy Balances 2019.* Internationales Energy Programm (IEA), 2020. Verfügbar unter: https://www.iea.org/statistics/coal/.

[W8] B. Kavalov. E. Peteves. *The future of coal*. EU-JRC-Studie, 2007. Verfügbar unter: http://publications.jrc.ec.europa.eu/repository/bitstream/JRC36671/6671%20EUR22744EN.pdf. Zuletzt geprüft am: 1.04.2021.

[W9] *Liste von größten Unglücken im Bergbau*. Wikipedia, 2020. https://de.wikipedia.org/wiki/Liste_von_Ungl%C3%BCcken_im_Bergbau. Zuletzt geprüft am: 1.04.2021.

[W10] *Green500 – die Liste von 500 energie-effizientesten HPC-Rechner der Welt*. Stand vom November 2019. Verfügbar unter: https://www.top500.org/green500/lists/2019/11/. Zuletzt geprüft am: 1.04.2021.

[W11] V. Koch. *Industrie 4.0. Chancen und Herausforderungen der vierten industriellen Revolution.* PwC Studie, 2014. Verfügbar unter: https://www.strategyand.pwc.com/de/de/studien/industrie-4-0.pdf. Zuletzt geprüft am: 1.04.2021.

[W12] *VHF Data Link*. Wikipedia. Verfügbar unter: https://de.wikipedia.org/wiki/VHF_Data_Link. Zuletzt geprüft am: 1.04.2021.

[W13] *Registered Jack*. Wikipedia. Verfügbar unter: https://de.wikipedia.org/wiki/RJ-Steckverbindung

[W14] *EtherNet/IP Network Devices. User Manual*. Rockwell Automation Publication ENET-UM006A-EN-P - March 2019, 2019. Verfügbar unter: https://literature.rockwellautomation.com/idc/groups/literature/documents/um/enet-um006_-en-p.pdf. Zuletzt geprüft am: 1.04.2021.

[W15] *Schneider Modbus RTU TCP/IP Device Specification*. PCL Connection Guide, Schneider Electric. Verfügbar unter: https://euroec.by/assets/files/weintek/plc_connection_guide/Schneider_MODBUS_TCP_IP.pdf. Zuletzt geprüft am: 1.04.2021.

[W16] *EtherCAT – ultra-fast communication standard*. Beckhoff Automation, 2019. Verfügbar unter: https://download.beckhoff.com/download/document/catalog/Beckhoff_EtherCAT_e.pdf. Zuletzt geprüft am: 1.04.2021.

[W17] Z. Lin, S. Pearson. *An inside look at industrial Ethernet communication protocols*. Texas Instrumental, July 2018. 2018. Verfügbar unter: http://www.ti.com/lit/wp/spry254b/spry254b.pdf?ts=1590420658291. Zuletzt geprüft am: 1.04.2021.

[W18] Web-Seite vom Industrie 4.0 Forschungsprojekt ADAPTATION. Verfügbar unter: https://adaption-projekt.de. Zuletzt geprüft am: 1.04.2021.

[W19] Web-Seite des BMBF-Projekts Basissystem Industrie 4.0 (BASYS). Verfügbar unter: https://www.basys40.de/. Zuletzt geprüft am: 1.04.2021.

[W20] *Rückblick vom Forum Bergbau 4.0.* DMT, 2016. Verfügbar unter: https://www.dmt-group.com/fileadmin/redaktion/documents/DMT_Events/DMT_Forum_Bergbau_Einladung_2015_lepo_RZ.pdf. Zuletzt geprüft am: 1.04.2021.

[W21] J. Rodig. *Erfolgreiche IoT-Geschäftsmodelle. Chancen im Internet der Dinge und der Industrie 4.0 nutzen*. White Paper, 2017. Verfügbar unter: http://fs-media.nmm.de/ftp/ITI/ITP/files/vortraege/2017/jan-rodig-tresmo.pdf. Zuletzt geprüft am: 1.04.2021.

[W22] C. Kleijn. *Introduction to Hardware-in-the-Loop Simulation*. Controllab Products, 2017. Verfügbar unter: http://www.hil-simulation.com/images/stories/Documents/Introduction%20to%20Hardware-in-the-Loop%20Simulation.pdf. Zuletzt geprüft am: 1.04.2021.

[W23] Spezifikation vom Movidius' Myriad2 Embedded System. Movidius, 2017. Verfügbar unter: https://www.movidius.com/solutions/vision-processing-unit. Zuletzt geprüft am: 1.04.2021.

[W24] Spezifikation vom parallelen System-on-Chip Zynq-7000. XILINX, 2017. Verfügbar unter: https://www.xilinx.com/products/silicon-devices/soc/zynq-7000.html. Zuletzt geprüft am: 1.04.2021.

[W25] S. Gries et al. *Simulation. Cyber Physische Systeme*. Analyse, 2017. Verfügbar unter: https://cps-hub-nrw.de/knowledgebase/publikation/3681-simulation. Zuletzt geprüft am: 1.04.2021.

[W26] R. Rannacher. *Finite Element Methods for Navier-Stokes Equation*. Lecture notes on theoretical and numerical fluid mechanics in Vancouver, 2019. Verfügbar unter: https://ganymed.math.uni-heidelberg.de/Oberwolfach-Seminar/CFD-Course.pdf. Zuletzt geprüft am: 1.04.2021.

