RWTH AACHEN UNIVERSITY

RWTH THEMEN

Forschungsmagazin **1/2024**

Research Software Engineering

Inhalt

Foto: Peter Winandy

Markus Diesmann, Julia Kowalski, Bernhard Rumpe

Research Software an der RWTH Aachen

In the last ten years, research software engineering has emerged as a crucial capability to an innovative scientific community. Efficient development of high-quality software beyond mere prototypes is often essential for transforming ideas into impactful results, translating hypotheses into knowledge, and implementing this knowledge in applications for the benefit of society and industry. Successful and effective research, development, and teaching at a university like RWTH now rely heavily on software development that is designed for sustainability, efficiency, and technical compatibility for integrated research networks. These activities also require the capabilities to use high-quality software in a creative, solution-oriented, and reproducible manner.
Developing high-quality research software is essential for the success and sustainability of science and will increasingly impact teaching and learning as well.

Software nimmt in fast allen Bereichen der Natur- und Ingenieurwissenschaften eine Schlüsselrolle ein. Mittlerweile existieren grundsätzlich verschiedene Software-Arten, beispielsweise Algorithmen für die Simulation thermofluid-mechanischer Multiskalenprozesse, Methoden zu Konstruktion und Optimierung komplexer technischer Systeme, Skripte zur automatisierten Prozessierung extrem umfangreicher Datensätze aus Messkampagnen, Algorithmen zur Steuerung und Regelung komplexer Anlagen und Roboter, Wissenssysteme zur schnellen Identifikation von Lösungen bei selten auftretenden Ereignissen oder Störungen, Digitale Zwillinge zur integrierten Verwaltung und Kontrolle physikalischer, chemischer oder biologischer Forschungsexperimente, assistierende KI-basierte Systeme und nicht zuletzt operative Grundfunktionalitäten in Betriebssystemen, Datenspeichern und Kommunikationsinfrastrukturen. Die Verfügbarkeit und eine weit über den Prototypen hinausgehende Qualität der Software sind oft Voraussetzung dafür, aus Ideen Wirkung zu entfalten oder Hypothesen in Erkenntnisgewinn weiterzuentwickeln, diesen in Anwendungen umzusetzen

und damit der Gesellschaft und der Industrie verfügbar zu machen. Ohne Software würde es keine Navigation geben, es würde kein modernes Auto fahren oder gebaut werden können, keine Wettervorhersage oder Frühwarnung vor Naturgefahren existieren, keine moderne medizinische Diagnostik oder Energieversorgung möglich sein – unser Lebensstandard würde in dieser Form schlichtweg nicht existieren.

Erfolgreiche und wirkungsvolle Forschungs-, Entwicklungs- und Lehrtätigkeit an einer Universität wie der RWTH Aachen ist daher ohne eine auf Nachhaltigkeit, Effizienz und technische Kompatibilität für integrierte Forschungsverbünde ausgelegte Softwareentwicklung, und ohne die Kompetenz zur kreativen, lösungsorientierten und reproduzierbaren Nutzung hochwertiger Software, nicht mehr möglich. Offenheit und Zugänglichkeit von Software im Sinne des FAIR Paradigmas (Findable, Accessible, Interoperable, Reusable) sind weitere Faktoren, um einen software-integrierten Innovationsprozess und wissenschaftlichen Erkenntnisgewinn zu gewährleisten.

Weil Simulation heute als dritte Säule des Erkenntnisgewinns neben Experiment und Theorie etabliert ist und Experimente immer datenlastiger und rechenintensiver werden, macht die Bedeutung von Software und ihrer Entwicklung in den Natur-, Sozial- und Ingenieurwissenschaften derzeit eine ähnliche Entwicklung durch, wie die Softwareentwicklung im industriellen Kontext in den vergangenen 20 Jahren bereits durchlebt hat: Der Entwicklungsbereich eines Automobilherstellers wird heute von Software dominiert. Der Vorstand der Volkswagen AG hat beispielsweise Volkswagen als Software-Konzern bezeichnet. Software kann aufgrund ihrer Komplexität nicht mehr als Beiwerk behandelt werden, ihre Entwicklung muss organisatorisch durchdacht sein sowie technisch unterstützt und behandelt werden.

Software Engineering wurde bereits 1968 als wissenschaftliches Forschungsfeld gestartet und in den nachfolgenden Jahren in allen Informatik-Ausbildungen etabliert, um Komplexität, Effizienz, Nachhaltigkeit sowie die Qualitätsanforderungen adäquat zu adressieren. Der mittlerweile gut konsolidierte und in regelmäßiger Weiterentwicklung befindliche „Software Engineering Body of Knowledge“ (SWEBOK) unterteilt seine Aktivitäten in 15 Kapitel, die sich mit Anforderungen, Entwurf, Konstruktion, Testen, Wartung, Konfigurations- und Engineering Management, Entwicklungsprozessen, Modelle und Methoden, Qualität, Professional Practices, Entwicklungsökonomie sowie Computing, mathematischen und Engineering Grundlagen des Software Engineering beschäftigen. Das klassische Programmieren gehört zur Konstruktion und betrifft daher nur einen kleinen Teil des Wissens. Viele Innovationen, wie etwa Wikis, Versionskontrolle, Variantenmanagement, Ticketsysteme oder agile Entwicklungsmethoden verdanken ihren Ursprung dem Software Engineering. Um die spezifischen Herausforderungen zur Entwicklung von Forschungssoftware zu adressieren, hat sich zunächst in den angelsächsischen Ländern als Spezialisierung des Software Engineering das „Research Software Engineering“ (RSE) herausgebildet. RSE bezeichnet ein Fachgebiet, das die Prinzipien des Software Engineering mit den Bedürfnissen und Zielen der natur- und ingenieurwissenschaftlichen Forschung vereint. Ähnlich wie andere

Spezialisierungen des Software Engineering (etwa Automotive Software Engineering) existiert RSE nicht isoliert, sondern kombiniert seine Methoden und Best Practices, um die Domänenforschung durch hochwertige Software und Softwarekompetenz effektiver und wirkungsvoller werden zu lassen.
RSE hat sich in den letzten Jahren zu einem der wichtigsten – jedoch nicht sehr lauten – Zukunftsthemen in der wissenschaftlichen Gemeinschaft entwickelt. RSE reicht von der Entwicklung von „Fire-and-Forget"-Skripten, die von Einzelpersonen geschrieben werden und für den einmaligen Gebrauch gedacht sind, bis hin zu koordinierten, institutionsübergreifenden Softwareprojekten, die viele Forscher zusammenbringen, um an einem gemeinsamen Ziel zu arbeiten. Zunehmend mutiert dabei Software, die zunächst als Forschungsgegenstand entwickelt wurde, zu wiederverwendbarer und damit qualitativ hochverfügbarer Software als Infrastruktur. Software bekommt so ein immenses Potenzial, Innovationen und wissenschaftliche Entdeckungen schneller voranzutreiben. Allerdings ist aus dem Software Engineering bekannt, dass Wiederverwendbarkeit vorbereitet und organisiert werden muss – und auch welche Maßnahmen dafür nötig sind.
Ein gemeinsames Merkmal von RSE ist die Fokussierung auf einen bestimmten Anwendungsbereich und die damit verbundenen Auswirkungen auf alle Phasen des Software-Lebenszyklus. Beispielsweise ist der Prozess der Anforderungserhebung stark mit dem wissenschaftlichen Forschungsprozess verzahnt und die Korrektheit der Software relativ zu den veröffentlichten wissenschaftlichen Papieren beziehungsweise den darin enthaltenen Modellen zu sichern. Deshalb passen klassische Paradigmen des Software Engineering aus der Industrie oft nicht zur gelebten Praxis einer auf Forschung und Innovationsleistung ausgelegten Universität. Softwareentwicklung muss sich hier nahtlos in die jeweiligen Forschungsprojekte eingliedern, welche oft von internationalen, interdisziplinären Kooperationen, hoher personeller Fluktuation und leider vor allem fragmentierten Finanzierungskonstrukten geprägt ist. Gleichzeitig möchte man in der Forschung, also auch in der damit einhergehenden Softwareentwicklung, agil auf aktuelle Entwicklungen reagieren. Aufgrund dieser Anforderungen und der zunehmenden Komplexität der Algorithmen, der Integration von Datenmengen, KI-Techniken und des Wunsches nach energetischer Sparsamkeit rechenintensiver Prozesse ist der Stand der Praxis im klassischen Software Engineering für die Bedürfnisse der forschungsorientierten RSE noch deutlich ausbaubar. Wie in anderen Bereichen auch, passen die bestehenden Software-Engineering-Methoden, -Werkzeuge und -Best Practices nicht genau und müssen angepasst werden, um bessere Software und damit bessere Forschung zu ermöglichen.
Die RWTH ist dabei, sich hier noch besser zu positionieren. So wurde in der ersten großen Konferenz am 17. Mai 2023 eine RSE-Strategie diskutiert. Definiert wurde eine koordinierte Strategie für Nachhaltigkeit und Exzellenz in RSE, die aus einer Reihe von inhaltlichen, organisatorischen, technischen und auch Ausbildungs-Maßnahmen besteht und sowohl in Exzellenzclustern als auch der Gesamtstrategie der RWTH Aachen ein wichtiger Eckpfeiler sein wird. Zentral ist auch die stattgefundene Sensibilisierung für das Thema über die gesamte Universität hinweg. Ein weiterer Eckpfeiler ist die Etablierung einer Sammlung an Best Practices der Softwareentwicklung an der RWTH, indem RSE in die Lehre integriert wird. Darüber hinaus sind eine institutionelle RSE-Methodik, Coaching-On-The-Job und Tooling Support durch Experten, also Software Engineers, Architekten und Projekt Engineers im Aufbau. Komplex bleibt die Frage, wie interne und externe Finanzierungsmöglichkeiten nachhaltig geschaffen werden können. Hier ist neben der Universität insbesondere das Forschungssystem gefragt, um langlebige Software-Infrastruktur valide und gesund zu erhalten.
Es ist mittlerweile klar, dass RSE-assoziierte Herausforderungen die gesamte Wissenschaft im globalen Kontext betreffen. Deshalb ist die RWTH dabei, insbesondere institutionsübergreifend mit der Jülich Aachen Research Alliance (JARA), der Euregio und den vielen RWTH-nahen Forschungsverbünden eine gemeinsame RSE-Strategie zu erarbeiten und weiterzuentwickeln.

Diese Ausgabe der RWTH THEMEN zeigt eine Auswahl an Softwareprojekten, welche RSE-Paradigmen umsetzen. Dies beinhaltet Beiträge zur Softwareentwicklung selbst, in Form von Artikeln zur Robustheit numerischer Modelle gegenüber Parameteränderungen, der Methodik zur Nutzung expliziter, domänenspezifischer Modellierungssprachen, generische Optimierungswerkzeuge, Software zur Datenintegration unter Wahrung des Datenschutzes und der Nutzung fachübergreifende Softwareumgebungen für spezifische Modellklassen. Darauf aufbauend werden Anwendungen für spezifische Fachdomänen, wie der Neurowissenschaft, der Materialwissenschaft, der Geologie bis hin zur Luftfahrt beschrieben. Umrahmt werden diese

Beiträge von dem Artikel „Research Software Engineering – Vergangenheit, Gegenwart und Zukunft“, der die mittlerweile zehnjährige Geschichte und Entstehung des RSE beschreibt, siehe Seite 8, und dem Beitrag „Wissenschaftstheoretische Reflexionen zu Research Software Engineering – Konzeptuelle Analyse von Forschungssoftware“, der wissenschaftstheoretische Reflektionen vornimmt, siehe Seite 80.
Die Beiträge zeigen, welche Arten von Software an der RWTH entwickelt, gewartet, aktualisiert und für die Öffentlichkeit beziehungsweise die Fachcommunity frei verwendbar zur Verfügung gestellt wird und oft zur gemeinsamen Weiterentwicklung und Integration neuer Forschungserkenntnisse und Ideen geführt hat. Denn Softwareentwicklung in der Wissenschaft kann sich über Jahrzehnte erstrecken. Sie wandelt sich im Erfolgsfall typischerweise nach einer initialen, projekt getriebenen Phase von einer „Fire-and-Forget“-Software zu einer grundsätzlichen Infrastruktur, die nicht mehr von einer einzelnen Gruppe betreut sondern in Verbünden von universitären Professuren und Forschenden an nationalen Forschungseinrichtungen weiter genutzt und entwickelt wird. Ist eine Software erfolgreich, mutiert sie zur Infrastruktur, was einerseits einen Umbau in Bezug auf Anwendbarkeit und Robustheit, methodische Änderungen zur Weiterentwicklung im Konsortium erfordert und andererseits auch eine große Verantwortung induziert.
Darüber hinaus zeigt diese Ausgabe der RWTH THEMEN, dass Software nicht nur zum Treiber der Wissenschaft geworden ist. Vielmehr sind sich die Forschenden bewusst, dass Software den wissenschaftlichen Prozess selbst verändert und als dritte Säule des Erkenntnisgewinns neue Wege induziert, wie über ein wissenschaftliches Problem nachgedacht wird.

Im Sinne des institutionsübergreifenden Austausches sind wir froh, im JARA Center for Simulation and Data Scienc (CSD) Erfahrungen über die Fachdisziplinen hinweg von der Großforschung bis zur universitären Forschung austauschen zu können. So entsteht ein detailliertes und vollständiges Bild, was RSE ausmacht, wie es an einer Exzellenzuniversität gelebt, gelehrt und an zukünftige Generationen weitergegeben werden kann. Der Erfolg bei der Entwicklung von Forschungssoftware ist kritisch für den Erfolg und die Nachhaltigkeit der Wissenschaft als Ganzes und hat in Zukunft auch noch deutlich mehr Einfluss auf die Lehre. Wir sehen daher ein Zukunft, in welcher RSE nicht länger als neuartige Entwicklung angesehen wird, sondern RWTH-weit als gelebte Praxis in allen Disziplinen selbstverständlich geworden ist.

Autoren

Univ.-Prof. Dr.rer.nat. Markus Diesmann leitet das Lehr- und Forschungsgebiet Computational Neuroscience und ist Leiter des Instituts für Computational and Systems Neuroscience (IAS-6) am Forschungszentrum Jülich.
Univ.-Prof. Dr.sc.habil. Julia Kowalski ist Inhaberin des Lehrstuhls für Methoden der Modellbasierten Entwicklung in den Computergestützten Ingenieurwissenschaften.
Univ.-Prof. Dr.rer.nat. Bernhard Rumpe ist Inhaber des Lehrstuhls für Software Engineering (Informatik 3).

Robert Speck, Claire Wyatt

Research Software Engineering

Vergangenheit, Gegenwart und Zukunft

Did you start off as a researcher and now spend time developing software to progress your research? Or maybe you started off from a more conventional software-development background and are drawn to research by the challenge of using software to further research?
A growing number of people in academia combine expertise in programming with an intricate understanding of research, all while being a researcher. Although this combination of skills is extremely valuable, these contributions lack a formal place in the academic system. There is no easy way to recognise their contribution, to reward them, or to represent their views.
Without a job title, it was difficult for people to rally around a cause and there was no easy way to recognize their contribution, to reward them, or to represent their views. The realisation emerged that the lack of awareness, recognition and reward of the skills and contribution or a recognised job title for research developers was having a knock on effect making many activities difficult. Those in the role spoke of how it was difficult to recruit and difficult for developers to find a job.
So in 2012 at a workshop in the UK, the term Research Software Engineer was first coined. This gave a focus for the next ten years of support and has enabled developers to find their tribe online and in-person. The RSE movement around the world is now working to raise awareness of the role, bring RSEs together and advocate for more appropriate career recognition and promotion. This article will cover the history of the RSE movement, the achievements to date around the world, the goals and aims of the RSE movement looking to the future.

Die Entwicklung von Forschungssoftware ist eine wichtige und in fast allen Disziplinen genutzte Fähigkeit. Zwei Umfragen aus den Jahren 2014 und 2017 belegen dies: In Großbritannien[1] bestätigten 92 Prozent der Forschenden, dass sie Forschungssoftware verwenden, und 69 Prozent gaben an, dass diese für ihre Forschung von grundlegender Bedeutung ist. Die Umfrage aus den USA[2] ergab, dass 95 Prozent der Wissenschaftlerinnen und Wissenschaftler Forschungssoftware nutzen, und für 66 Prozent der Befragten war diese für ihre Forschung von grundlegender Bedeutung. Heute wären die Zahlen sicherlich höher.
Software ist für die Forschung also von entscheidender Bedeutung. Ihre Entwicklung sollte daher als Pfeiler der Wissenschaft und eigenständige Forschungsleistung betrachtet werden. Die Erstellung guter Codes erfordert Ausbildung, Erfahrung, Fachwissen sowie spezielle Kompetenzen. Diese müssten bereits im Rahmen des Studiums unabhängig vom Fach allen Studierenden vermittelt werden.
Viele der übergeordneten FAIR-Datenprinzipen können direkt auf Forschungssoftware angewendet werden, wenn man Software und Daten als vergleichbare digitale Forschungsobjekte behandelt. Dies bedeutet, dass die Auffindbarkeit, Zugänglichkeit, Interoperabilität und Wiederverwendbarkeit von Forschungssoftware gewährleistet sein muss.
In einer Welt, in der ein Großteil der Forschung durch Software vorangetrieben wird, ist die Anerkennung von Research Software Engineering von zentraler Bedeutung – eine Einsicht, die sich in dem Motto „Better Software, Better Research“ manifestiert. Dieses wurde vom Software Sustainability Institute in Großbritannien entwickelt und weltweit übernommen. Eine gängige Definition von Research Software Engineering ist „die Anwendung von Software-Engineering-Verfahren in Forschungsanwendungen“. Forschungssoftware reproduzierbar, nachhaltig und wiederverwendbar zu machen erfordert ein umfassen-

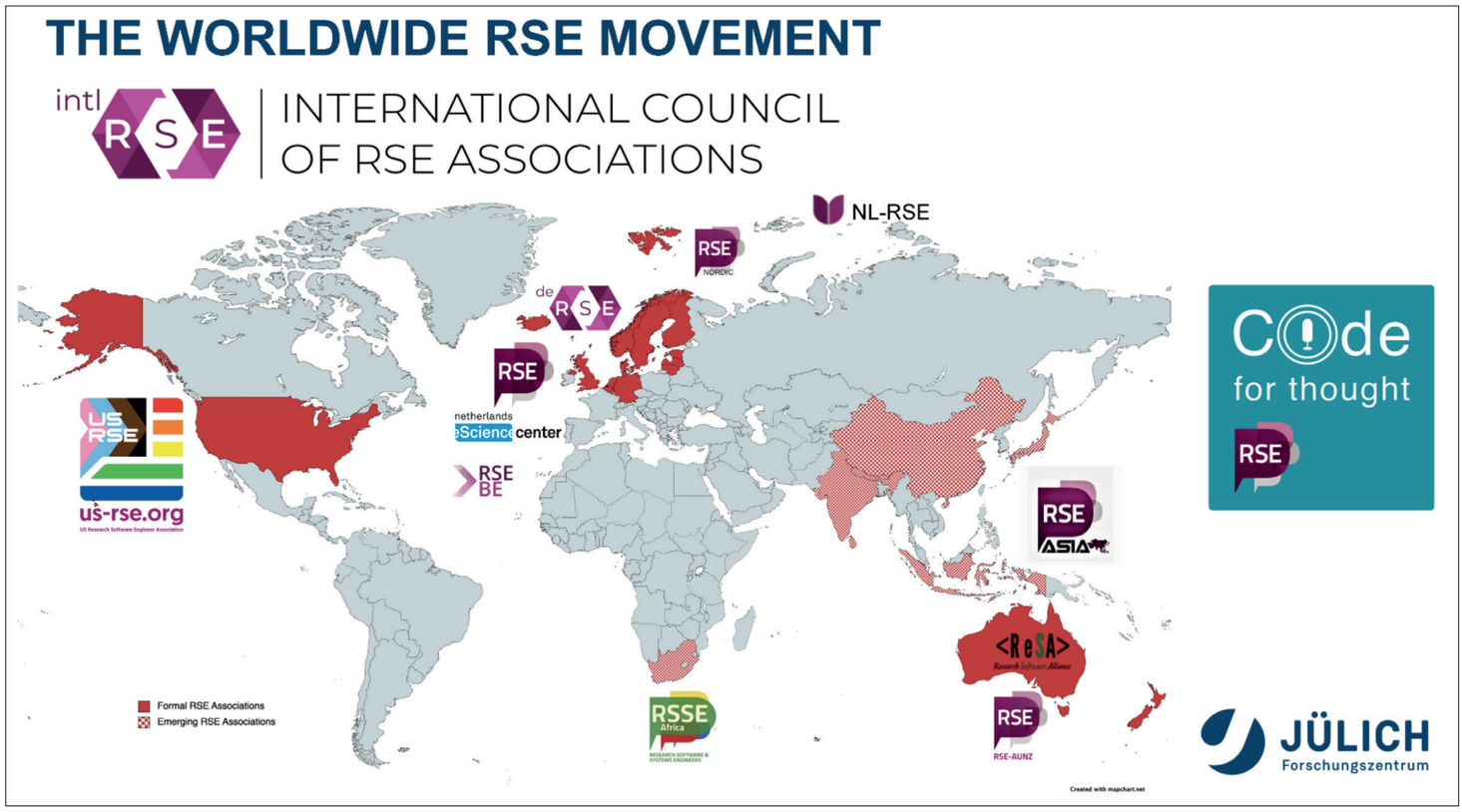


Bild 1: Karte der der weltweiten RSE-Bewegungen, sowie die auf RSE ausgerichtete Podcast-Reihe „Code4Thought“
Quelle: Ian Cosden (Princeton University) und Claire Wyatt (Forschungszentrum Jülich)

des Verständnis von Forschungsmethoden sowie Erfahrung in der Softwareentwicklung. Als Forschungssoftware kann „jede Software betrachtet werden, die zur Erzeugung, Verarbeitung und Analyse von Ergebnissen verwendet wird, die für eine Veröffentlichung gedacht sind“[3]. Diese Definition mag nicht unumstritten sein, da sie die Frage aufwirft, ob Forschungssoftware nur dazu genutzt werden soll, um Ergebnisse zu erzielen, die veröffentlicht werden. Argumente gegen die Definition wären beispielsweise, dass manchmal lediglich ein Konzeptnachweis erbracht oder die Grundlage für einen Förderantrag erarbeitet wird. Forschungssoftware kann alles Mögliche sein, von ein paar Codezeilen bis hin zu einem kompletten Softwarepaket.

