Energieforschungsprogramm

Publizierbarer Endbericht

Programmsteuerung:
Klima- und Energiefonds (KLIEN) und Bundesministerium für Klimaschutz, Umwelt, Energie, Mobilität, Innovation und Technologie (BMK)

Programmabwicklung:
Österreichische Forschungsförderungsgesellschaft mbH (FFG)

Endbericht
erstellt am

31/12/2025

Projekttitel:DES-CERT

Projektnummer:57285140

Ausschreibung	2024. Ausschreibung Energieforschungsprogramm
Projektstart	01/01/2025
Projektende	31/12/2025
Gesamtprojektdauer (in Monaten)	12 Monate
ProjektnehmerIn (Institution)	AIT Austrian Institute of Technology GmbH
AnsprechpartnerIn	Denis Vettoretti
Postadresse	Giefinggasse 2, 1210 Wien
Telefon	050 5500
Fax	...
E-mail	denis.vettoretti@ait.ac.at
Website	https://www.ait.ac.at/

DES-CERT

Scalable Digital Energy Systems: Evaluation and Testing towards Precertification

AutorInnen:
Denis Vettoretti

Clemens Korner

Catalin Gavriluta

Barbara Herndler

Filip Pröstl Andren

Mihai-Teodor Stanusoiu

1 Inhaltsverzeichnis

2 Einleitung

2.1 Ziel

Der Elektrizitätssektor stellt eine zentrale Komponente moderner Infrastruktur dar und wird voraussichtlich eine zunehmend zentrale Rolle im europäischen Energiesystem einnehmen. Im Kontext des European Green Deal, der eine Reduktion der Treibhausgasemissionen um 55 % bis 2030 und Klimaneutralität bis 2050 anstrebt, werden zusätzliche Sektoren – insbesondere Verkehr, Wärme und Industrie – umfassend elektrifiziert, während der Anteil erneuerbarer Erzeugung weiter zunimmt. Diese Kombination aus steigender Nachfrage und grundsätzlich variabler erneuerbarer Einspeisung wird die Flexibilitätsanforderungen zukünftiger Stromsysteme erheblich erhöhen.

Gleichzeitig entwickelt sich das Smart Grid zu einem hochkomplexen cyber-physischen System (CPS), das durch hohe Durchdringungen von Distributed Energy Resources (DERs), flexibler Nachfrage und Möglichkeiten zur aktiven Steuerung gekennzeichnet ist. Etwa 60 % der gesamten steuerbaren erneuerbaren Erzeugung werden voraussichtlich in virtuellen Kraftwerken aggregiert sein, und mehr als 65 Millionen Elektrofahrzeuge werden bis 2050 auf Smart Charging- oder Vehicle-to-Grid-Dienste angewiesen sein[2].

Konventionelle Validierungspraktiken, die primär auf Offline-Simulationen und isolierten Geräteprüfungen basieren, sind für dieses aufkommende Paradigma unzureichend. Sie vernachlässigen das ganzheitliche Systemverhalten und erfassen nicht die Phänomene, die aus Wechselwirkungen zahlreicher verteilter Regler und Kommunikationsabhängigkeiten entstehen. Das Projekt DES-CERT adressiert diese Herausforderung durch die Entwicklung skalierbarer und automatisierter Methoden zur Präzertifizierung digitaler Energiesysteme. Zu den zentralen Zielen gehört die Erreichung hoher Genauigkeit und hoher Skalierbarkeit in einer reproduzierbaren und automatisierten Weise, sodass glaubwürdige Nachweise für Präzertifizierungsprozesse erzeugt werden können.

2.2 Projekt-Fokus

Das Projekt konzentriert sich auf Anwendungsfälle, die mit Analysen überlagerter Regler im Stromnetz zusammenhängen und auf Zeitskalen von Sekunden bis Stunden operieren. Dazu gehört Sekundärregelleistung, flottenweite Koordination vom Laden von Elektrofahrzeugen, Bewertung des Nutzens für Energiegemeinschaften sowie Untersuchungen der strategischen Netzplanung. Szenarien mit inneren Regelschleifen wie fault ride-through oder virtuelle Schwungmasse liegen außerhalb des primären Projektfokus, das in diesem Projekt erstellte Framework ist jedoch so ausgelegt, dass es in Richtung solcher Szenarien erweiterbar ist.

Das DES-CERT-Projekt untersucht eine Reihe von Orchestrierungslösungen, die Container-Orchestratoren, Workflow-Engines und ereignisgesteuerte Skalierungsmechanismen umfassen, sowie mehrere Power-System-Modelling-and-Analysis (PSMA)-Tools, die sich in ihrer Modelltreue, Performance-Eigenschaften und Schnittstellen unterscheiden. Diese Werkzeuge werden hinsichtlich Automatisierungsmöglichkeiten, Skalierbarkeit, Interoperabilität, einfacher Integration über standardisierte APIs und Übereinstimmung mit CI/CD (Continuous Integration/Continuous Deployment)-getriebenen Vorvalidierungs-Workflows bewertet.

2.3 Einordnung innerhalb des Programms

DES-CERT ist im Energieforschungsprogramm verankert, das vom Klima- und Energiefonds (KLIEN) und dem Bundesministerium für Klimaschutz, Umwelt, Energie, Mobilität, Innovation und Technologie (BMK) verwaltet wird. Das Projekt trägt unmittelbar zu den Zielen des Programms bei, den Übergang zu einem sicheren, nachhaltigen und digitalen Energiesystem zu unterstützen, indem Methoden und Werkzeuge für die systematische Validierung und Präzertifizierung digitaler Energieservices entwickelt werden.

2.4 Verwendete Methoden

Das Projekt nutzt eine Kombination aus Literaturrecherche, Einbindung von Stakeholdern, Technologieevaluierung und Prototypentwicklung. Eine Stakeholder-Befragung wurde durchgeführt, um Erwartungen, Herausforderungen und Erfahrungen im Zusammenhang mit aktuellen Validierungs- und Zertifizierungspraktiken von Verteilernetzbetreibern, Technologieanbietern und Forschungseinrichtungen in ganz Europa zu erfassen. Auf technischer Seite wurden Orchestrierungstools (Kubernetes, Docker Swarm, Terraform) und PSMA-Tools (pandapower, ANDES, GridLAB-D) Benchmark-Tests und Kompatibilitätsbewertung mit containerisierten Workflows evaluiert.

Die zentrale technische Entwicklung ist das Automated Cyber-Physical Testing and Validation (ACTV)-Framework, das Containerisierung (Docker), Orchestrierung (Kubernetes) und ein einheitliches Cyber-Physical Model (CPM) auf Basis eines erweiterten Common Information Model (CIM) nutzt. Das Framework wurde durch Use-Case-Experimente an Benchmark-Netzen des Simbench-Datensatzes validiert.

3 Darstellung des Inhalts

3.1 Stand der Technik

Abbildung 1 zeigt eine qualitative Bewertung bestehender Ansätze zum Testen und Validieren von Anwendungen auf Komponentenebene (z. B. einer neuen Regelungsstrategie für einen leistungselektronischen Wechselrichter oder einen Generator). Historisch bestand eine erhebliche Lücke zwischen Offline-Simulationen und Feldtests, wodurch der iterative Prozess zwischen Entwurf und Testen langsam und ressourcenintensiv war. Neuere Entwicklungen in der RTDS-Technologie haben diese Lücke jedoch deutlich verringert. Zwar sind die Ressourcenanforderungen für diese Echtzeitlösungen erheblich höher als für rein simulationsbasierte Tests, sie bleiben aber für die praktische Anwendung durchführbar. Infolgedessen hat sich ein intermediäres HIL-Testing, umgesetzt als Controller Hardware-in-the-Loop (C-HIL), Power Hardware-in-the-Loop (P-HIL) oder einer Kombination davon, zum Stand der Technik für hochauflösende Validierung auf Komponentenebene [4] entwickelt und ist ein zentrales Bauelement für strukturierte Präzertifizierungs-Workflows.

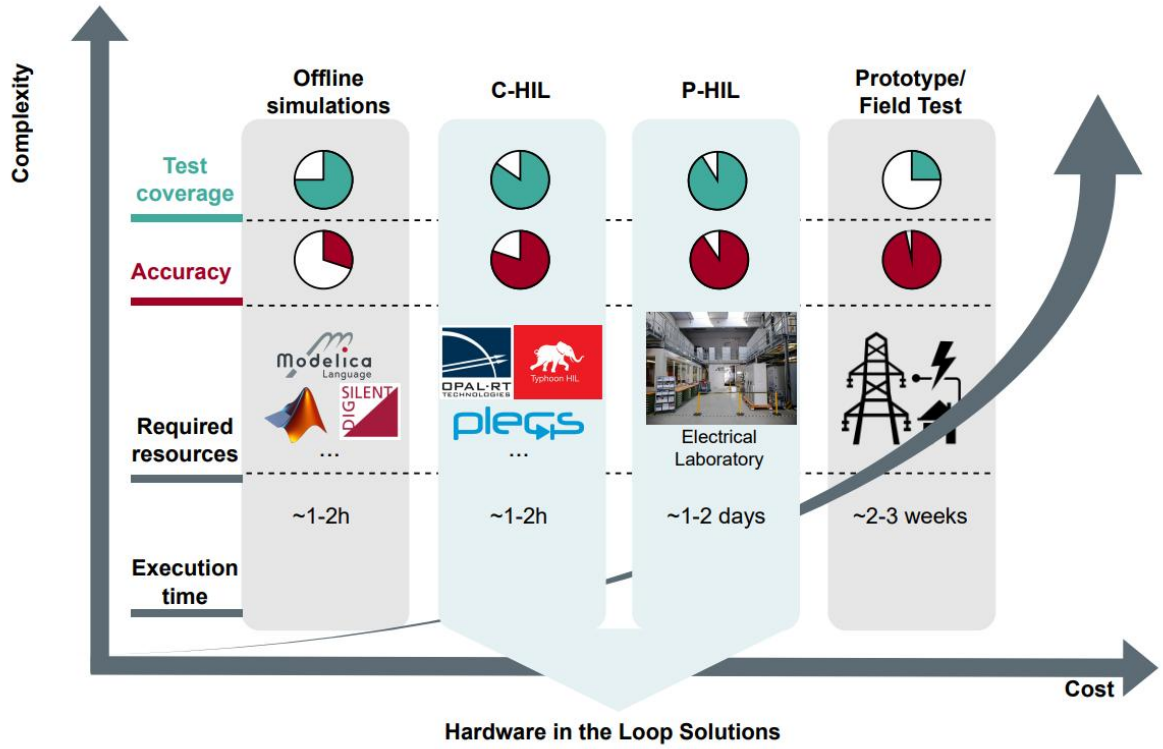

Abbildung 1 Ansätze für das Testen und Validieren von Anwendungen auf Komponentenebene.

Offline-Simulationen sind am besten geeignet, um Kernkonzepte zu validieren und deren grundlegende technische Machbarkeit zu bewerten. In finalen Anwendungen stellen diese Kernideen jedoch typischerweise nur einen kleinen Bruchteil des gesamten Software- und Hardware-Stacks dar. Dagegen ermöglichen C-HIL- und P-HIL-Experimente eine deutlich breitere Testabdeckung und erfassen den vollständigen Software- und Hardware-Stack einschließlich Nebenmodulen und deren komplexe Wechselwirkungen. Aus Sicht der Testabdeckung liefern Feldtests die umfassendsten Erkenntnisse, wobei HIL-Tests dicht dahinter rangieren. In bestimmten Fällen bietet HIL sogar Vorteile gegenüber Labor- oder Feldexperimenten, da wichtige, aber wenig wahrscheinliche Szenarien mithilfe von RTDSs sicher und realistisch emuliert werden können. Offline-Simulationen bieten bei geringster Testabdeckung die größte Flexibilität: Modellaufbau, Szenarienerzeugung und Testorchestrierung lassen sich weitgehend automatisieren. Diese Flexibilität nimmt generell ab, je näher man an vollumfängliche Feldtests heranrückt.

Auf Systemebene wurde ein vergleichbares Reifegradniveau noch nicht erreicht, wie in Abbildung 2 dargestellt. Es fehlt weiterhin an effizienten und generalisierbaren Methodiken zur umfassenden Validierung großskaliger, softwareintensiver Smart-Grid-Anwendungen unter realistischen Bedingungen vor der Feldbereitstellung. Übersichten zu Smart-Grid-CPS-Testbeds [5,6] heben durchgehend den Bedarf an realistischen, skalierbaren und wiederverwendbaren Plattformen hervor. Zwar wurden verschiedene existierende Lösungen vorgeschlagen, die von reinen Software-Simulationen und hardwarebasierten Testbeds bis hin zu Echtzeit- und Hybridansätzen reichen, doch kommen diese Studien zu dem Schluss, dass die meisten Plattformen projektspezifisch oder auf bestimmte Use-Cases zugeschnitten sind und sich häufig nur auf Cyber-Security-Aspekte fokussieren. Dieser Mangel an Allgemeingültigkeit und systematischer Wiederverwendbarkeit schränkt ihre Eignung als Grundlage für standardisierte Präzertifizierungs-Frameworks wie jene, die in DES-CERT vorgesehen sind, ein.

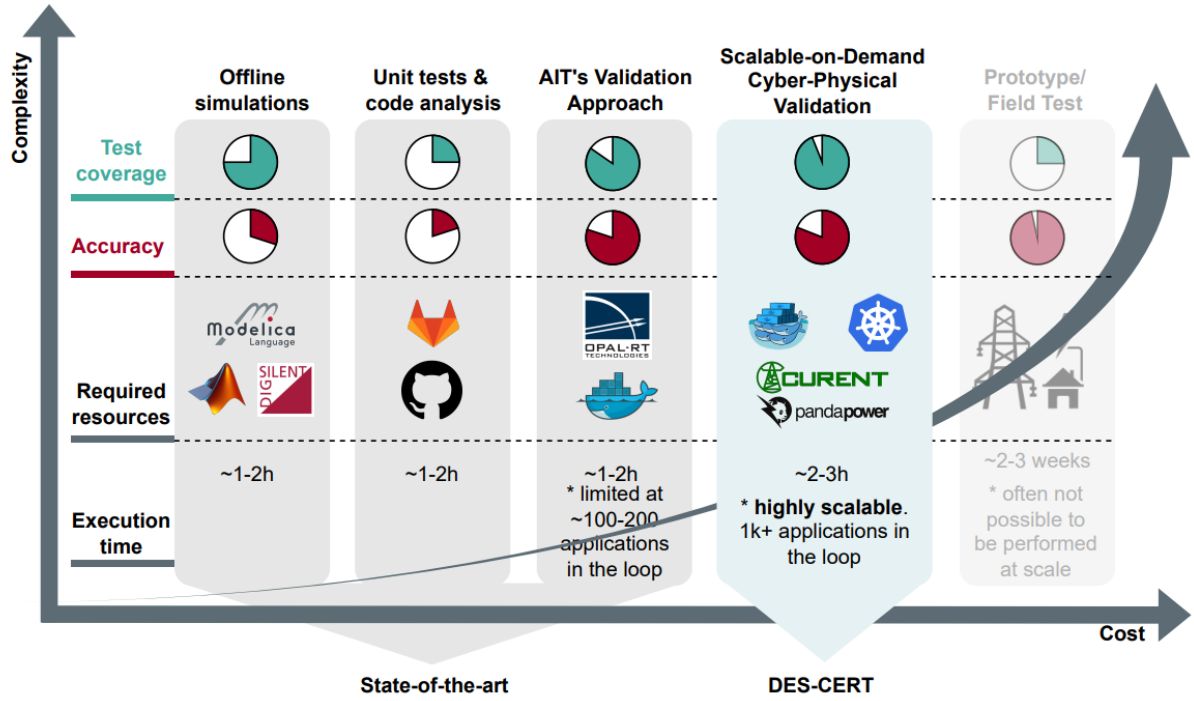

Abbildung 2 Verfügbare Ansätze für das Testen und Validieren von Anwendungen auf Komponentenebene.

3.1.1 Co-Simulations-Frameworks

Co-Simulations-Frameworks orchestrieren mehrere Simulationswerkzeuge, um eine integrierte Analyseumgebung bereitzustellen. Zahlreiche solcher Ansätze wurden in den letzten Jahren entwickelt. Der Functional Mock-up Interface (FMI)-Standard unterstützt Co-Simulation über Functional Mock-up Units (FMUs), und Werkzeuge wie PyFMI [7] ermöglichen abstrahierte Co-Simulationsplattformen. Zusätzlich wurden mehrere Co-Simulationsumgebungen speziell für Anwendungen im Bereich elektrischer Energiesysteme vorgeschlagen. Zu den in den letzten Jahren am weitesten verbreiteten gehören HELICS [8] und Mosaik [9]. Das Mosaik-Ökosystem wird überwiegend für Lastfluss- und Energiemanagementstudien verwendet, während HELICS eine breitere Palette von Anwendungen adressiert, einschließlich dynamischer und multidomänen Analysen.

Für HELICS wurde gezeigt, dass es auf große und detaillierte gekoppelte Übertragungs- und Verteilernetz-Modelle skalierbar ist. Beispielsweise zeigt [10] einen Anwendungsfall mit mehr als 160 000 Knoten und über 10 000 detaillierten Wechselrichtermodellen, der auf der Hochleistungsrecheninfrastruktur des Pacific Northwest National Laboratory eingesetzt wurde, um die Auswirkungen netzbildender Wechselrichter zu untersuchen. In [11] wird HELICS als Kopplungsschicht zwischen GridLAB-D und NS3 eingesetzt, um die Cyber-Resilienz in einem 9500-Knoten-Netz zu untersuchen. Ähnlich stellt [12] eine Cyber-Sicherheitsbewertungsplattform vor, in der Mosaik verwendet wird, um pandapower mit virtuellen Edge-Geräten zu integrieren. Diese Frameworks bieten wertvolle Fähigkeiten für Systemebenenanalysen, und ihre Skalierbarkeit und Interoperabilität machen sie zu möglichen Kandidaten für die Einbindung in DES-CERT-konforme Validierungstoolchains.

3.1.2 Hardware-in-the-Loop

HIL-basierte Plattformen stellen eine weitere wichtige Klasse von Ansätzen zur Modellierung und Simulation cyber-physischer Energiesysteme dar. Ob als C-HIL, P-HIL oder rein RTDS-basierte Emulation der physischen Schicht umgesetzt, legen diese Testbeds Wert auf hohe Genauigkeit sowohl im Energie- als auch im Cyber-Bereich. Die Cyber-Seite kann reale Netzwerkstacks, Firewalls, Sicherheitsmechanismen und sogar realistische Angreifer

umfassen und unterstützt damit eine umfassende CPS-Validierung.

In [13] wird eine Plattform zur Validierung großskaliger DER-Koordinationsstrategien vorgestellt, bei der ein RTDS zur Modellierung des Übertragungsnetzes der Vermont Electric Power Company verwendet wird. Während diese Konfiguration eine hohe Genauigkeit für die physische Schicht bietet, ist die Cyber-Darstellung vereinfacht, um Skalierbarkeit zu ermöglichen: DERs werden als C++-Threads modelliert, die auf einer Hochleistungsrechenplattform ausgeführt werden. Andere Ansätze, wie in [14], [15] und [16] beschrieben, zeigen Testbeds, die um RTDSs in Kombination mit Flotten kostengünstiger Einplatinencomputer aufgebaut sind. Obwohl diese Plattformen hohe Genauigkeit erreichen, ist ihre Skalierbarkeit begrenzt; typische Studien beschränken sich auf wenige Dutzend Einheiten (z. B. 27 Controller in [14]), was ihre Anwendbarkeit für großskalige Systemvalidierung und Präzertifizierung einschränkt.

Um Skalierbarkeitsbeschränkungen zu überwinden, schlagen neuere Arbeiten [17,18] containerbasierte Ansätze in Verbindung mit RTDSs vor. In beiden Fällen werden die Plattformen anhand relativ kleiner Fallstudien demonstriert, die sich auf Stromnetze auf Schiffen konzentrieren. Die physische Schicht umfasst ein detailliertes Modell eines Mittelspannungs-AC/DC-Hybridnetzes mit Synchrongeneratoren, AC-Lasten, AC/DC-Wandlern und DC-Lasten. Auf der Cyber-Schicht werden Software-Applikationen als Docker-Container instanziiert, die über eine auf NS3 basierende Emulation des Informations- und Kommunikationstechnologie-(IKT-)Netzwerks miteinander verbunden sind. Eine deutlichere Illustration der durch Containerisierung ermöglichten Skalierbarkeit wird in [19] präsentiert, wo 400 virtuelle Intelligent Electronic Devices (IEDs) als Docker-Container auf einem einzigen Server bereitgestellt werden.