[W27] Message-Passing Interface Standart. MPI Forum, 2020. Verfügbar unter: https://www.mpi-forum.org/. Zuletzt geprüft am: 1.04.2021.

[W28] Anleitung zur Installation von OpenFOAM Toolboxes mittels eines Docker-Containers. OpenFOAM, 2020. Verfügbar unter: https://www.openfoam.com/download/install-binary-linux.php. Zuletzt geprüft am: 1.04.2021.

[W29] A. Sharma. *Apache Kafka: Next Generation Distributed Messaging System*. Architecture & Design, 2014. Verfügbar unter: https://www.infoq.com/articles/apache-kafka/ Zuletzt geprüft am: 1.04.2021.

[W30] W. Regulski. *CFD Modeling using MATLAB*. Mathworks, 2018. Verfügbar unter: https://blogs.mathworks.com/racing-lounge/2018/06/20/cfd-modeling-using-matlab/. Zuletzt geprüft am: 1.04.2021.

[W31] M. Fowler. *MicroServices: A definition of this new architectural term*. 2014. Verfügbar unter: https://martinfowler.com/articles/microservices.html. Zuletzt geprüft am: 1.04.2021.

[W32] MS-Sim: Referenzplattform zur Implementierung der Simulationsservices anhand OpenMPI. Verfügbar auf GitHub unter: https://github.com/alexey-cheptsov/MS-Sim

[W33] Definition der Echtzeitsimulation. Wikipedia, 2020. Verfügbar unter: https://en.wikipedia.org/wiki/Real-time_simulation. Zuletzt geprüft am: 1.04.2021.

[W34] S. Miller et al. *Echtzeitsimulation physikalischer Systeme mit Simscape.* MathWorks, 2020. Verfügbar unter: https://de.mathworks.com/company/newsletters/articles/real-time-simulation-of-physical-systems-using-simscape.html. Zuletzt geprüft am: 1.04.2021.

[W35] Spezifikation des Odroid-XE4-Systems. Pollin. Verfügbar unter: https://www.pollin.de/p/odroid-xu4-einplatinen-computer-samsung-exynos-5422-2-gb-2x-usb-3-0-810409. Zuletzt geprüft am: 1.04.2021.

[W36] Spezifikation des HLRS-Systems Vulcan (Stand – Anfang 2020). Verfügbar unter – https://www.hlrs.de/systems/nec-cluster-vulcan/. Zuletzt geprüft am: 1.04.2021.

Lebenslauf

Curriculum Vitae — M. Sc. Alexey CHEPTSOV

Date of birth: ***1 September 1980***

Place of birth: ***Donezk, Ukraine***

Contact: cheptsov@gmail.com

General Information	
Employment history :	**01/2008 - present** – Researcher at the High-Performance Computing Center Stuttgart (HLRS), University of Stuttgart (Prof. Michael Resch) ▪ Management, conduction, and technical coordination of EU and national projects (currently – PHANTOM and HPC-Europa EU-H2020 projects, in the past – JUNIPER, DreamCloud, LarKC, DORII, Int.EU.Grid EU-FP7 projects); ▪ Project acquisition (lately, H2020-ICT4 with 3 proposals, 1 of those - funded); ▪ Support of lectures on topics related to parallel programming and simulation. 09/2005 – 12/2007 – Associate professor at the Department of Computer Science, Donetsk National University of Technology (Prof. Vladimir Svjatnyj) ▪ Conduction of lectures on Parallel Simulation Technologies. 10/2004 – 08/2005 – Research scientist at the Institute of Industrial Automation and Software Engineering, University of Stuttgart (Prof. Peter Göhner) ▪ Development of extension for UML-2 standard in the frame of a DAAD-funded project grant. 9/2002 – 09/2004 – Research fellow at the Department of Computer Science, Donetsk National University of Technology (Prof. Gudula Rünger) ▪ Support of lectures on Hardware Engineering and System Programming
Qualification and Education :	12/2003 – 11/2006 – Kandidat Technichnih Nauk (at the Research Institute of Simulation Problems in Power Engineering of the National Academy of Science of Ukraine), with honour. Topic of the thesis: "The Simulation and Service Center for the Coal Industry". Supervisor: Prof. Dr.-Ing. Vladimir Svjatnyj 09/1997 – 07/2002 – Master of Computer Science (at Donetsk National University of Technology, Department of Computer Science), with honour 09/2001 – 01/2005 – Practice on system programming at Chemnitz University of Technology 09/1999 – 07/2003 – Master of Economics (in Donetsk National University of Technology, Economical Faculty), 2nd education of choice
Languages:	Ukrainian (native), Russian (native), German (C2), English (C2)
Research Experience and Competence	
Professional and research interests :	▪ Research topics: Simulation Technologies for Engineering Sciences, Big Data, Data Management, Parallelisation, Performance Optimization, Semantic Web, Reasoning, Energy-Efficiency of HPC Infrastructures. ▪ Programming languages: Java, C, C++ (major), C# (secondary), Fortran (basics) ▪ Parallelisation techniques: MPI, OpenMP, PGAS, MapReduce/Hadoop, CUDA ▪ E-Infrastructures: Cloud (Open Stack and Open Nebula), Grid (globus, gLite) ▪ Performance analysis and application optimisation: Zabbix, Scalasca, Vampir ▪ Other: Embedded systems and FPGA technologies with Xilinx tools

Stuttgart, 04/2021