Historische Entwicklung

Im März 2012 traf sich eine kleine Gruppe zu einem „Collaborations Workshop“ am Software Sustainability Institute des Queen’s College Oxford. Dort wurde der Begriff „Research Software Engineer“ als Berufsbezeichnung und „Research Software Engineering“ für die Praxis der in diesem Bereich Tätigen geprägt. Ziel war, über den Mangel an Karrieremöglichkeiten für Softwareentwicklerinnen und -entwickler in der Wissenschaft zu diskutieren, dem Berufsbild mehr Anerkennung zu verschaffen sowie den Mangel an Karrieremöglichkeiten und Arbeitsplatzsicherheit zu. Am ersten RSE-Workshop an der Universität Oxford 2013 nahmen gerade einmal 56 Personen teil. Nach diesem Workshop wurde die UK RSE Association gegründet, wobei das Software Sustainability Institute (SSI) die noch junge Gemeinschaft vertritt. Im Jahr 2016 veranstaltete das SSI die erste RSE-Konferenz und 2017 das erste internationale Treffen der RSE-Führungskräfte in Großbritannien. Dies inspirierte die Gründung der nationalen RSE-Verbände in den USA und in Deutschland, denen bald weitere Gruppen in Skandinavien, Australien und Neuseeland folgten.

Im Jahr 2019 entstand die gemeinnützige Gesellschaft „The Society of Research Software Engineering“, um dem Berufsstand und der Gemeinschaft eine Organisation als Impulsgeber für die notwendigen Veränderungen zur Seite zu stellen. Aus dieser Kampagne hat sich inzwischen eine internationale RSE-Gemeinschaft entwickelt; in Deutschland formierte sich 2018 die nationale Gruppe „de-RSE e.V.“. Ihre Ziele sind vielfältig, die Mitgliedschaft steht allen Interessierten offen. de-RSE veranstaltet jährlich eine Konferenz und bietet im Online-Community-Bereich Möglichkeiten zur Vernetzung und Austausch. Eine Vision besteht darin, das Forschungsumfeld weiter zu sensibilisieren, so dass die zentrale Rolle von Software in Forschung und Wissenschaft noch höhere Anerkennung erfährt.

https://www.fz-juelich.de/en/rse/about-rse

Literatur

[1] Hettrick, S.J., et al, UK Research Software Survey 2014, DOI:10.5281/zenodo.1183562
[2] Nangia, U., Katz, D. S., Surveying the US National Postdoctoral Association Regarding Software Use and Training in Research. (WSSSPE5.1), 2017
[3] UK Research Software Survey 2014, Hettrick et al, https://zenodo.org/record/14809

Autoren

Dr. Robert Speck ist einer der stellvetretenden Institutsleiter des Jülich Supercomputing Centre und leitet dort die Abteilung „Mathematik und Ausbildung“.
Claire Wyatt ist RSE Community Managerin am Forschungszentrum Jülich.

Dominik Bongartz, Jannik Lüthje, Alexander Mitsos, Jaromił Najman, Artur M. Schweidtmann, Clara Witte

Optimierung und maschinelles Lernen

Entwicklung und Anwendungen der deterministischen globalen Software MAiNGO und der Toolbox MeLOn

Mathematical optimization algorithms are used in many areas of science and engineering. In the field of energy and process engineering, for example, chemical plants are optimized to make them more efficient. However, the optimization of complex processes is challenging. In particular, the optimization problems that arise are often nonconvex, motivating the use of deterministic global optimization methods. Unfortunately, these methods are very computationally expensive. A remedy can be provided by hybrid models, combining mechanistic equations (in particular physical laws) with data-driven model components. For this purpose, the MAiNGO software and the MeLOn toolbox were developed at the Institute of Process Systems Engineering at RWTH Aachen University.

Mathematische Optimierungsalgorithmen werden in vielen Bereichen von Wissenschaft und Technik angewendet. Aber auch im alltäglichen Leben begegnet uns die Optimierung häufig, zum Beispiel bei der Benutzung eines Navigationssystems. Dabei sucht dieses die schnellste oder auch die umweltfreundlichste Route von Punkt A nach B. Es wird die aktuelle Verkehrssituation berücksichtigt, und häufig können die Nutzenden Vorlieben oder Stopps mit angeben. Unter Einbeziehung aller Vorgaben schlägt das Navigationssystem mithilfe von Optimierung eine Route vor. Ähnliche Aufgabenstellungen ergeben sich auch in der Energie- und Verfahrenstechnik. Extrem wichtig ist dort beispielsweise die Optimierung von Prozessen für die Herstellung von Chemikalien oder für die Energiewandlung, mit dem Ziel diese effizienter oder kostengünstiger zu gestalten.

Globale Optimierung eines Fließbildes

In der Verfahrenstechnik werden häufig sogenannte Prozessfließbilder, eine abstrakte Darstellung einer verfahrenstechnischen Anlage, genutzt und optimiert. Das Fließbild enthält dabei verfahrenstechnische Grundoperationen, wie zum Beispiel Reaktoren oder Destillationskolonnen. Diese Grundoperationen werden durch Stoff- und Energieströme miteinander verbunden und in Beziehung gesetzt. Bei der Optimierung des Prozessfließbildes werden nun Werte für die Betriebs- oder Auslegungsvariablen ermittelt, die eine bestimmte Größe reduzieren soll, beispielsweise die Betriebskosten oder die Umweltbeeinträchtigung. Die zu minimierende Funktion wird auch Zielfunktion genannt. Ein allgemeines Optimierungsproblem besteht aus einer Zielfunktion und Nebenbedingungen, welche in Form von Gleichungen und

Bild 1: In der Bioraffinerie der Aachener Verfahrenstechnik wird an Aufschluss und Konversion von Biomasse zu Plattformchemikalien geforscht. Solche Prozesse sind Anwendungsbeispiele für die Optimierung mittels MAiNGO und MeLOn.
Foto: Peter Winandy

Ungleichungen gegeben werden. Physikalische Zusammenhänge zur Beschreibung von Grundoperationen und Flüsse können in dem Optimierungsproblem als Gleichungen in den Nebenbedingungen formuliert werden. Als Ungleichungen tauchen die Grenzen der Prozessgrößen, beispielsweise die Angabe eines maximalen Druckes, auf. Des Weiteren können Ungleichungen benutzt werden, um gewünschte Eigenschaften des Prozesses oder des Produktes, etwa maximale Anlagengröße oder Produktreinheit, zu garantieren. Die daraus formulierten Optimierungsprobleme sind häufig nichtkonvex. Da nichtkonvexe Optimierungsprobleme oft mehrere lokale Optima aufweisen, wird die Verwendung deterministischer globaler Optimierungsmethoden angestrebt. Im Gegensatz zu lokalen und stochastischen globalen Methoden garantieren deterministische globale Optimierungsverfahren ein globales Optimum in endlicher Zeit innerhalb einer gegebenen Toleranz. Trotz dieser Garantie gibt es für die praktische Anwendung solcher Optimierungsverfahren eine große Herausforderung: die Rechenzeit ist zwar endlich, kann aber je nach Problem dennoch sehr groß sein und möglicherweise Jahre betragen. Dadurch ist bisher der Einsatz dieser Verfahren für viele praxisrelevante Probleme unmöglich.
Ein Forschungsziel besteht darin, die Rechenzeit von deterministischen globalen Optimierungsmethoden weiter zu reduzieren. Am Lehrstuhl für Systemverfahrenstechnik wurde daher die Software McCormick-based Algorithm for mixed-integer Nonlinear Global Optimization, kurz MAiNGO, entwickelt. MAiNGO ist ein deterministischer globaler Optimierer und basiert auf dem Branch-and-Bound-Algorithmus. Somit kann die Software eingesetzt werden, um gemischt-ganzzahlige nichtlineare Programme, wie auch das oben genannte Beispiel, zu lösen. Eine algorithmische Besonderheit der Software ist die Durchführung der Berechnung in dem Raum der ursprünglichen Optimierungsvariablen. Das bedeutet, dass im Optimierungsprozess keine Hilfsvariablen eingeführt werden, wie es bei anderen kommerziellen Lösern gemacht wird. Dies kann bei einigen Optimierungsproblemen zu einer drastischen Reduzierung der Rechenzeit führen[1, 2].

Globale Optimierung mit eingebetteten künstlichen neuronalen Netzen

Trotz aktueller Fortschritte in der Entwicklung deterministischer globaler Optimierungsmethoden stellen viele Optimierungsprobleme aus der Verfahrenstechnik weiter eine Herausforderung dar. Beispiele sind Prozesse

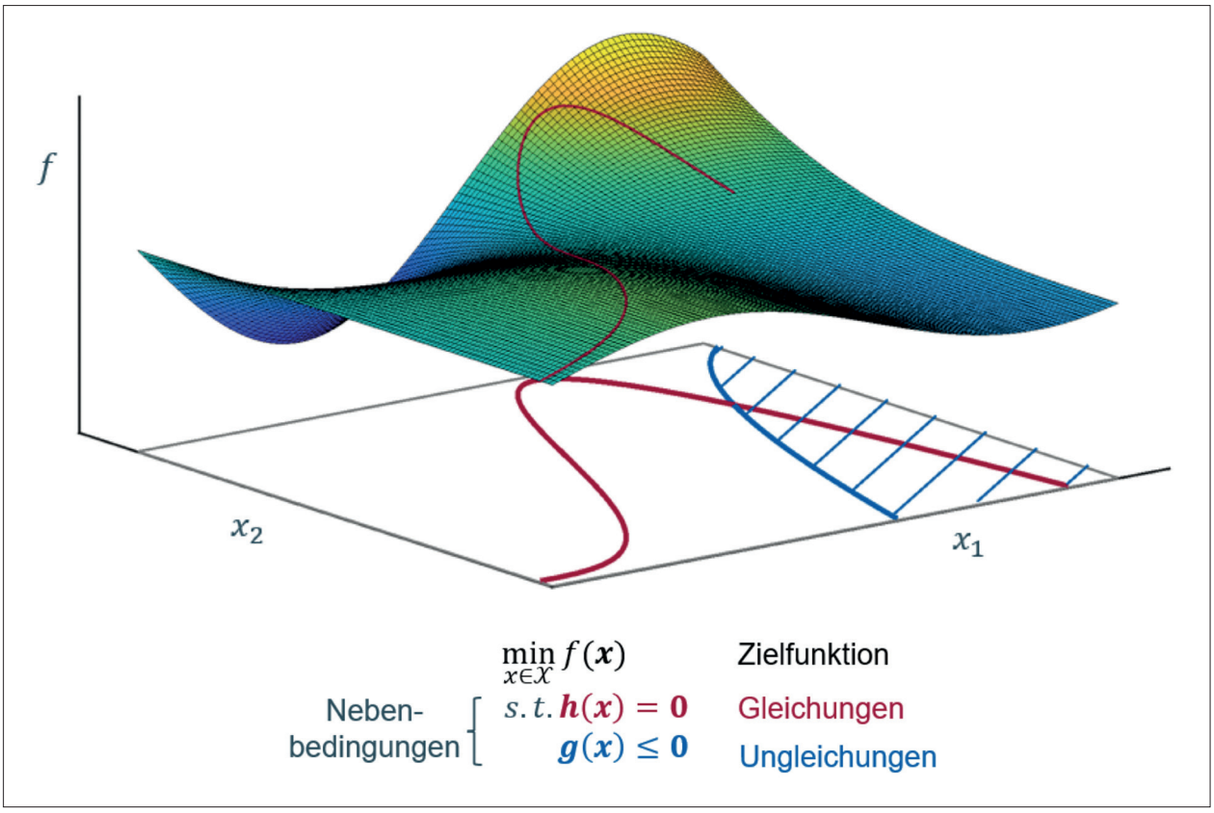


Bild 2: Grafische Darstellung eines nichtlinearen, nichtkonvexen Optimierungsproblems und allgemeine mathematische Schreibweise eines Optimierungsproblems.

wie ein Kraftwerk oder eine Entsalzungsanlage, welche detaillierte thermodynamische Zusammenhänge erfordern. Diese sind häufig noch zu komplex für die genannten Optimierungsmethoden. Abhilfe kann die Verwendung von hybriden Modellen schaffen. Diese kombinieren mechanistische Gleichungen, wie zum Beispiel physikalische Gesetze, mit datengetriebenen Modellbausteinen aus dem maschinellen Lernen. In Zusammenarbeit zwischen dem Lehrstuhl für Systemverfahrenstechnik und dem Lehrstuhl für Chemische Verfahrenstechnik wurde ein solcher hybrider Ansatz zur Entwicklung eines Membransystems eingesetzt. Dabei wurden experimentelle Daten mit etablierten mechanistischen Transportgleichungen kombiniert. Mithilfe der Daten wurde ein künstliches neuronales Netz trainiert. Das aufgestellte hybride Modell wurde anschließend genutzt, um das Membransystem zu optimieren und konnte gute Ergebnisse hinsichtlich Rechenzeit sowie des optimalen Betriebspunkt erzielen[3, 4, 5].
Um die Entwicklung und Optimierung solcher hybriden Modelle zu unterstützen, wurde am Lehrstuhl für Systemverfahrenstechnik die Toolbox MeLOn (Machine Learning models for Optimization) entwickelt. MeLOn stellt neben den künstlichen neuronalen Netzen verschiedene Modelle des maschinellen Lernens bereit, wie zum Beispiel Gauß-Prozesse oder Support Vector Machines. Diese Modelle können über eine Schnittstelle mit bekannten Werkzeugen wie Keras (Python) oder Matlab trainiert werden. Für die Lösung der Optimierungsprobleme, in die die Modelle eingebettet werden, steht eine Schnittstelle zum Optimierer MAiNGO zur Verfügung. Dabei hat sich für künstliche neuronale Netze und Gauß-Prozesse die Berechnung im Raum der ursprünglichen Optimierungsvariablen, ähnlich wie bei der oben beschriebenen Fließbildoptimierung, als erheblicher Vorteil gezeigt. Dies kann automatisch in MeLOn ausgewählt werden und ermöglicht eine effizientere Optimierung[6].

Entwicklung von MAiNGO und MeLOn

MAiNGO und MeLOn wurden am Lehrstuhl für Systemverfahrenstechnik entwickelt. Beide Programme sind Open-Source-Software, lizenziert unter EPL2.0. Ihr Code ist auf der GitLab-Instanz der RWTH Aachen für die Öffentlichkeit zugänglich. MAiNGO ist dabei ein deterministischer globaler Optimierungslöser[7]. MeLOn ist eine Toolbox, die Modelle

des maschinellen Lernens in ein MAiNGO-Optimierungsproblem integrieren kann[8]. MAiNGO entwickelte sich aus Forschungsarbeiten zweier Doktoranden ab dem Jahre 2015. Die Forschungsarbeit eines weiteren Doktoranden legte zwei Jahre später den Grundstein für MeLOn. Die Software-Pakete wurden mit Unterstützung von anderen Mitarbeitenden und studentischen Hilfskräften entwickelt sowie implementiert. Ende 2019 wurden beide Softwares über GitLab veröffentlicht, am Institut für Systemverfahrenstechnik weiterentwickelt und bei Forschungs- und Industrieprojekten eingesetzt. Außerhalb der RWTH werden die Softwares unter anderem an der KU Leuven (Belgien), dem Imperial College London (Großbritannien) und dem National Institute of Standards and Technology (USA) genutzt. Mit dem Imperial College und der KU Leuven arbeitet die RWTH inzwischen eng zusammen, um die Software und die dahinterstehenden mathematischen Methoden weiterzuentwickeln. MAiNGO und MeLOn wurden auch in einer aktuellen Veröffentlichung der ETH Zürich angewendet[9].

Beide Programme sind in C++ implementiert und plattformübergreifend anwendbar. Die Software MAiNGO bietet für den Nutzer vielfältige Schnittstellen an und kann über C++, C, Python oder Textdateien angesprochen werden. Hierzu wird die Hilfsbibliothek libALE genutzt, um mathematische Texteingaben für den Nutzer zu vereinfachen. Dabei dient libALE der Modellierung komplexer Optimierungsprobleme und ist ebenfalls am Lehrstuhl für Systemverfahrenstechnik entstanden und wird dort stetig weiterentwickelt. MAiNGO und MeLOn nutzen zudem umfassend die Bibliothek MC++, die am Imperial College London entsteht. In dieser Software sind Relaxierungen implementiert, welche später in MAiNGO verwendet werden. Zudem wurde die MC++ durch die Wissenschaftlerinnen und Wissenschaftler der RWTH mit spezifischen Funktionen für die Verfahrenstechnik erweitert. Daneben nutzt MAiNGO eine Reihe weiterer Bibliotheken wie FADBAD, für die Berechnung von Ableitungen durch automatische Differentiation, sowie verschiedenste Löser für lineare sowie nichtlinear Optimierungsprobleme.

Literatur

[1] Bongartz, D., Mitsos, A., Deterministic global optimization of process flowsheet in a reduced space using McCormick relaxations, Journal of Global Optimization, Nr. 69(4), pp. 761-796, 2017a

[2] Bongartz, D., Mitsos, A., Deterministic global flowsheet optimization: Between equation-oriented and sequential-modular methods, AiChE Journal, Nr. 65(3), pp. 1022-1034, 2019

[3] Rall, D., Menne, D., Schweidtmann, A., Kamp, J., von Kolzenberg, L., Mitsos, A., Wessling, M., Rational design of ion separation membranes, Journal of Membrane Science, Nr. 569, pp. 209-219, 2019

[4] Rall, D., Schweidtmann, A., Aumeier, B., Kamp, J., Karwe, J., Ostendorf, K., Mitsos, A., Wessling, M., Simultaneous rational design of ion separation membranes and processes, Journal of Membran Science, Nr. 600, p. 117860, 2020

[5] Rall, D., Schweidtmann, A., Kruse, M., Evdochenko, E., Mitsos, A., Wessling, M., Multi-scale membrane process optimization with high-fidelity ion transport models through machine learning, Journal of Membrane Science, p. 118208, 2020

[6] Schweidtmann, A., Mitsos, A., Deterministic Global Optimization with Artificial Neural Networks Embedded, Journal of Optimization Theory and Applications, Nr. 180, pp. 925-948, 2019

[7] https://git.rwth-aachen.de/avt-svt/public/maingo

[8] https://git.rwth-aachen.de/avt-svt/public/MeLOn

[9] Forster, T., Vázquez, D., Guillén-Gosálbez, G., Algebraic surrogate-based process optimization using Bayesian symbolic learning, AIChE Journal, Nr. 69(8), 2023

Autoren

Univ.-Prof. Alexander Mitsos, Ph.D., ist Inhaber des Lehrstuhls für Systemverfahrenstechnik, Direktor des Institutsbereichs Energiesystemtechnik am Forschungszentrum Jülich und Projektleiter der Software MAiNGO und Toolbox MeLOn.

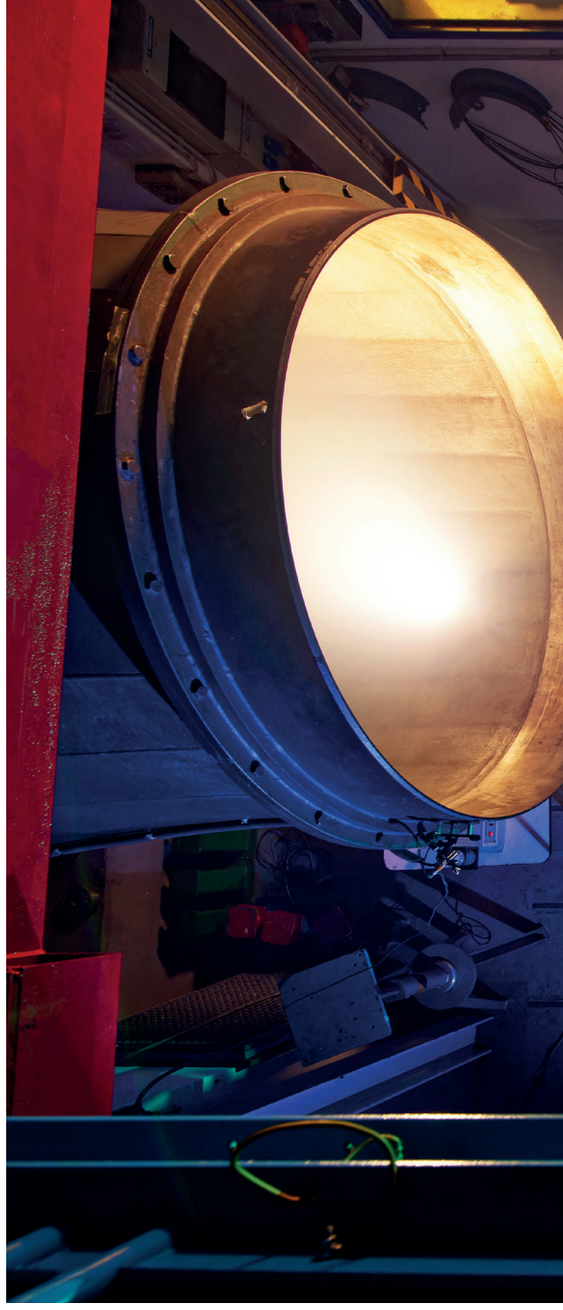
Jannik Lüthje, M.Sc., und Clara Witte, M.Sc., sind wissenschaftliche Mitarbeitende am Lehrstuhl für Systemverfahrenstechnik. Lüthje ist Ansprechperson der Toolbox MeLOn, Witte Entwicklerin der Software MAiNGO.

Ass.-Prof. Dr.-Ing. Dominik Bongartz ist Hauptentwickler der Software MAiNGO und Assistant Professor am Lehrstuhl für chemische und biochemische Reaktortechnik und -sicherheit der KU Leuven.

Dr.rer.nat. Jaromił Najman ist ehemaliger Hauptentwickler der Software MAiNGO.

Ass.-Prof. Dr.-Ing. Artur M. Schweidtmann ist Initiator der Software MeLOn und Assistant Professor am Lehrstuhl für Verfahrenstechnik der TU Delft.