3.1.3 Geographically Distributed Testbeds (GDS)

Geographically Distributed Testbeds (GDS) verbinden mehrere Labore oder Simulatoren über Kommunikationsnetze zu einer einzigen, virtuellen großskaligen Testumgebung. Ziel ist es, die Ressourcenbeschränkungen einzelner Labore durch die Föderation ihrer Ressourcen zu überwinden; zum Beispiel durch die Kopplung von RTDS-Installationen an verschiedenen Standorten, die jeweils einen Teil eines großen Stromnetzes in zeitlich synchronisierter Weise simulieren. GDS-Ansätze wurden untersucht, um weitreichende Steuerungs- und Schutzschemata zu validieren, die sich über mehrere Regionen erstrecken. Prinzipiell können GDS-Konfigurationen sowohl hohe Genauigkeit (durch detaillierte lokale Modelle und HIL-Integration an jedem Standort) als auch hohe Skalierbarkeit (durch Aggregation vieler Knoten über Standorte hinweg) erreichen.

Trotz dieses Potenzials blieb die praktische Verbreitung von GDS begrenzt, vor allem aufgrund von Bedenken hinsichtlich Stabilität, Latenz und Konsistenz der Ergebnisse. Eine präzise Synchronisation geografisch verteilter Simulatoren ist technisch herausfordernd: Kommunikationsverzögerungen und Uhrendrift an den Schnittstellen können Instabilitäten oder Ungenauigkeiten verursachen [20]. Die Bewältigung dieser Herausforderungen ist eine Voraussetzung dafür, GDS-Konzepte als Teil eines robusten, standardisierten Präzertifizierungsumgebung für digitale Energiesysteme zu verwenden.

3.1.4 Einordnung unseres Beitrags

Unsere Arbeit stellt einen Ansatz vor, der mehrere offene Herausforderungen bei der Prävalidierung von CPS-Software für intelligente Stromnetze adressiert und die Ziele des DES-CERT-Projekts direkt unterstützt, das skalierbare und systematische Methoden zur Präzertifizierung digitaler Energiesysteme etablieren möchte. Im Gegensatz zu Validierungsplattformen, die für hochrealistische Hardware-in-the-Loop-Experimente auf RTDS setzen, ist unser Ansatz explizit um moderne Software-Orchestrierungstechnologien, insbesondere Kubernetes, herum konzipiert. Diese Designentscheidung spiegelt den DES-CERT-Fokus auf frühphasige, softwarezentrierte Prävalidierung wider und nicht auf spätere

Labor- oder Feldtests.

Unser Framework nutzt Containerisierung und Orchestrierung, um cyber-physische Testumgebungen vollständig automatisiert und reproduzierbar bereitzustellen, zu konfigurieren und zu betreiben. Im Unterschied zu bestehenden Ansätzen, die umfangreiches manuelles Aufsetzen von Simulatoren, Schnittstellen und experimentspezifischen Konfigurationen erfordern, ermöglicht die vorgeschlagene Lösung End-zu-End-Automatisierung und integriert sich stark in zeitgemäße Softwareentwicklungs-Workflows. Dies reduziert den Aufwand zum Durchführen großer Mengen systematischer Validierungsexperimente erheblich und bringt den Prävalidierungsprozess in Einklang mit etablierten DevOps-Praktiken — ein Aspekt, der in der aktuellen Literatur weiterhin unzureichend adressiert ist.

Wir führen das Konzept einer cyber-physischen Staging-Umgebung ein, die in einem Kubernetes-Cluster instanziiert, skaliert und wieder abgebaut werden kann. Diese Staging-Umgebung bietet eine sichere, aber realistische Sandbox für die Bewertung digitaler Energieservices vor deren Einsatz in realen Infrastrukturen. Entscheidend ist, dass sie so gestaltet ist, dass sie sich nahtlos in Continuous Integration- und Continuous Deployment-(CI/CD)-Pipelines integrieren lässt. Dadurch können neue Softwareversionen automatisch umfassende Systemtests unter repräsentativen Netzbedingungen auslösen und reproduzierbare sowie prüfbare Nachweise erzeugen, die sich gut für DES-CERT-artige Präzertifizierungs-Workflows eignen.

Ein zentrales Element dieser Automatisierung ist die Verwendung eines standardisierten cyber-physischen Datenmodells, basierend auf einer erweiterten Version des Common Information Model (CIM). CIM dient als semantische Basis des Testbeds und beschreibt Netztopologie, elektrische Komponenten, Assets und deren Beziehungen in einer herstellerneutralen und interoperablen Art und Weise. Konfigurationsparameter wie Leitungsimpedanzen, Transformatorbewertung, Lastprofile oder Geräteeigenschaften werden konsistent innerhalb des CIM-basierten Modells ausgedrückt und können direkt von den bereitgestellten Softwarediensten und Simulatoren abgefragt werden. Dieser modellgetriebene Ansatz verbessert die Konfigurierbarkeit, Nachverfolgbarkeit und Wiederverwendbarkeit erheblich: Neue Komponenten oder Algorithmen können durch Verwendung des CIM-Schemas integriert werden, ohne ad-hoc-Datenabbildungen oder proprietäre Integrationslogik verwenden zu müssen.

Wie in Abbildung 2 dargestellt, nimmt das vorgeschlagene Framework eine Zwischenposition zwischen rein softwarebasierten Tests und vollumfänglicher Labor- oder Feldvalidierung ein. Es zielt nicht darauf ab, RTDS-basierte oder Hardware-in-the-Loop-Lösungen zu ersetzen, bietet jedoch ein hohes Maß an Automatisierung, Skalierbarkeit und Realitätsnähe, das für frühe und mittlere Validierungsstadien gut geeignet ist. Der strukturierte End-to-End-Workflow unterstützt eine effiziente Konfiguration, Bereitstellung und Ausführung komplexer Szenarien und senkt damit die Einstiegshürde für systematische Prävalidierungskampagnen erheblich. Diese Ausrichtung steht im Einklang mit dem Bedarf nach zugänglichen, cloud-nativen und benutzerfreundlichen Validierungsplattformen, die Automatisierung, Reproduzierbarkeit und Integration mit modernen Softwareentwicklungspraktiken betonen. Im Kontext von DES-CERT stellt unser Framework somit einen konkreten Schritt in Richtung einer standardisierten und automatisierten Roadmap für die Prävalidierung und Präzertifizierung digitaler Energiesystemslösungen dar.

3.1.5 Klassifizierung der Anwendungsfälle und DES-CERT-Fokus

Abbildung 3 stellt eine Klassifikation repräsentativer Simulations- und Validierungsanwendungsfälle über mehrere Dimensionen dar, einschließlich der Steuerungsebene, der Zeitgranularität, des Kommunikationsschemas, der Lastflussabstraktion und unterstützenden Werkzeugen. DES-CERT fokussiert sich bewusst

auf Anwendungsfälle, die mit Analysen auf hoher Steuerungsebene verbunden sind. Diese Anwendungsfälle arbeiten typischerweise auf Zeitskalen von Sekunden bis Stunden, basieren auf asynchronen oder ereignisgesteuerten Kommunikationsmustern und verwenden stationäre oder abstrahierte Netzmodelle. Typische Beispiele sind Strategien für Sekundärregelleistung, der Koordination von Ladeinfrastruktur von Elektrofahrzeugen, Analysen von Energiegemeinschaften und strategische Netzertüchtigungsuntersuchungen.

Niedrigere Steuerungsebenen, wie Fault Ride-Through oder virtuelle Schwungmasse, sind durch Mikrosekunden- bis Millisekunden-Dynamik, synchrone Kommunikation und strikte Echtzeitvorgaben gekennzeichnet und werden daher üblicherweise eher mit RTDS- oder Hardware-in-the-Loop-Plattformen adressiert. Obwohl solche Szenarien außerhalb des primären Anwendungsbereichs von DES-CERT liegen, ist das vorgeschlagene Framework bewusst so konzipiert, dass es über das gesamte in Abbildung 3 dargestellte Spektrum an Anwendungsfällen hinweg erweiterbar bleibt. Durch die Verwendung von Modularität, standardisierten Schnittstellen und Interoperabilität vermeidet das Framework eine starre Kopplung an spezifische Simulationsparadigmen oder Timing-Annahmen. Folglich kann die Architektur, obwohl der aktuelle Fokus auf höheren Steuerungebenen und softwarezentrierter Vorvalidierung liegt, schrittweise erweitert werden, um zusätzliche Klassen von Anwendungsfällen zu unterstützen, einschließlich niedrigerer Steuerungsebenen und hybrider Validierungssetups, wenn sich zukünftige Anforderungen in diese Richtung entwickeln.

Aufbauend auf der oben eingeführten Use-Case-Klassifikation ist der nächste Schritt innerhalb von DES-CERT eine systematische Evaluierung der Basistechnologien, die erforderlich sind, um derartige Szenarien skalierbar und interoperabel zu realisieren. Insbesondere motiviert die in Abbildung 3 hervorgehobene Vielfalt an Zeitskalen und Ausführungsarten eine strukturierte Bewertung sowohl von Orchestrierungswerkzeugen als auch von PSMA-Werkzeugen.

		Explored		Explored
Use-case	**Low-level control**	**High-level control**	**Faster-than-Realtime**	**Parallelize Computation**
Example	Fault-Ride-Through Synthetic Inertia	Secondary Control, Fleet EV Controller	Energy Community Yearly Benefit	Grid Reinforcement
Time granularity	µs, ms	s	min, h	-
Communication schema	Synchronous	Asynchronous	Synchronous, Event	-
Load-flow type	Phasor, EMT	Steady-State	Steady-State	- (Black-Box)
Tools	OpalRT, Typhoon HIL	OpalRT, CRON scheduling	Mosaik, HELICS, PowerFactory, pandapower	Keda, Dagster, Argo WFs
Main challenge	Timing, Communication	Interfaces (API etc.)	Co-simulation	Distributed Computing

Abbildung 3 Klassifizierung von Anwendungsfällen

Ziel dieser Evaluierung ist es nicht, für alle Anwendungsfälle eine einzige optimale Lösung zu identifizieren, sondern die mit unterschiedlichen Toolchains verbundenen Kompromisse

zu verstehen und deren Eignung für die Integration in eine gemeinsame Vorvalidierungsplattform zu bestimmen. Dementsprechend untersucht DES-CERT eine Bandbreite an Orchestrierungslösungen, die von Container-Orchestratoren über Workflow-Engines bis hin zu ereignisgesteuerten Skalierungsmechanismen reichen, sowie mehrere PSMA-Werkzeuge, die sich in Modelltreue, Leistungsmerkmalen und Integrationsschnittstellen unterscheiden. Diese Werkzeuge werden anhand von Kriterien wie Automatisierungsfähigkeiten, Skalierbarkeit, Interoperabilität, Integrationsaufwand über standardisierte APIs und Ausrichtung an CI/CD-getriebenen Vorvalidierungs-Workflows evaluiert. Das Ergebnis dieser Analyse fließt direkt in die Auswahl und Zusammensetzung der Kernkomponenten der Plattform ein und stellt sicher, dass die resultierende Architektur die angestrebten Steuerungs- und schneller-als-Echtzeit-Anwendungsfälle effektiv unterstützen kann, während sie zugleich gegenüber zusätzlichen Szenarien erweiterbar bleibt, während sich die Plattform weiterentwickelt.

3.2 Bewertung moderner Orchestrierungswerkzeuge

Die digitale Transformation der Energiesysteme erfordert Simulations- und Validierungsplattformen, die skalierbar, automatisiert und eng in moderne Software-Engineering-Praktiken integriert sind. Container-basierte Orchestrierung hat sich in diesem Zusammenhang als ein wichtiger Enabler herauskristallisiert, da sie die Replikation komplexer cyber-physischer Umgebungen, die automatische Bereitstellung von Simulationsworkflows und das zuverlässige sowie reproduzierbare Management großer Zahlen heterogener Simulationen ermöglicht. Moderne Orchestrierungsplattformen, allen voran Kubernetes, sind explizit darauf ausgelegt, im großen Maßstäben zu operieren und die Ausführung großer Zahlen gleichzeitiger Jobs über verteilte Rechenressourcen hinweg zu unterstützen, wobei Fehlertoleranz und hohe Verfügbarkeit gewährleistet werden [21]. Cloudbasierte Virtualisierungs- und Orchestrierungstechnologien haben bereits ihr Potenzial zur Verbesserung von Simulationen des Stromsystems demonstriert, etwa durch das Auslagern rechenintensiver Netzüberwachungs- und Steuerungsfunktionen in Cloud-Infrastrukturen [22].

3.2.1 Technologielandschaft

Wir haben führende Orchestrierungs- und Infrastruktur-Tools untersucht, mit Schwerpunkt auf Kubernetes, Docker Swarm und komplementäre Provisioning-Plattformen (z. B. Terraform). Tabelle 1 und die anschließende Diskussion fassen die Merkmale jedes Tools zusammen und untersuchen dabei die Aspekte Skalierbarkeit, Zuverlässigkeit und Integrationsfähigkeit.

- **Kubernetes (K8s):** Ein Open-Source, produktionsreifer Container-Orchestrator der Cloud Native Computing Foundation (CNCF). Er unterstützt deklarative Konfiguration und selbstheilende Workloads. Wichtige Merkmale sind automatisierte Bereitstellung, horizontales Auto-Scaling von Containern, Rolling Updates und Multi-Master-Hochverfügbarkeit. Kubernetes unterstützt konsistente Bereitstellung über On-Premise-, Hybrid- und Cloud-Umgebungen [23]. Sein großes Ökosystem (Helm, Operators, Service-Meshes usw.) ermöglicht Interoperabilität und einfache Integration mit CI/CD-Pipelines [23]. Studien bestätigen, dass Kubernetes für die Verwaltung großskaliger Anwendungen gut verwendbar ist; z. B. stellt eine Analyse fest, dass K8s „a powerful tool for managing large-scale containerized applications“ ist [24]. In Edge- oder IoT-Szenarien wurden Erweiterungen wie K3s oder KubeEdge entwickelt, um Kubernetes für ressourcenbeschränkte Geräte anzupassen [21]. Allerdings wurde angemerkt, dass eine Einschränkung von Kubernetes darin besteht, dass es eine steile Lernkurve hat und erhebliche Cluster-Ressourcen erfordert.

- **Docker Swarm:** Dockers native Clustering-Lösung. Sie ist einfacher einzurichten und

hat einen geringeren Overhead als Kubernetes und hat einen einfachen Docker-Compose–artigen Workflow. Swarm bietet grundlegende Scheduling-, Skalierungs- und Load-Balancing-Funktionen für Docker-Services. Seine Feature-Palette und sein Ökosystem sind jedoch eingeschränkter. In quantitativen Tests auf IoT-Gateways zeigte Swarm langsamere Container-Starts und einen höheren Ressourcenverbrauch als Kubernetes (K8s erreichte eine um 3 s schnellere Container-Erstellung und 14,5 % geringere CPU-Auslastung) [25]. Zusammendfassend bietet Swarms eine einfache Handhabung von kleinen Clustern, aber auch eine begrenzte Robustheit für großskalige Simulationen. Manche HPC-Forschung hat Swarm sogar zur Auto-Skalierung von Rechenknoten erprobt [6], doch bietet Kubernetes im Allgemeinen reichhaltigere Steuerungsmechanismen und breitere Community-Unterstützung.

- **HashiCorp Nomad:** Ein einfacherer Multi-Workload-Orchestrator, der Docker und nicht-containerisierte (raw exec) Jobs nativ unterstützt. Nomad ist horizontal skalierbar und kann sowohl Microservices als auch Batch-Jobs verwalten. Seine Architektur ist im Vergleich zu K8s leichtgewichtiger, was den Betrieb im großen Maßstab erleichtert. Vorteile: Schnelles Scheduling, unkomplizierte Cluster-Einrichtung. Einschränkungen: Kleineres Ökosystem; es fehlen einige Kubernetes-Funktionen wie automatisches Bin-Packing oder umfassende API-getriebene Erweiterbarkeit.

- **Terraform (IaC) and Related Tools:** Terraform ist ein Infrastructure-as-Code-Tool, das Cloud- oder On-Prem-Ressourcen (VMs, Netzwerke, Speicher) bereitstellt. Obwohl es selbst kein Orchestrator ist, kann Terraform die Erstellung der zugrunde liegenden Infrastruktur für einen Simulationscluster automatisieren. Zum Beispiel kann man die Bereitstellung eines Kubernetes-Clusters, von Speichervolumes und Netzwerkregeln per Terraform skripten. Dies gewährleistet die Reproduzierbarkeit der Simulationsumgebung. Terraform ergänzt die Orchestrierung, indem es die Infrastrukturkonfiguration kodifiziert, verwaltet jedoch keine laufenden Container.

Tabelle 1 fasst zusammen, wie diese Werkzeuge mit ausgewählten Anforderungen an Energiesimulationen übereinstimmen.

Tool	**Skalierbarkeit**	**Zuverlässigkeit / HA**	**Integration & Ökosystem**	**Hinweise / Use Cases**
Kubernetes	Extrem hoch (Pods & Nodes); für elastische Cluster konzipiert	Eingebaute Replikation, Multi-Master-HA; selbstheilende Pods	Umfangreiche APIs, Helm-Charts, Service Meshes, native CI/CD-Integration	Bevorzugt für große, containerisierte Workflows; ideal für Microservices
Docker Swarm	Mittel; einfacherer Cluster aus Docker-Hosts	Grundlegende Redundanz	Docker-CLI-Integration; weniger Erweiterungen	Gut für kleine bis mittlere Anwendungen; schnelle Bereitstellung; im IoT getestet
Nomad	Hoch; dezentralisiertes Modell	Unterstützung für Multi-Region-Cluster; Failover	Integration mit Consul, Vault (HashiCorp-Suite); Plugins	Sinnvoll für Kombination aus Container- und

			für Singularity/HPC	Legacy-HPC-Jobs in einem Scheduler
Terraform	N/A (provisioniert Infrastruktur)	Deklarative Provisionierung sorgt für konsistente Setups	Funktioniert mit Cloud-Providern, Kubernetes usw.	Automatisiert Cluster-/Infrastrukturaufbau; stellt Reproduzierbarkeit sicher

Tabelle 1 Vergleich von Orchestrierungs- und Infrastrukturwerkzeugen: Werkzeugauswahl und Funktionsanalyse

Basierend auf Tabelle 1 bleibt Kubernetes die geeignetste Orchestrierungsplattform für großskalige Energiesystemsimulationen aufgrund seiner Skalierbarkeit, Erweiterbarkeit und seines starken Ökosystems. Tiefgehende Kubernetes (K8s)-Tests konzentrierten sich auf Funktionen, die sich für die Erstellung von cyber-physischen Sandbox-Umgebungen eignen:

- **Horizontal Pod Autoscaling**: Kubernetes skaliert Pods automatisch für parallele Simulationskomponenten in Reaktion auf CPU-Metriken und hielt den Durchsatz bei variabler Last aufrecht.

- **Persistent Storage**: Simulierte Geräte und Controller erfordern häufig die Berücksichtigung des Zustands der Applikartion. Kubernetes' persistent volumes (bereitgestellt von netzwerkbasiertem Speicher) ermöglichten es Containern, Simulationsdaten zwischen Neustarts zu behalten, wodurch die Reproduzierbarkeit verbessert wurde.

- **Networking and Service Mesh**: Wir evaluierten Service-Meshes (z. B. Istio), um Microservices sicher zu vernetzen. Diese hatten einen geringen Overhead, boten jedoch wertvolle Funktionen wie mutual TLS und feingranulare Verkehrssteuerung.

- **CI/CD Integration**: Wir verknüpften den K8s-Cluster mit einer GitOps-Pipeline: Codeänderungen an Simulationsmodellen lösten einen automatisierten Build-and-Deploy-Prozess von Docker-Images gefolgt von einem Kubernetes-Rollout aus. Dies bestätigte, dass Container-Orchestrierung zu Continuous-Integration-Workflows passt und Validierungsiterationen beschleunigt [23].

3.2.2 Kompatibilitätsbewertung

Anschließend haben wir die Anforderungen aus dem Energiebereich den Merkmalen der gewählten Orchestrierungsplattform zugeordnet. Die wichtigsten Anforderungen und die K8s-Kompatibilität sind nachfolgend aufgelistet:

- **Skalierbarkeit von Simulationen**: Untersuchungen von Energiesystemen kann eine sehr hohe Anzahl an Szenarien umfassen. Kubernetes unterstützt elastische Skalierung (bedarfsorientiertes Hinzufügen von Nodes/Pods) [23]. Tests bestätigen, dass K8s zuverlässig Hunderte paralleler Job-Pods starten kann; Cloud- oder HPC-Ressourcen können über den Cluster-Autoscaler für Lastspitzen hinzugefügt werden. Als zustandslose Pods ausgelegte Workloads (MPI (Message Passing Interface)-basierte oder Python-Simulatoren) profitierten von der horizontalen Skalierung durch Kubernetes.

- **Hohe Zuverlässigkeit und Resilienz**: Viele Netzsimulationen sind als kritisch einzuordnen; Softwarefehler dürfen gesamte Tests nicht zum Erliegen bringen.

Kubernetes-self-healing (automatisches Neustarten fehlgeschlagener Pods) und Replikationscontroller stellen Ziel-Servicezahlen sicher. In unserem Testbed haben wir Multi-Master-Control-Planes für High-Availability eingesetzt (als empfohlene Praxis) [23]. Die K8s-Control-Plane selbst verwendete den RAFT-Konsensalgorithmus über die Master hinweg und bot Fehlertoleranz. Wir setzten auf eingebaute Liveness-Probes und Readiness-Checks, um nicht reagierende Dienste zu erkennen. Diese Fähigkeiten entsprechen dem Ziel der Resilienz: Der Cluster tolerierte Ausfälle von Worker-Nodes ohne Datenverlust oder signifikante Ausfallzeiten.

- **Interoperabilität mit existierenden Werkzeugen**: Interoperabilität ist eine Schlüsselanforderung in der modernen Energiesystemforschung, in der Simulationen häufig heterogene Werkzeuge und externe Hardware oder Plattformen einbinden. Das System ist so ausgelegt, dass externe Komponenten wie RTDS-Einheiten, spezialisierte Simulatoren oder andere Cloud-Plattformen mit minimaler Anpassung angebunden werden können. Die Kommunikation kann über REST APIs, MQTT oder andere standardisierte Protokolle erfolgen, wodurch flexible Co-Simulationen möglich werden, in denen mehrere Simulationsumgebungen Daten in Echtzeit austauschen. Dieser Ansatz stellt sicher, dass vorhandene Modellierungswerkzeuge und externe Systeme nahtlos integriert werden können, ohne größere architektonische Änderungen zu erfordern.

- **Sicherheit und Isolation**: Cyber-physische Energiesysteme erfordern strikte Sicherheitsgrenzen, kontrollierte Kommunikationsflüsse und eine zuverlässige Handhabung von Zugangsdaten. Kubernetes stellt hierfür Grundbausteine bereit: Namespaces und Network Policies sorgen dafür, dass Simulationsumgebungen isoliert bleiben, während Role-Based Access Control (RBAC) einschränkt, welche Benutzer oder Dienste Workloads bereitstellen, ändern oder einsehen dürfen. Mutual TLS ist für alle API-Server-Kommunikationen aktiviert, sodass sich Cluster-Komponenten authentifizieren und den Datenverkehr Ende-zu-Ende verschlüsseln. Frühere Arbeiten (z. B. die SCADA-Fallstudie in [23]) haben gezeigt, dass Kubernetes auch Übersetzungsdienste hosten kann, die industrielle Protokolle wie DNP3 oder IEC 61850 an moderne Middleware wie Data Distribution Service anbinden.

- **Automatisierung und CI/CD**: Die Architektur ist so konzipiert, dass CI/CD als Dienst angeboten werden kann, wodurch externe Forschungsgruppen oder Institute ihre eigenen Softwarekomponenten innerhalb großskaliger Simulationsszenarien validieren können. Durch CI/CD-Integration können Nutzer neue Software-Releases einreichen und automatisiert Testausführungen in der Simulationsumgebung auslösen. Beispielsweise kann ein Institut, das einen verteilten Algorithmus zur Steuerung der Ladeleistungen von verteilten E-Ladestationen entwickelt, eine neue Version in die CI/CD-Pipeline einspielen, welche dann das Container-Image baut, in den Simulationscluster deployt und vordefinierte großskalige Testszenarien ausführt. Dieser serviceorientierte CI/CD-Ansatz ermöglicht reproduzierbare Validierung, systematischen Vergleich von Softwareversionen und schnelles Experimentieren in großem Maßstab, ohne dass externe Nutzer die Kubernetes-Infrastruktur selbst betreiben müssen. Er verwandelt die Simulationsplattform effektiv in eine gemeinsame Validierungsumgebung, in

der diverse Algorithmen, Agenten und Steuerungsstrategien unter konsistenten und realistischen Bedingungen getestet werden können.

- **Unterstützung für Edge- und verteilte Topologien**: Zukünftige Arbeiten könnten die Integration von Simulationen mit Feldgeräten oder verteilten Testbeds untersuchen. Lightweight-Orchestratoren wie K3s oder KubeEdge ermöglichen Kubernetes-Deployments auf Edge-Nodes [21]. Unsere Architektur ist mit solchen Erweiterungen kompatibel; beispielsweise könnten an entfernten Edge-Standorten deployte Pods Daten in den zentralen Cluster streamen. Wir weisen jedoch darauf hin, dass Public-IP-Beschränkungen die Orchestrierung über Netzgrenzen hinweg erschweren können, welches ein gängiges Problem im Edge-Computing darstellt [21]. Daher würde eine praktische Implementierung sorgfältige Netzkonfigurationen wie VPNs oder Proxy-Lösungen erfordern.

3.2.3 Umgebungsbeschreibung

Um den Wartungsaufwand für die Infrastruktur zu reduzieren, wurde entschieden, Kubernetes nicht selbst zu hosten. Stattdessen wird ein verwalteter Kubernetes-Dienst genutzt, nämlich Azure Kubernetes Service (AKS). Azure stellt einen vollständig verwalteten und skalierbaren Kubernetes-Cluster für den Betrieb verteilter Simulations-Workloads bereit. Container-Images werden über Docker Hub gespeichert und verwaltet und sich nahtlos in AKS zum Herunterladen und Bereitstellen von Simulationsmodulen integriert. AKS profitiert von Azures umfangreichem Cloud-Ökosystem und bietet eine Vielzahl von On-Demand-Ressourcen wie elastischer Rechen-Skalierung, verwalteten Speicheroptionen, automatisierten Backups und integrierten Sicherheitsfunktionen einschließlich Netzwerkrichtlinien, Key Vault-Integration und einem Schutz vor externen Angriffen. Diese Fähigkeiten gewährleisten eine stabile, sichere und flexible Umgebung, ohne dass das zugrunde liegende Cluster-Management erforderlich ist. Um einen Vendor-Lock-in zu reduzieren und Portabilität zu erhalten, wurde die Nutzung proprietärer Azure-Dienste, wo immer möglich, bewusst reduziert. Stattdessen wurden Open-Source- und cloud-agnostischen Tools der Vorzug gegeben. Beispielsweise wurden zur Verwaltung sensibler Konfigurationsdaten Sealed Secrets verwendet, eine Open-Source-Lösung, die eine sichere Verschlüsselung von sensitiven Daten (beispielsweise Passwörtern) und deren Speicherung in versionskontrollierten Repositories ermöglicht. Dieser Ansatz stellt sicher, dass das System über Cloud-Anbieter hinweg portierbar bleibt und nicht von Azure-spezifischen Secret-Management-Angeboten abhängig ist. In Kombination mit Azures Monitoring- und Logging-Diensten bietet das AKS-Setup eine robuste und zugleich flexible Grundlage für die Ausführung verteilter Energiesystem-Simulationen bei gleichzeitiger Minimierung der Cloud-Provider-Abhängigkeiten.

Kein einzelnes Tool ist fehlerfrei; es wurde festgestellt, dass Kubernetes sorgfältig konfiguriert werden muss (z. B. durch die Verwendung eines dedizierten Namespace pro Simulation) und der Cluster-Zustand überwacht werden muss. Trotzdem machte die Flexibilität des Tools und sein Ökosystem es zum best-geeignetsten Kandidaten. Docker Swarm wurde für Vergleichszwecke in kleineren Szenarien beibehalten, wird jedoch nicht als primärer Orchestrator für großskalige Simulationen dienen.

Zusammenfassend erfüllt Kubernetes die meisten Anforderungen für großskalige Softwaresimulationen, indem es robuste Bereitstellungs-, Skalierungs- und Netzwerkmöglichkeiten bereitstellt, die die Validierung von Energiesystemkomponenten mit minimalen Anpassungen unterstützen. Es eignet sich gut für DES-CERT-Anwendungen und auf Grundlage der oben dargestellten Punkte wurde es als Basis unseres

Architekturdesignprozesses ausgewählt.

3.3 Bewertung von Werkzeugen zur Modellierung und Analyse von Energiesystemen

Dieser Absatz konzentriert sich auf die Bewertung von State-of-the-Art Power System Modeling and Analysis (PSMA)-Tools zur Unterstützung des Ziels des DES-CERT-Projekts einer skalierbaren cyber-physischen Validierungsumgebung von Energiesystemen. Konkret bewerten wir die Open-Source-Simulatoren ANDES, pandapower und GridLAB-D anhand für großskalige Netzsimulationen relevante Kriterien: Rechen- und Laufzeiteffizienz, Skalierbarkeit, Interoperabilität und Kompatibilität mit containerisierter Orchestrierung. Unsere Methodik umfasst eine Literaturanalyse, die praktische Erprobung jedes Tools, Benchmark-Tests an repräsentativen Netzen und das Design von Schnittstellen für Co-Simulation. Wichtige Ergebnisse umfassen eine vergleichende Bewertung der Tools, die Auswahl der für unsere Anforderungen am besten geeigneten PSMA-Engine und Prototyp-Schnittstellen, die einen durchsatzstarken Datenaustausch in einer containerisierten Umgebung ermöglichen.

3.3.1 PSMA-Werkzeuglandschaft

Wir haben drei prominente Open-Source-PSMA-Werkzeuge mit unterschiedlichen Designfokusen evaluiert:

- **pandapower [26]**: pandapower ist eine Python-basierte Bibliothek für statische (bzw. quasi-statische) Netzsimulationen. Es bietet Routinen für Lastflüsse, optimale Lastflüsse, Kurzschlussrechnungen und Zustandsschätzungen und verwendet das PYPOWER-Solver-Backend [27]. Das Netzmodell ist elementbasiert (Knoten, Leitungen, Transformatoren, Lasten, Erzeuger usw.) mit Parametern, die in Pandas-Tabellen definiert sind. Es ist BSD-lizensiert, leicht über Python erweiterbar und verfügt über eine große Nutzerbasis. pandapower eignet sich am besten für stationäre oder quasi-stationäre Studien; es fehlt an integrierter Unterstützung für detaillierte dynamische Simulationen oder unsymmetrische Analysen (außer symmetrischen Äquivalenten), kann jedoch große Netze mittels effizienter Sparse-Solver verarbeiten. Beispielsweise enthält pandapower ein 2224-Knoten-Modell des Übertragungsnetzes von Großbritannien für Benchmarks. Da es „pythonic“ ist, integriert pandapower sich nahtlos in Datenanalyse- und Optimierungs-Workflows, läuft jedoch typischerweise auf einem einzelnen Rechner (Parallelität beschränkt sich auf das Ausführen mehrerer unabhängiger Simulationen).

- **ANDES (Advanced Network Dynamic Simulation) [28]:** ANDES ist ein Open-Source-Python/C++-Tool, das für dynamische, cyber-physische Simulation großer Übertragungssysteme entwickelt wurde. Es erzeugt zur Laufzeit kompilierten Code zur Simulation der elektromechanischen Dynamik und unterstützt Generatoren, Regler, usw. Im Gegensatz zu pandapower behandelt ANDES nativ dynamische Netzsimulationen und eignet sich damit für Transienten-Stabilitätsstudien und Szenarien mit Kommunikation/Regelungs-Co-Simulation. ANDES enthält Schnittstellen zu Co-Simulations-Frameworks wie HELICS [29]. Das Werkzeug wurde in Forschungstestbeds als Übertragungssimulator eingesetzt (z. B. einer cyber-physischen Simulation unter Verwendung von HELICS und ANDES gekoppelt mit Verteilungsmodellen). Für stationäre Lastflüsse bietet ANDES Newton-Raphson-Algorithmen an und bietet Interoperabilität mit pandapower über eine Konvertierungs-API. Mit anderen Worten: Ein ANDES-Modell kann zur Validierung

oder für hybride Workflows in ein pandapower-Netz exportiert werden. ANDES ist MIT-lizensiert und für hohe Performance ausgelegt: Sein numerischer Solver ist parallelisierbar (z. B. durch Multiprocessing in Python), sodass es auf HPC-Clustern nutzbar ist. Nachteile sind eine steilere Lernkurve und weniger Beispielnetzmodelle im Vergleich zu pandapower.

- **GridLAB-D [30]:** GridLAB-D ist ein Open-Source-Verteilernetz-Simulator, entwickelt vom PNNL. Er legt den Schwerpunkt auf detaillierte Zeitreihen- und unsymmetrische Modellierung von Verteilernetzen, einschließlich Lasten, Wechselrichtern und Reglern. GridLAB-D arbeitet als ereignisgesteuerte Simulation und eignet sich ideal für Aufgaben wie Power-Quality-Analysen, Demand-Response oder Studien zur Integration verteilter Energiequellen (DER). Es unterstützt beliebige zeitliche Auflösungen und kann große Anzahl von Endpunkten (z. B. tausende Haushaltslasten) mit realitätsnahem Detailheitsgrad simulieren. Allerdings ist der Kernsolver von GridLAB-D in der Regel single-threaded und nicht für umfangreiche stationäre Lastflussberechnungen großer Übertragungsnetze optimiert. Seine Stärke liegt stattdessen in der Detailtreue der Verteilernetzmodellierung. Es kann über die Middleware FNCS oder HELICS mit anderen Werkzeugen co-simulieren, wodurch z. B. Umspannwerks-Lastflüsse mit einem Übertragungsnetz-Simulator ausgetauscht werden können. GridLAB-D ist C++-basiert und verfügt über eine TCL/Python-API. Es ist BSD-lizensiert und für Verteilungsstudien hochgradig ausgereift, aber die Skalierung auf sehr große Netze (z. B. > 10 000 Knoten) erfordert sorgfältige Szenarien-Partitionierung oder Multi-Instanz-Orchestrierung. In der Community wird an der Parallelisierung von GridLAB-D und der Verbesserung seiner HPC-Nutzung gearbeitet, aber in seiner aktuellen Form ist es am besten als Komponente für Verteilernetz- Co-Simulationen statt als monolithisches Werkzeug für großvolumige Netzlösungen geeignet. Tabelle 2 fasst du Kernaspekte zusammen:

Kriterium	**pandapower**	**ANDES**	**GridLAB-D**
Domäne	Stationär (AC & DC Lastfluss), OPF, Kurzschluss	Dynamische und stationäre Simulationen	Zeitreihen-Verteilernetzsimulation (unsymmetrisch)
Modellumfang	Symmetrische Übertragungs- und Verteilernetze	Übertragungungs- & Verteilernetze	Detaillierte Verteilernetze
Lizenz / Programmier-sprache	BSD / Python (NumPy, Pandas)	MIT / Python + C++	BSD / C++ (mit TCL- und Python-Schnittstellen)
Solver	Sparse Newton-Raphson (KLU, SuperLU), DC-Approximierung	Sparse Newton-Raphson; parallelisierte Dynamik-Solver	Ereignisgetriebener Zeitbereichs-Solver
Parallelisierung	Multi-Case (MPI oder Multi-Process), keine native Thread-	Multi-Domain (MPI/Threads für Dynamik)	Nein; nur mehrere Instanzen extern parallel

	Beschleunigung		
Community & Reife	Sehr aktiv, gut dokumentiert	Aufstrebend (universitätsgeführt), weniger Dokumentation	Etabliert (DOE/PNNL), große Nutzerbasis für Verteilernetze
Interoperabilität	APIs für gängige Formate (MATPOWER, OpenDSS)	Konvertierung zu/von pandapower; HELICS-Support	Co-Simulation via FNCS/HELICS, Tabellen etc.
Large-Scale Performance	Benchmarks <1 s für 14-Busse; ~O(1 s) für 100–1k Busse (Single-Core)	Vergleichbare Laufzeiten; skaliert über parallele Multi-Runs	10k+ Knoten möglich, lange Laufzeiten; ideal für Verteilernetze
Besondere Features	Mitgelieferte Netze (IEEE, CIGRE), GIS-Daten, 3-Phasen-Modul	Dynamische Regler, Schnittstellen für Co-Simulationen	Detaillierte Modelle (HVAC, Boiler), unsymmetrischer Lastfluss

Table 2 Vergleich von Open-Source-Werkzeugen zur Modellierung und Analyse von Energiesystemen (PSMA) anhand wichtiger Kriterien.

3.3.2 Werkzeugauswahl und Bewertung

Basierend auf der obigen Analyse wurde pandapower als primäre PSMA-Engine gewählt aufgrund seiner Benutzerfreundlichkeit, seinen umfassenden Funktionen für stationäre Analysen und der Unterstützung für großskalige Netze – wodurch es am besten für die umfangreichen Validierungsszenarien in DES-CERT geeignet ist. ANDES bleibt ein geeigneter Kandidat für dynamische Untersuchungen und cyber-physikalische Co-Simulationen und wird in zukünftige Erweiterungen des Frameworks integriert. GridLAB-D, obwohl gut geeignet für detaillierte Verteilernetzmodellierung, ist für spezifische Szenarien vorbehalten und wird bei Bedarf eingesetzt.

3.3.3 Kompatibilitätsbewertung

Mit den ausgewählten Simulationsengines haben wir deren Kompatibilität mit containerbasierten Orchestrierungswerkzeugen bewertet, um skalierbare, automatisierte Prävalidierungs-Workflows zu unterstützen, wie sie im DES-CERT-Projekt vorgesehen sind. Zu diesem Zweck haben wir Schnittstellenkomponenten entwickelt, die Power-System-Modeling-and-Analysis-(PSMA)-Werkzeuge als First-Class-Komponenten in einem Orchestrierungsframework integrieren. Das primäre Ziel besteht darin, eine effiziente Ausführung, Koordination und Skalierung von Simulationstasks sowie eine nahtlose Interaktion mit Cyber-Schicht-Services wie Steuerungs-, Optimierungs- und Analyse-Modulen zu ermöglichen. Die Schlüsselaspekte dieser Kompatibilitätsbewertung sind nachfolgend zusammengefasst:

- **Containerisierte Bereitstellung**: Wir haben Docker-Images für pandapower erstellt, sodass es als vollständig containerisierte Simulationskomponente ausgeführt werden kann. Da pandapower Open Source und Python-basiert ist, eignet es sich gut für die Containerisierung und kann konsistent über heterogene Hardware- und Ausführungsumgebungen bereitgestellt werden.

Die resultierenden Container-Images wurden erfolgreich auf einem Kubernetes-Cluster bereitgestellt, wodurch Simulationsinstanzen automatisiert und reproduzierbar instanziiert, skaliert und verwaltet werden können.

- **Skalierbarkeit und parallele Ausführung**: Das containerisierte Design ermöglicht die parallele Ausführung mehrerer Instanzen der Simulationsengine und unterstützt damit groß angelegte Szenarioerkundungen und Parameterstudien. Orchestrierungsmechanismen wie Pod-Replikation und Batch-Job-Scheduling erlauben es DES-CERT, zahlreiche Prävalidierungs-Szenarien gleichzeitig zu bewerten, was für eine systematische und statistisch aussagekräftige Bewertung digitaler Energieservices essenziell ist.
- **Ressourcenmanagement und Isolation**: Die Kompatibilität mit Orchestrierungsplattformen ermöglicht die Ausführung von Simulations-Workloads mit expliziten Ressourcenbeschränkungen (z. B. CPU- und Arbeitsspeicher), wodurch vorhersehbare Leistungskennwerte und Isolation zwischen gleichzeitigen Simulationen sichergestellt werden. Diese Fähigkeit ist insbesondere in geteilten oder cloudbasierten Infrastrukturen wichtig, in denen mehrere Validierungsexperimente gleichzeitig ausgeführt werden können.
- **Automatisierung und Lifecycle-Management**: Durch die Integration der PSMA-Werkzeuge in das Orchestrierungsframework kann der gesamte Lebenszyklus von Simulationsaufgaben – einschließlich der Bereitstellung, Ausführung, Überwachung und Abschaltung – automatisiert werden. Dies unterstützt reproduzierbare und prüfbare Prävalidierungs-Workflows und ermöglicht eine nahtlose Integration in CI/CD-Pipelines, im Einklang mit DES-CERTs Schwerpunkt auf softwarezentrierter Validierung.
- **Interoperabilität mit Cyber-Layer-Komponenten**: Der containerbasierte Ansatz erleichtert standardisierte Schnittstellen zwischen der physischen Simulation und Cyber-Layer-Komponenten, wodurch Steuerungs-, Optimierungs- und Analyseservices flexibel zusammengesetzt werden können. Diese Modularität stellt sicher, dass verschiedene PSMA-Werkzeuge oder Steuerungsalgorithmen ausgetauscht oder erweitert werden können, ohne die Gesamtarchitektur der Plattform zu stören.