Kai-Uwe Schröder, Eike Stumpf, Maurice Zimmnau



Software für eine zukunftsfähige Luftfahrt

Mit UNICADO und ASAMI forschen und lehren

Climate impact of aviation has drastically to be reduced. In order to still close the business case for the main air transport stakeholders, i.e. aircraft manufacturers, aircraft leasing companies and airlines this requires aircraft configurations with very low energy demand and new concepts of operation. Thus, design environments coping with novel requirements, design processes and technologies and delivering the requested kind of air mobility system have to be developed. Following this path, at RWTH Aachen University the University Conceptual Aircraft Design and Optimization environment (UNICADO) and the Aachen Structural Analysis Multiscale Integration software (ASAMI) are brought to life. UNICADO aims at coupling all relevant aircraft design disciplines and it is developed jointly together with all German aircraft design universities as open source software product. In contrast, ASAMI bridges the different scales of modelling during design.

Die Luft- und Raumfahrtbranche ist dafür bekannt, das technisch Machbare auszuloten. Gleichzeitig ist die Branche aber durchaus konservativ, da zum einen die Gewinnmargen überschaubar sind und zum anderen hohe Sicherheitsanforderungen im Betrieb gestellt werden.

Herausforderungen durch den Klimawandel

Der menschengemachte Klimawandel zwingt die Branche nun, den evolutionären Entwicklungspfad zumindest teilweise zu verlassen und stattdessen revolutionäre Konzepte in Betracht zu ziehen. In der Luftfahrt steht aller Voraussicht nach ein Wechsel des Energieträgers – von fossilen Kohlenwasserstoffen (Kerosin) hin zu Wasserstoff und synthetisch produzierten Kohlenwasserstoffen (Sustainable Aviation Fuel, SAF) – an[1]. Ebenso wird sich die Auslegung und Konfiguration der Flugzeuge ändern: Schlanke Flügel mit großer Spannweite aus Kompositwerkstoffen werden unter anderem in den nächsten Jahren an einem neuen X-Plane der NASA getestet. Des Weiteren haben Blended-Wing-Body-Konfigurationen, kurz BWB, siehe Bild 3, eine Kombination aus Nurflügler und breitem, auftriebsproduzierenden Flugzeugrumpf, sowohl in den USA als auch in Europa wieder den Weg auf die Forschungsagenda gefunden. Eine BWB-Konfiguration ist unter den geänderten Randbedingungen überaus vielversprechend: Sie verbindet den Vorteil einer deutlich reduzierten umspülten Oberfläche und damit eines geringeren Reibungswiderstands (bis zu 30 Prozent weniger umspülte Oberfläche im Vergleich zu einer klassischen Konfiguration) mit einem ausreichenden Platzangebot für die voluminösen Tanksysteme für flüssigen Wasserstoff. Auch die etablierten Betriebskonzepte im Lufttransport stehen auf dem Prüfstand. So werden bislang für den zivilen Personentransport noch nicht zugelassene Abläufe wie Formationsflug und Luftbetankung untersucht[1]. All diese Ansätze

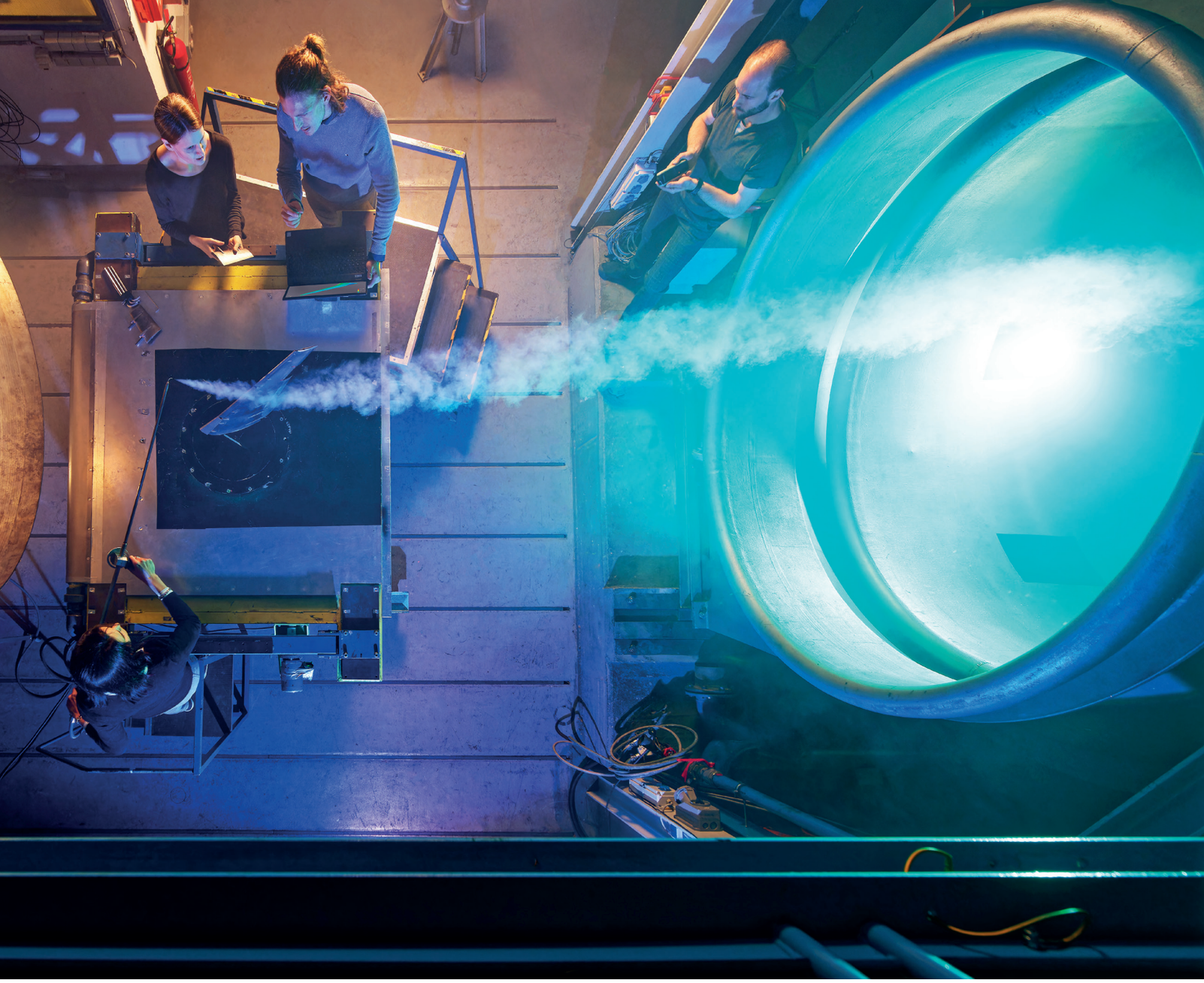


Bild 1: Softwareseitige Strömungssimulation bedarf experimenteller Validierung, hier die qualitative Untersuchung des Strömungsverhaltens eines Flügels in Hochauftriebskonfiguration
Foto: Peter Winandy

stellen eine massive Änderung des Status quo dar. Für keinen dieser Ansätze gibt es bisher erprobte Entwurfswerkzeuge, keiner ist umfassend numerisch sowie experimentell getestet und natürlich hat keiner den Luftfahrt-Zertifizierungsprozess durchlaufen. Neuland zu betreten ist für Ingenieurinnen und Ingenieure der Luft- und Raumfahrt an sich nicht ungewöhnlich, der vorgegebene Zeithorizont lässt das Unterfangen aber zu einer großen Herausforderung werden. Es wurden allerdings – wenn man an das Apollo-Programm mit der bemannten Mondlandung innerhalb eines Zeitraums von weniger als zehn Jahren denkt – schon größere Herausforderungen gemeistert, wenngleich damals mit quasi unlimitiertem Budget. Erklärtes Ziel ist diesmal der Erstflug eines wasserstoffgetriebenen Passagierflugzeugs bis 2035[2] und die reguläre Markteinführung von weitgehend klimaneutralen Flugzeugkonfigurationen und Flugbetriebsprozessen bis 2050. Sollte die Transformation hin zu einem nachhaltigen Lufttransportsystem nicht gelingen, wird die Luftfahrt, aufgrund von absehbaren Reduktionspotenzialen in anderen Branchen, von einem heute kleinen zum branchenübergreifend größten Emittenten von Treibhausgasen im Jahr 2050.

Die Neuartigkeit und der Zeitdruck sind bei dieser Aufgabe Chance und Risiko zugleich. Das Forschungs- und Entwicklungsrisiko kann auf der einen Seite zu erheblichen Mehraufwänden führen und die zeitgerechte Fertigstellung gefährden. Auf der anderen Seite kann der Zeitplan nur gehalten werden, wenn die Möglichkeiten der digitalen Transformation umfassend ausgeschöpft werden. Dies bedeutet eine große Aufgabe für die Wissenschaft. Ein zentraler Aspekt ist die Umsetzung eines digitalen Fadens und eines digitalen Zwillings schon auf den niedrigen Technologieentwicklungsstufen (Technology Readiness Level, TRL) im Bereich der angewandten Forschung. Der digitale Faden verbindet chronologisch und virtuell die aufeinanderfolgenden Stationen der Forschung und Entwicklung bezüglich eines Produktes oder einer Dienstleistung entlang des kompletten Lebenszyklus und stellt den Datenfluss sicher. Der Vorteil besteht darin, dass nachgelagerte Prozessschritte vollständig auf das zuvor spezifisch erzeugte Konvolut aus Daten, Informationen und Wissen zugreifen zu können. Anders als heute lassen sich so die Verluste an den Schnittstellen zwischen angewandter Forschung und Industrie und innerhalb der industriellen Entwicklung zwischen den Fachabteilungen vermeiden und die Entscheidungen und Festlegungen im Entwurf (das sogenannte „Design Freeze“) können später im Prozess auf Basis eines umfassenderen aufgebauten Wissens getroffen werden. Benötigt wird dafür unter anderem eine möglichst genaue Beschreibung der Technologie in einer Entwicklungsumgebung. Die initiale Beschreibung wächst im Zuge der Forschung und Entwicklung zu einer vollständigen Beschreibung, dem „digitalen Zwilling“,

Bild 2: Vor quantitativen Messungen kommen bei der Untersuchung von Strömungszuständen in der Regel qualitative Visualisierungsverfahren zum Einsatz, welche im Nachgang mit Strömungssimulationsergebnissen abgeglichen werden.
Foto: Peter Winandy

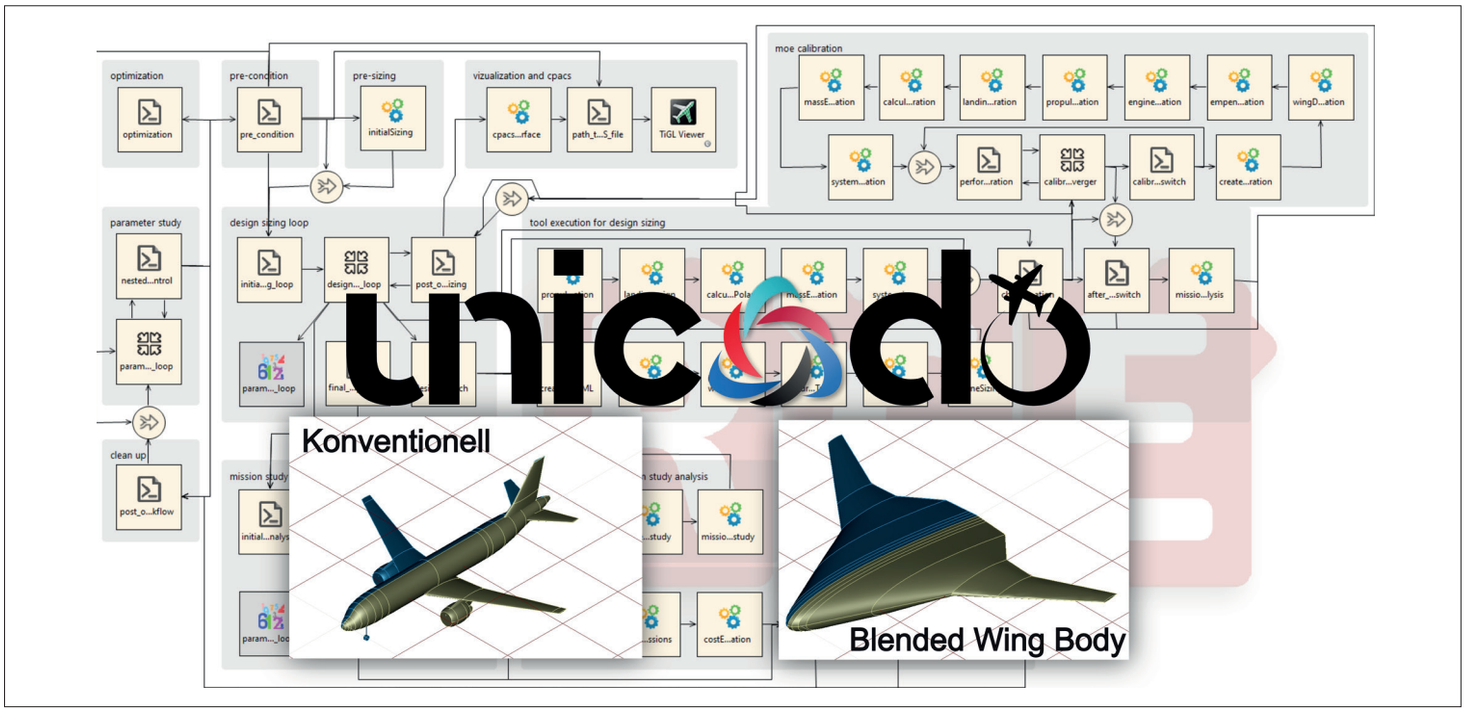


Bild 3: Entwurfsplattform UNICADO für konventionelle und unkonventionelle Flugzeugkonfigurationen

heran. Befüllt wird der digitale Zwilling im Bereich der sehr niedrigen TRL vor allem mit simulationsbasierten Entwurfsergebnissen, erst in späteren Phasen kommt tatsächlich Hardware zum Einsatz.

In der Luft- und Raumfahrt beginnt der Forschungs- und Entwicklungsprozess für ein neues Vehikel mit der Konzeptfindung. Auf Basis der Ergebnisse von einfachen Berechnungen wird eine geringe Anzahl an Konzepten ausgewählt, die in der Vorentwurfsphase weiter ausgearbeitet, optimiert und bewertet werden. Nach der Konfigurationsauswahl wird im detaillierten Entwurf sukzessive die Granularität der Simulationen und des Entwurfs gesteigert.

Für die neuen technologischen Ansätze, die helfen sollen, die ambitionierten Nachhaltigkeitsziele zu erreichen, braucht es neue Softwaremodelle und simulationsbasierte Entwurfsumgebungen – sowohl für den Vorentwurf wie auch den detaillierten Entwurf. Hierbei spielen die Erkenntnisse und die Prozesse des Research Software Engineering – wie Versionierung, Softwaretest und -pflege, Programmiervorgaben – eine maßgebliche Rolle. Zudem werden Ingenieurinnen und Ingenieure benötigt, die die neuen Entwurfswerkzeuge bedienen und (weiter)entwickeln können.

UNIversity Conceptual Aircraft Design and Optimization – UNICADO

Seit den späten 80er-Jahren wird in der deutschen Forschungslandschaft an softwarebasierten Entwurfswerkzeugen für Flugzeuge gearbeitet. Am Institut für Luft- und Raumfahrtsysteme ist dabei die universitäre Softwareumgebung MICADO (Multidisciplinary Integrated Conceptual Aircraft Design and Optimization) für den konzeptionellen Flugzeugentwurf entstanden.

Bei hochmultidisziplinären Problemstellungen wie dem ganzheitlichen softwaretechnischen Flugzeugentwurf ist auch der softwareentwicklungsseitige Aufwand hoch, entsprechend groß müssen die Entwicklerteams sein. Eine weitere Herausforderung ist, dass die Fluktuation der wissenschaftlich Mitarbeitenden an den Forschungsinstituten unausweichlich einen Verlust von Wissen nach sich zieht. Um diesen Herausforderungen zu begegnen, wurde die Initiative UNICADO[2] (UNIversity Conceptual Aircraft Design and Optimization) ins Leben gerufen. Hier bündeln die sechs führenden deutschen Luftfahrt-Universitäten RWTH Aachen, TU Berlin, TU Braunschweig, TU Hamburg, TU München und Universität Stuttgart die Entwicklungs- und Entwurfskompetenzen der akademischen Forschungslandschaft im Bereich des Flugzeugentwurfs. UNICADO basiert auf dem Entwurfstool MICADO des RWTH-Instituts für Luft- und Raumfahrt, finanziert wird die Entwicklung durch das Luftfahrtforschungsprogramms der Bundesregierung (LuFo-Projekte UNICADO I/UNICADO II).

Mit UNICADO soll eine langlebige, wartbare, robuste und öffentlich zugängliche Software (Open Source nach GNU General Public License Version 3.0) für den Flugzeugvorentwurf aufgebaut werden, die sich in der akademischen Forschung und der universitären Lehre als Standard etabliert. Über die gemeinsame Entwicklung einer Software für den Flugzeugentwurf wird der Entwicklungsaufwand auf die beteiligten Universitäten verteilt. Die geplante Veröffentlichung der Software als Open-Source-Produkt potenziert die Anzahl möglicher Nutzender und Entwickelnder.

Anhand von UNICADO werden künftige Ingenieurinnen und Ingenieure der Luft- und Raumfahrt ausgebildet. Darüber hinaus werden mit UNICADO die Lehrstühle einzeln und im Verbund dazu befähigt, die Luftfahrtindustrie hinsichtlich der Erforschung und Bewertung von innovativen Technologien auf Gesamtflugzeugebene zu unterstützen. Letztlich soll mit UNICADO ein wichtiger Beitrag zur Minimierung der Emissionen der Luftfahrt auf dem Weg zur Klimaneutralität geleistet werden.

Kollaborative akademische Softwareentwicklung

Für die Veröffentlichung von UNICADO wurden zunächst die Software MICADO von den Partnern analysiert und Regeln in Bezug auf Programmiersprache, Programmierstil und Formatierung definiert. Die Regeln zielen auf eine Erhöhung der Wartbarkeit, der Verständlichkeit und der Lesbarkeit des Codes sowie der Erweiterbarkeit der Software. Wie derzeit in der Softwareentwicklung üblich, wird auf

die Versionierung der Software durch das Versionskontrollsystem Git gesetzt. Mit der steigenden Anzahl an Entwickelnden wurden Qualitätssicherungsmaßnahmen notwendig, um eine kontinuierliche Integration von Softwareänderungen zu ermöglichen. Ein erstes Quality Gate, welches implementiert wurde, ist ein verpflichtendes Code-Review. Jede Codeänderung muss überprüft und genehmigt werden, bevor die Änderung zurück in die Codebasis auf dem Server gespielt werden kann. Das im Code-Review üblicherweise verwendete Vieraugenprinzip ist eine erste Maßnahme, verhindert Fehler aber nicht zuverlässig. Daher wurden Softwaretests mit einer Automatisierung ins Leben gerufen. Überprüft wird die Software auf unterschiedlichen Ebenen: von der Funktionseinheit, die der Entwickelnde geändert hat, über die Entwurfsdisziplin, an welcher gearbeitet wurde (beispielsweise Flügelentwurf), bis hin zur Gesamtflugzeugebene. Als zweites Quality Gate wird somit jede Codeänderung durch dafür geeignete Tests automatisiert überprüft.
An der RWTH wird UNICADO bereits in der Lehre eingesetzt, beispielsweise in der Veranstaltung „Flugzeugkonzeptstudien“. Hier sammeln die Studierenden Programmiererfahrung, produzieren Entwurfsergebnisse und liefern der Entwicklergruppe wertvolles Feedback.
Häufig wird Software an einer Universität durch einzelne wissenschaftlich Mitarbeitende entwickelt. Selten ist die Software ausreichend kommentiert und dokumentiert. Mit dem Ausscheiden der Mitarbeitenden ist das Wissen für den Lehrstuhl dann verloren.
Dies ist im UNICADO-Verbund anders: Über die Kooperation kann dauerhaft eine „kritische Masse“ in den universitätsübergreifenden Entwicklungsteams erhalten und der Wissenserhalt sichergestellt werden. Einen Beitrag dazu leisten zudem die Studierenden und die Open-Source-Community.

Aachen Structural Analysis Multiscale Integration – ASAMI

Wird durch die Entwicklung von UNICADO die Zusammenarbeit zwischen verschiedenen Disziplinen im Flugzeugentwurf ermöglicht, greift die Software ASAMI (Aachen Structural Analysis Multiscale Integration) die skalenübergreifende Modellierung auf. Aufgrund der Komplexität des Systems Flugzeug stellt die Auslegung von Flugzeugstrukturen eine Besonderheit im Strukturentwurf dar. Die Komplexität äußert sich dabei in der Kopplung des Verformungsverhaltens der Bauteile untereinander. Anders als sonst bei Strukturen technischer Systeme üblich, ist es beim Flugzeug dadurch kaum möglich, die Bauteile einzeln zu betrachten, jeweils für sich auszulegen und am Ende in die Gesamtstruktur zu integrieren. Stattdessen wird der Entwurfsprozess mit Modellen relativ geringer Granularität gestartet, die das Gesamtverhalten der Struktur beschreiben und es erlauben, das Tragkonzept, das Material und die Topologie des Tragwerks festzulegen sowie erste Abmessungen und damit Massen zu bestimmen. Diese Ebene der Modellierung wird als Makroebene bezeichnet und befindet sich in der sogenannten Konzept- und Vorentwurfsphase des Entwurfsprozesses. Wichtig ist hierbei die permanente Einbindung in den UNICADO-Auslegungsprozess, wodurch die Konsistenz zu den anderen Disziplinen des Flugzeugentwurfs hergestellt und sichergestellt wird. Insbesondere die Massen und ihre Verteilung stellen wesentliche Entwurfsparameter für das Gesamtsystem Flugzeug dar und müssen zu einem frühen Zeitpunkt bekannt sein.
Nach der Festlegung der Entwurfsvariablen auf der Makroebene wird die Granularität der Modellierung erhöht und die Geometrie der Struktur weiter festgelegt. Hierunter fallen jetzt die Querschnittsformen und die Abmessungen der Bauteile. Die Herausforderung ist die Unstetigkeit, die durch den Modellwechsel bedingt mit der höheren Detaillierung einhergeht. Dies ist vergleichbar mit einem Mikroskop, mit dem zunächst die Struktur untersucht und schließlich der Vergrößerungsfaktor erhöht wird. Neue Strukturen werden sichtbar, diese unterscheiden sich von gewachsenen Organismen: die detaillierteren Strukturen müssen erst noch entworfen werden. Da die Entwicklung vom großen zum kleinen Maßstab erfolgt, kann die Konsistenz der Modelle nicht über eine Homogenisierung erfolgen. Vielmehr müssen diese über geeignete Datenstrukturen und definierte Datenanschlusspunkte sichergestellt werden. Diese Datenstrukturen aufzubauen und die Modelle der unterschiedlichen Skalen auf Konsistenz zu prüfen beziehungsweise entsprechende Randbedingungen für die unterschiedlichen Skalen abzuleiten, ist Aufgabe von ASAMI.
Die Entwicklung, Wartung und Pflege der Software stellen damit das Rückgrat für zukünftige Forschungsprojekte in der Strukturanalyse dar und verlangen ein professionelles Vorgehen. In der Softwareentwicklung werden aus diesem Grund dieselben Prinzipien wie bei UNICADO verfolgt. Sowohl neue Bauweisen als auch Materialien und deren Modellansätze aus verschiedenen Quellen können dann über die Anbindung an ASAMI in die Gesamtstruktur des Flugzeugs integriert werden. Dadurch ist es möglich, Technologiebewertungen schon früh im Entwurfsprozess vorzunehmen und Optimierungsläufe auf der Ebene der Gesamtstruktur durchzuführen.