Abschließend ist anzumerken, dass der oben beschriebene Integrationsansatz nicht auf pandapower beschränkt ist, sondern auf eine breitere Klasse von PSMA-Werkzeugen mit ähnlichen Eigenschaften anwendbar ist. Insbesondere PSMA-Werkzeuge wie ANDES, die ebenfalls Open Source, Python-basiert und für programmatische Interaktion ausgelegt sind, können entsprechend denselben Prinzipien containerisiert und orchestriert werden. Durch die Kapselung dieser Werkzeuge in standardisierten Container-Images und die Bereitstellung von standardisierten Schnittstellen können sie einheitlich vom Orchestrierungsframework bereitgestellt, skaliert und verwaltet werden.

Aufbauend auf dieser generalisierten Integrationsfähigkeit verlagert sich der Fokus nun von der Werkzeugorchestrierung hin zum systemweiten Rahmenwerk, das diese Fähigkeiten für automatisierte Analyse und Validierung nutzt. In diesem Zusammenhang ist das vorgeschlagene Automated Cyber-Physical Testing and Validation (ACTV)-Framework als Proof of Concept zu verstehen, das demonstriert, wie ein solcher Orchestrierungsansatz systematisch erweitert werden kann, um integrierte cyber-physische Studien zu unterstützen.

3.4 Automated Cyber-Physical Testing and Validation (ACTV) Framework

Die Grundlage des ACTV-Frameworks ist das Cyber-Physical Model (CPM), das als einheitliche Darstellung der elektrischen, Kommunikations- und Software-Schichten des Systems dient. Der folgende Abschnitt beschreibt die Struktur des CPM und seine Rolle innerhalb des ACTV-Frameworks und beschreibt, wie es eine zentrale Quelle für Simulationen etabliert und dynamische Co-Simulationsstudien unterstützt.

3.4.1 Cyber-Physical Modell

Ein zentrales Systemmodell ist für die Ermöglichung von Automatisierung, Interoperabilität und Reproduzierbarkeit innerhalb des ACTV-Framework unerlässlich. Da das Framework die Simulation cyber-physischer Energiesysteme zum Ziel hat, muss das Modell nicht nur das elektrische Netz, sondern auch die zugehörige Kommunikationsinfrastruktur und die Softwarekomponenten abbilden. Die Integration dieser heterogenen Elemente in ein einheitliches CPM liefert eine konsistente und umfassende Systemrepräsentation, bietet eine zentrale Informationsquelle für alle Ebenen des Frameworks und unterstützt genaue, wiederholbare Simulationen. Diese Arbeit baut auf unseren früheren Beiträgen in [32] auf und erweitert sie, indem die CPM-Struktur formalisiert und ihr Umfang zur Unterstützung von dynamischen und Co-Simulations-Szenarien ausgeweitet wird.

Das CPM verwendet ein maschinenlesbares Format, das sowohl die Funktionalität einzelner Komponenten als auch die Interaktionen zwischen ihnen beschreibt, wodurch die automatisierte Instanziierung der Szenarien ermöglicht wird und das Risiko von Inkonsistenzen reduziert wird. Durch die Standardisierung der Systemrepräsentation steigert das CPM außerdem die Interoperabilität zwischen verschiedenen Simulationsengines, Analysemodulen und Steuerungsanwendungen, was eine zentrale Voraussetzung für das modulare Design von ACTV darstellt.

Um Flexibilität zu wahren und benutzerdefinierte Erweiterungen zu erleichtern, ist das CPM in drei unterscheidbare Schichten organisiert: die Physische Schicht, die elektrische Elemente wie Erzeuger, Lasten, Energiespeicher und die Netztopologie modelliert; die Kommunikationsschicht, die Datenaustauschmechanismen, Netzwerkprotokolle und die Infrastruktur erfasst, die physische Geräte mit Monitoring- und Steuerungssystemen verbindet; und die Anwendungsschicht, die Softwarekomponenten, Algorithmen und Steuerungslogik definiert, die auf der physischen- und der Kommunikationsschicht ausgeführt werden. Dieser Schichtenansatz unterstützt nicht nur eine modulare Entwicklung und einfache Integration neuer Komponenten, sondern schafft auch eine klare Trennung der Verantwortungsbereiche, ermöglicht sowohl stationäre als auch dynamische Simulationen und gewährleistet skalierbare sowie robuste cyber-physische Test- und Validierungsabläufe.

3.4.2 Physische Schicht

Die physiche Schicht (Physical Layer (PL)) repräsentiert das elektrische Netz und dient als grundlegende Komponente für alle Simulationen innerhalb des ACTV-Frameworks. Sie definiert die Topologie, elektrische Parameter und betrieblichen Eigenschaften des Netzes und stellt die strukturelle Basis dar, auf der dynamische Verhaltensweisen und Steuerungsinteraktionen modelliert werden. Die meisten Werkzeuge zur Simulation und Analyse von Energienetzen unterstützen eine Vielzahl maschinenlesbarer Formate zum Import und Export von Netzmodellen. Während einige dieser Formate proprietär oder werkzeugspezifisch sind, ermöglichen standardisierte Formate wie die Common Grid Model Exchange Specification (CGMES) [31] Interoperabilität zwischen verschiedenen Werkzeugen und Plattformen und erleichtern eine konsistente Darstellung und den Austausch von Netzdaten.

CGMES, entwickelt von ENTSO-E, ist ein spezialisiertes Profil des IEC Common Information Model (CIM), das speziell für den Austausch von Netzmodellen der Übertragungsnetzbetreiber (TSO) ausgelegt ist. Es definiert ein detailliertes Schema für elektrische Komponenten, Konnektivität, betriebliche Einschränkungen und Metadaten und erlaubt die exakte Rekonstruktion von Netzmodellen in kompatiblen Softwareumgebungen. Innerhalb des ACTV-Frameworks wird der PL durch den Import von Netzdaten im CGMES-Format aufgebaut, wodurch sichergestellt wird, dass das resultierende Modell einem anerkannten Standard entspricht. Durch die Übernahme von CGMES bietet die PL nicht nur eine robuste und konsistente Grundlage für Simulationen, sondern verbessert auch die Interoperabilität und ermöglicht die Integration mit mehreren PSMA-Werkzeugen, Analysemodulen und zukünftigen Erweiterungen des Frameworks. Darüber hinaus erleichtert die Nutzung dieses Standards die Validierung anhand realer Übertragungsnetz-Datensätze und stellt sicher, dass Simulationen mit den Branchenpraktiken übereinstimmen, wodurch die Zuverlässigkeit und Anwendbarkeit der Ergebnisse des Frameworks verbessert werden.

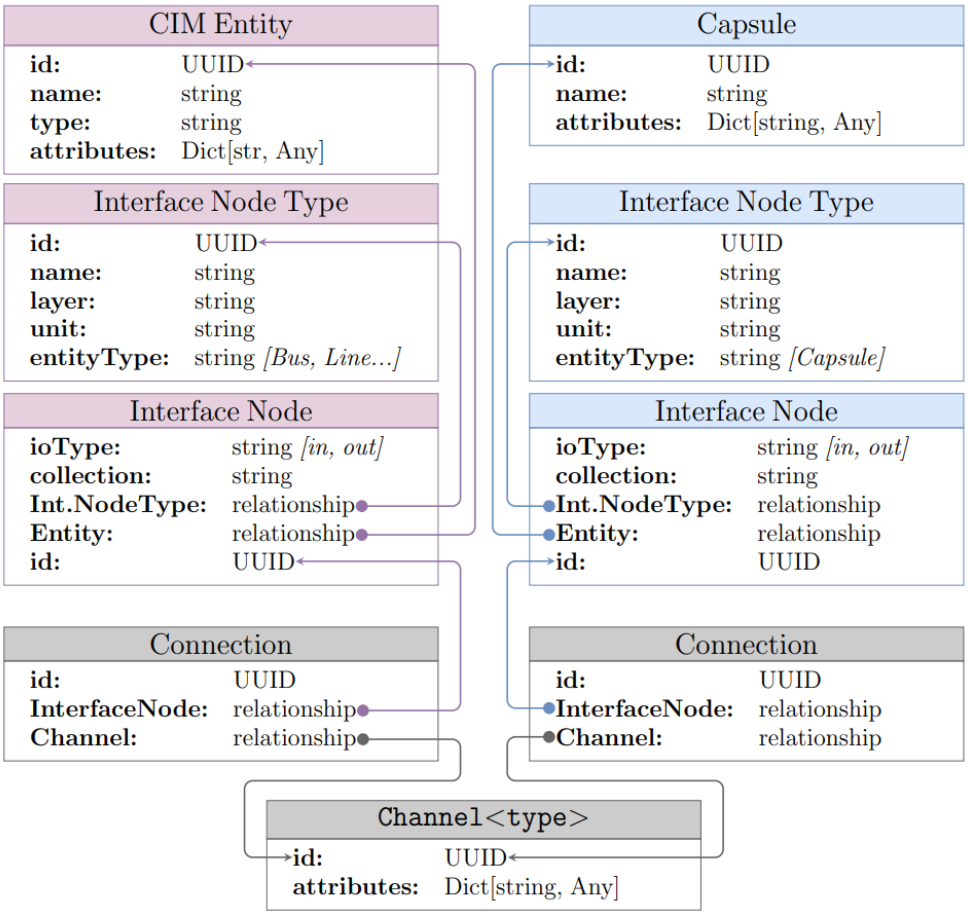

Abbildung 4 Instanziierte Entitäten, die den Informationsaustausch zwischen einem Messwert oder Sollwert eines Netzmodellelements (oben links) und einer Anwendung (oben rechts) darstellen.

3.4.3 Kommunikationsschicht

The Kommunikationsschicht (Communication Layer (CL)) definiert die Kommunikationsschnittstellen und -kanäle innerhalb des Systems und legt fest, wie Komponenten in der Kommunikationsschicht Messwerte und Sollwerte mit der Netzsimulation austauschen sowie wie sie Daten untereinander teilen. Insbesondere abstrahiert der CL Implementierungsdetails der Nachrichtenübertragung (z. B. Message-Queue-Technologien, Publish/Subscribe-Busse oder Request/Response-APIs) in eine Menge generischer Entitäten, die für unterschiedliche Kommunikations-Backends instanziiert werden können. Diese Abstraktion ermöglicht eine klare Trennung der Verantwortlichkeiten: die Physical- und Application-Layers können unabhängig von der konkreten Kommunikations-Middleware modelliert werden, während der CL eine Zuordnung von logischen Signalen (z. B.

Busspannungen) zu konkreten Kommunikationsartefakten (z. B. Topics, Keys oder Nachrichtenfeldern) bereitstellt. Um Kommunikation und Informationsaustausch systematisch und wiederverwendbar zu modellieren, werden vier Entitäten definiert:

Interface Node Type: Stellt eine eindeutige Art von Messgrößen oder Sollwerten dar. Sie spezifiziert 1) die Art der Komponente, auf die sie sich bezieht, 2) den mit dieser Komponente verbundenen Signaltypen und 3) die Einheit des Signals. Beispielsweise erfasst sie für Leitungen deren Auslastung mit der Einheit %. Allgemeiner kann ein Interface Node Type durch ein Tupel bezeichnet werden

$$INT = (entityType, signalType, unit)$$

wobei entityType sich auf eine CIM-Klasse wie ACLineSegment oder EnergyConsumer bezieht. signalType unterscheidet beispielsweise zwischen Wirkleistung, Blindleistung, Strombetrag oder Spannungsbetrag, und unit ist gemäß einem vereinbarten Einheitensystem definiert (z. B. SI-Einheiten oder Per-Unit-System). Die Verwendung von Interface Node Types gewährleistet semantische Konsistenz im Gesamtsystem, da alle Applikationen, die sich auf denselben Typ beziehen, eine identische Interpretation der zugrunde liegenden Größe teilen.

Interface Node: Verknüpft spezifische Komponenten mit ihren entsprechenden Interface Node Types. Beispielsweise wird zur Bereitstellung von Spannungsmessungen für zwei Busse ein Interface Node Type für Busspannungsmessungen definiert, und für jeden Bus wird ein Interface Node erstellt, das diesem Typen zugeordnet ist. Formell stellt ein Interface Node eine Relation her

$$IN: Component \rightarrow INT$$

wobei Component eine Instanz entweder im PL (z. B. ein bestimmter Bus, identifiziert durch seine CIM ID) oder in der Applikationsschicht (z. B. eine bestimmte Eingangs- oder -Ausgangsgröße einer Applikation) ist. Dieses Design unterstützt Many-to-One-Abbildungen, bei denen mehrere Interface Nodes (entsprechend verschiedenen Komponenten) auf denselben Interface Node Type verweisen können und damit erzwingen, dass alle dieselbe Signalsemantik teilen (z. B. „Betrag der Busspannung in V“).

Channel: Definiert, wie Informationen ausgetauscht werden, einschließlich des Kommunikationsprotokolls und etwaiger erforderlicher Konfigurationsattribute. Beispielsweise ein Redis-Key oder ein MQTT-Topic, das zur Übertragung eines Messwerts oder Sollwerts verwendet wird. Ein Channel kapselt transportbezogene Eigenschaften wie das verwendete Protokoll, z. B. MQTT, Redis, HTTP.

Connection: Verbindet Channels und Interface Nodes und ermöglicht den eigentlichen Informationsaustausch zwischen den von den Interface Nodes referenzierten Komponenten. Eine Connection verknüpft ein oder mehrere Interface Nodes mit einem gegebenen Channel und legt die Richtung des Informationsflusses fest (z. B. Messwerte von der Simulation zur Anwendung oder Sollwertvorgaben von der Anwendung zur Simulation). Dadurch können nicht nur Punkt-zu-Punkt-Verbindungen, sondern auch Broadcast- oder Publish/Subscribe-Semantiken modelliert werden, bei denen ein einzelner Channel mit mehreren konsumierenden Interface Nodes verbunden ist. Connections können zusätzlich Meta-Informationen wie Datenratenbeschränkungen, Abonnementfilter oder Routing-Richtlinien tragen, die für die Validierung oder die automatisierte Bereitstellung der Kommunikationsinfrastruktur verwendet werden können.

Abbildung 4 veranschaulicht, wie der Austausch eines Messwertes oder Sollwertes zwischen einem Netzmodell-Element (CIM Entity) und einer Anwendung (Capsule) dargestellt wird. Pfeile geben die Referenzen zwischen den instanziierten Modellen an. Wir modellieren diesen Datenaustausch mithilfe von zwei verschiedenen Interface Node Types: einem mit entityType

entsprechend dem Typ des Netzmodell-Elements im PL (im linken Teil der Abbildung gezeigt) und einem weiteren in der Applikationsschicht, das auf die Capsules (Anwendungen) verweist. Auf diese Weise stellt der CL eine bidirektionale Abbildung zwischen der cyber-physischen Repräsentation des Energiesystems und der logischen Ein-/Ausgabestruktur von Anwendungen her: Messwerte werden von CIM Entities über Channels zu den entsprechenden Capsule-Eingängen weitergeleitet, während Sollwerte in die entgegengesetzte Richtung übertragen werden, von Capsule-Ausgängen zu CIM Entities. Diese explizite Darstellung von Interface Node Types, Interface Nodes, Channels und Connections erleichtert formales Schlussfolgern über Signalvollständigkeit, Einheitenkonsistenz und Konnektivität und unterstützt automatisierte Prüfungen (z. B. Erkennung fehlender Messwerte, inkompatibler Einheiten oder nicht angeschlossener Sollwerte) vor der Ausführung von Simulationen.

3.4.4 Anwendungsschicht

Die Anwendungsschicht (Application Layer (AL)) beinhaltet die zu testende Applikation und ist mit Terminologien bezeichnet, die aus Containerisierungs- und Orchestrierungsumgebungen stammen. Insbesondere folgt sie Konzepten, die denen moderner cloud-nativer Plattformen wie Docker und Kubernetes ähneln, wo Applikationen als Container-Images gebüdelt und als isolierte Laufzeitinstanzen bereitgestellt werden. Innerhalb des ACTV-Frameworks werden Applikationen durch vier Kernentitäten repräsentiert, wie in Abbildung 5 dargestellt:

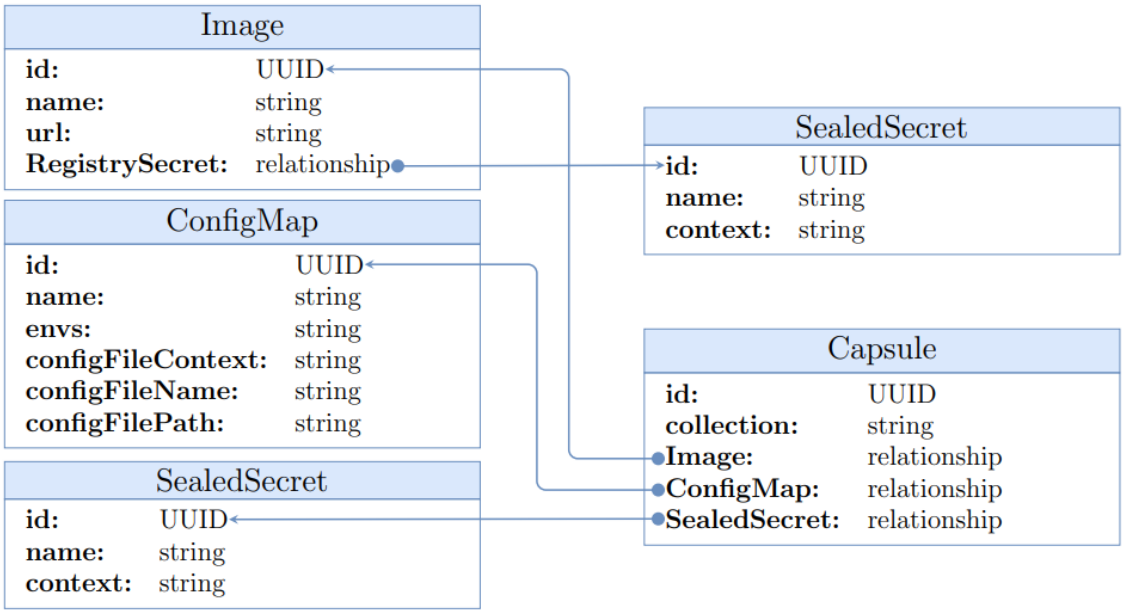

Abbildung 5 Die zu testenden Anwendungen werden durch eine Reihe von Entitäten modelliert. Jede Anwendung wird als Blackbox behandelt und durch eine Capsule (die Laufzeitinstanz), ihr entsprechendes Image (das Container-Image), eine ConfigMap für Konfigurationsdaten und SealedSecrets zur Verwaltung sensibler Informationen dargestellt.

Image: Repräsentiert das Container-Image der getesteten Anwendung. Es enthält die ausführbare Anwendung oder den Quellcode sowie die Ausführungsumgebung und etaige Abhängigkeiten (z. B. Betriebssystem-Bibliotheken, Laufzeitumgebungen für Programmiersprachen und Drittanbieter-Pakete). Diese Abstraktion ermöglicht reproduzierbare und portable Deployments und stellt sicher, dass eine Anwendung sich über heterogene Ausführungsumgebungen hinweg konsistent verhält. Jede Anwendung ist mit einem eigenen Image verknüpft, das versioniert werden kann (z. B. mittels Tags), um unterschiedliche Releases oder Konfigurationen abzubilden.

Capsule: Bezeichnet eine Instanziierung eines Images, analog zu einem Pod in Kubernetes oder einem Container in Docker. Capsules kapseln Laufzeitinstanzen von Anwendungen,

einschließlich ihres Prozesszustands, ihrer Ressourcenallokationen (wie CPU- und Arbeitsspeicherlimits) und ihrer Netzwerkkonfiguration. Eine Capsule kann kurz- oder langlebig sein, abhängig vom Testszenario, und verweist auf die zur Ausführung erforderliche Konfiguration, wie Umgebungsvariablen und gemountete Volumes. Mehrere Capsules können aus demselben Image instanziiert werden, um Skalierbarkeit oder paralleles Testen zu unterstützen.

ConfigMap: Speichert Konfigurationsdaten, die für die Ausführung der Anwendung erforderlich sind, wie Parameter, Umgebungsvariablen, dateibasierte Konfigurationseinträge und benutzerdefinierte Einstellungen, die vom Anwendungs-Binary getrennt bleiben sollen. Durch die Trennung von Konfiguration und Code unterstützen ConfigMaps das Twelve-Factor-App-Prinzip der strikten Trennung zwischen Konfiguration und Build-Artefakten und erleichtern die flexible Neukonfiguration von Capsules in unterschiedlichen Testszenarien, ohne dass ein Neubauen des Images erforderlich ist. ConfigMaps sind mit Capsules verknüpft und müssen vom Benutzer explizit definiert werden.

SealedSecret: Bietet einen sicheren Mechanismus zum Verwalten sensibler Informationen, einschließlich Passwörtern, Tokens, API-Schlüsseln und privatem kryptografischem Inhalt. Im Gegensatz zu Klartextkonfigurationen stellen SealedSecrets sicher, dass solche Daten im Ruhezustand und während der Übertragung verschlüsselt sind und nur zur Laufzeit durch einen kontrollierten Entschlüsselungsprozess der entsprechenden Capsule zugänglich werden. Dieses Design entspricht den empfohlenen Praktiken beim Umgang mit sensitiven Informationen, indem es die versehentliche Offenlegung von Zugangsdaten in Konfigurations-Repositories, Logdateien oder Testartefakten verhindert und gleichzeitig die automatisierte, reproduzierbare Bereitstellung von Testumgebungen ermöglicht.

Das Framework behandelt die Anwendung als Black Box und bleibt gegenüber Programmiersprache, internen Abhängigkeiten und Implementierungsdetails agnostisch. Formal sei I die Menge der Images und C die Menge der Capsules; dann referenziert jede Capsule $c \in C$ genau ein Image $i \in I$, zusammen mit einem Konfigurations-Tupel (cfg, sec), das eine ConfigMap und ein optionales SealedSecret umfasst. Da die Anwendung als Container-Image gekapselt ist, muss der Quellcode nicht offengelegt werden, was besonders vorteilhaft ist, um proprietäre oder Drittanbieter-Komponenten unter realistischen Bedingungen zu testen, dabei geistiges Eigentum zu wahren und Testanforderungen von Implementierungsdetails zu isolieren.

3.5 Implementierung des Frameworks

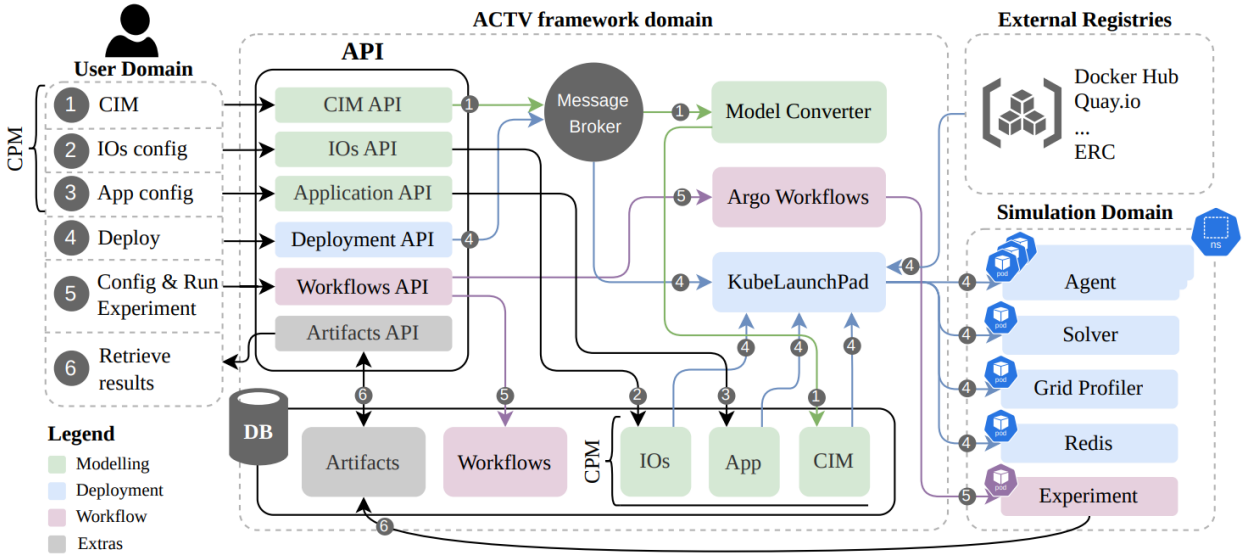

Abbildung 6 Softwarearchitektur des ACTV-Frameworks, implementiert in Python und bereitgestellt auf Kubernetes. Das Diagramm ist in die Bereiche Benutzer (User), Framework und Simulation unterteilt, wobei die Farbkodierung funktionale Gruppen hervorhebt: Modellierung (grün) und Bereitstellung (blau).

Die Softwarearchitektur des ACTV-Frameworks folgt einem modularen, serviceorientierten Design, ist in Python implementiert und wird auf einem Kubernetes-Cluster deployt, um Container-Orchestrierung und automatisches Skalieren zu nutzen. Abbildung 6 veranschaulicht die Gesamtarchitektur. Das Diagramm ist in drei Bereiche unterteilt: die Nutzerdomäne, die ACTV-Framework-Domäne und die Simulations-Domäne. Die ACTV-Framework-Domäne stellt den Kern des Systems dar und bietet Orchestrierung, Koordination und Lifecycle-Management von Simulationen, während die Nutzer- und Simulations-Domänen ergänzende Komponenten bereitstellen, die dessen Betrieb unterstützen und die Fähigkeiten gegenüber externen Nutzern exponieren.

In der Nutzer-Domäne zeigt das Diagramm, wie Nutzer mit dem Cluster interagieren, und skizziert die Schrittfolge, die erforderlich ist, um eine Simulation eines cyber-physischen Modells (CPM) auszuführen. Nutzer stellen zunächst das Netz in Form eines Common Information Models (CIM) und dessen I/O-Spezifikationen bereit und konfigurieren diese, danach liefern sie eine Applikationskonfiguration, die das CPM an eine spezifische Simulations- oder Analyseaufgabe bindet. Diese Konfigurationsartefakte werden über die öffentlichen APIs an das Framework übermittelt. Die nummerierten Pfeile in Abbildung 6 fassen diesen Prozess zusammen: (1) Erstellung und Upload von CIM- und I/O-Konfigurationen, (2) Spezifikation von applikationsbezogenen Parametern, (3) Absenden von Deployment-Anfragen, (4) Konfiguration und Ausführung von Experimenten und (5–6) Abruf von Simulationsresultaten und abgeleiteten Ergebnissen.

Die Simulations-Domäne hebt die auf dem Cluster bereitgestellten Ressourcen hervor, einschließlich Kernlaufzeitkomponenten wie dem Solver, der die CPM-Gleichungen numerisch integriert, und dem Grid Profiler, der Lastprofile auf die verschiedenen Knoten im Netz anwendet. Diese Komponenten laufen in isolierten Kubernetes-Pods und können über mehrere Cluster-Knoten repliziert werden, um Parallelisierung zu nutzen. Zusätzliche Dienste, wie ein Key-Value-Store (z. B. Redis) für transiente Zustände und ein Service zur Verwaltung von Experiment-Metadaten und Checkpoints, ergänzen den Solver-Stack. Container-Images für diese Dienste werden aus externen Registries bezogen (z. B. Docker Hub, Quay.io), was reproduzierbare und portable Deployments über Umgebungen hinweg gewährleistet.

Ein kritischer Aspekt des Designs ist die Inter-Modul-Kommunikation. Um hohen

Nachrichtendurchsatz, geringe Latenz und horizontale Skalierbarkeit sicherzustellen, setzt das Framework auf Apache Pulsar als Message Broker. Pulsar bietet eine verteilte, partitionierte Log-Abstraktion und unterstützt topic-basierte Publish-/Subscribe-Semantiken, wodurch asynchrone Kommunikation zwischen Modulen ermöglicht wird. Dieses Design unterstützt Backpressure-Handling und Replay-Semantiken unter hoher Last. Durch die Entkoppelung von Produzenten und Konsumenten auf Messaging-Ebene kann das Framework einzelne Dienste elastisch skalieren, um auf variierende Last zu reagieren, bei gleichzeitiger Wahrung der Gesamtreaktionsfähigkeit des Systems.

Einstiegspunkt des Frameworks ist eine Sammlung von APIs, die seine Funktionalität externen Nutzern und Diensten bereitstellen. Diese APIs sind in einer dedizierten API-Gateway-Schicht gruppiert und umfassen unter anderem die CIM API, IOs API, Application API, Deployment API, Workflows API und Artifacts API. Intern orchestrieren diese APIs die Ausführung verschiedener Module in der ACTV-Framework-Domäne, von denen jeweils eines eine spezifische Funktion übernimmt: Der Model Converter transformiert vom Nutzer bereitgestellte Modelle in interne Repräsentationen, die mit dem Solver-Stack kompatibel sind; Argo Workflows verwaltet komplexe, mehrstufige Workflows wie Multi-Szenario-Studien oder Parameter-Sweeps; und KubeLaunchPad übernimmt Low-Level-Deployment-Logik auf Kubernetes, einschließlich Pod-Instanziierung, Namespace-Management sowie Bindung von ConfigMaps und Secrets. Die folgenden Unterabschnitte beschreiben jedes dieser Module detailliert, gruppiert nach ihren funktionalen Rollen.

Die Farbkodierung im Diagramm dient der Gruppierung von Modulen mit ähnlichen Funktionalitäten und trägt so zur visuellen Verständlichkeit der Architektur bei: Modellierungs-Module sind grün dargestellt (z. B. CIM-, IOs- und Applikationskonfigurationsdienste), Deployment-Module blau (z. B. Deployment API, KubeLaunch-Pad), Workflow-Module lavendelfarben (z. B. Argo Workflows, Workflows API) und Simulation-Module grau innerhalb der Simulations-Domäne. Zusätzliche Komponenten wie Datenbanken und externe Registries sind als „Extras“ hervorgehoben, um ihre unterstützende Rolle im Gesamtsystem zu betonen, wobei sie dennoch essentiell für Persistenz, Versionierung von Artefakten und Integration mit externen Ökosystemen sind.

3.5.1 Modellierung

Die Modeling-APIs enthalten die CIM API (1), IOs API (2) und Application API (3) und bilden zusammen die Grundlage zur strukturierten und erweiterbaren Beschreibung des Cyber-Physical Model (CPM). Im Kern dient eine MongoDB-Datenbank als persistenter Speicher-Backend für das CPM. Das flexible, JSON-ähnliche Dokumentmodell von MongoDB ermöglicht es, heterogene Datenstrukturen zu speichern und zu erweitern, ohne disruptive Schema-Migrationen durchführen zu müssen, was insbesondere für Energiesystemmodelle vorteilhaft ist, deren Strukturen sich im Laufe der Zeit durch die Einführung neuer Gerätekategorien, Attribute oder Softwareentitäten weiterentwickelt. Die Collections in MongoDB spiegeln die hierarchische Struktur des im CGMES-Standard verwendeten CIM-Formats wider. Dies gewährleistet semantische Übereinstimmung mit CGMES und ermöglicht einen standardisierten Austausch von Netzmodellen zwischen Tools und Organisationen.
Jenseits der CIM-basierten Repräsentation des physikalischen Energiesystems sind zusätzliche Collections definiert, um den CL und den AL darzustellen und damit das CPM um sowohl kommunikationstopologische Objekte (z. B. Netzwerk-Schnittstellen, Nachrichtenkanäle, Protokolle) als auch Software-Artefakte (z. B. Applikationen, Services und deren Konfigurationen) zu erweitern. Auf diese Weise sind physische Netzkomponenten und Cyber-Entitäten konsistent in einer einheitlichen Datenbank verknüpft. Beispielsweise kann ein Applikationsobjekt in der AL sowohl auf seine Ausführungsumgebung (z. B. ein Container-Image und Bereitstellungskonfiguration) als auch auf den Satz von IOs verweisen, die es in der CL konsumiert oder produziert, wodurch Rückverfolgung von hochrangigen Use Cases bis hin zu einzelnen Signalen und Prozessen ermöglicht wird.

In der User Domain wird beim Import eines aus mehreren XML-Dateien zusammengesetzten CIM-Modells (Schritt (1) in Figure 6) ein dediziertes Modell Konvertierungs-Modul aufgerufen. Diese Komponente parst die CIM-XML-Dateien und transformiert die CIM-Repräsentation in JSON-ähnliche Objekte, die für die Speicherung in MongoDB geeignet sind. Während dieser Transformation wird eine Schema-Validierung durchgeführt, um die Konformität mit den CIM-Definitionen zu verifizieren, fehlende oder inkonsistente Referenzen zu erkennen und grundlegende Integritätsbedingungen durchzusetzen (z. B. Eindeutigkeit von Identifikatoren und Typkorrektheit von Attributen). Diese Pipeline kann konzeptionell als Abbildung gesehen werden

$$f_{CIM \to CMP}: M_{CIM} \to M_{CPM}$$

wobei M_{CIM} die Menge CIM-konformer XML-Modellinstanzen und M_{CPM} die entsprechenden JSON-basierten CPM-Instanzen in MongoDB bezeichnet. Die Abbildung $f_{CIM \to CPM}$ ist darauf ausgelegt, informations-erhaltend hinsichtlich der CIM-Semantik zu sein, sodass importierte Daten mit den offiziellen CIM-Definitionen übereinstimmen und zugleich nativ vom übrigen Framework genutzt werden können.

Nach dem CIM-Import wird die CPM-Beschreibung inkrementell erweitert, indem zusätzliche Entitäten innerhalb der CL und der AL definiert werden. Die IOs-API (2) verwaltet die Create-, Read-, Update- und Delete-(CRUD-)Operationen für in Abschnitt 3.4.3 beschriebene Kommunikationsschicht-Objekte. Dazu gehört beispielsweise das Konfigurieren von Nachrichtentopics oder -queues sowie das Zuordnen von IOs zu bestimmten CIM-Elementen (wie Messwerten oder Steuerbefehlen). Entsprechend stellt die Application-API (3) CRUD-Operationen für in Abschnitt 3.4.4 beschriebene Applikationsschicht-Objekte bereit. Diese APIs kapseln Validierungslogik und Beziehungsmanagement und verhindern damit, dass Client-Anwendungen inkonsistente oder partielle CPM-Zustände erzeugen.

Diese Organisation stellt sicher, dass die Modellierungsschicht als einziger Punkt der Wahrheit für die Gesamtsystembeschreibung fungiert und dabei das Stromnetz, die Kommunikationsinfrastruktur und die Applikationen umfasst. Da alle nachfolgenden Workflow-Stufen – einschließlich des Deployments (über die Deployment-API), Workflow-Orchestrierung (über die Workflows-API und Argo Workflows) und die Versuchsausführung in der Simulations Domäne – auf CPM-Instanzen arbeiten, die über diese Modellierungs-APIs abgerufen werden, ist Konsistenz domänenübergreifend gewährleistet. Darüber hinaus erleichtert die Verwendung einer zentralen CPM-Repräsentation die Reproduzierbarkeit von Experimenten, das Versionieren von Systemkonfigurationen und den systematischen Vergleich unterschiedlicher Szenarien, da jedem Experiment ein spezifischer in der Datenbank gespeicherter CPM-Snapshot zugeordnet werden kann.

3.5.2 CPM User Interface (CPM UI)

Um die Nutzung zu erleichtern und die mit dem Framework verbundene Lernkurve zu reduzieren, wurde eine dedizierte Benutzerschnittstelle, dargestellt in Abbildung 7, entwickelt, um die intuitive Analyse und das Verständnis des cyber-physischen Modells zu unterstützen. Die Schnittstelle stellt eine interaktive Umgebung zur Visualisierung des Modells als geschichteten Graphen bereit, die eine schrittweise Analyse der Systemkomplexität ermöglicht. Ein dediziertes Panel erlaubt es Benutzern, die Physische-, Kommunikations- und Applikations-Schicht unabhängig voneinander ein- und auszuschalten, wodurch es möglich wird, zu beobachten, wie sich das Modell schrittweise entwickelt, wenn zusätzliche Ressourcen eingeführt werden.

Ausgehend vom PL ergänzt die Visualisierung den Graphen sukzessive um Kommunikationsknoten und -verbindungen und schließlich um Komponenten auf Anwendungsebene, die alle in einem gemeinsamen räumlichen Kontext gerendert werden, um die strukturellen Beziehungen zwischen den Schichten zu bewahren. Ein synchronisiertes Datenpanel listet die zugrunde liegenden Graph-Elemente mit Filtermöglichkeiten auf und

gewährleistet somit die Rückverfolgbarkeit zwischen visuellen Knoten und deren Metadaten. Dieser gestaffelte, Schicht-für-Schicht-Ansatz unterstützt die Analyse von Abhängigkeiten und Interaktionen und hilft Benutzern zu verstehen, wie Ressourcen systematisch zum CPM hinzugefügt und in dieses integriert werden.

Neben den unterschiedlichen Schichten stellt die 3D-Visualisierung explizit den Informationsfluss innerhalb des cyber-physischen Modells dar. Richtungsangaben werden standardmäßig durch Pfeile angezeigt, um Datenübertragungswege zwischen Knoten über die Schichten hinweg zu kennzeichnen, sodass Benutzer unmittelbar wahrnehmen können, wie Informationen durch das System fließen. Diese Pfeile sind farblich codiert, um die Art der ausgetauschten Daten zu unterscheiden: Sollwerte, die im AL entstehen und über den CL in den PL propagiert werden, werden mit roten Pfeilen dargestellt, während Messwerte, die in umgekehrter Richtung fließen — vom physischen elektrischen Netz über die Kommunikationsinfrastruktur hinauf zur Applikationsschicht — mit grünen Pfeilen dargestellt werden. Diese Visualisierung des Datenflusses unterstützt ein intuitives Verständnis von Steuerungs- und Rückkopplungsmechanismen und macht die funktionalen Beziehungen zwischen den Schichten deutlich.

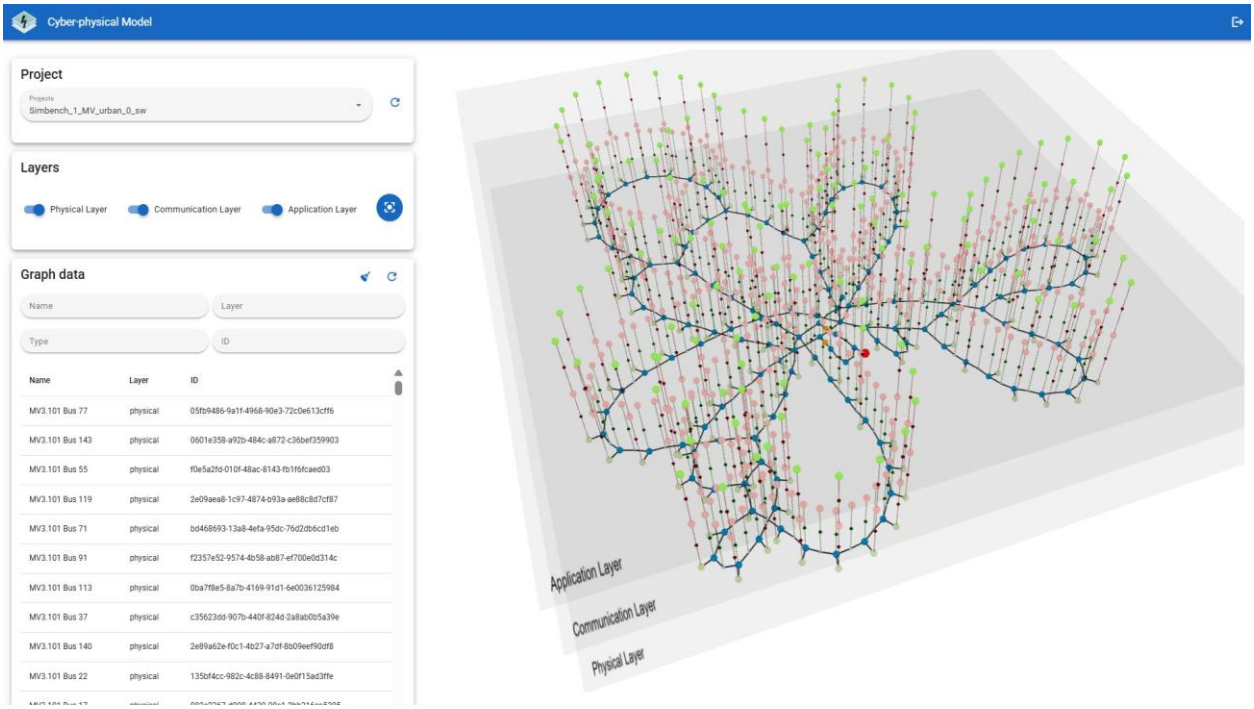

Abbildung 7 Benutzeroberfläche des cyber-physischen Modells.