Knowledge. Impact. Networks.

Der Ansatz der großen universitären Konsortien zur Softwareentwicklung im Entwurf hat sich bewährt. Mit UNICADO und ASAMI entstehen wegweisende Softwareumgebungen, die das Potenzial haben, die universitäre Lehre nachhaltig zu verbessern und die akademische Forschung im Bereich Flugzeugentwurf und Technologieintegration und -bewertung effizient mit den Aktivitäten der Industrie zu verbinden.

Literatur

[1] Air Transport Action Group, Waypoint 2050 - Balancing growth in connectivity with a comprehensive global air transport response to the climate emergency: a vision of net-zero aviation by mid-century, Genf, 2021
[2] Zimmnau, M., Schültke, F., Stumpf, E., UNICADO: multidisciplinary analysis in conceptual aircraft design, CEAS Aeronautical Journal, Band 14, Heft 1, pp. 75-89, 2022

Autoren

Univ.-Prof. Dr.-Ing. Kai-Uwe Schröder ist Inhaber des Lehrstuhls und Leiter des Instituts für Strukturmechanik und Leichtbau.
Univ.-Prof. Dr.-Ing. Eike Stumpf ist Inhaber des Lehrstuhls und Leiter des Instituts für Luft- und Raumfahrtsysteme.
Maurice Zimmnau, M.Sc., ist wissenschaftlicher Mitarbeiter am Institut für Luft- und Raumfahrtsysteme.

Marek Behr, Norbert Hosters, Fabian Key

In Raum und Zeit durch das Bottleneck computergestützter Mechanik

Der Multi-Physics-Code XNS

Continuous progress in all technical areas is leading to ever more complex products. Predictions that are as accurate as possible in all phases of the development process are essential for application-related optimization. In principle, numerical simulation methods offer an attractive tool for dealing with these challenges. At the same time, the constant increase in available computing power fuels the hope that the requirements can be fulfilled without restriction. However, this is only partially the case, as partitioning is limited to spatial dimensions. The simulation of dynamic processes can currently only benefit from the increase in computing power to a very limited extent. To address this bottleneck in computational mechanics, CATS has been working on the development of space-time methods since its foundation. In contrast to classical methods, these methods not only consider the spatial dimensions, but also a so-called space-time continuum. The additional dimension makes it possible to integrate this into the partitioning of the problem. A complete utilization of computing clusters is thus theoretically possible. The article discusses the stated limitations, a solution approach, and the software development of a corresponding finite element solver at CATS.

Stetiger Fortschritt und höhere Anforderungen in allen technischen Bereichen führen zu immer komplexeren Produkten. So erfordern beispielsweise ressourceneffiziente Produkte aufwendiges Design unter möglichst geringem Materialeinsatz. Dabei sind für eine Optimierung in Bezug auf den jeweiligen Anwendungszweck möglichst akkurate Vorhersagen in allen Stufen des Entwicklungsprozesses absolut notwendig. Durch hohen Kostendruck und immer kürzere Entwicklungszyklen ist der Wunsch nach hochpräzisen, aber gleichzeitig schnelleren Vorhersagetools allgegenwärtig. Grundsätzlich bieten numerische Simulationsverfahren ein attraktives Werkzeug, um diesen Herausforderungen gerecht werden zu können. Die ebenfalls kontinuierlich zunehmende verfügbare Rechenleistung nährt dabei gleichzeitig

Bild 1: Mitarbeiter des CATS analysieren Ergebnisse aus der Raum-Zeit Strömungssimulation um eine Herzklappe.
Foto: Peter Winandy

die Hoffnung, die Anforderungen uneingeschränkt erfüllen zu können. In der Realität gilt dieser Zusammenhang aber nur bedingt. Besonders die Simulation dynamischer Prozesse kann aktuell nur sehr eingeschränkt von der Zunahme der Rechenressourcen profitieren. Im Folgenden werden die Gründe für diese Einschränkungen sowie eine vom Lehrstuhl für Computergestützte Analyse technischer Systeme erforschte Methode zur Lösung vorgestellt. Darüber hinaus wird die Softwareentwicklung zur Umsetzung dieser Ansätze präsentiert.

Das Bottleneck computergestützter Mechanik

Im Gegensatz zu Computeranimationen in Filmen oder Computerspielen, welche ausschließlich eine möglichst realistische Anmutung der dargestellten Physik erzielen sollen, wird in der computergestützten Mechanik versucht, mit Hilfe von Simulationen möglichst exakt Fragen in Bezug auf technische Problemstellungen zu beantworten. Welchen Auftrieb und Widerstand haben neu entworfene Flugzeugflügel? Wie viel Leistung kann grundsätzlich aus Windkraftanlagen gewonnen werden? Welche Eingriffe am Herzen sind sinnvoll? Die notwendige Rechenzeit solcher Simulationen wird von vier Faktoren bestimmt, welche in Bild 2 schematisch dargestellt sind. Die Komplexitätszunahme technischer Anwendungen ist unmittelbar mit einem Anstieg der Anzahl an Freiheitsgraden verbunden, deren Bestimmung Ziel der Simulationen ist. Gleichzeitig steigt die Leistungsfähigkeit von Hochleistungsrechenzentren exponentiell an, ablesbar an der Leistung der schnellsten Hochleistungsrechner. Dies wird durch zwei Faktoren ermöglicht: Dem Anstieg der Leistungsfähigkeit einzelner Recheneinheiten auf Computerprozessoren und den komplexeren Infrastrukturen zur effizienten Integration von immer mehr Prozessoren. Damit liegt die Folgerung nah, dass die steigende Problemkomplexität mit steigenden Rechenressourcen kompensiert werden kann. Der Idee „Divide-and-Conquer“ folgend, können im System vorhandene Freiheitsgrade auf eine immer höhere Anzahl zur Verfügung stehender Recheneinheiten verteilt werden, der sogenannten Partitionierung. Im Idealfall muss je Recheneinheit nur ein Freiheitsgrad gelöst werden. Diese sogenannte ideale starke Skalierbarkeit ist allerdings in der Praxis nicht umsetzbar. Die gegenseitige Abhängigkeit der Freiheits-

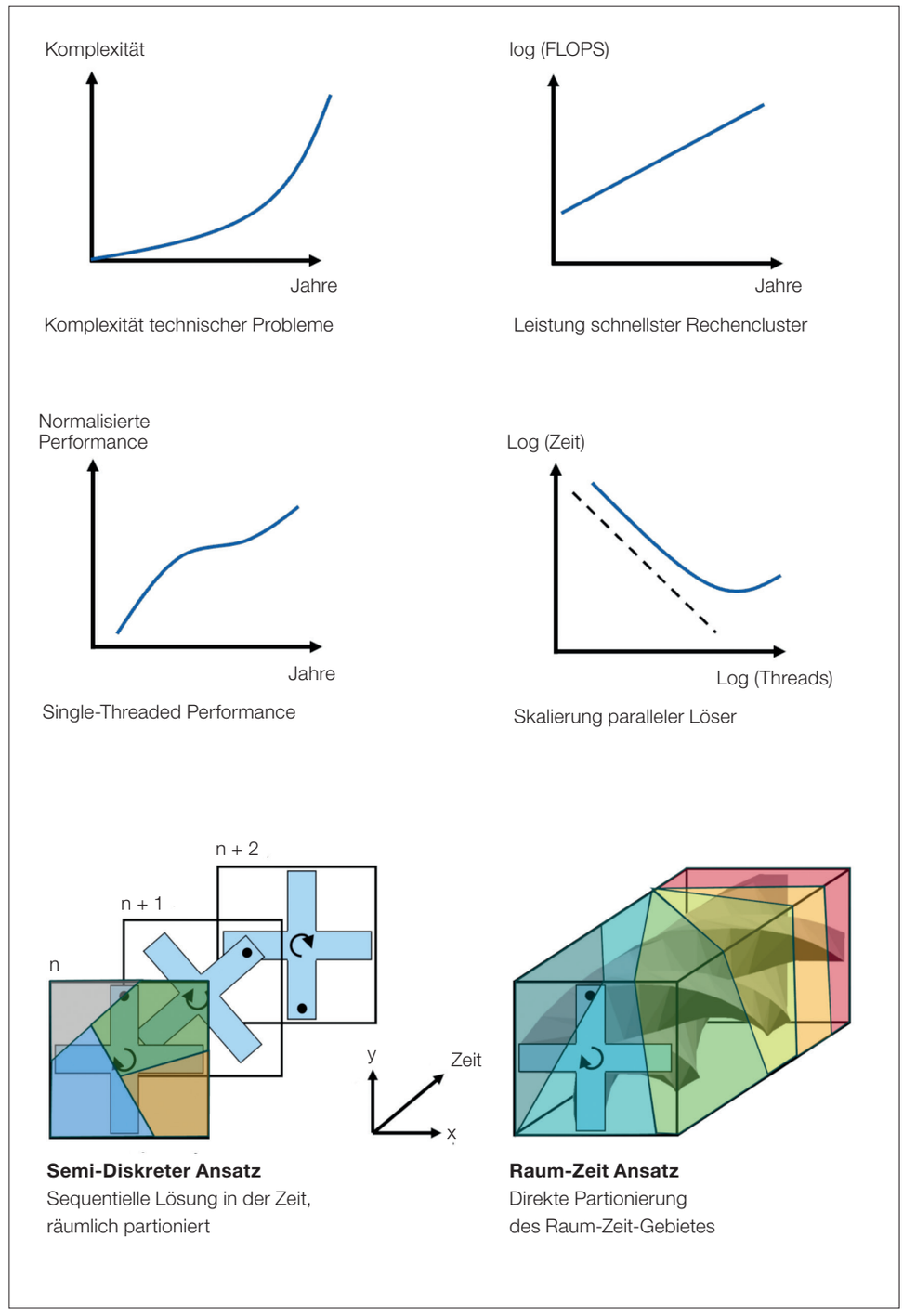
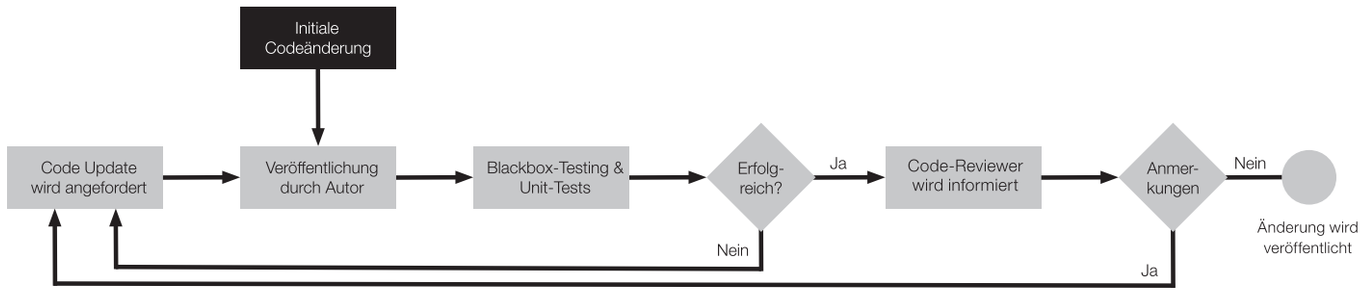


Bild 2: Einflussfaktoren auf die Simulationszeit und Lösungs- beziehungsweise Partitionierungsansätze

grade bedingt einen notwendigen Austausch an Informationen, die resultierende Kommunikation bremst die Lösungsgeschwindigkeit. Daraus folgt eine minimale Anzahl an Freiheitsgraden pro Recheneinheit, deren Unterschreitung zu ineffizienten Lösungsverfahren führt. Das feststehende Minimum hat nun zur Folge, dass die Rechenzeit zur Problemlösung nur noch durch zunehmende Prozessorleistungen gesteigert werden kann und die gegebenenfalls zusätzlich verfügbaren Rechenressourcen nicht genutzt werden können. Für zeitlich unveränderliche Probleme ist diese Charakteristik unproblematisch. Vorausgesetzt, dass das beschriebene Minimum unabhängig von der involvierten Partitionsanzahl ist, können größere Probleme einfach auf mehr Partitionen verteilt werden. Problematisch ist dagegen die Analyse von dynamischen Problemstellungen, welche einem Großteil der technischen Systeme entspricht. Beispiele sind drehende Windkraftanlagen, Böenanalysen von Flugzeugen oder die Analyse des schlagenden Herzes. Hier gilt es, über die Zeit veränderliche Probleme zu analysieren, für deren heutige kommerzielle und große Open-Source-Simulationswerkzeuge auf semidiskrete Verfahren setzen. Diese lösen das Problem sequenziell in der Zeit, das heißt pro definiertem Zeitschritt wird ein räumliches Problem gelöst, unter Verwendung einer ebenfalls räumlichen Partitionierung. Bedingt durch den beschränkten Skalierungseffekt kann die Lösungsgeschwindigkeit pro Zeitschritt und die resultierende Gesamtlaufzeit zur Lösung aller Zeitschritte bei Erreichen des Minimums nur noch durch die Geschwindigkeit der einzelnen Recheneinheiten beeinflusst werden. Auch für diesen Fall bedeutet also eine größere Anzahl an zur Verfügung stehenden Recheneinheiten zwar die Möglichkeit, größere Probleme berechnen zu können. Die Rechenzeit zur Lösung einzelner Zeitschritte bleibt bei konstanter Rechenkernleistung allerdings unverändert. Konstant große dynamische Probleme können also ebenfalls nicht von zusätzlichen Recheneinheiten profitieren. Reduzierungen der Rechenzeit ergeben sich nur aus der Rechenleistung neuer Rechenkerngenerationen. Kleine Zeitschrittweiten und eine hohe notwendige Zeitspanne machen deshalb eine Analyse von vielen dynamischen Problemen mit den etablierten Ansätzen auf absehbare Zeit unmöglich und führen damit zum Bottleneck der computergestützten Mechanik. Dieses Bottleneck ist nur mit alternativen Methoden zu bewältigen.

Vereinfachte Workflow der XNS Entwicklung



Rollenverteilung

Progress

Developer

Entwickelt neue
· Methoden
· Dokumentation
· Testfälle
· Änderungen am RSE Framework

Diskutiert potentielle Verbesserungen
Begutachtet Änderungen anderer **Entwickler**

Veröffentlicht genehmigte Änderungen, schlägt fundamentale Änderungen vor

Control

Maintainer

Verantwortet spezifische Funktion

Begutachtet final Änderungsvorschläge der **Developer**

Organisiert Kommunikation der genehmigten Änderungen

Berichtet potentiell bedeutende Änderungen und gegebenenfalls entstehende Konflikte an **Steering Committee**

Dokumentiert Diskussionen öffentlich

Aber: Ist nicht zuständig Fehlerkorrekturen, Testfälle, Ideen, etc. zu liefern

Guidance

Steering Committee

Behält Übersicht über Funktionalität

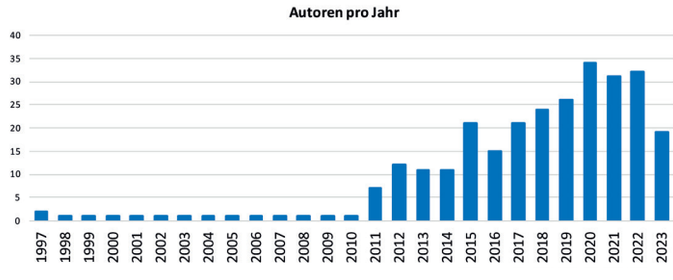
Entscheidet langfristige Strategie

Entscheidet über die von **Maintainern** berichtete bedeutende Änderungen und Konflikte

Ernennt **Maintainer**

Kommuniziert Entscheidungen

Beiträge zur Softwareentwicklung



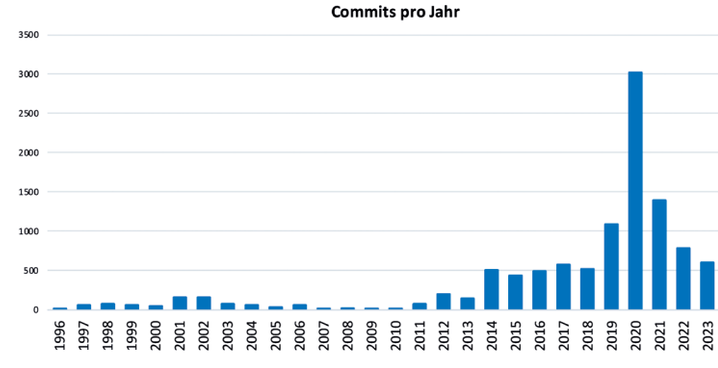


Bild 3: Übersicht Softwareentwicklung und Statistiken

Bild 4: Mit Hilfe der 3D-Powerwall können komplexe Ergebnisse visualisiert und diskutiert werden.
Foto: Peter Winandy

Raum-Zeit-Ansatz

Um dem beschriebenen Bottleneck in der computergestützten Mechanik entgegentreten zu können, beschäftigt sich der Lehrstuhl für Computergestützte Analyse technischer Systeme schon seit seiner Gründung mit der Entwicklung von Raum-Zeit-Verfahren. Diese betrachten im Gegensatz zu klassischen Verfahren nicht ausschließlich die räumlichen Dimensionen, sondern ein sogenanntes Raum-Zeit-Kontinuum. Wie in Bild 2 illustriert, ergibt sich für einen drehenden Rotor nicht mehr eine schrittweise veränderliche Konfiguration, sondern eine kontinuierliche Problembeschreibung. Für dynamische Probleme werden aus zweidimensionalen Problemen im Raum dreidimensionale Probleme im Raum-Zeit-Gebiet, aus 3D wird 4D.

Wie kann mit diesem Ansatz nun die Laufzeit von Simulationen signifikant reduziert werden? Die zusätzliche Dimension erlaubt es, diese auch in die Partitionierung des Problems zu integrieren. Einerseits haben alle Freiheitsgrade eine zusätzliche Zeitkomponente, zusätzlich ergibt sich durch die Raum-Zeit-Betrachtung ebenfalls eine Steigerung der Anzahl der Freiheitsgrade. Obwohl die zulässige Freiheitsgradeanzahl pro Partition auch bei diesem Ansatz beschränkt ist, kann das gesamte Raum-Zeit-Gebiet dennoch nun in mehr Partitionen unterteilt werden. Die Vorteile durch das Prinzip „Divide-and-Conquer“ entsprechen nun denen von stationären Problemen. Damit ist die notwendige Rechenzeit zur Lösung dynamischer Probleme nicht mehr durch die minimal zulässige Partitionsgröße beschränkt, sondern durch die Größe des Rechenclusters. Eine vollständige Ausnutzung solcher Rechencluster ist damit theoretisch denkbar. Diese theoretische Skalierbarkeit konnte grundsätzlich demonstriert werden, sie enthält allerdings weiterhin großen Forschungsbedarf. Während für die Lösung von Diffusionsgleichungen, zum Beispiel bei Wärmeleitungsproblemen, die beschriebenen Überlegungen bestätigt werden konnten, ist für Advektions-Diffusions-Probleme, etwa klassische Probleme aus der Strömungsmechanik, die theoretische Skalierbarkeit allerdings schon durch mathematische Nachweise widerlegt. Dennoch kann auch in solchen Fällen der Raum-Zeit-Ansatz signifikante Beschleunigungen, beispielsweise durch die lokale Raum-Zeit-Verfeinerung, liefern, welche mit klassischen Ansätzen nur sehr eingeschränkt realisierbar sind. Diese Ansätze stehen im Mittelpunkt von Forschungsarbeiten an unstrukturierten Raum-Zeit-Finite-Elementen des Lehrstuhls für Computergestützte Analyse technischer Systeme. Weitere Forschungsfragen ergeben sich durch Löser, Vernetzungsmethoden oder Parallelisierung, welche am hauseigenen Finite-Elemente-Löser XNS untersucht werden.

Multi-Physics-Code XNS

Das eingesetzte Simulationsframework basiert auf dem Multi-Physics-Code XNS, welcher schon 1996 von Marek Behr in Minnesota initiiert wurde. Die Abkürzung ergibt sich dabei aus dem „X“ als damals üblichen Präfix für ausführbare Dateien und „NS“ für die Navier-Stokes-Gleichungen, deren Lösung zur Untersuchung von Strömungsphänomenen ursprüngliches Ziel der Software gewesen ist. Auch heute noch ist dieser Fortran-Code für UNIX-basierte Systeme ausgelegt. Nach einem Refactoring unter Ausnutzung der maximal möglichen Objektorientierung gilt mittlerweile der Fortran2008-Standard. Auch wenn Fortran heutzutage ein sehr angestaubtes Image hat, ermöglicht diese Programmiersprache einen einfachen und insbesondere

schnellen Einstieg für eine breite Zielgruppe möglicher Mitarbeiterinnen und Mitarbeiter des Lehrstuhls, insbesondere für diejenigen ohne vertiefte Programmiererfahrung. So ist gewährleistet, dass diese neue Methoden schnell realisieren können, welche durch entsprechende Veröffentlichungen gegebenenfalls dann Berücksichtigung in komplexeren Softwaresystemen finden.