3.5.3 Simulationskomponenten

Redis

Redis wird als Kommunikationsmiddleware über die Schichten hinweg genutzt und gewährleistet einen latenzarmen Austausch zwischen allen Simulationskomponenten. Im ACTV-Framework fungiert Redis als gemeinsames In-Memory-Daten-Framework für zeitkritische Informationen wie Stellgrößen, Messwerte und Statusflags. Mit Bibliotheken für mehr als 50 Programmiersprachen bietet es breite Interoperabilität und einfache Integration mit heterogenen Werkzeugen und Diensten, die von numerischen Solvern bis hin zu Orchestrierungs-Utilities reichen.
Durch die Nutzung atomarer Operationen auf Befehls-Ebene (z. B. Einzelschlüssel-Updates oder Mehrschlüssel-Transaktionen) verhindert Redis Race Conditions bei gleichzeitigen Lese-/Schreibzugriffen und trägt zur Bewahrung der Integrität von Messwerten und Sollwerten bei.

Dies ist besonders bei hoher Parallelität relevant, etwa wenn mehrere Regelungsalgorithmen Sollwerte anpassen, während der Solver gleichzeitig neue Netz-Zustände versendet. Darüber hinaus ermöglicht die Unterstützung von Datenstrukturen wie Hashes, Lists und Streams eine effiziente Darstellung von Zeitreihendaten oder gebündelten Simulationsergebnissen und erleichtert die Implementierung von publish/subscribe-Nachrichtenpatterns für ereignisgesteuerte Workflows.
Zusätzlich unterstützt Redis horizontale Skalierbarkeit durch Clustering und Sharding, wodurch die Verteilung von Daten und Arbeitslast über mehrere Knoten möglich ist. Dies erlaubt es der Simulation, auf große Netze mit vielen Agenten und hoher zeitlicher Auflösung zu skalieren, ohne durch I/O-begrenzt zu werden. Im Simulationskontext erleichtert Redis somit die Kommunikation über die Schichten hinweg.

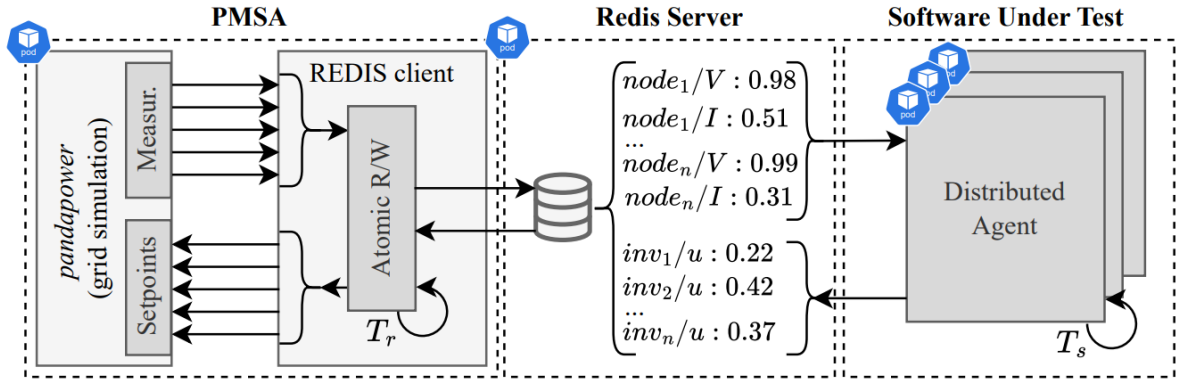

Abbildung 8 Implementierung der Kommunikationsschicht mit Redis. Das Diagramm zeigt den Datenfluss zwischen den Hauptkomponenten der Simulationen, z. B. dem Netzberechnungsprogramm (Grid Solver) und den zu testenden Anwendungen.

Abbildung 8 stellt Redis als CL dar, die den Informationsaustausch zwischen dem Netzsolver (PL) und der AL ermöglicht. In dieser Architektur fungiert PMSA als Netzsolver, der netzbezogene Daten erzeugt und liest, während die Anwendungsschicht aus der zu testenden Software besteht. Redis ist zwischen diesen beiden Schichten positioniert und implementiert den Kommunikationsmechanismus, indem es gemeinsame Daten wie Sollwerte und Messungen speichert und verteilt. Der Solver und die verteilten Anwendungskomponenten interagieren über Redis-Clients mit Redis und verwenden atomare Lese- und Schreiboperationen, um Konsistenz und korrekte Synchronisation sicherzustellen. Auf diese Weise entkoppelt Redis den Netzsolver von der Anwendungslogik und bietet gleichzeitig ein zuverlässiges, latenzarmes Medium für den Datenaustausch zwischen den beiden Schichten.

Solver

Der Solver ist eine der Kernkomponenten des ACTV-Frameworks und Teil der Simulations Domäne wie in Abbildung 6 gezeigt, wo er für die Ausführung von Lastfluss-Simulationen verantwortlich ist. Er basiert auf pandapower [5] und ist als Docker-Image gebündelt, wodurch eine nahtlose Integration in die Kubernetes-Umgebung und reproduzierbare Ausführung über verschiedene Deployments hinweg sichergestellt wird. Durch die Containerisierung werden alle Solver-Abhängigkeiten gekapselt, was die Versionsverwaltung vereinfacht und eine unabhängige Skalierung von Solver-Instanzen ermöglicht.

Bei der Initialisierung ruft der Solver die CPM-Beschreibung aus der Datenbank ab. Mittels des in pandapower integrierten CIM-Importers übersetzt er die PL-Beschreibung der CPM in interne Datenstrukturen, wie etwa Busse-, Leitungen-, Transformator- und Lasttabellen. Parallel dazu liest er die CL-Definitionen, um zu bestimmen, welche I/O-Endpunkte ihn mit dem AL verbinden, und stellt damit eine klare Abbildung zwischen Simulationsvariablen (z. B. Sollwerte, Messwerte) und Redis-Keys oder -Streams her. Nach Abschluss dieser Initialisierungsphase tritt der Solver in eine periodische Simulationsschleife mit einer konfigurierbaren Schrittzeit Δt ein, die von sub-sekündlicher Auflösung für schnelle Dynamiken bis zu mehreren Minuten für langfristige Planungsstudien reichen kann. In jedem Zyklus führt

der Solver die folgenden Schritte aus:

1. Er liest Netzkomponenten-Sollwerte und Randbedingungen aus Redis (z. B. Wirkleistungs-/Blindleistungseinspeisungen, Transformator-Stufen-Positionen oder DER-Kontrollsignale).
2. Der Solver aktualisiert entsprechend das interne pandapower-Netzmodell.
3. Anschließend berechnet er einen Lastfluss mittels pandapower's Newton–Raphson- oder anderen verfügbaren Algorithmen.
4. Zuletzt schreibt der Solver die resultierenden Messwerte – wie Busspannungen, Leitungsauslastung, Transformatorauslastung oder Verluste – zurück nach Redis.

Das Ausführungsintervall des Solvers ist somit vollständig konfigurierbar, wodurch er sowohl für Studien mit hoher zeitlicher Auflösung als auch für langsamere, systemweite Analysen (z. B. Day-Ahead-Planung oder Szenario-Bewertungen) geeignet ist. Darüber hinaus kann der Solver mehrfach parallel instanziiert werden, etwa um verschiedene Szenarien oder Parametersätze gleichzeitig zu evaluieren, wobei Kubernetes die Pod-Scheduling- und Ressourcenallokation übernimmt.
Dank seiner Bereitstellung als Docker-Image und modularen Gestaltung kann der Solver mit alternativen Engines erweitert werden, z. B. ANDES [3] für dynamische Stabilitätsanalysen, Wrapper für Real-Time-Digital-Simulatoren (RTDSs) oder zusätzliche Kommunikationsprotokolle wie MQTT oder OPC UA, ohne das bestehende Framework zu beeinträchtigen. Diese Erweiterbarkeit ermöglicht es dem ACTV-Framework, eine breite Palette von Simulationstypen (stationär, dynamisch, Co-Simulation) zu unterstützen und gleichzeitig eine einheitliche Integrations- und Orchestrierungsoberfläche beizubehalten.

Grid Profiler

Das Grid Profiler-Modul ist Teil der Simulationskomponenten, die vom ACTV-Framework bereitgestellt werded, und ergänzt den Solver durch die Bereitstellung zeitlich variabler Eingangssignale. Da Profile (Zeitserien) in den meisten Simulationen verwendet werden, kann der Grid Profiler zusammen mit dem Solver und den Applikationskomponenten während des Deployment-Schrittes bereitgestellt werden, wie in Abbildung 6 dargestellt. Er ermöglicht Anwendern, dynamische Grid-Profile zu definieren und anzuwenden; diese werden in der zentralen Datenbank gespeichert und bei der Initialisierung der Simulation abgerufen.
Profile können mit einer konfigurierbaren Periodizität angewendet werden, um ihre zeitliche Auflösung und Ausrichtung am Zeitschritt des Solvers Δt auszurichten. Beispielsweise können Anwender Profile mit 1 min Auflösung für Verteilnetzstudien oder mit 15 min Auflösung für marktorientierte Analysen spezifizieren, mit der Option zur Interpolation oder Aggregation bei Bedarf. Diese Profile werden typischerweise verwendet, um Haushaltslasten, Ladevorgänge von Elektrofahrzeugen oder Photovoltaik-Erzeugungskurven abzubilden, können aber auch andere exogene Signale kodieren, wie temperaturabhängige Energienachfrage, Informationen zum Energiespeichereinsatz oder makrtorientiertes Verhalten.
Operativ aktualisiert der Grid Profiler periodisch die relevanten Sollwert-Aktualisierungen in Redis entsprechend dem aktiven Profil und der aktuellen Simulationszeit. Er dient als zentraler Mechanismus zur Verwaltung dieser Eingaben innerhalb der Simulations Domäne und ermöglicht reproduzierbare Szenariendefinitionen, systematische Sensitivitätsanalysen sowie die Kopplung von Simulationsläufen mit externen Datenquellen (z. B. historischen Messdaten oder synthetischen Profilen, die von dedizierten Tools erzeugt werden). In Kombination mit dem Solver und der Redis-basierten Middleware bildet er einen zentralen Baustein für skalierbare, datengesteuerte Experimente innerhalb des ACTV-Frameworks.

3.5.4 Deployment

Die in der Deployment-Phase beteiligten Module und Komponenten sind in Abbildung 6 blau hervorgehoben. In der vorgeschlagenen Architektur ist die Deployment-Phase dafür verantwortlich, eine abstrakte Versuchsspezifikation in eine Menge konkreter, ausführbarer

Artefakte auf einem Kubernetes-Cluster zu transformieren. Diese Transformation wird vom Dienst KubeLaunchPad orchestriert, der die Logik für die Ressourcenbereitstellung, das Konfigurationsmanagement und die Lebenszykluskontrolle der deployten Workloads kapselt. Durch die Zentralisierung dieser Belange entkoppelt KubeLaunchPad die abstrakteren CPM-Beschreibungen von den Detailaspekten der Container-Orchestrierungsplattform.

Operativ werden alle Ressourcen auf einem Kubernetes-Cluster bereitgestellt und von KubeLaunchPad verwaltet, das sowohl die Zuweisung als auch die Konfiguration dieser Ressourcen automatisiert. Wenn ein Benutzer eine Deployment-Anfrage über die Deployment API (4) stellt, wird die Anfrage über die API-Schicht und den Message Broker an KubeLaunchPad weitergeleitet. Der Dienst holt dann die entsprechende CPM-Beschreibung aus der Datenbank, validiert sie und übersetzt sie in eine Menge von Kubernetes-Manifests. Diese Manifests umfassen typischerweise, jedoch nicht ausschließlich, Pods (definieren die Container und deren Laufzeitspezifikationen), Services, ConfigMaps (speichern nicht-sensitive Konfigurationsdaten wie Simulationsparameter, Topologie-Beschreibungen oder Logging-Einstellungen) und Secrets (speichern sensitive Informationen wie Zugriffstokens, Zugangsdaten für externe Registries oder Verschlüsselungsschlüssel). Die Erzeugung dieser Manifests kann formalisiert werden als eine Abbildung

$$f_{deploy} : C_{CPM} \rightarrow M_{K8s},$$

wobei C_{CPM} den Raum gültiger CPM-Konfigurationen und M_{K8s} den Raum von Kubernetes-Ressourcenspezifikationen bezeichnet. Die Funktion f_{deploy} ist in KubeLaunchPad implementiert und garantiert, dass die syntaktischen und semantischen Vorgaben sowohl des CPM-Schemas als auch der Kubernetes-API erfüllt werden.

Um Multi-Tenancy durchzusetzen und Rückwirkungen zwischen gleichzeitig laufenden Experimenten zu vermeiden, werden die Ressourcen für jedes Projekt in einem dedizierten Kubernetes-Namespace instanziiert. Diese Namespace-Ebene Isolation bietet mehrere Vorteile: (i) klare Trennung der Verantwortungsbereiche zwischen Benutzer:innen und Experimenten, (ii) fein granulare Zugriffskontrolle durch Role-Based Access Control (RBAC)-Policies, (iii) unabhängige Ressourcenzuweisungen zur Verhinderung von Ressourcenverknappung und (iv) vereinfachtes Lifecycle-Management, da alle mit einem bestimmten Experiment assoziierten Ressourcen als eine einzige logische Einheit erstellt, überwacht und gelöscht werden können. Namespaces bilden damit die primäre Isolationsgrenze zwischen verschiedenen Simulationsinstanzen in der Simulationsdomäne von Abbildung 6.

Innerhalb jedes Namespaces umfassen die bereitgestellten Ressourcen den pandapower-basierten Solver, eine Redis-Instanz für die Inter-Modul-Kommunikation sowie eine Reihe von Capsules, die benutzerdefinierte Applikationen instanziieren. Der pandapower-Solver wird als Pod bereitgestellt, der die numerische Lastfluss-Simulations-Engine kapselt. Redis wird als In-Memory-Datenbank und Message-Broker eingesetzt, um den latenzarmen Austausch von Zwischenergebnissen, Konfigurationsupdates und Steuerungsnachrichten zwischen der zu testenden Software, Solver-, Grid-Profiler- und Experiment-Komponenten zu unterstützen. Capsules werden als Pods realisiert, die aus Docker-Images erstellt werden, welche aus externen Registries (z. B. Docker Hub, Quay.io oder ERC) bezogen werden. Während der Instanziierung wird jede Capsule über eingehängte ConfigMaps und Secrets mit den entsprechenden Konfigurationsdaten verdrahtet, wodurch sichergestellt wird, dass dasselbe Container-Image in mehreren Experimenten mit unterschiedlichen Parametrisierungen wiederverwendet werden kann. Sobald alle Manifeste generiert sind, wendet KubeLaunchPad diese mittels der Kubernetes-API auf den Ziel-Cluster an. Die resultierenden Objekte werden anschließend kontinuierlich durch Kubernetes überwacht und es wird sicherstellt, dass der im CPM-abgeleiteten Manifests ausgedrückte Soll-Zustand mit dem Ist-Zustand des Clusters übereinstimmt. In Kombination mit darauch ausetzenden Workflow-Orchestrierung (z. B. via Argo Workflows) ermöglicht dieser Mechanismus eine End-zu-End-Automatisierung von der Modellspezifikation bis zur großskaligen, containerisierten Ausführung von Simulationen

innerhalb des ACTV-Frameworks.

3.5.5 Experimente

Das ACTV-Framework stellt eine dedizierte API-Schicht zum Konfigurieren und Ausführen von Experimenten auf laufenden Simulationen bereit. Die in diese Experimente involvierten Ressourcen und Module sind in Abbildung 6 lavendelfarben hervorgehoben. Innerhalb dieser Architektur dienen Experimente primär dazu, das Verhalten der zu testenden Software – in Abbildung 6 als Agents bezeichnet – unter einem breiten Spektrum von Betriebsbedingungen, Fehlerszenarien und Profilen zu bewerten. Durch systematisches Variieren dieser Bedingungen können Benutzer Validierungs-, Regressions- und Robustheitstests in kontrollierter und reproduzierbarer Weise durchführen.
Experimente werden mithilfe von Argo Workflows implementiert, die sie als Kubernetes-Jobs organisieren, die in Directed Acyclic Graphs (DAGs) angeordnet sind. Diese Workflow-Abstraktion ermöglicht sowohl sequentielle als auch parallele Ausführung von Aufgaben und bietet damit die Flexibilität, komplexe Validierungsszenarien zu definieren, die aus mehreren Experimentenstufen bestehen. Jeder Knoten im DAG entspricht einem diskreten Experimentierschritt, wie z. B. dem Bereitstellen oder Neukonfigurieren eines Agenten, dem bewussten Stören der Simulationsumgebung oder dem Sammeln von Metriken und der Nachverarbeitung. Diese Schritte können mit anderen Simulationskomponenten (z. B. dem Solver, Grid Profiler und Agent-Pods in der Simulation Domain) interagieren, über Redis auf gemeinsamen Zustand und Zwischenergebnisse zugreifen und Artefakte produzieren, die von nachfolgenden Aufgaben konsumiert werden. Der explizit im DAG kodierte Daten- und Kontrollfluss erleichtert zudem die Rückverfolgbarkeit der Experimentenlogik, wodurch sich Reproduzierbarkeit, Debugging und der Vergleich unterschiedlicher Experimentkonfigurationen vereinfachen.
Experimente werden über die Workflows API (5) konfiguriert, die Operationen zum Registrieren, Instanziieren und Verwalten von Argo-Workflow-Spezifikationen bereitstellt. Über diese APIs erlaubt das ACTV-Framework den Nutzern, (i) den Fortschritt einzelner Workflow-Knoten und des gesamten Experiments zu kontrollieren, (ii) Fehler zu behandeln, indem Kubernetes- und Argo-Retry-Policies oder bedingte Verzweigungen im DAG genutzt werden, und (iii) die Ausführung der Experimente elastisch über den Kubernetes-Cluster zu skalieren, indem die Parallelitätsgrenzen und Ressourcenanforderungen für jede Aufgabe angepasst werden. Dieses Design unterstützt sowohl Einzellauf-Experimente als auch große Experiment-Batches, etwa bei Parameterstudien, Sensitivitätsanalysen oder Monte-Carlo-artigen stochastischen Auswertungen.
Während der Experimentenausführung werden Daten kontinuierlich aufgezeichnet und in der zentralen Datenbank abgespeichert, wie in Abbildung 6 angezeigt. Diese Daten können direkte Ausgaben der Simulation, Logs der Agenten, Ereignisse, Performance-Counter und aggregierte Indikatoren umfassen, die von Analyse-Schritten innerhalb des Workflows berechnet werden. Am Ende eines Experiments können Benutzer die vollständige Kollektion an Artefakten über die Artifacts API (6) abrufen. Diese API stellt ein einzelnes (ZIP-)Archiv bereit, das alle relevanten Ergebnisse und Metadaten bündelt, die während des Laufs erzeugt wurden, wie Konfigurationsdateien, Workflow-Spezifikationen, Logs und Ergebnisdateien. Die Verfügbarkeit eines selbstenthaltenden Artefaktpakets vereinfacht die Reproduzierbarkeit von Experimenten und die anschließende Analyse erheblich, sodass Benutzer Experimentenläufe archivieren, mit Kollaboratoren teilen und in externe Tools für Visualisierung, statistische Nachbearbeitung oder machine-learning-basierte Modellkalibrierung integrieren können.