Das von Beginn an parallelisierte Tool hat sich über die Jahre zu einem Multi-Physik-Code entwickelt, dessen Kernentwicklerteam am Lehrstuhl für Computergestützte Analyse technischer Systeme angesiedelt ist. Beiträge des Forschungszentrums Jülich, der TU Wien, der Chuo University in Tokio, der Seoul National University, und anderen Forschungsgruppen demonstrieren die nationale und internationale Anerkennung der Softwareentwicklung. Um Möglichkeiten zur Zusammenarbeit mit anderen Forschungsgruppen zu schaffen, bietet die Software außerdem ein Multilinguales Interface zu Fortran, C, Matlab und Python. Dies erlaubt beispielsweise durch die mögliche Kopplung mit externen Strukturlösern die kollaborative Lösung von Problemen aus dem Bereich Fluid-Struktur-Interaktion.

Literatur

[1] von Danwitz, M., Karyofylli, V., Hosters, N., Behr, M., Simplex space-time meshes in compressible flow simulations. International Journal for Numerical Methods in Fluids, 91(1), 29-48, 2019

[2] Karyofylli, V., Wendling, L., Make, M., Hosters, N., Behr, M., Simplex space-time meshes in thermally coupled two-phase flow simulations of mold filling. Computers & Fluids, 192, 104261, 2019

[3] von Danwitz, M., Antony, P., Key, F., Hosters, N., Behr, M., Four-dimensional elastically deformed simplex space-time meshes for domains with time-variant topology. International Journal for Numerical Methods in Fluids, 93(12), 3490-3506, 2021

[4] von Danwitz, M., Voulis, I., Hosters, N., Behr, M., Time-continuous and time-discontinuous space-time finite elements for advection-diffusion problems. International Journal for Numerical Methods in Engineering, 124(14), 3117-3144, 2023

[5] Behr, M., Simplex space–time meshes in finite element simulations. International journal for numerical methods in fluids, 57(9), 1421-1434, 2008

Autoren

Univ.-Prof. Marek Behr, Ph.D., ist Inhaber des Lehrstuhls für Computergestützte Analyse technischer Systeme.

Dr.-Ing. Norbert Hosters ist Oberingenieur am Lehrstuhl für Computergestützte Analyse technischer Systeme.

Dr.-Ing. Fabian Key war wissenschaftlicher Mitarbeiter am Lehrstuhl für Computergestützte Analyse technischer Systeme und arbeitet als Post-Doc am Institut für Leichtbau und Strukturbiomechanik der TU Wien.

Matthias Meinke, Wolfgang Schröder

Analyse von Multiphysik-Problemen mit der Simulationssoftware m-AIA

Over the past 15 years, the m-AIA (multiphysics Aerodynamisches Institut Aachen) simulation library has been developed and today comprises over 400,000 lines of program code. The framework has been developed in close collaboration between RWTH's Institute of Aerodynamics, Forschungszentrum Jülich, and the High-Performance Computing Center (HLRS) at the University of Stuttgart. m-AIA is a multiphysics simulation code employed in numerous national and EU-funded projects to address engineering challenges with the help of simulations. It includes various solvers capable of predicting flows, heat conduction, combustion, aeroacoustics, and fluid-structure interactions.

Das Aerodynamische Institut und der Lehrstuhl für Strömungsmechanik haben umfangreiche Erfahrung in der Entwicklung und Anwendung von numerischen Methoden unter anderem in den Gebieten der numerischen Strömungsmechanik, Aeroakustik und Verbrennung. In den vergangenen 15 Jahren wurde die Simulationsbibliothek m-AIA (multiphysics Aerodynamisches Institut Aachen) mit mehr als 400.000 Zeilen Programmcode erstellt. Das Framework wird in enger Zusammenarbeit mit dem Forschungszentrum Jülich und dem High Performance Computing Center (HLRS) der Universität Stuttgart entwickelt. m-AIA ist ein Multiphysik-Simulationscode, der erfolgreich in zahlreichen nationalen und von der EU finanzierten Projekten für die Simulation von ingenieurtechnischen Problemen eingesetzt wurde. Er enthält verschiedene Löser, die für die Vorhersage von Strömungen, Wärmeleitung, Verbrennung, Aeroakustik und Fluid-Struktur-Interaktion verwendet werden können. Die numerischen Lösungsmethoden umfassen unter anderem:

- Finite-Volumen-Methoden zur Vorhersage von Strömungen und Wärmeleitung
- Lattice-Boltzmann-Methoden für Strömungen und Temperaturfelder bei niedriger Mach-Zahl
- Discontinous Galerkin-Methoden zur Vorhersage der Ausbreitung von Schallwellen
- Finite-Cell-Methoden für Strukturmechanik
- Level-Set-Methoden zur Verfolgung von Oberflächen
- Lagrange-Modelle für Spraymodelle und Verfolgung von Partikeln mit sechs Freiheitsgraden

Die meisten Löser sind für kartesische hierarchische Gitter formuliert, die den Vorteil einer vollautomatischen Gittergenerierung auch für komplexe Geometrien bieten[1]. Ein weiteres Merkmal des Simulations-Frameworks ist die Fähigkeit, Simulationen mit beliebig beweglichen Objekten durchzuführen. Aufgrund der adaptiven Gitterverfeinerung, die für eine genaue Auflösung lokaler Strömungsphänomene bei gleichzeitiger Reduktion der Anzahl der Gitterzellen essenziell ist, sind dynamische Lastverteilungstechniken erforderlich. Eine raumfüllende Kurve wird verwendet, um die Rechenlast auf Hochleistungsrechnersystemen gleichmäßig zu verteilen. Ein integrierter paralleler Gittergenerator erzeugt automatisch Gitter für Multiphysik-Domänen basierend auf STL-Definitionen der Domänengrenzen und der eingebetteten Geometrien. Alle Methoden verwenden eine hybride OpenMP/MPI-Parallelisierung. In letzter Zeit wurden Portierungsaktivitäten in Zusammenarbeit mit NVIDIA durchgeführt, um das Simulations-Framework mithilfe der parallelen Standard Template Library in neueren C++-Standards auf Grafikprozessoren zu implementieren. Das Paket m-AIA wird zukünftig als Open-Source-Software zur Verfügung gestellt.

Im Folgenden werden einige Lösungen von m-AIA näher diskutiert. In Bild 1 ist die Geometrie zusammen mit einigen Gitterdetails einer axialen Turbine mit 1,5 Stufen dargestellt[2, 3]. Das Gitter hat etwa 1 Milliarde Zellen und wird vollautomatisch generiert. Die Einströmung von heißem Gas in den Radseitenraum zwischen den Rotor- und Statorscheiben wird analysiert, was im Hinblick auf die Lebensdauer und die thermodynamische Effizienz von axialen Gasturbinen, die in Kraftwerken oder Flugzeugen verwendet werden, wichtig ist. Solche Simulationen erfordern aufgrund der sich zeitlich langsam entwickelnden Strömung im Radseitenraum erhebliche Rechenressourcen und können nur auf

Hochleistungsrechnersystemen wie dem HAWK-System am HLRS Stuttgart mit etwa 16.000 CPU-Kernen in einem vertretbaren Zeitrahmen durchgeführt werden. Um die Oberflächen der rotierenden Schaufeln zu verfolgen, wird eine Level-Set-Methode verwendet, bei der eine vorzeichenbehaftete Abstandsfunktion effizient die Positionen der im Gitter eingebetteten, rotierenden Oberflächen auf parallelen Rechensystemen bestimmt. Eine Cut-Cell-Formulierung stellt physikalisch korrekte Bedingungen in den kartesischen Gitterrandzellen sicher, und gewährleistet gleichzeitig die Erhaltung von Masse, Impuls und Energie[4].
Die Kopplung der unterschiedlichen Löser wird durch den Austausch von Termen innerhalb verschiedener Teilmengen eines gemeinsamen kartesischen Gitters implementiert. Dies ermöglicht eine effiziente Kopplung mit einem Datenaustausch zwischen den Lösungsverfahren ohne zusätzlichen Kommunikationsoverhead auf parallelen Systemen. Eine solche gekoppelte Methode wird verwendet, um aerodynamisch erzeugten Schall durch eine direkte hybride Strömungsmechanik/Aeroakustik-Methode vorherzusagen[5]. In diesem Fall wird ein Lösungsverfahren der Navier-Stokes-Gleichungen mit einer numerischen Methode zur Lösung der akustischen Störungsgleichungen gekoppelt. Die beiden Methoden sind durch akustische Quellterme miteinander verbunden, die aus der Lösung des turbulenten Strömungsfeldes bestimmt und in die akustischen Störungsgleichungen eingefügt werden. Die direkt gekoppelte hybride Strömungsmechanik/Aeroakustik-Methode erlaubt einen In-Memory-Transfer dieser Quellterme, sodass kein Disk-IO erforderlich ist, um die Daten zwischen den Lösungsverfahren zu übertragen. Diese gekoppelte Lösungsmethode wurde um Modellterme erweitert, um die Strömung in porösen Medien zu beschreiben. Anhand der numerischen Vorhersage kann das Potenzial von porösem Material zur Reduktion des an der Hinterkante von Tragflügeln erzeugten Lärms beurteilt werden.
Ein Beispiel für aerodynamisch erzeugten Lärm, der in vielen technischen Anwendungen, wie zum Beispiel Flugzeugen oder Windturbinen, auftritt, ist in Bild 2 dargestellt. Auf der linken Seite ist die Geometrie der verzahnten Hinterkante zu sehen, während auf der rechten Seite das Strömungs- und das Schallfeld abgebildet sind[6]. Die Lärmreduktion durch eine poröse gezackte Hinterkante wird anhand der direkten hybriden Strömungsmechanik/Aeroakustik-Methode bestimmt. Die Kombination aus gezackter

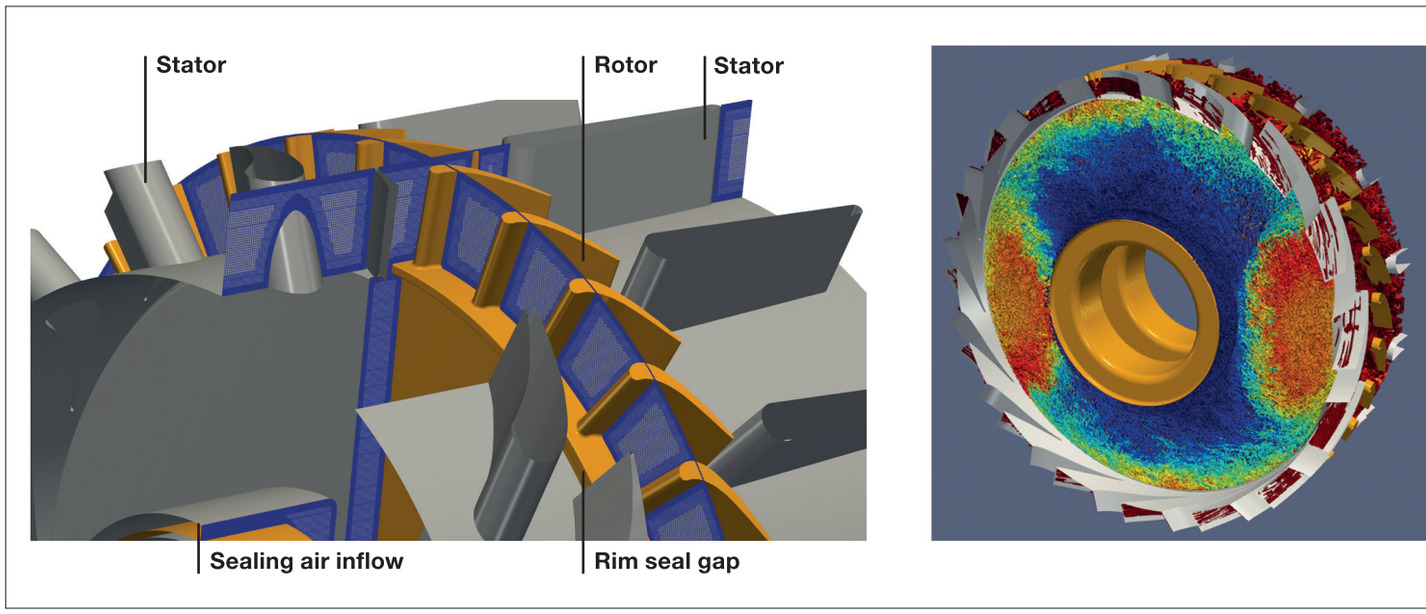


Bild 1: Kartesisches Gitter für eine 1,5-stufige Axialturbine (links) und Eintritt von heißem Gas in den Radseitenraum. Die Farbe zeigt die Konzentration des heißen Gases, wobei die Farbe Rot eine hohe und Blau eine niedrige Konzentration von Heißgas anzeigt (rechts).

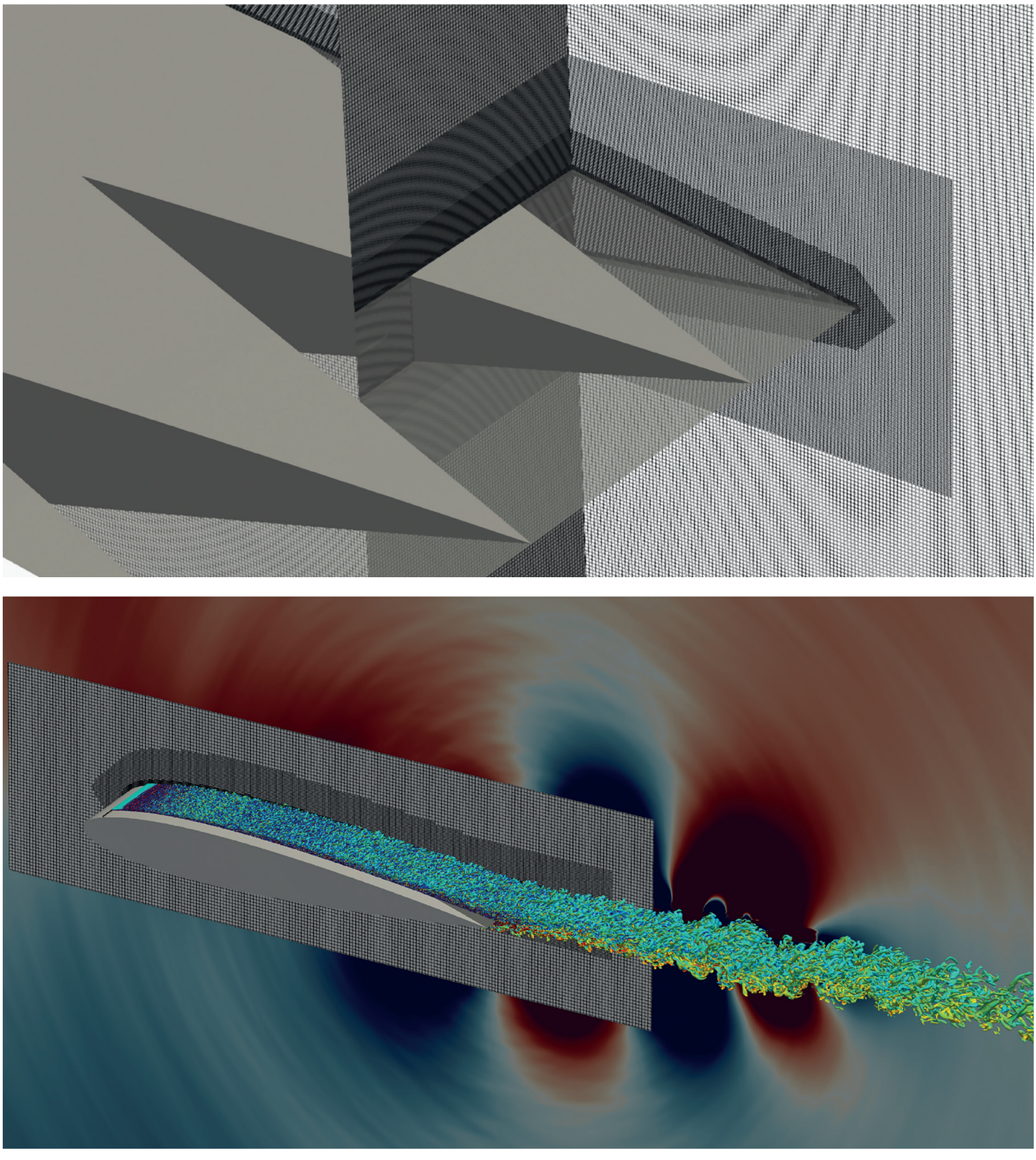


Bild 2: Aeroakustische Analyse des durch eine poröse zackige Hinterkante eines Tragflügelprofils erzeugten Schallfeldes. Die Geometrie der Hinterkante ist links dargestellt. Rechts erkennt man die turbulenten Strukturen in der Grenzschicht und die erzeugten Schallwellen in einer Ebene parallel zur Strömungsrichtung.

Hinterkante und porösem Material kann insbesondere für niedrige und mittlere Frequenzen den abgestrahlten Lärm um mehrere dB reduzieren. Eine Anwendung von m-AIA für Mehrphasenströmungen wird in Bild 3 dargestellt, wo der Energie- und Impulsaustausch einer großen Anzahl frei beweglicher, ellipsoider Partikel in isotroper Turbulenz simuliert wird[7]. Das Verhalten derartiger Partikelgeometrien in turbulenten Strömungen ist im Kontext der Verbrennung von Biomasse wichtig, die die klassische Kohle ersetzt. Da Biomasse größtenteils aus Fasern besteht, interagiert sie auf unterschiedliche Weise mit Turbulenz, was in den durchgeführten Simulationen analysiert wird. Das Ziel dieser Untersuchungen mit detaillierten Simulationen besteht darin, reduzierte Modellformulierungen zu bestimmen, die in Vorhersagen für vollständige Biomasseverbrennungskammern verwendet werden können.

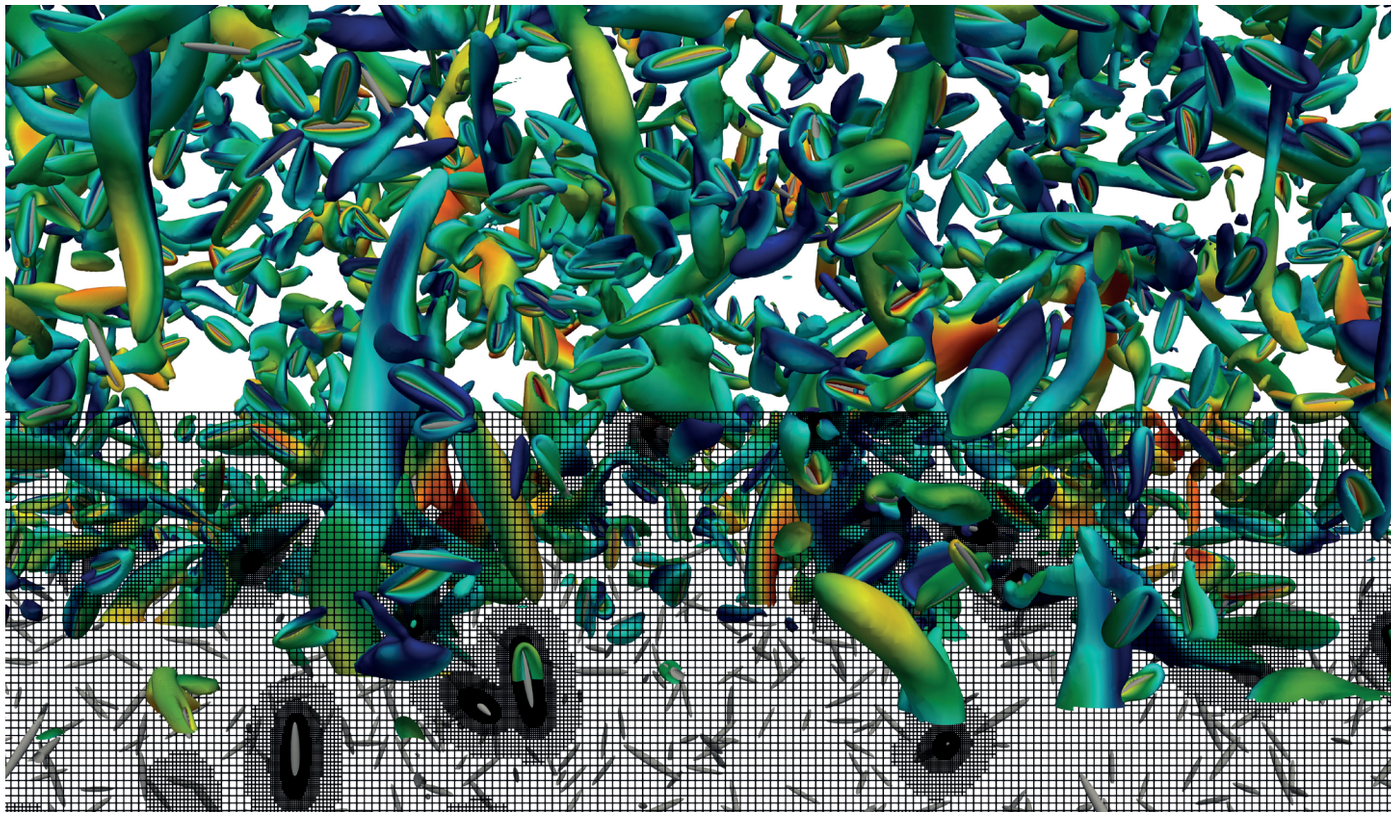


Bild 3: Abnahme isotroper Turbulenz mit etwa 60.000 frei beweglichen, vollständig aufgelösten ellipsoiden Partikeln. Die Isoflächen visualisieren die turbulenten Wirbelstrukturen, die farblich mit der lokalen Mach-Zahl codiert sind. Das adaptiv verfeinerte kartesische Gitter um die Partikel ist in der unteren Hälfte der Abbildung dargestellt.

Literatur

[1] Lintermann, A., Meinke, M., Schröder, W., Zonal Flow Solver (ZFS): A highly efficient multi-physics simulation framework, International Journal of Computational Fluid Dynamics 34, https://doi.org/10.1080/10618562.2020.1742328, 2020

[2] Pogorelov, A., Meinke, M., Schröder W., Cut-Cell Method Based Large-Eddy Simulation of Tip-Leakage Flow, Physics of Fluids, 27(7), 075106, https://doi.org/10.1063/1.4926515, 2015

[3] Hösgen, Th., Meinke, M., Schröder, W., Analysis of Single Blade Passage and Full Circumference Large-Eddy Simulations of Turbine Rim Seal Flows, Journal of Turbomachinery, https://doi.org/10.1115/GT2023-101688, 2023

[4] Schneiders, L., Günther, C., Meinke, M., Schröder, W., An efficient conservative cut-cell method for rigid bodies interacting with viscous compressible flows, Journal of computational physics, 311:62-86, https://doi.org/10.1016/j.jcp.2016.01.026, 2016

[5] Niemöller, A., Schlottke-Lakemper, M., Meinke, M., Schröder, W., Dynamic Load Balancing for Direct-Coupled Multiphysics Simulations, Computers & Fluids 199(2):104437, https://doi.org/10.1016/j.compfluid.2020.104437, 2020

[6] Satcunanathan, S., Meinke, M., Schröder, W., Impact of porous media on boundary layer turbulence, Fluids 2022, 7(4), 139, https://doi.org/10.3390/fluids7040139, 2022

[7] Schneiders, L., Fröhlich, K., Meinke, M., Schröder, W., The decay of isotropic turbulence carrying non-spherical finite-size particles, Journal of fluid mechanics, 875:520-542, https://doi.org/10.1017/jfm.2019.516, 2019

Autoren

Dr.-Ing. Matthias Meinke leitet die Numerische Abteilung des Aerodynamischen Instituts.

Univ.-Prof. Dr.-Ing. Wolfgang Schröder ist Inhaber des Lehrstuhls für Strömungslehre und Leiter des Aerodynamischen Instituts.

Daniel Döhring, Michael Schlottke-Lakemper

Höchstleistungs-rechnen leicht(er) gemacht

Nachhaltige wissenschaftliche Softwareentwicklung mit Trixi.jl

Modern scientific software development faces the challenge of reconciling complexity in models and methods with diversified high-performance computing hardware. Heterogeneous supercomputing systems are crucial for energy-efficient scientific computing, but challenge the development process from prototype to production code. To remedy this, the software package Trixi.jl aims to strike a balance between user-friendliness and high performance. By combining a modular architecture with the beneficial properties of the Julia programming language, it facilitates collaboration and transparency in the software development process. Such novel approaches to research software engineering will be essential for a sustainable use of resources in the future.

Wissenschaftliche Softwareentwicklung steht vor der ständigen Herausforderung, moderne Simulationsmethoden mit immer größeren und vielfältigeren Rechnerarchitekturen zu verknüpfen. Die Komplexität der verwendeten Modelle zur Beschreibung realer Prozesse in Physik oder Ingenieurwissenschaften, sowie die numerischen Verfahren, die zur Lösung dieser Modelle eingesetzt werden, erfordern tiefgreifendes Wissen in einer zunehmenden Zahl von Fachdisziplinen. Dies trifft insbesondere auf die Simulation von Multiphysik-Systemen zu, bei denen mehrere gekoppelte physikalische Systeme untersucht werden. Beispiele hierfür sind gasdynamische Prozesse unter dem Einfluss von Gravitation, wie sie in der Astrophysik vorkommen, Plasmasysteme im erdnahen Weltraum oder technische Strömungen, bei denen zudem die Lärmentwicklung bestimmt werden muss. Gleichzeitig gewinnen hybride Höchstleistungsrechner im Vergleich zu klassischen Hardwarekonfigurationen zunehmend an Bedeutung. Dabei kommen zusätzlich zu den Hauptprozessoren auch besonders spezialisierte Beschleunigungskomponenten wie Grafikkarten zum Einsatz. Prozessoren und Grafikkarten eignen sich jedoch unterschiedlich gut für verschiedene Problemstellungen. Umfangreiche Kenntnisse über die jeweiligen Hard- und Softwaresysteme sowie über die Eigenschaften der simulierten Systeme werden für schnelles und energieeffizientes wissenschaftliches Rechnen also immer wichtiger. Aufgrund dieser Vielzahl an Herausforderungen existiert typischerweise eine deutliche Lücke im Bereich des wissenschaftlichen Rechnens zwischen experimentellen Software-Implementierungen und Simulationsprogrammen für den produktiven Einsatz bei realen Problemen. Viele kleinere bis mittelgroße Softwareprojekte werden für experimentelle Untersuchungen von mathematischen Modellen oder numerischen Methoden

Bild 1: Schnelles Prototyping mit Trixi.jl: Frische Ideen können einfach implementiert und direkt auf dem Hochleistungsrechner zum Einsatz gebracht werden.
Foto: Peter Winandy

erstellt. Diese Codes wurden entwickelt, um möglichst einfach einzelne Aspekte der Implementierung zu verändern oder komplett auszutauschen, etwa um neue Modelle hinzuzufügen oder um Methoden zu evaluieren. Im Gegensatz dazu stehen Forschungscodes für den produktiven Praxiseinsatz, die einen Fokus auf schnelle Ausführung und große Datenmengen legen. Diese Programme haben typischerweise eine enge Kopplung zwischen den Modellierungsaspekten, den Details der numerischen Methoden sowie den geschwindigkeitsrelevanten Teilen der Implementierung. Die enge Verzahnung macht es deutlich aufwendiger, einzelne Komponenten des Codes auszutauschen oder mit neuen Verfahren zu experimentieren. Es existiert also auf der einen Seite der Wunsch nach einfachen Möglichkeiten zur experimentellen Evaluierung neuartiger Modelle oder Algorithmen, während andererseits die Fähigkeit zur Durchführung großskaliger Simulationen auf komplexen Hardwaresystemen notwendig ist.

Mit diesen Herausforderungen im Blick ging 2020 das Softwareprojekt Trixi.jl[1-3] an den Start, welches von rund 20 Wissenschaftlerinnen und Wissenschaftlern in Europa und den USA entwickelt wird. Hierbei ist auch der Lehrstuhl für Angewandte und Computergestützte Mathematik (ACoM) beteiligt. Trixi.jl ist ein Software-Framework für die adaptive numerische Simulation[1, 3, 5] von strömungsmechanischen Fragestellungen und Multiphysik-Systemen, siehe Bild 3. Adaptiv bedeutet hier, dass die verwendeten Modelle und Methoden, wie etwa die Auflösung des Rechengitters, dynamisch während der Simulation verändert werden, um jederzeit einen optimalen Ressourceneinsatz sicherzustellen. Trixi.jl wird vor allem für die Forschung im Bereich neuartiger numerischer Verfahren und effizienter Algorithmen genutzt[4].

Der Hauptfokus bei der Entwicklung von Trixi.jl liegt auf einer einfachen Verwendungsfähigkeit für neue oder unerfahrene Nutzende, während gleichzeitig hohe Rechenleistungen für Produktionssimulationen möglich sind. Trotzdem erhält Trixi.jl die einfache Erweiterbarkeit, die erforderlich ist, um neue Modelle oder Methoden im Rahmen des Forschungsprozesses schnell umsetzen und evaluieren zu können. Erreicht werden diese Ziele vor allem aufgrund der folgenden drei Maßnahmen: Die Verwendung der modernen Programmiersprache Julia, eine von Grund auf modular konzipierte Softwarearchitektur, ein Entwicklungsprozess, der konsequent auf einen unkomplizierten Einstieg für unerfahrene Nutzende und die zügige Realisierbarkeit neuer Projekte ausgerichtet ist.

Die Programmiersprache Julia wurde als leicht zu erlernende Sprache für wissenschaftliche Forschungssoftware konzipiert. Sie verbindet ein umfangreiches, modulares

Bild 2: Brainstorming zu „Continuous Visualization“, also automatisierte on-the-fly Visualisierung der Simulationsergebnisse von Trixi.jl.
Foto: Peter Winandy


Cx
Soennecken

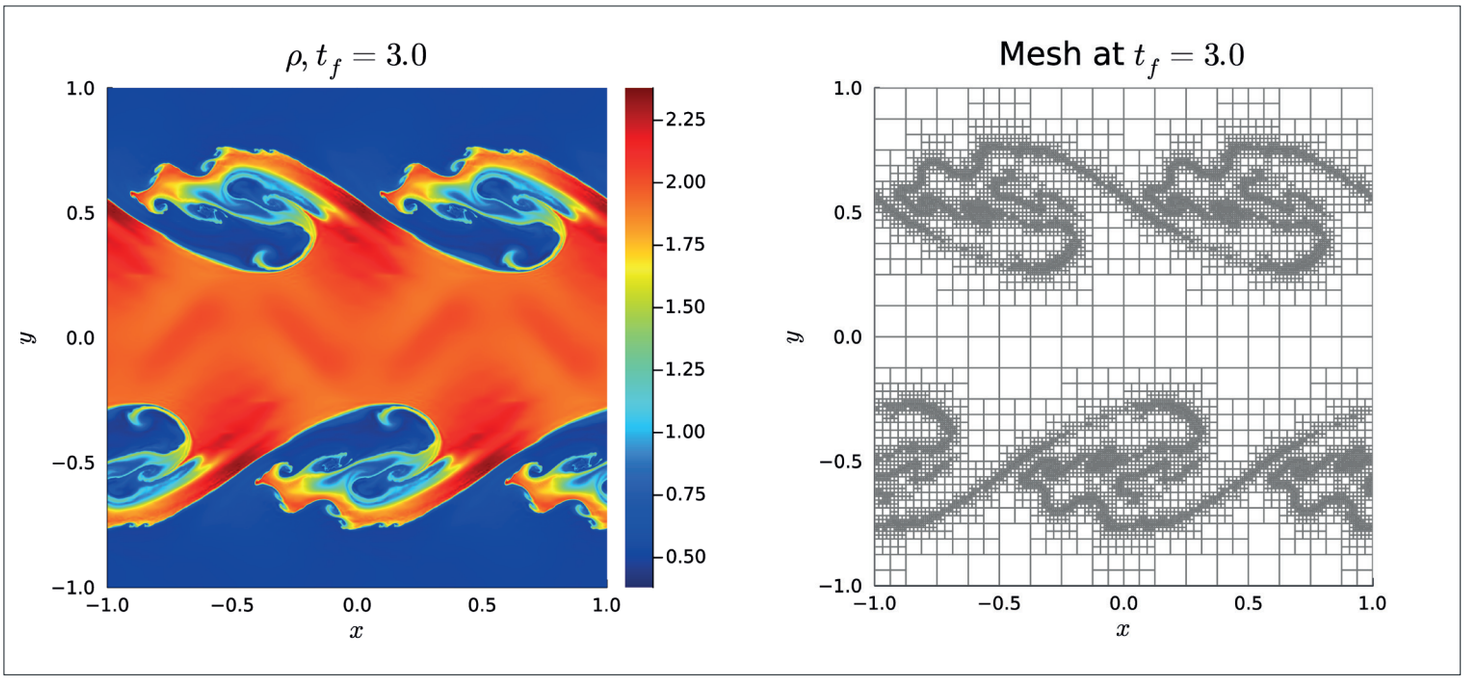


Bild 3: Berechnung einer Kelvin-Helmholtz-Instabilität mit Trixi.jl[6]. Dargestellt ist das Dichtefeld der simulierten Strömung (links). Das Rechengitter wird in Bereichen großer Dichteänderung dynamisch verfeinert (rechts).

Datentypensystem mit einer hohen Geschwindigkeit, wodurch sie auch für die Ausführung auf Höchstleistungsrechnern geeignet ist, siehe Bild 4. Aus diesem Grund lässt sich Julia von der explorativen Phase bis hin zur Produktionsphase während des gesamten Forschungszyklus sinnvoll einsetzen. Damit können Wissenschaftlerinnen und Wissenschaftler flexibel zwischen den Phasen wechseln, ohne dass unterschiedliche Softwarecodes oder Entwicklungsumgebungen zum Einsatz kommen müssen. Dies spart Zeit, vermeidet Fehler bei der Portierung und führt zu nachhaltigerer Software von höherer Qualität.

Um die Integration von neuen Fähigkeiten auch langfristig zu ermöglichen, wurde Trixi.jl als Programmbibliothek mit weitgehend unabhängigen, jedoch zueinander kompatiblen Bausteinen entworfen. So stellen das physikalische Modell, das numerische Lösungsverfahren oder das zugehörige Rechengitter jeweils eigenständige Komponenten dar. Neben diesen Grundfunktionen werden auch fortgeschrittene Features separat betrachtet und implementiert, wie etwa Dateiausgaberoutinen, die dynamische Anpassung des Rechengitters oder die Visualisierung der Simulationsergebnisse während der Laufzeit. Dieser konsequente Modularisierungsansatz ermöglicht es einerseits, dass Entwicklerinnen und Entwickler mit geringem Aufwand neue Funktionen hinzufügen können ohne zugleich komplexe Interaktionen mit anderen

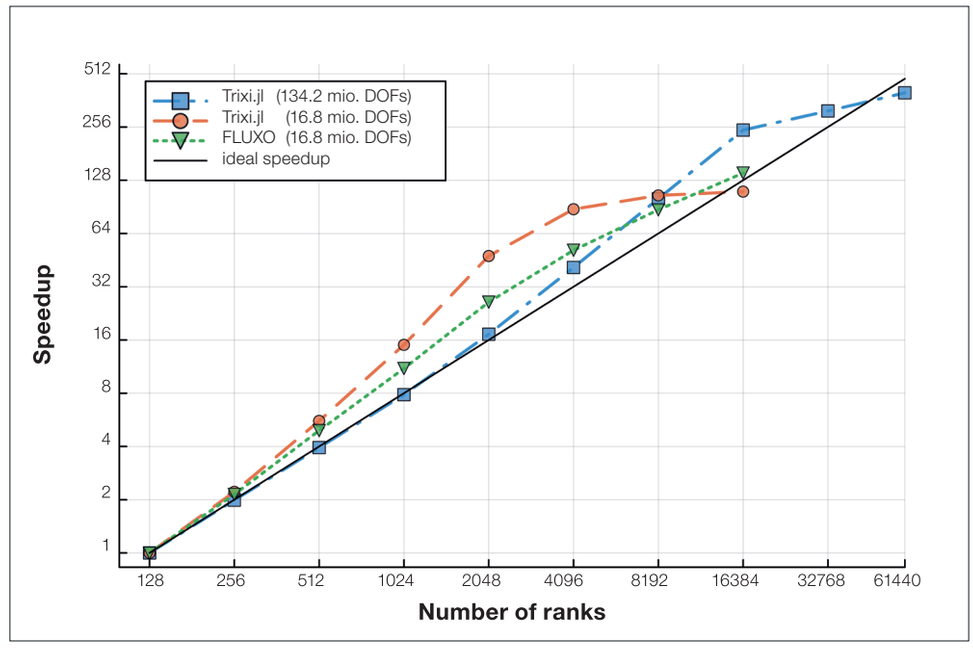


Bild 4: Parallele Skalierung einer Trixi.jl-Simulation auf JURECA, einem Hochleistungsrechner am Forschungszentrum Jülich, im Vergleich zu einem klassischen, in Fortran geschriebenen Simulationscode FLUXO. Höhere Werte zeigen eine bessere Laufzeiteffizienz an.

Programmteilen beachten zu müssen. Auch die Integration von bestehenden Softwaremodulen, die in anderen Forschungsgruppen entwickelt wurden, wird so deutlich vereinfacht. Andererseits verbessert die Modularisierung auch die Anwendungsfreundlichkeit der Software, da nicht verwendete Funktionalitäten zunächst ausgeblendet werden können.

Als Open-Source-Software ist der Quellcode sowie der Entwicklungsprozess von Trixi.jl öffentlich einsehbar. Um Interessierten den Einstieg so leicht wie möglich zu machen, werden darüber hinaus Ressourcen für die unkomplizierte Einbindung neuer Nutzerinnen und Nutzer mit unterschiedlichem Erfahrungsstand oder fachlichem Hintergrund bereitgestellt. Neben umfangreicher Dokumentation, die sowohl technische Aspekte als auch die zugrunde liegenden Modelle und Methoden umfasst, stehen Videoanleitungen zu Verfügung. Ein Katalog von Tutorials er-

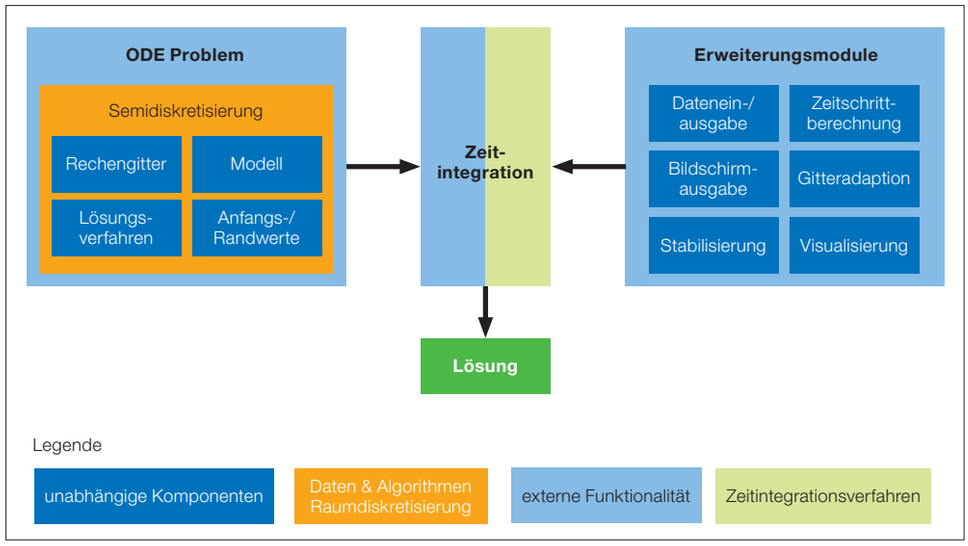


Bild 5: Modulare Softwarearchitektur von Trixi.jl.

laubt es, sich viele Themen von den Grundlagen zur Verwendung bis hin zu fortgeschrittenen Fragen bei der Weiterentwicklung anzueignen. Zudem gibt es ein chatbasiertes Forum, in dem bei der Einarbeitung in die Software oder bei der Umsetzung von eigenen Projektideen unterstützt wird.

Um bei einem heterogenen Entwicklungsteam dauerhaft eine hohe Qualität der Software zu gewährleisten, wird auf ein strukturiertes Peer-Review-System gesetzt. Neue Erweiterungen der Funktionalität werden zunächst einer Vielzahl automatisierter Tests unterzogen, um die korrekte Funktionsweise und die effiziente Ausführung sicherzustellen. Anschließend werden die Code-Änderungen begutachtet und mögliche Verbesserungen gemeinsam mit den Autoren erarbeitet.

Dieser Prozess ermöglicht es auch Wissenschaftlerinnen und Wissenschaftlern mit weniger Erfahrung, schnell sinnvolle Beiträge zu liefern. So wurden bereits über 15 Abschlussarbeiten – von der Bachelorarbeit bis zur Dissertation – mit Trixi.jl erfolgreich durchgeführt. Durch die Bereitstellung von entsprechenden Software-Repositorien wird bei Publikationen zudem eine einfache und robuste Reproduzierbarkeit der mit Hilfe von Trixi.jl gewonnenen Ergebnisse sichergestellt. Dies wiederum eröffnet neue Möglichkeiten der Kooperation mit anderen Forschungsgruppen, da es die Hürden für eine gemeinsame Nutzung von Softwarepaketen senkt.

Die Kombination von umfassender Erweiterbarkeit, einfacher Nutzbarkeit und hoher Effizienz stellt eine anspruchsvolle Herausforderung für Forschungssoftware dar. Gleichzeitig bildet sie jedoch eine wichtige Voraussetzung dafür, dass trotz steigender Komplexität bei den Modellen und Methoden die heterogener werdenden Höchstleistungsrechner für ein breites wissenschaftliches Publikum nutzbar sind. Um dieses Ziel zu erreichen, müssen in der wissenschaftlichen Softwareentwicklung neue Wege beschritten und das Research Software Engineering, also die professionalisierte Softwareentwicklung im Forschungskontext, deutlich stärker in den Vordergrund gerückt werden. Das Research Software Engineering unterstützt den nachhaltigen Einsatz von Ressourcen, indem es eine Brücke schlägt zwischen dem Expertenwissen in den wissenschaftlichen Fachdisziplinen und modernen Softwareentwicklungspraktiken. Die Verbindung von Anwendungsfreundlichkeit und Spitzenleistung in der wissenschaftlichen Softwareentwicklung ist dabei ein wichtiger Faktor und bleibt ein herausforderndes Forschungs- und Entwicklungsziel.

Literatur

[1] Schlottke-Lakemper, M., Gassner, G., Ranocha, H., Winters, A. R., Chan, J., Trixi.jl: Adaptive high-order numerical simulations of hyperbolic PDEs in Julia. Code Repository, 2021, https://github.com/trixi-framework/Trixi.jl, doi: 10.5281/zenodo.3996439

[2] Schlottke-Lakemper, M., Winters, A. R., Ranocha, H., Gassner, G., A purely hyperbolic discontinuous Galerkin approach for self-gravitating gas dynamics,Journal of Computational Physics 442, 2021, doi: 10.1016/j.jcp.2021.110467

[3] Ranocha, H., Schlottke-Lakemper, M., Winters, A. R., Faulhaber, E., Chan, J., Gassner, G., Adaptive numerical simulations with Trixi.jl: A case study of Julia for scientific computing, Proceedings of the JuliaCon Conferences 1, 2022, doi: 10.48550/arXiv.2102.06017

[4] Ranocha, H., Schlottke-Lakemper, M., Chan, J., Rueda-Ramírez, A. M., Winters, A. R., Hindenlang, F., Gassner, G. J., Efficient implementation of modern entropy stable and kinetic energy preserving discontinuous Galerkin methods for conservation laws,ACM Trans. Math. Softw., 2023, doi: 10.1145/3625559

[5] Ranocha, H., Winters, A. R., Castro, H. G., Dalcin, L., Schlottke-Lakemper, M., Gassner, G. J., Parsani M., On Error-Based Step Size Control for Discontinuous Galerkin Methods for Compressible Fluid Dynamics,Commun. Appl. Math. Comput.,2023, doi: 10.1007/s42967-023-00264-y

[6] Doehring, D., Schlottke-Lakemper, M., Gassner, G. J., Torrilhon, M., Multirate Time-Integration based on Dynamic ODE Partitioning through Adaptively Refined Meshes for Compressible Fluid Dynamics, Submitted to Journal of Computational Physics, 2024, doi: 10.48550/arXiv.2403.05144

Autoren

Daniel Döhring, M.Sc., ist wissenschaftlicher Mitarbeiter am Lehrstuhl für Angewandte und Computergestützte Mathematik.
Prof. Dr.-Ing. Michael Schlottke-Lakemper war Professurvertreter für Computational Mathematics an der RWTH Aachen.
Seit 1. Juni 2024 ist er Professor für High-Performance Scientific Computing an der Universität Augsburg.