3.6 Bewertung von Anwendungsfall und Skalierbarkeit

Dieser Abschnitt stellt einen zweiteiligen Use Case vor, um die Anwendung und Skalierbarkeit des vorgeschlagenen Frameworks im DES-CERT-Kontext zu veranschaulichen. Zunächst beschreiben wir das schrittweise Vorgehen zur Konfiguration und Ausführung einer Simulation ausgehend von einem CPM. Zweitens demonstrieren wir die Skalierbarkeit des Frameworks,

indem wir dasselbe experimentelle Vorgehen für ein wesentlich größeres digitales Energiesystem wiederholen.

In konventionellen Stand-derTechnik-Workflows erfordert ein derartiger Übergang zu einem größeren System typischerweise umfangreiche manuelle Konfigurationsaufwände. Im Gegensatz dazu erfordert der vorgeschlagene DES-CERT-orientierte Ansatz lediglich eine Modifikation der CIM-Datei zu Beginn des Aufsetzens der Simulation, wodurch der manuelle Eingriff erheblich reduziert wird, und die effiziente Vorzertifizierung skalierbarer digitaler Energiesysteme unterstützt wird.

3.6.1 Cyber-physische Simulation

Abbildung 9 zeigt den ACTV-Workflow in Form eines Pseudo-Codes innerhalb der Nutzerdomäne (linke Seite des Diagramms). Der Pseudo-Code ist als Abfolge von Schritten strukturiert, die erforderlich sind, um eine cyber-physische Simulation eines durch ein CPM spezifizierten Systems auszuführen. Für jeden Schritt wird die entsprechende Wirkung in der CPM Domäne abgebildet, sodass Benutzer nachverfolgen können, welche Ressourcen instanziiert werden und wie sie zum fortschreitenden Aufbau des CPM beitragen. Die Cluster Domain auf der rechten Seite veranschaulicht, wie diese Ressourcen schließlich in der Recheninfrastruktur bereitgestellt und ausgeführt werden. Zur Unterstützung der Verständlichkeit und Nachvollziehbarkeit über die Domänen hinweg wird ein konsistentes Farbschema angewendet, sodass identische Ressourcen im gesamten Workflow leicht verfolgt werden können.

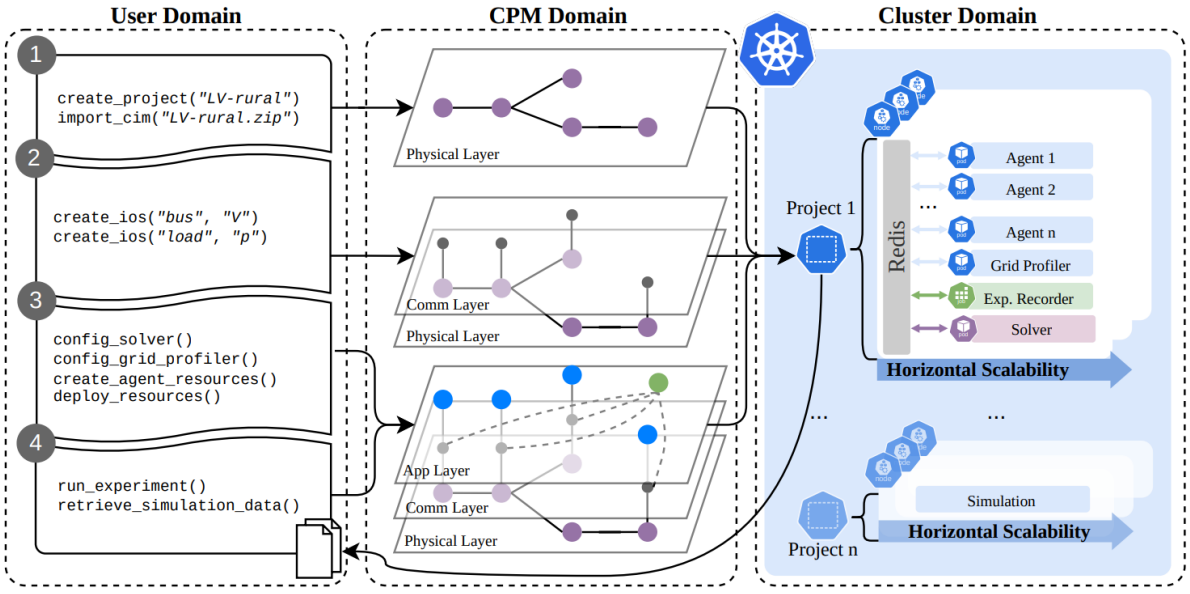

Abbildung 9 ACTV-Workflow, dargestellt als Pseudocode in der Benutzerdomäne (Nutzerdomäne), mit entsprechenden Auswirkungen, die in der CPM-Domäne visualisiert werden, und deren Bereitstellung in der Cluster-Domäne gezeigt wird. Eine konsistente Farbkodierung über alle Domänen hinweg ermöglicht die einfache Rückverfolgbarkeit von Ressourcen.

Der Workflow beginnt damit, dass der Benutzer ein neues Projekt für ein spezifisches Stromnetz anlegt und dessen Topologie im CIM-Format importiert (Schritt (1)). Dieser Vorgang erzeugt den PL, das die Verbindungen zwischen Netzkomponenten abbildet und im CPM-Domän visualisiert wird. Mit Hilfe der CPM-UI sind die Ergebnisse in Abbildung 10 dargestellt. In Schritt (2) legt der Benutzer den CL fest, indem er Interface Nodes, Channels und Connections definiert. Die API des ACTV-Frameworks stellt Endpunkte bereit, die die Erstellung dieser Entitäten erleichtern, etwa indem Eingangs- und Ausgangsschnittstellen für Busspannungen oder Lastleistungen in einer einzigen Operation definiert werden können. Im Diagramm durch graue Knoten und Linien dargestellt, formalisiert der CL die Struktur des Informationsaustauschs innerhalb des CPM. Die 3D-Visualisierung des CLs ist in Abbildung 11 gezeigt.

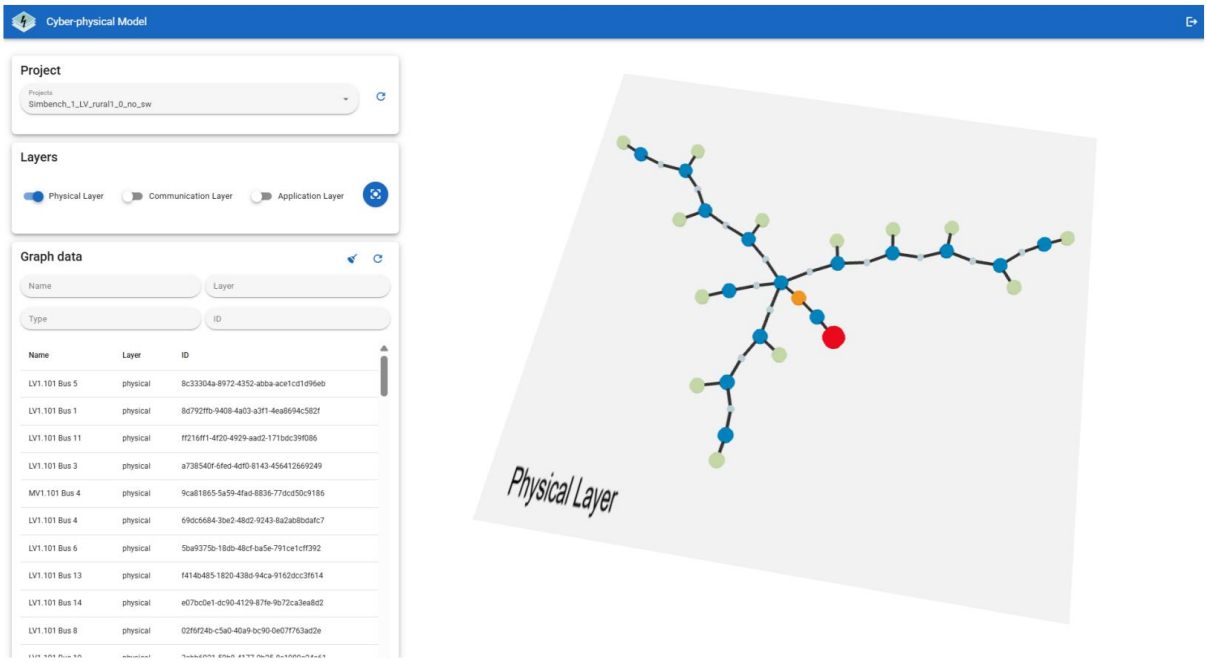

Abbildung 10 3D-Darstellung der Physischen Schicht.

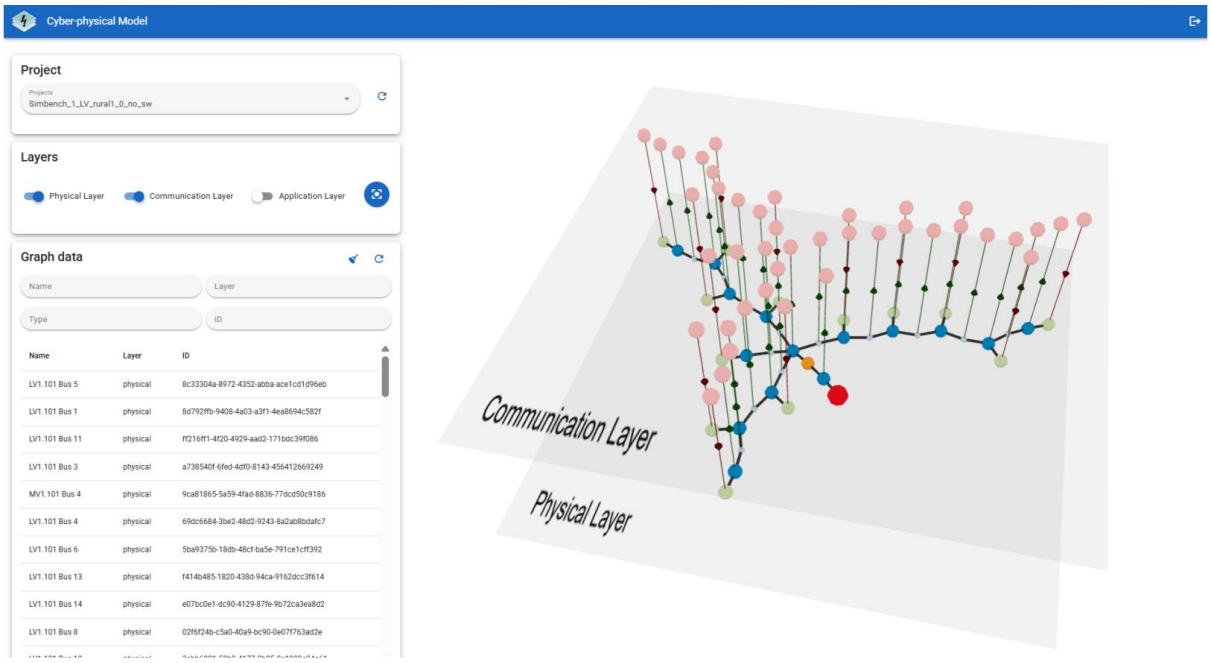

Abbildung 11 3D-Darstellung der Physischen und Kommunikationsschicht.

Sobald der CL definiert ist, konfiguriert der Benutzer in Schritt (3) die AL. Diese Schicht kapselt die zu testende Software und zeichnet sich dadurch aus, dass sie ihr Image, Capsules, ConfigMaps und Sealed-Secrets bereitstellt. Neben der AL werden auch weitere Ressourcen spezifiziert, wie der Lastfluss Solver, dessen Konfiguration und die Netz-Profile. Die 3D-Visualisierung des ALs ist in Abbildung 12 dargestellt.

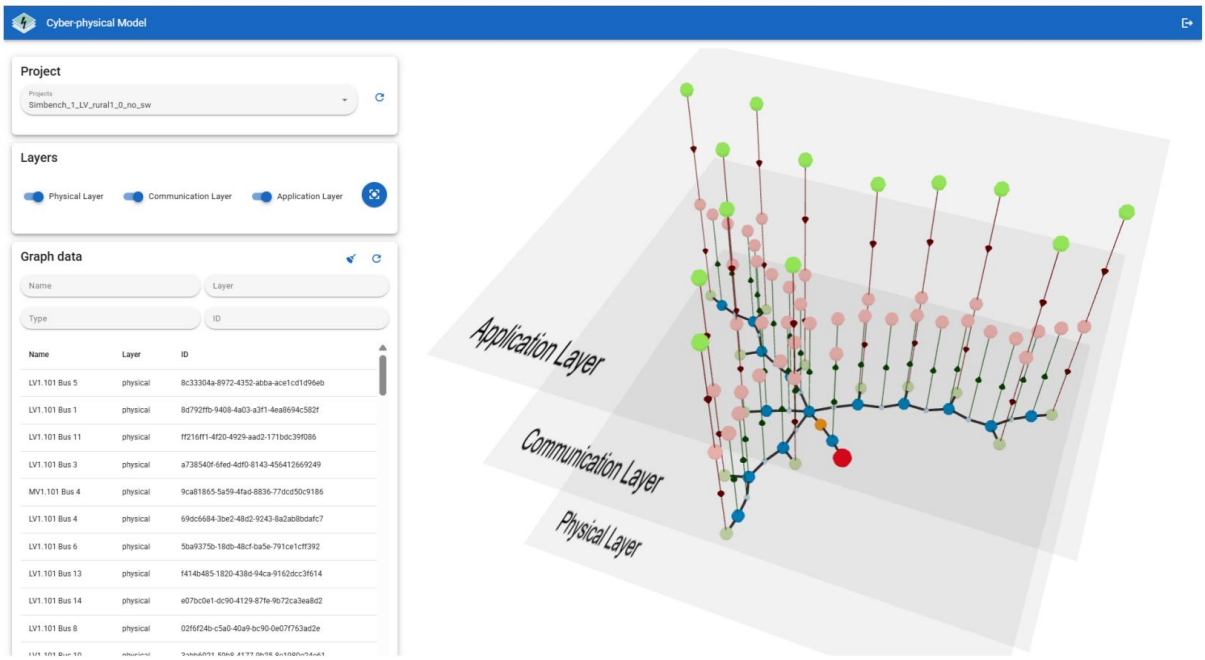

Abbildung 12 3D-Darstellung der Physical, Communication und Application layers

Diese Ressourcen sind für die Durchführung der Simulation notwendig, gehören jedoch nicht zur Anwendungslogik selbst. Nachdem das CPM vollständig definiert ist, werden die benötigten Projektressourcen provisioniert und im Cluster bereitgestellt. In diesem Stadium wird die Simulationsumgebung automatisch instanziiert, wodurch sichergestellt wird, dass alle Abhängigkeiten, Konfigurationen und Laufzeitkomponenten korrekt initialisiert sind. Mit der vorhandenen Umgebung wechselt der Workflow in die Ausführungsphase, in der die konfigurierte Simulation gestartet, gemonitort und über die Clusterinfrastruktur verwaltet wird.

Schließlich definiert und führt der Benutzer in Schritt (4), ein oder mehrere Experimente auf der laufenden Simulation durch. Jedes Experiment wird als Job im Cluster angelegt und kann mehrere parallele Aufgaben enthalten. Das konkrete Design des Experiments ist anwendungsfallabhängig und kann beispielsweise das Anwenden vordefinierter Sollwerte zur Bewertung des Verhaltens der zu testenden Software (im Diagramm als Agents bezeichnet) umfassen. Während der Ausführung werden kontinuierlich Daten erhoben, die nach Abschluss zur nachgelagerten Analyse abgerufen werden können. Dieser strukturierte Workflow unterstützt direkt das Ziel von DES-CERT, systematische, reproduzierbare und skalierbare cyber-physische Simulationen als Grundlage für die Vorzertifizierung digitaler Energiesysteme zu ermöglichen.

Zur Ergänzung der schematischen Darstellung in Abbildung 9 wurde eine detailliertere Demonstration des Prozesses aufgezeichnet und ist via [6] öffentlich abrufbar.

3.6.2 Skalierbarkeit und Simulationsergebnisse

Um die Skalierbarkeit des ACTV-Frameworks im Kontext digitaler Energiesysteme zu bewerten, wurden zwei Benchmark-Netzmodelle aus dem Simbench-Datensatz [4] verwendet:

- **Simbench-1-LV-rural1-0-no-sw (LV1):** ein Niederspannungs-Ruralnetz mit 0,4 kV, bestehend aus 13 Knoten, versorgt durch einen 160 kVA Transformator.
- **Simbench-1-MV-urban-0-sw (MV1):** ein Mittelspannungs-Stadtnetz mit 10 kV, bestehend aus 134 Knoten, versorgt durch zwei 63 MVA Transformatoren.

Diese beiden Netze wurden ausgewählt, um die horizontale Skalierbarkeit des ACTV-Frameworks zu demonstrieren, von einem kleinen ländlichen Niederspannungsnetz bis hin zu einem deutlich größeren Mittelspannungsnetz städtischen Charakters.

Mit zunehmender Anzahl von Knoten und Agenten steigt entsprechend die Rechenlast. Das Framework adressiert dies durch die dynamische Orchestrierung zusätzlicher Kubernetes-Pods, wodurch größere und komplexere Simulationen ohne Neuimplementierungen oder zusätzlichen Konfigurationsaufwand ausgeführt werden können. Dieses elastische Verhalten ist insbesondere für DES-CERT relevant, da es die Bewertung digitaler Energiesysteme über verschiedene Skalen hinweg unterstützt, von lokalen Verteilernetzen bis zu regionalen Netzen, innerhalb einer konsistenten Präzertifizierungsumgebung.
Das Experiment modelliert die Interaktion verteilter Agenten, die einzelne netzgekoppelte Einheiten repräsentieren, wie Haushalte, Ladepunkte für Elektrofahrzeuge und Photovoltaik-Anlagen. Jeder Agent koordiniert sich mit dem Gesamtsystem durch Nachrichtenaustausch zu lokalem Verbrauch und Erzeugung. Während Agenten periodisch den aktuellen Leistungsbedarf aller aktiven Agenten abrufen, ist ihr Betrieb asynchron statt strikt synchronisiert. Überschreitet die aggregierte Netzauslastung eine vordefinierte globale Grenze, reagieren die Agenten gemeinsam, indem sie ihren individuellen Verbrauch proportional reduzieren. Dieser Mechanismus erzeugt Limits für das Gesamtsystem und bewahrt gleichzeitig Fairness unter den Teilnehmern, wodurch ein skalierbares Koordinationskonzept entsteht, das für die Präzertifizierung verteilter Regelungsstrategien in digitalen Energiesystemen relevant ist.

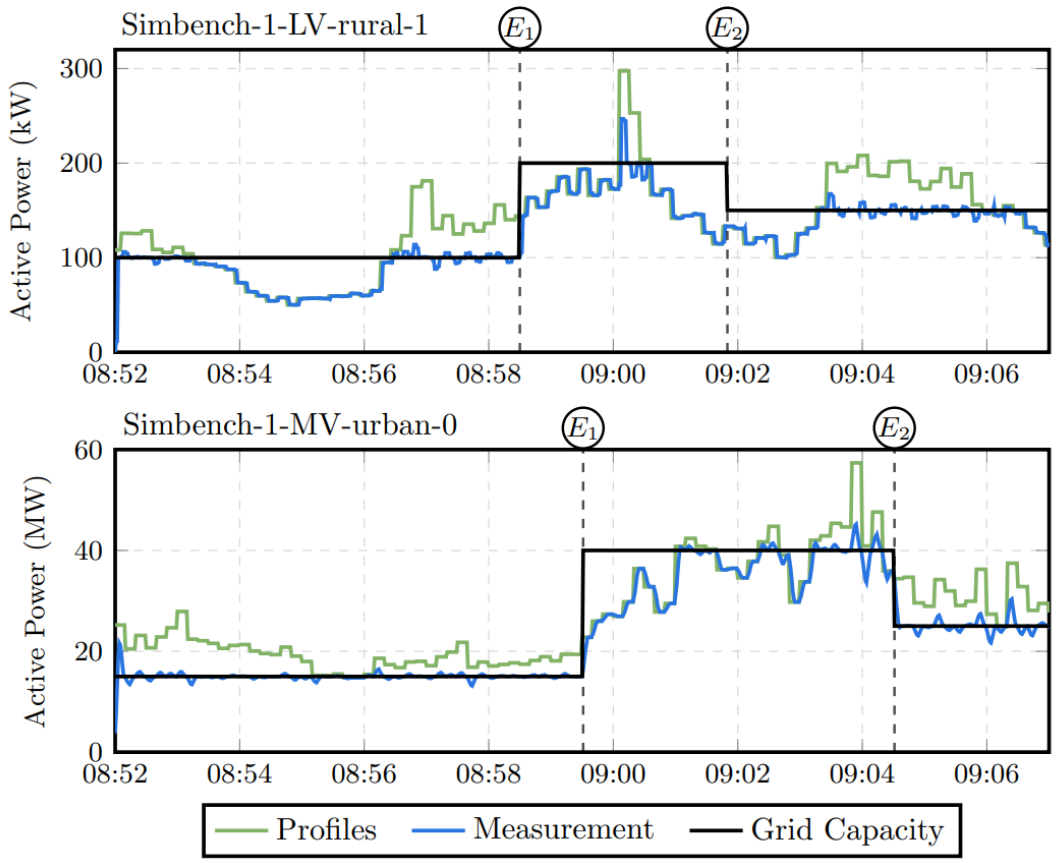

Abbildung 13 Simulationsergebnisse für die Anwendungsfälle LV1 und MV1. In LV1 folgte die Systemlast mit geringer Abweichung der Netzkapazität mit kurzen transienten Überschreitungen, aber ohne anhaltende Überlastungen, wobei das System auf einem einzelnen Knoten lief. Ein vergleichbares Verhalten wurde bei MV1 in größerem Maßstab beobachtet, wobei Kubernetes einen zweiten Knoten hinzufügte, um 142 Pods zu hosten.