Marco Davidovic, Fabian Fröde, Michael Gauding, Terence Lehmann, Heinz Pitsch

Mit CIAO zu effizienten und emissionsarmen Energiesystemen

CIAO is a massive-parallel high-fidelity solver for the reacting and non-reacting Navier-Stokes equations. It provides advanced numerical methods to accurately model turbulence and chemistry using both direct numerical and large-eddy simulations. With exceptional scalability on supercomputers, CIAO facilitates investigations into turbulent and laminar flames, multiphase flows, and nano-particle formation. The code's versatility extends to simulations in combustion engines, gas turbines, and industrial burners, making it a valuable tool for developing novel combustion models and advancing energy systems toward carbon-free energy carriers.

Die Navier-Stokes-Gleichungen, die das Verhalten von Fluidströmungen beschreiben, spielen eine zentrale Rolle bei der Gestaltung unseres Alltagslebens. Obwohl diese Gleichungen wie ein komplexes mathematisches Konstrukt erscheinen, zeigt sich ihr Einfluss in einer Vielzahl praktischer Anwendungen. Von der Vorhersage von Wettermustern, über den Blutkreislauf des Körpers bis hin zum Entwurf effizienter Energie- und Transportsysteme dienen die Navier-Stokes-Gleichungen als Eckpfeiler für das Verständnis und die Beeinflussung der Dynamik von Fluiden. Dieses Verständnis ist für die notwendige schnelle Transformation von fossilen, hin zu effizienten, kohlenstofffreien und emissionsarmen Energiesystemen essenziell. Für diese Problemstellung wird am Institut für Technische Verbrennung der Simulationscode CIAO entwickelt. CIAO steht für Compressible/ Incompressible Advanced Reactive Turbulent Simulations with Overset und ist ein vielsei-

Bild 1: Untersuchung zur emissionsarmen Verbrennung: Experimente am Brennerprüfstand des Instituts für Technische Verbrennung.
Foto: Peter Winandy

tiger Navier-Stokes-Löser für die Simulation laminarer und turbulenter reagierender Strömungen in komplexen Geometrien. Die Entwicklung erstreckt sich über mehr als ein Jahrzehnt und wird von einem Team unter der Leitung von Heinz Pitsch in enger Zusammenarbeit mit Expertinnen und Experten an verschiedenen internationalen Universitäten vorangetrieben.

Numerische Lösungen der Navier-Stokes-Gleichungen

Die numerische Lösung der reaktiven Navier-Stokes-Gleichungen stellt hohe Anforderungen an die verwendeten Methoden. Verschiedene Lösungsstrategien haben sich daher entwickelt, um diesen Herausforderungen gerecht zu werden. Eine dieser Strategien ist die direkte numerische Simulation (DNS), die ohne Modellierung der Turbulenz auskommt. Sie löst sämtliche relevanten chemischen und turbulenten Strukturen und Prozesse unmittelbar auf dem genutzten Rechengitter. Bei reaktiven Strömungen beinhaltet dies auch die direkte Lösung der Chemie, die in der Regel mithilfe eines reduzierten Reaktionsmechanismus dargestellt wird. Dieser Ansatz ermöglicht eine nahezu exakte Simulation der Strömung und wird daher auch als numerisches Experiment bezeichnet. Die erzielten Daten gewähren einzigartige Einblicke in die physikalischen Prozesse reaktiver Strömungen und sind unverzichtbar für die Entwicklung innovativer Verbrennungsmodelle, insbesondere für die Simulation von Wasserstoffflammen[1]. Jedoch erfordert die Durchführung direkter numerischer Simulationen erhebliche Rechenressourcen und ist selbst für Laborflammen nur auf den leistungsfähigsten Supercomputern unter Einsatz effizienter Parallelisierungsstrategien möglich. Gleichzeitig sind präzise und robuste numerische Lösungsmethoden erforderlich, um die kleinsten chemischen und turbulenten Strukturen genau zu berechnen. Die Herausforderung besteht darin, die numerische Diffusion zu minimieren, ohne künstliche Oszillationen der Lösung in Regionen mit hohen Gradienten zu verursachen. Zu diesem Zweck werden in CIAO Finite-Differenzen-Verfahren höherer Ordnung und halb-implizite Methoden eingesetzt. Aus Gründen der Stabilität erfolgt die Berechnung des nicht-linearen Konvektionsterms skalarer Größen, wie den Massenbrüchen und der Temperatur, mittels gewichteter wesentlich nicht-oszillierende Verfahren (WENO)[2].

Eine kostengünstigere Alternative zur direkten numerischen Simulation stellen skalenauflösende Simulationstechniken, wie zum Beispiel die Large-Eddy-Simulation (LES) dar. Hierbei werden die großen Skalen direkt berechnet, während die kleineren Skalen modelliert werden. Die kleineren Skalen zeigen in turbulenten Strömungen häufig ein universelles Verhalten und sind daher durch relativ einfache

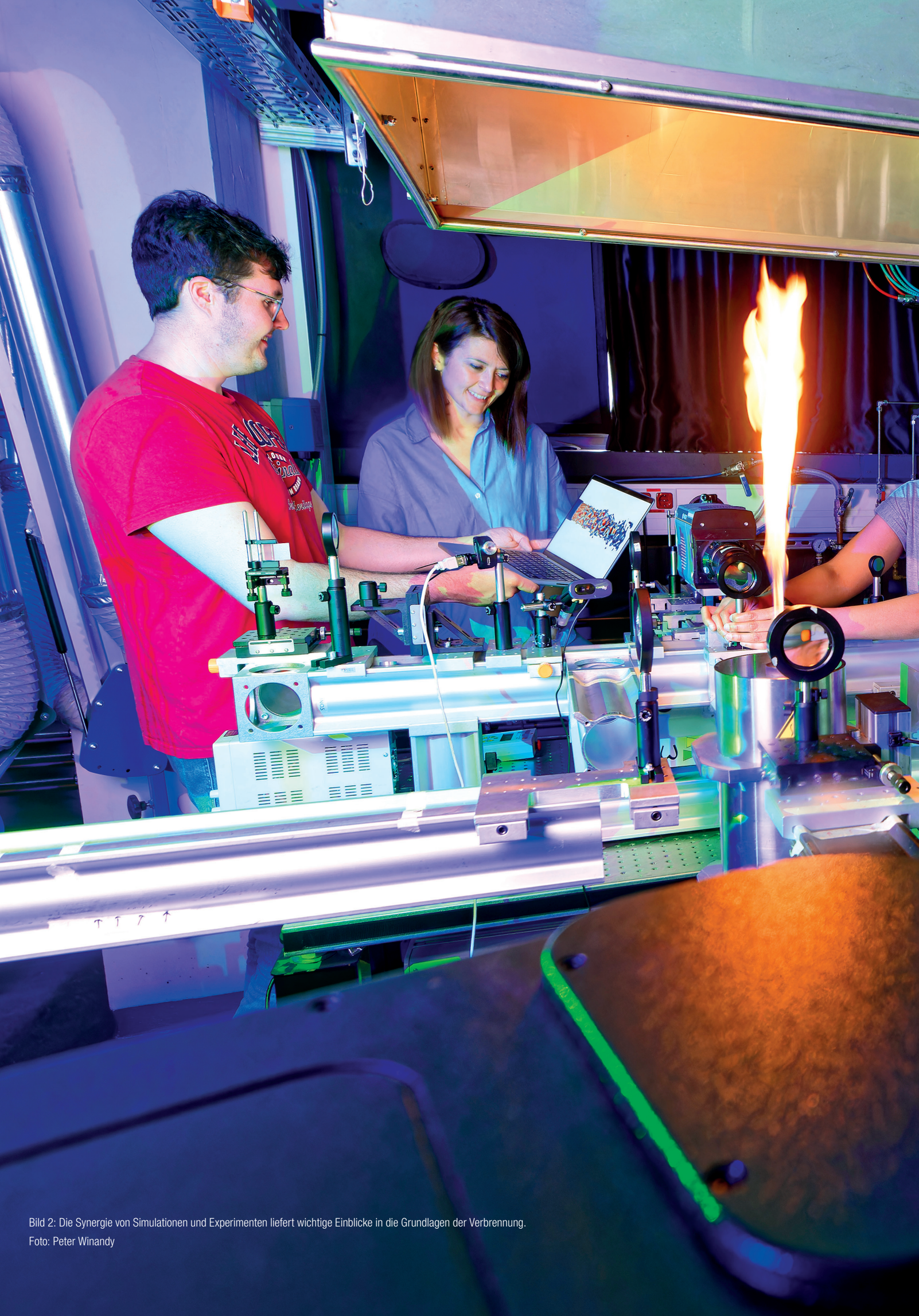


Bild 2: Die Synergie von Simulationen und Experimenten liefert wichtige Einblicke in die Grundlagen der Verbrennung.
Foto: Peter Winandy

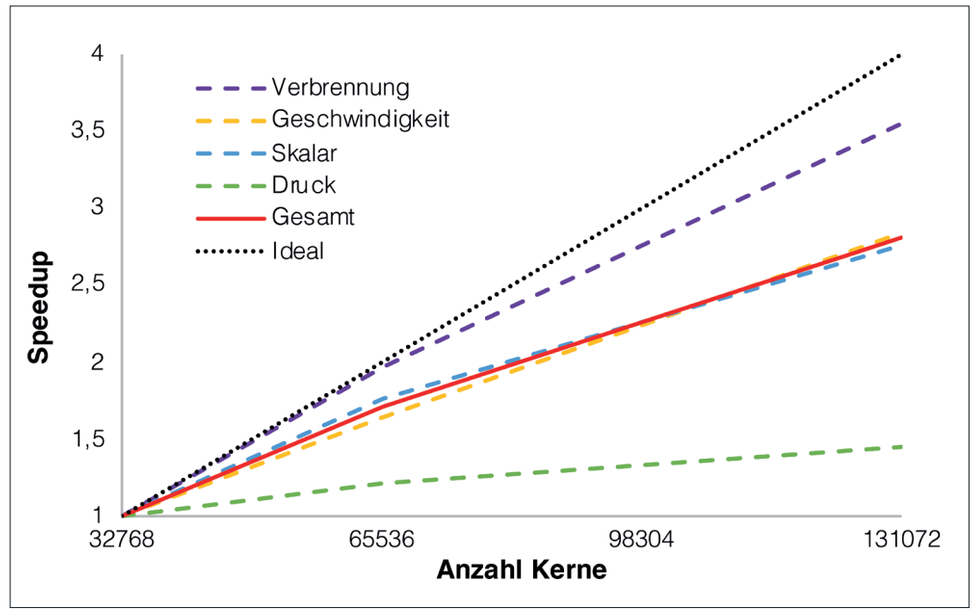


Bild 3: Skalierung von CIAO und seiner zentralen Bestandteile auf dem Supercomputer SuperMUC-NG am Leibniz-Rechenzentrum der Bayerischen Akademie der Wissenschaften in Garching.

Modelle darstellbar. Dieser Ansatz reduziert die notwendigen Rechenressourcen erheblich und erlaubt die Simulation komplexer Strömungen[3]. Dennoch sind Large-Eddy-Simulationen reaktiver Strömungen weiterhin Gegenstand der aktuellen Forschung. Die turbulenten Transportprozesse in reaktiven Strömungen sind hochkomplex und hochgradig nicht-universell, was die Modellierung erschwert. Bei wasserstoffhaltigen Brennstoffen kann es beispielsweise lokal zu differentieller Diffusion kommen, da das Wasserstoffmolekül im Vergleich zur Temperatur eine höhere Diffusionsgeschwindigkeiten aufweist. Dieser Vorgang hat eine komplexe Abhängigkeit von der Krümmung und Streckung der Flammenoberfläche und kann in mageren Gemischen zu Flammeninstabilitäten führen, die die Flammenausbreitungsgeschwindigkeit um ein Vielfaches erhöhen. Neben klassischen LES-Modellen für die turbulente Strömung verfügt CIAO über unterschiedliche Ansätze zur Modellierung der turbulenten Verbrennung. Ein weit verbreiteter Ansatz sind Flamelet-Modelle mit einer Tabellierung des gefilterten chemischen Quellterms in Abhängigkeit weniger transportierter Größen. Damit kann zum einen die Anzahl der zu transportierenden Skalare reduziert und zum anderen die Sub-Filter-Verteilung des Quellterms berücksichtig werden. Hier verfügt CIAO über erweiterte Modelle, die differentielle Diffusion und Thermodiffusion mittels mischungsgewichteter Ansätze berücksichtigen und die auf Transportgleichungen für den Mischungsbruch und der Fortschrittsvariable basieren. Studien von Wasserstoffflammen mit unterschiedlicher Krümmung und Streckung konnten eine exzellente Übereinstimmung der Modelle mit Experimenten und DNS nachweisen.

Massiv-parallele Simulationen mit CIAO

CIAO ist ein massiv-paralleler, hochgenauer Finite-Differenzen-Code. Im Rahmen des Codes steht ein vollständig kompressibler Löser zur Verfügung, der ein explizites fünfstufiges Runge-Kutta-Verfahren für die Zeitintegration verwendet. Für den inkompressiblen Löser stehen darüber hinaus eine semi-implizite Crank-Nicolson-Zeitintegration mit einem iterativen Prädiktor-Korrektor-Schema zur Verfügung. Zur Verbesserung der numerischen Genauigkeit werden räumliche und zeitliche versetzte Gitter eingesetzt. Die Poisson-Gleichung für den Druck wird mithilfe des Multi-Grid-HYPRE-Solvers gelöst, während für die Gleichungen für Temperatur und Spezies ein WENO-Schema verwendet wird, um Oszillation der Lösung zu verhindern. Für eine detaillierte Betrachtung der Chemie wird zusätzlich der symmetrische Operator-Splitting-Ansatz nach Strang verwendet. Der chemische Quellterm wird dabei mit dem Differentialgleichungslöser CVODE gelöst, der Teil von SUNDIALS ist. Alternativ steht eine institutseigene Bibliothek zur Lösung des Quellterms auf Graphikkarten zur Verfügung. CIAO ist größtenteils in Fortran programmiert, zur Datenspeicherung kommt HDF5 (Hierarchical Data Format) als standardisiertes Dateiformat zum Einsatz. HDF5 nutzt das parallele Dateisystem und erreicht beispielsweise auf dem Supercomputer SuperMUC-NG am Leibniz-Rechenzentrum der Bayerischen Akademie der Wissenschaften in Garching mit 3072 Rechenknoten Lese- und Schreibraten von mehr als 200 GB pro Sekunde.

Unabdingbar für die Simulation von multiphysikalischen Mehrskalenproblemen ist eine effiziente Parallelisierung, um moderne Supercomputer nutzen zu können. CIAO ist vollständig mittels MPI (Message-Passing Interface) parallelisiert und hat seine sehr gute Skalierbarkeit auf den schnellsten tier-0 Supercomputern bewiesen. Die Skalierung von CIAO für einen exemplarischen Fall einer turbulenten Wasserstoffflamme ist in Bild 3 gezeigt. Die Operationen mit dem größten Rechenaufwand ergeben sich aus der Lösung der elliptischen Druckgleichung, dem Löser für den Skalartransport und dem Löser für den chemischen Quellterm. Letzterer macht etwa 75 Prozent der gesamten Rechenzeit aus. Da der Chemie-Löser CVODE keinen Informationsaustausch zwischen Prozessen erfordert, wird dennoch eine gute Skalierbarkeit beobachtet. Die größten Simulationen mit CIAO von turbulenten vorgemischten Wasserstoffflammen umfassen mehr als 10 Milliarden Gitterpunkten und nutzen reduzierte kinetische Mechanismen mit 10 chemischen Komponenten und 21 Reaktionen. Die erforderliche Rechenzeit hierfür beträgt etwa 100 Millionen Core-Stunden, oder anders formuliert etwa 11.400 Core-Jahre. CIAO ist Mitglied des High-Q-Clubs, einer Gruppe von Codes, die für ihre außergewöhnliche Skalierbarkeit auf Supercomputern wie JUQUEEN am Forschungszentrum Jülich bekannt sind.

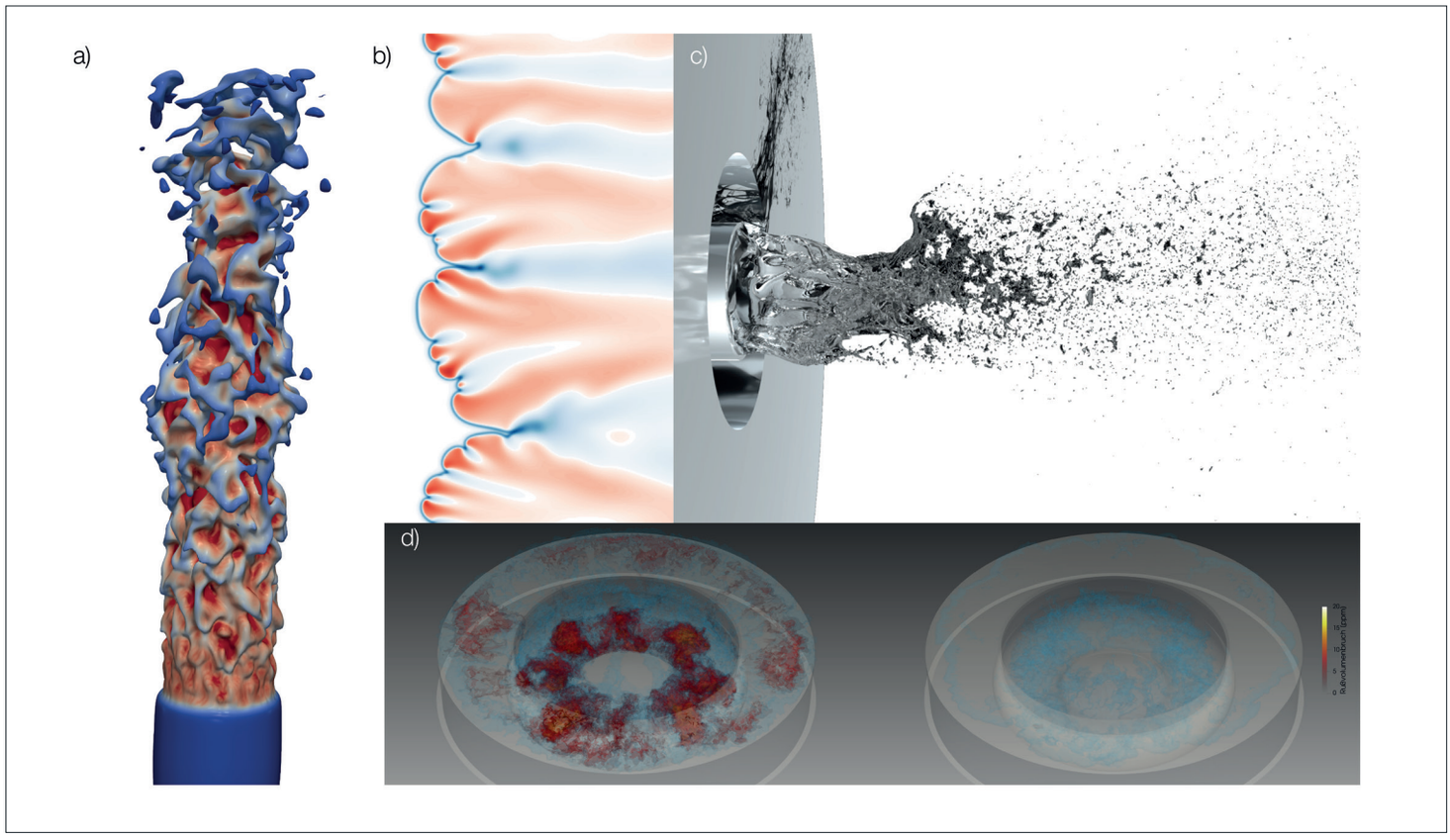


Bild 4: Visualisierung einzelner, mit CIAO durchgeführter Simulationen: a) Runde, turbulente Wasserstoff-Freistrahlflamme, b) Detailansicht einer laminaren Flammenausbreitung, c) Primärzerfall eines koaxialen Zerstäubers, d) Schadstoffentstehung im Motor für unterschiedliche Kraftstoffe.

Entwicklung neuer Verbrennungskonzepte

Das Simulationspaket CIAO kommt aufgrund seines großen Funktionsumfangs in verschiedenen Fragestellungen zum Einsatz. Dazu zählen beispielsweise die Untersuchung von Flammeninstabilitäten, welche bei der Verbrennung von wasserstoffbasierten Kraftstoffen auftreten[1, 4]. Die Ergebnisse dieser Simulationen legen den Grundstein für eine genaue Modellierung der Verbrennungskonzepte und somit die effiziente Entwicklung neuartiger Brennerkonzepte. Im Bereich der Mehrphasenströmungen werden mithilfe von CIAO detaillierte Simulationen des Primärzerfalls von Flüssigkeitsinjektionen durchgeführt[5]. Die Ergebnisse finden ihre Anwendung in der motorischen Verbrennung, aber auch auf dem Gebiet der Nanopartikelsynthese. Im Bereich der motorischen Verbrennung bietet die Simulationssoftware die Möglichkeit einen tiefgreifenden Einblick in die Emissionsbildung von unterschiedlichen Kraftstoffen in verschiedenen Motorgeometrien zu erhalten und somit Emissionen effektiv zu reduzieren. Hierfür werden mittels Overset-Gitter die Simulation von Strömungen in komplexen und bewegten Geometrien unterstützt[3]. Exemplarische Simulationsergebnisse sind in Bild 4 dargestellt.

Literatur

[1] Berger, L., Attili, A., Pitsch, H., Synergistic interactions of thermodiffusive instabilities and turbulence in lean hydrogen flames, Combustion and Flame 244 (2022) 112254
[2] Desjardins, O., Blanquart, G., Balarac, G., Pitsch, H., High order conservative finite difference scheme for variable density low Mach number turbulent flows, Journal of Computational Physics 227 (2008) 7125-7159
[3] Falkenstein, T., Kang, S., Davidovic, M., Bode, M., Pitsch, H., Kamatsuchi, T., Nagao, J., Arima, T., LES of Internal Combustion Engine Flows Using Cartesian Overset Grids, Oil & Gas Science and Technology – Rev. IFP Energies Nouvelles 72 (2017) 36
[4] Chu, H., Berger, L., Grenga, T., Wu, Z., Pitsch, H., Effects of differential diffusion on hydrogen flame kernel development under engine conditions, Proceedings of the Combustion Institute 39 (2023) 2129-2138
[5] Fröde, F., Grenga, T., Le Chenadec, V., Bode, M., Pitsch, H., A three-dimensional cell-based volume-of-fluid method for conservative simulations of primary atomization, Journal of Computational Physics 465 (2022) 111374

Autoren

Dr.-Ing. Marco Davidovic, Fabian Fröde, M.Sc., und Terence Lehmann, M.Sc., sind wissenschaftliche Mitarbeiter am Institut für Technische Verbrennung.
Dr.-Ing. Michael Gauding ist Oberingenieur am Institut für Technische Verbrennung.
Univ.-Prof. Dr.-Ing. Heinz Pitsch ist Inhaber des Lehrstuhls und Leiter des Instituts für Technische Verbrennung.