Die Simulationsergebnisse sind in Abbildung 13 zusammengefasst. Im Szenario LV1 blieb die gesamte Systemlast nahe an der verfügbaren Netzkapazität, welche ein globales Limit darstellt, das vom Netzbetreiber definiert und beispielsweise zur Berücksichtigung zusätzlicher erneuerbarer Einspeisungen (wobei E1 und E2 Netzreaktionen auf Kapazitätsänderungen

entsprechen
) angepasst werden kann. Wann immer dieses Kapazitätslimit verändert wurde, bewirkte der Koordinationsmechanismus eine proportionale Verbrauchsreduktion über alle Agenten hinweg. Aufgrund der asynchronen Kommunikation erforderte das System nach einer Laständerung mindestens zwei Iterationen, um zu konvergieren. Folglich überschritt die aggregierte Last gelegentlich für kurze Intervalle den vordefinierten Schwellenwert, bevor sich die Agenten gemeinsam auf einen neuen angepassten Betriebspunkt stabilisierten. Trotz dieser transienten Überschüsse verhinderte die Koordination erfolgreich anhaltende Überlastungen. Für LV1 wurde die gesamte Simulation auf einem einzigen Kubernetes-Knoten ausgeführt, wobei alle Pods zusammen 41 mCPU und 769 MiB Speicher verbrauchten.

Im Szenario MV1 wurde das Framework auf eine deutlich größere Anzahl von Agenten und eine signifikant höhere Gesamtnachfrage skaliert. Der asynchrone Update-Mechanismus zeigte ein Konvergenzverhalten, das mit dem in LV1 beobachteten vergleichbar war. Für MV1 wurde das Cluster-Limit pro Knoten auf 110 Pods gesetzt. Da das Experiment 142 Pods erforderte (139 Agenten, Redis-Instanz, Grid Profiler und Pandapower-Solver), skalierte das Kubernetes-Cluster automatisch durch Hinzufügen eines zweiten Knotens, um die erhöhte Arbeitsbelastung handzuhaben. Der entsprechende Ressourcenverbrauch betrug 182 mCPU und 4682 MiB Speicher.

Diese Experimente bestätigen die Skalierbarkeit und Robustheit des ACTV-Frameworks selbst. Der Übergang von einem kleinen Niederspannungsnetz zu einem großen Mittelspannungsnetz erforderte keine Änderungen an der Agentenlogik, dem Redis-basierten Koordinationsmechanismus oder den API-Schnittstellen. Die verteilte Architektur ermöglichte ein nahtloses Skalieren von wenigen Agenten auf mehrere hundert, ohne zentrale Engpässe einzuführen, und bewahrte dabei konsistente Konvergenzeigenschaften über verschiedene Netzgrößen hinweg. Das ACTV-Framework wurde mit Netzen von bis zu 1000 Knoten getestet, und die zugrunde liegende Architektur ist darauf ausgelegt, noch größere Systeme zu unterstützen, was für das Ziel von DES-CERT, eine skalierbare Testumgebung für die Präzertifizierungsprüfung digitaler Energiesysteme bereitzustellen, wesentlich ist.

Darüber hinaus steht ein öffentlich verfügbares Video zur Verfügung, das einen detaillierten Walkthrough der erforderlichen Schritte zum Skalieren der Simulationen und zur Implementierung der entsprechenden Konfigurationsanpassungen bietet [36] und damit Transparenz und Reproduzierbarkeit im DES-CERT-Evaluationsworkflow unterstützt.

4 Ergebnisse und Schlussfolgerungen

Dieser Bericht stellt das Design, die Implementierung und die Evaluierung eines skalierbaren und automatisierten Frameworks zur Validierung digitaler Energiesysteme vor, das im Rahmen des DES-CERT-Projekts entwickelt wurde. Das Automated Cyber-Physical Testing and Validation (ACTV) Framework unterstützt systematische Validierungs-Workflows über verschiedene Stromnetzgrößen, Kommunikationsinfrastrukturen und Software-Deployments hinweg und adressiert das Fehlen integrierter und automatisierter Ansätze zur Konfiguration und Durchführung cyber-physischer Validierungsstudien in der aktuellen Praxis.

Das Framework basiert auf dem Cyber-Physical Model (CPM), das eine einheitliche Darstellung der physischen, Kommunikations- und Anwendungsschicht eines digitalen Energiesystems bietet. Durch die Kombination von Netztopologien, dem Informationsaustausch und Anwendungskonfigurationen in einem einzigen maschinenlesbaren Modell ermöglicht das CPM die automatisierte Bereitstellung und Ausführung von Simulationen bei gleichzeitiger Konsistenz über alle Framework-Komponenten hinweg. Die geschichtete Struktur erlaubt eine unabhängige Modellierung elektrischer Elemente, Kommunikationsschnittstellen und Softwarekomponenten, während explizite Beziehungen zwischen diesen erhalten bleiben. Dieses Design unterstützt

Reproduzierbarkeit und reduziert den Konfigurationsaufwand beim Skalieren von kleinen zu großen Systemen.

Die Implementierung des ACTV-Frameworks demonstriert die Anwendbarkeit cloud-nativer Technologien auf cyber-physische Validierungs-Workflows. Containerbasierte Lösungen und Kubernetes-Orchestrierung werden verwendet, um Simulationskomponenten und Anwendungen in isolierten Umgebungen bereitzustellen, was parallele Ausführung und kontrollierte Ressourcenvergabe ermöglicht. Die In-Memory-Datenbank Redis bietet einen leichtgewichtigen und latenzarmen Mechanismus zum Austausch von Messwerten und Sollwerten zwischen Simulations- und Anwendungskomponenten, während Argo Workflows die strukturierte Ausführung von aus mehreren Schritten bestehenden Experimenten ermöglicht. Zu validierende Anwendungen werden als Blackbox-Komponenten behandelt, sodass eine Validierung ohne Zugriff auf den Quellcode möglich ist und Szenarien mit proprietärer oder Drittsoftware unterstützt werden.

Die vorgestellten Anwendungsfälle und Skalierungsexperimente zeigen, dass das Framework auf Netze unterschiedlichster Größen ohne Änderungen an der Anwendungslogik oder dem Validierungs-Workflow angewendet werden kann. Das Skalieren von einem Niederspannungsnetz in ländlicher Umgebung zu einem Mittelspannungsnetz in städtischer Umgebung erforderte lediglich eine Modifikation des Eingang-Netzmodells (CIM), während Bereitstellung, Orchestrierung und Versuchsausführung unverändert blieben. Die Kubernetes-basierte Architektur bewältigte die erhöhte Rechenlast durch Zuweisung zusätzlicher Ressourcen nach Bedarf. Das beobachtete Verhalten der verteilten Agenten blieb in beiden Szenarien konsistent, was darauf hindeutet, dass das Framework skalierbare Experimente unterstützt, ohne zentrale Koordinationsengpässe einzuführen.

Im Rahmen des DES-CERT-Projekts liefert diese Arbeit eine technische Grundlage für strukturierte Validierungs-Workflows, die für Nutzer:innen mit unterschiedlichen Hintergründen zugänglich sind. Durch die Formalisierung der Systemkonfiguration mittels CPM und die Automatisierung von Bereitstellungs- und Ausführungsschritten senkt das ACTV-Framework die Einstiegshürde für die Durchführung komplexer cyber-physischer Validierungsstudien und reduziert das Risiko von Konfigurationsfehlern. Dies ist insbesondere relevant für iterative Testszenarien, in denen Systemkonfigurationen häufig und konsistent angepasst werden müssen.

Zukünftige Arbeiten werden sich auf die weitere Reduzierung des Nutzeraufwands und die Verbesserung der Benutzbarkeit konzentrieren. Geplante Erweiterungen umfassen die Unterstützung zusätzlicher Simulationsparadigmen, wie dynamische und Echtzeitsimulationen, sowie die Integration weiterer Kommunikationstechnologien (z. B. OPC UA oder MQTT). Darüber hinaus ist eine vielversprechende Richtung die Einführung KI-basierter Agenten zur Unterstützung der Nutzer:innen bei der Projektkonfiguration und der Einrichtung der Validierung. Solche Agenten könnten beim Definieren von CPM-Komponenten Anleitung bieten, unvollständige oder inkonsistente Konfigurationen automatisch erkennen und geeignete Standardwerte oder Templates basierend auf dem Validierungsziel und den Systemcharakteristika vorschlagen. Diese Unterstützung könnte den Nutzerworkflow weiter beschleunigen und die Robustheit konfigurierter Experimente verbessern.

Abschließend zeigt Deliverable D5.1, dass eine automatisierte und skalierbare cyber-physische Validierung digitaler Energiesysteme durch einen einheitlichen Modellierungsansatz und containerbasierte Ausführung erreichbar ist. Die vorgestellten Konzepte und Werkzeuge bilden eine konsistente Grundlage für nachfolgende DES-CERT-Aktivitäten und zukünftige Projekte, die sich mit der Validierung und Präzertifizierung digitaler Energiesystemlösungen befassen.

5 Literaturverzeichnis

[1] D Koolen, M De Felice, and S Busch. Flexibility requirements and the role of storage in future european power systems. Technical Report KJ-NA-31-239-EN-N (online), Publications Office of the European Union, Luxembourg (Luxembourg), 2023.

[2] M Antretter, M Klobasa, M Kühnbach, M Singh, K Knorr, J Schütt, J Boer, O Rolser, D Hernandez Diaz, F Fitzschen, A Garcerán, R Reina, S Stemmer, J Steinbach, and E Popovski. Digitalisation of energy flexibility. Technical report, European Commission and Directorate-General for Energy. Publications Office of the European Union, 2022.

[3] Filip Pröstl Andrén, Catalin Gavriluta, Denis Vettoretti, and Marco Mittelsdorf. Automated Cyber-Physical Validation Framework for Rapid Development of Smart Grid Controls. In *IECON 2024 -50th Annual Conference of the IEEE Industrial Electronics Society*, pages 1–6, November 2024.

[4] Georg Lauss and Kai Strunz. Accurate and stable hardware-in-the-loop (hil) real-time simulation of integrated power electronics and power systems. *IEEE Transactions on Power Electronics*, 36(9):10920–10932, 2021.

[5] Mehmet Hazar Cintuglu, Osama A. Mohammed, Kemal Akkaya, and A. Selcuk Uluagac. A survey on smart grid cyber-physical system testbeds. *IEEE Communications Surveys & Tutorials*, 19(1):446–464, 2017.

[6] Rajaa Vikhram Yohanandhan, Rajvikram Madurai Elavarasan, Rishi Pugazhendhi, Manoharan Premkumar, Lucian Mihet-Popa, Junbo Zhao, and Vladimir Terzija. A specialized review on outlook of future cyber-physical power system (cpps) testbeds for securing electric power grid. *International Journal of Electrical Power & Energy Systems*, 136:107720, 2022.

[7] Christian Andersson, Johan Åkesson, and Claus Führer. PyFMI: A Python Package for Simulation of Coupled Dynamic Models with the Functional Mock-up Interface.

[8] Trevor D. Hardy, Bryan Palmintier, Philip L. Top, Dheepak Krishnamurthy, and Jason C. Fuller. HELICS: A Co-Simulation Framework for Scalable Multi-Domain Modeling and Analysis. *IEEE Access*, 12:24325–24347, 2024.

[9] Cornelius Steinbrink, Marita Blank-Babazadeh, André El-Ama, Stefanie Holly, Bengt Lüers, Marvin Nebel-Wenner, Rebeca P. Ramírez Acosta, Thomas Raub, Jan Sören Schwarz, Sanja Stark, Astrid Nieße, and Sebastian Lehnhoff. CPES Testing with mosaik: Co-Simulation Planning, Execution and Analysis. *Applied Sciences*, 9(5):923, January 2019.

[10] Yuan Liu, Renke Huang, Wei Du, Ankit Singhal, and Zhenyu Huang. Highly-Scalable Transmission and Distribution Dynamic Co-Simulation With 10,000+ Grid-Following and Grid-Forming Inverters. *IEEE Transactions on Power Delivery*, 39(1):578–590, February 2024.

[11] Oceane Bel, Joonseok Kim, William J. Hofer, Manisha Maharjan, Burhan Hyder, Sumit Purohit, and Shwetha Niddodi. Co-Simulation Framework for Network Attack Generation and Monitoring. *IEEE Access*, 12:142227–142240, 2024.

[12] Immanuel Hacker, Johannes Lenzen, Florian Schmidtke, Dennis van der Velde, and Andreas Ul-big. A co-simulation environment to evaluate cyber resilience in active distribution grids utilising behind-the-meter assets. *Electric Power Systems Research*, 230:110254, May 2024.

[13] Catalin Gavriluta, Cedric Boudinet, Friederich Kupzog, Antonio Gomez-Exposito, and Raphael Caire. Cyber-physical framework for emulating distributed control systems in smart grids. *International Journal of Electrical Power & Energy Systems*, 114:105375, 2020.

[14] Tung-Lam Nguyen, Yu Wang, Quoc-Tuan Tran, Raphael Caire, Yan Xu, and Catalin Gavriluta. A distributed hierarchical control framework in islanded microgrids and its agent-based design for cyber–physical implementations. *IEEE Transactions on Industrial Electronics*, 68(10), 2021.

[15] Hao Tu, Yuhua Du, Hui Yu, Abhishek Dubey, Srdjan Lukic, and Gabor Karsai. Resilient information architecture platform for the smart grid: A novel open-source platform for microgrid control. *IEEE Transactions on Industrial Electronics*, 67(11):9393–9404, 2020.

[16] Tung Lam Nguyen, Abdulmueen Alrashide, and O. Mohammed. A Cyber-Physical Platform to Assess Power System Operation With Communication Network and Heterogeneous Controllers. *IEEE Transactions on Industry Applications*, 60(3):4776–4785, May 2024.

[17] Linh Vu, Tung-Lam Nguyen, Mahmoud S. Abdelrahman, Tuyen Vu, and Osama A. Mohammed. A cyber-hil for investigating control systems in ship cyber physical systems under communication issues and cyber attacks. *IEEE Transactions on Industry Applications*, pages 1–11, 2023.

[18] Van Hoa Nguyen, Houda Abil, David Ramsey Herrera, Rémi Thomas, and Thierry Coste. Containerized IEC 61850-based protection in smart grid – Lab demonstration and performance assessment for large scale deployment. *Electric Power Systems Research*, 249:111999, December 2025.

[19] Mazheruddin Syed, Tran The Hoang, Alkistis C. Kontou, Alexandros G. Paspatis, Graeme M. Burt, Quoc Tuan Tran, Efren Guillo-Sansano, Steffen Vogel, Ha Thi Nguyen, and Nikos D. Hatziargyriou. Applicability of geographically distributed simulations. *IEEE Transactions on Power Systems*, 38(4):3107–3122, 2023.

[20] Ivan Cilic, Petar Krivic, Ivana Zarko, and Mario Kusek. Performance evaluation of container orchestration tools in edge computing environments. *Sensors*, 23:4008, 04 2023.

[21] Oleg Ivanchenko, Eugene Brezhnev, Ihor Kliushnikov, and Boris Moroz. Cloud simulation and virtualization for testing of critical energy infrastructure components. *International Journal of Computing*, pages 119–128, 03 2021.

[22] Taehun Kim, Hojung Kim, SeungKeun Cho, YongSeong Kim, ByungKwen Song, and Jincheol Kim. Real-time kubernetes-based front-end processor for smart grid. *Electronics*, 14(12), 2025.

[23] Vedran Dakić, Mario Kovač, and Jurica Slovinac. Evolving high-performance computing data centers with kubernetes, performance analysis, and dynamic workload placement based on machine learning scheduling. *Electronics*, 13(13), 2024.

[24] Syed M. Raza, Jaeyeop Jeong, Moonseong Kim, Byungseok Kang, and Hyunseung Choo. Empirical performance and energy consumption evaluation of container solutions on resource constrained iot gateways. *Sensors*, 21(4), 2021.

[25] L. Thurner, A. Scheidler, F. Schafer, J. H. Menke, J. Dollichon, F. Meier, S. Meinecke, and

M. Braun. pandapower - an open source python tool for convenient modeling, analysis and optimization of electric power systems. *IEEE Transactions on Power Systems*, 2018.

[26] A. Fernández-Guillamón, A. Molina-García, K. Das, and M. Altin. Comparison of different tools for power flow analysis with high wind power integration. In *2019 International Conference on Clean Electrical Power (ICCEP)*, pages 82–86, 2019.

[27] Hantao Cui and Fangxing Li. Andes: A python-based cyber-physical power system simulation tool. In *2018 North American Power Symposium (NAPS)*, pages 1–6, 2018.

[28] Trevor D. Hardy, Bryan Palmintier, Philip L. Top, Dheepak Krishnamurthy, and Jason C. Fuller. Helics: A co-simulation framework for scalable multi-domain modeling and analysis. *IEEE Access*, 12:24325–24347, 2024.

[29] D. P. Chassin, K. Schneider, and C. Gerkensmeyer. Gridlab-d: An open-source power systems modeling and simulation environment. In *2008 IEEE/PES Transmission and Distribution Conference and Exposition*, pages 1–5, 2008.

[30] Adil Khurram, Mahraz Amini, Luis A. Duffaut Espinosa, Paul D. H. Hines, and Mads R. Almas-salkhi. Real-time grid and der co-simulation platform for testing large-scale der coordination schemes. *IEEE Transactions on Smart Grid*, 13(6):4367–4378, 2022.

[31] Energy management system application program interface (ems-api) — part 600-1: Common grid model exchange standard (cgmes) — structure and rules, 2021.

[32] Filip Pröstl Andrén, Catalin Gavriluta, Denis Vettoretti, and Marco Mittelsdorf. Automated cyber-physical validation framework for rapid development of smart grid controls. In *IECON 2024 - 50th Annual Conference of the IEEE Industrial Electronics Society*, pages 1–6, 2024.

[33] Hantao Cui and Fangxing Li. Andes: A python-based cyber-physical power system simulation tool. In *2018 North American Power Symposium (NAPS)*, pages 1–6, 2018.

[34] S. Meinecke, D. Sarajlic, Simon Ruben Drauz-Mauel, A. Klettke, L. Lauven, C. Rehtanz, A. Moser, and Martin Braun. Simbench - a benchmark dataset of electric power systems to compare innovative solutions based on power flow analysis, 2020.

[35] L. Thurner, A. Scheidler, F. Schafer, J. H. Menke, J. Dollichon, F. Meier, S. Meinecke, and M. Braun. pandapower - an open source python tool for convenient modeling, analysis and optimization of electric power systems. *IEEE Transactions on Power Systems*, 2018.

[36] Denis Vettoretti, Clemens Korner, Catalin Gavriluta, Filip Pröstl Andrén, and Barbara Herndler. Towards intelligent simulation platforms for scalable validation of digital energy systems, September 2025.

6 Kontaktdaten

Denis Vettoretti
AIT Austrian Institute of Technology GmbH
Giefinggasse 2, 1210 Wien, 050 5500,
denis.vettoretti@ait.ac.at,
https://www.ait.ac.at/