Julius Berges, Joerg Berroth, Gregor Höpfner, Georg Jacobs, Stefan Wischmann

Virtuelle Absicherung von mechatronischen Systemen

Physikalische Verhaltensmodelle und Model-Based Systems Engineering

As lifecycles of technical products continue to decrease, the development process of these products needs to create innovative, mature products faster while minimizing costs and resources. At the same time, the complexity of technical products increases, mainly driven by the increasing integration of software, which in turn is driven by digitalization. In this context, methods of virtual product development, particularly virtual testing, could play a pivotal role in increasing the efficiency of the product development process.
Thus, software tools and methods for virtual product development are developed at the Institute of Machine Elements and Systems Engineering. Having started off as simulation software for the design and analysis of the dynamic behavior of drive trains, today, the software aims to cover the entire product development process using Model-Based Systems Engineering (MBSE) methods. As a logical next step in product development, the MBSE-method motego, an innovative approach that enables clear, interconnected development across multiple domains, is being developed at MSE. Motego comprises virtual models and provides a language profile allowing for automation, error reduction, and significant enhancement of efficiency, with the added benefit of early virtual validation, all of which are crucial to agile and sustainable product development.

Die Lebenszyklen technischer Produkte verkürzen sich fortwährend. Im Entwicklungsprozess müssen daher in kürzerer Zeit und unter Minimierung von Kosten und Ressourcen Innovationen zur Produktreife gebracht werden. Gleichzeitig steigt die Komplexität der Produkte, unter anderem durch zunehmende Integration von Software in vormals mechanische Systeme. Diesem Zielkonflikt zwischen steigender Komplexität und kürzeren Zyklen kann durch neue Methoden der virtuellen Produktentwicklung begegnet werden. Ziel ist, das Verhalten des Produkts über virtuelle Modelle vorherzusagen und unter Berücksichtigung der Wechselwirkungen zwischen den regelungstechnischen, fluidtechnischen, elektrotechnischen, mechanischen und Software-Teilsystemen zu optimieren. Dies gilt insbesondere für frühe Entwicklungsphasen, in denen Änderungen schnell und kostengünstig umsetzbar sind und keine zeit- und kostenintensiven physischen Prototypen verfügbar sind.
Die modellbasierte Produktentwicklung wird am Institut für Maschinenelemente und Systementwicklung seit langem erforscht. Entwicklungs- und Simulationsmethoden werden über entsprechende Software zur Verfügung gestellt.

Bild 1: Planung experimenteller Untersuchungen mit strukturdynamischen Modellen in den Softwaremethoden des Instituts für Maschinenelemente und Systementwicklung.
Foto: Peter Winandy

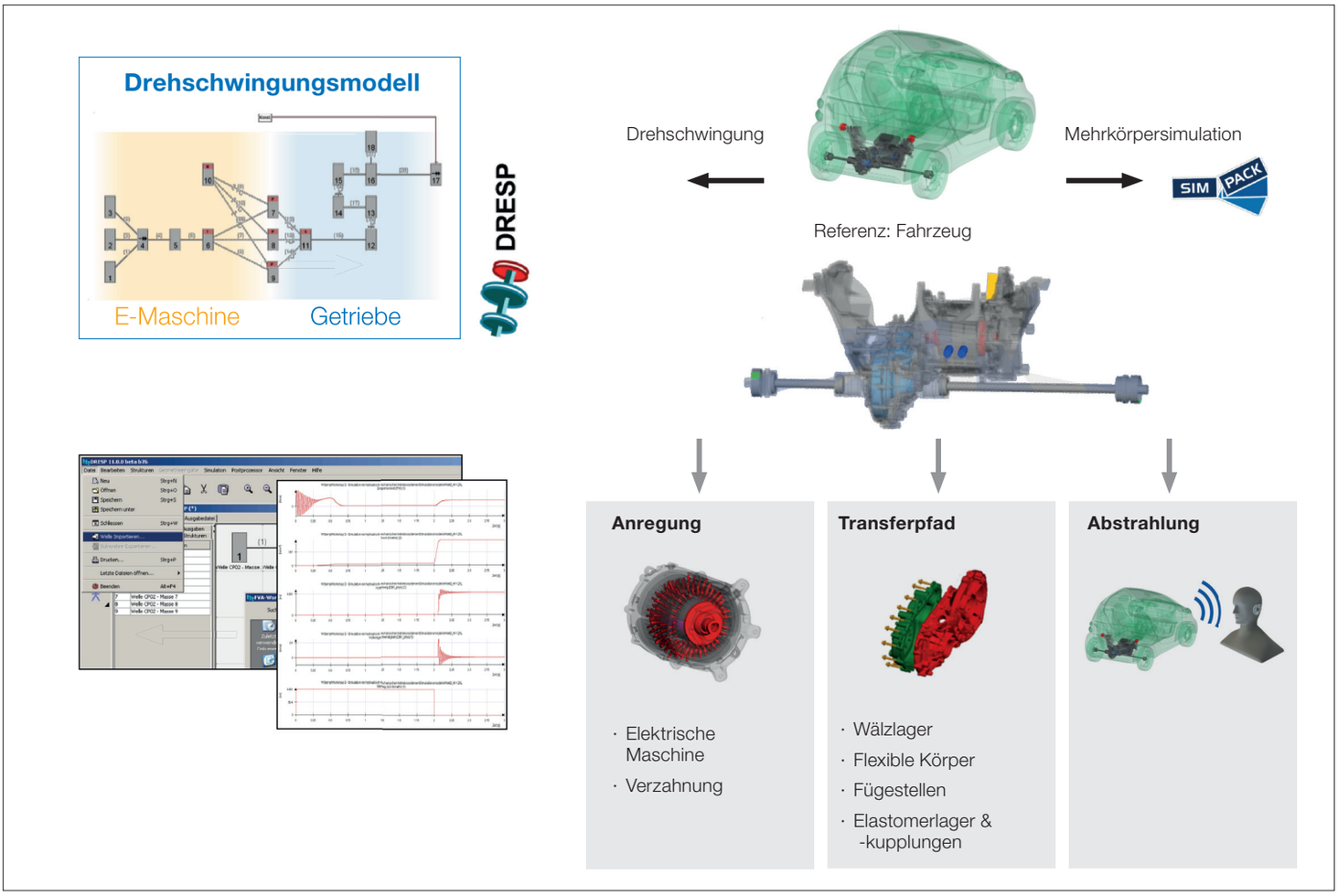


Bild 2: Interdisziplinäre Simulation des Verhaltens dynamischer Systeme im DREhschwingungsSimulationsProgramm, kurz DRESP, und der Mehrkörpersimulations-(MKS)-Software SIMPACK

DRESP, DYLA und SIMPACK

Ausgangspunkt waren Methoden zur Vorhersage des dynamischen Verhaltens hochbelasteter mechanischer Antriebsstränge im Rahmen der Maschinendynamik und -akustik. Dabei sind vor allem Untersuchungen der dynamischen Lasten im Drehfreiheitsgrad (dynamische Momente und Torsionsschwingungen) für die erste Auslegung relevant. Diese werden von den angrenzenden Systemen anderer Domänen wie Elektromotoren, Hydraulik und regelungstechnischen Systemen stark beeinflusst. Daher wurde bereits in den 1980er Jahren mit der Entwicklung einer effizienten Simulationssoftware zur Abbildung der Wechselwirkungen zwischen mechanischen Systemen im Drehfreiheitsgrad, elektrischen, hydraulischen und regelungstechnischen Systemen begonnen, die im DREhschwingungsSimulationsProgramm, kurz DRESP[1], mündete. Mit DRESP können das Gesamtsystemverhalten frühzeitig bewertet und potenziell kritische Betriebspunkte in der Auslegung identifiziert werden. In die Module von DRESP sind bis heute 14 Dissertationen und Kosten von etwa 2,5 Millionen Euro eingeflossen. DRESP wird über die Forschungsvereinigung Antriebstechnik (FVA) bereitgestellt und in mehr als 60 Firmen genutzt. Dabei wurde der Rechenkern in Fortran implementiert und über ein graphisches Interface in Java einfach zugänglich gemacht. Mit zunehmender Ausdetaillierung der mechanischen Antriebe wird verstärkt eine Betrachtung der Freiheitsgrade über den Drehfreiheitsgrad hinaus relevant. Mit DYLA (Simulationsprogramm zur Ermittlung der dynamischen Lagerkräfte) wurde eine Erweiterung in alle sechs mechanischen Freiheitsgrade erreicht.

Die auf Basis von DRESP entwickelten Berechnungsmodule werden heute für die stark am Markt vertretene Mehrkörpersimulations-(MKS)-Software SIMPACK weiterentwickelt und können als eigenständige Teile einer Modellbibliothek eingebunden werden[2].

Mit ihrer validierten, hohen Abbildungsgüte lassen sich so sehr gute Prognosefähigkeiten des dynamischen und akustischen Verhaltens mechatronischer Antriebssysteme realisieren. Der Fokus der Module richtet sich nach der aktuellen Transition hin zu emissionsfreien Antrieben in der Elektromobilität und umfasst unter anderem Abbildungen des Anregungsverhaltens von Verzahnungen und elektrischen Maschinen sowie des nichtlinearen Übertragungsverhaltens von Fügestellen, Wälzlagern und Elastomerkupplungen. Somit wurde eine umfassende Softwarelandschaft zur physikbasierten Simulation des dynamischen Verhaltens mechatronischer Antriebsstränge über verschiedene Entwicklungsdomänen und Zeitpunkte hinweg geschaffen, siehe Bild 2.

motego – Vernetzung von physikalischen Verhaltensmodellen

Die Erfahrung mit DRESP, DYLA und der MKS wird heute genutzt, um den Entwicklungsprozess mechatronischer Systeme mithilfe der Methoden des Model-Based Systems Engineerings zu virtualisieren und zu verbessern. Dabei spielt nicht nur die Integration und durchgängige Nutzung physikalischer Verhaltensmodelle in die modellbasierte Entwicklung eine Rolle, sondern auch die Steigerung der Effizienz der Anwendung dieser Modelle durch Wiederverwendung

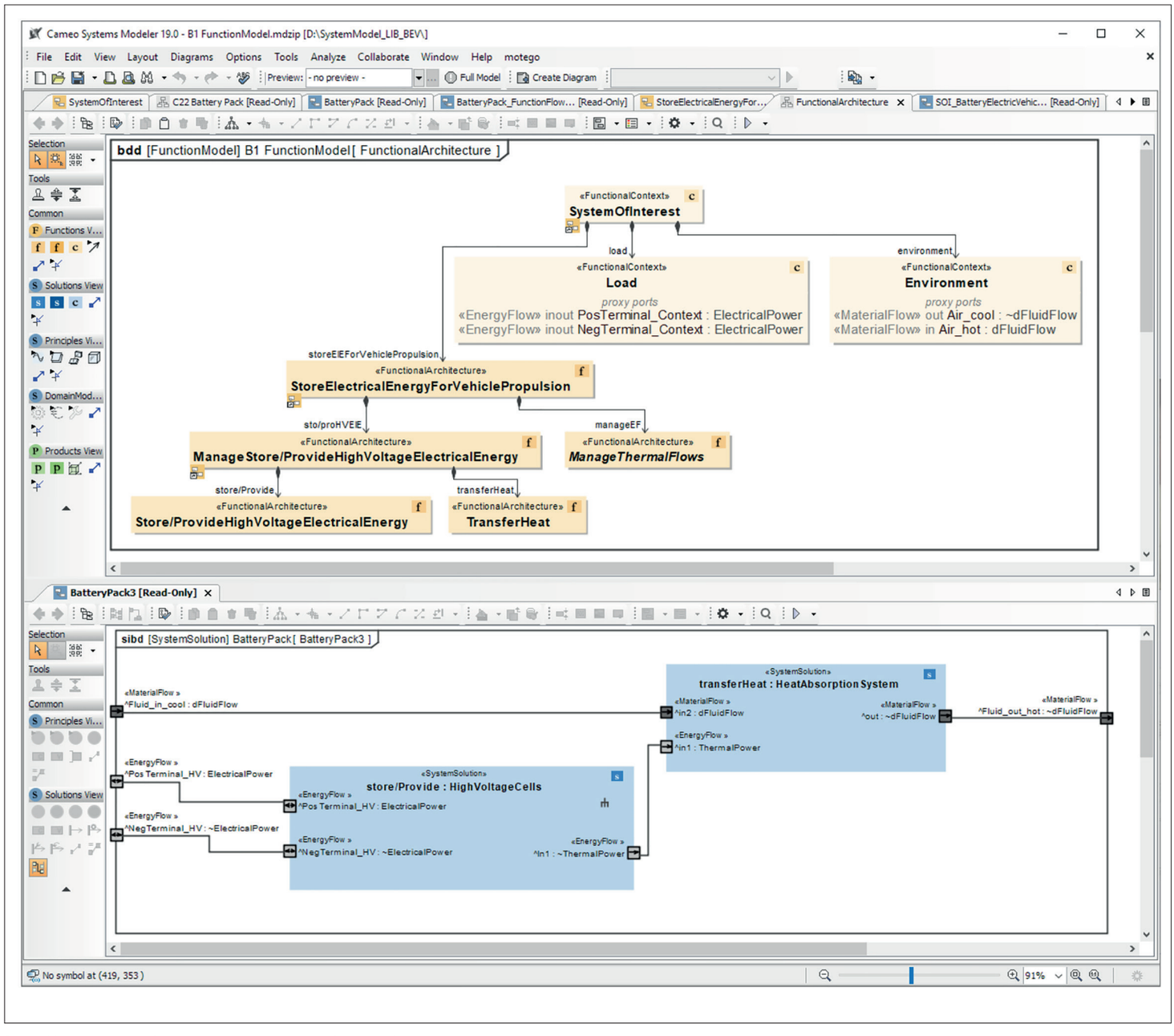


Bild 3: Mechatronisches System modelliert im motego-Sprachprofil

und das dazu notwendige methodische Rahmenwerk zur Strukturierung und Klassifikation der Modelle. Über die Verhaltenssimulation hinaus wird eine modellierte Systemarchitektur in den Mittelpunkt gestellt. Hierzu bedarf es Modellierungssprachen und Softwaretools, die die unterschiedlichen Aspekte eines interdisziplinären Systems in einer Systemarchitektur abbilden können. Ein Schritt ist die Entwicklung und Verbreitung von motego, einer Modellierungsmethode mit einem frei verfügbaren Sprachprofil, das auf der standardisierten Modellierungssprache für technische Systeme „Systems Modeling Language“ (SysML) aufbaut[3]. motego ist eine Methode des Model-Based Systems Engineering, die darauf abzielt, komplexe mechatronische Systeme auf eine klare, vernetzte und flexible Weise zu entwickeln. Die Methode folgt einem Strukturschema, das sich auf vier Pfeiler stützt: Anforderungen, Funktionen, technische Lösungen und das Produkt. Jede Säule wird in der motego-Methode spezifiziert, um eine einheitliche digitale Modellierung auf Parameterebene sicherzustellen. Ein Vorteil von motego liegt in der durchgängigen Vernetzung sämtlicher Systemelemente über diese vier Säulen hinweg. Dadurch werden komplexe Zusammenhänge zwischen den einzelnen Systemelementen über das gesamte zu entwickelnde System hinweg abgebildet. Die motego-Methode zeichnet sich durch ihre systematische Integration und Vernetzung von physikalischen Verhaltensmodellen, modelliert in CAx-Tools wie DRESP, DYLA, der MKS und weiteren, aus. So können die entwickelten Systeme durchgängig virtuell getestet werden. Über die entstehende Struktur für technische Lösungen können physikalische Verhaltensmodelle toolübergreifend verknüpft und in Modellbibliotheken wiederverwendbar gemacht werden. Dies wird über das motego-Plug-in ermöglicht, welches aktuell für die Software Cameo Systems Modeler entwickelt und in Java implementiert ist, siehe Bild 2. Die Verbindung zu weiteren Tools wird über die im Cameo Systems Modeler vorhande-

Bild 4: Die Kombination aus Methoden des Model-Based Systems Engineerings in motego mit validierten physikalischen Modellen und experimentellen Untersuchungen ermöglicht die interdisziplinäre Optimierung antriebstechnischer Systeme.
Foto: Peter Winandy

nen Schnittstellen zu unter anderem Matlab implementiert. Die Methode ermöglicht so die kontinuierliche Erweiterung um zusätzliche Modelle und erlaubt zudem die Organisation unterschiedlicher Detailgrade für verschiedene Entwicklungsstadien[4]. Diese Flexibilität ist entscheidend, um den unterschiedlichen Anforderungen hinsichtlich erforderlicher und realisierbarer Modellgüte gerecht zu werden, die in den verschiedenen Phasen der Produktentwicklung auftreten können.
Die motego-Methode ermöglicht die Abbildung von automatisierten Workflows, in denen mehrere Modelle hintereinander angesprochen werden, um Aufgaben, wie die Absicherung eines Antriebsstranges, zu automatisieren[5]. Durch diese Automatisierung werden menschliche Fehler minimiert und die Effizienz im Entwicklungsprozess gesteigert. Die Vernetzung von Modellen und die Automatisierung von Aufgaben ermöglichen zudem die Arbeit in kurzzyklische Sprints. In diesen können mechatronische Systeme frühzeitig auf Grundlage der Modellbibliotheken virtuell abgesichert werden, ein entscheidender Fortschritt zu heutigen Produktentwicklungsmethoden, da sie Entwickelnden erlaubt, schnell und effizient auf Probleme zu reagieren und Anpassungen am System vorzunehmen. Mit motego wird die virtuelle Absicherung und Entwicklung mechatronischer Systeme am Institut für Maschinenelemente und Systementwicklung ausgehend von den einzelnen Berechnungstools aus DRESP, DYLA und den MKS-Berechnungstools in die durchgängig vernetzte, virtuelle Produktent-

wicklung erweitert und so nicht nur ein effizienter Prozess geschaffen, sondern auch die Qualität und Nachhaltigkeit der entwickelten Produkte gesteigert. Die Methode schafft eine Grundlage für eine agile und transparente Produktentwicklung, die den Herausforderungen komplexer mechatronischer Systeme gewachsen ist. Das Sprachprofil motego wird als frei zugängliches Plug-in für die MBSE-Softwareumgebung Cameo Systems Modeler angeboten. So können Entwicklerinnen und Entwickler die Methoden in eigene Projekte integrieren und von den genannten Vorteilen zu profitieren. Das Sprachprofil kann über die Website www.motego.info abgerufen werden. Durch die stetige Weiterentwicklung der MBSE-Methode motego leistet das Institut für Maschinenelemente und Systementwicklung einen entscheidenden Beitrag zur Gestaltung der zukünftigen modellbasierten Produktentwicklung.

www.motego.info

Literatur

[1] Jacobs, G., Schelenz, R., Juretzki, B., Flock, S., Benutzerhandbuch DRESP 13.0. (FVA-Forschungsheft 485) Forschungsvereinigung Antriebstechnik: Frankfurt a.M., Aachen, 2012

[2] Jacobs, G., Wegerhoff, M., Andary, F., DRESP2SIMPACK II (FVA-Forschungsheft 1176) Forschungsvereinigung Antriebstechnik: Frankfurt a.M., Aachen, 2016

[3] Spütz, K., Jacobs, G., Zerwas, T. et al., Modeling language for the function-oriented development of mechatronic systems with motego. Forsch Ingenieurwes 87, 387–398, 2023, https://doi.org/10.1007/s10010-023-00623-4

[4] Jacobs, G., Konrad, C., Berroth, J. et al, Function-oriented model-based product development. In: Krause D, Heyden E (eds) Design methodology for future products: data driven, agile and flexible, 1st edn. Springer, Cham, pp 243–263, 2022

[5] Berges, J.M., Spütz, K., Zhang, Y. et al., Reusable workflows for virtual testing of multidisciplinary products in system models. Forsch Ingenieurwes 87, 339–351,2023, https://doi.org/10.1007/s10010-023-00621-6

Autoren

Julius Berges, M.Sc., ist Gruppenleiter Systems Modeling and Optimization am Institut für Maschinenelemente und Systementwicklung.

Dr.-Ing. Joerg Berroth ist geschäftsführender Oberingenieur Systems Engineering and Off-Highway am Institut für Maschinenelemente und Systementwicklung.

Gregor Höpfner, M.Sc., ist Bereichsleiter Systems Engineering – Modeling and Simulation am Institut für Maschinenelemente und Systementwicklung.

Univ.-Prof. Dr.-Ing. Georg Jacobs ist Inhaber des Lehrstuhls und Leiter des Instituts für Maschinenelemente und Systementwicklung.

Stefan Wischmann, M.Sc., ist Gruppenleiter Noise Vibration Harshness am Institut für Maschinenelemente und Systementwicklung.

Susanne Kunkel, Markus Diesmann

Entwicklung des Research Software Engineering am Beispiel von NEST

Wissenschaftliche Software ist wissenschaftliche Infrastruktur

The scientific software NEST is a simulation code capable of representing biological neuronal networks at full density, meaning with their natural number of contact points, called synapses, between the nerve cells. The software scales from laptops to the largest supercomputers available today and is the core simulation engine of the European ICT infrastructure EBRAINS for this level of description. The code originates from the middle of the 1990s where it was developed under the name SYNOD in the context of a Master's thesis. At the time, Computational Neuroscience had already been around for a decade. From the beginning, a motivation of the NEST project was to move code development in science out of the "dark ages" and make it a proper part of the scientific method. Today NEST is governed by a community organization, and it is used by scientists around the world to produce and publish new scientific findings. NEST is used as a simulation engine inside other applications, and, continuously, peer-reviewed papers are published, describing the technological advancements of the tool. Therefore, NEST is well suited to trace back the evolution of the practice of Research Software Engineering (RSE) over the past 25 years.

Die wissenschaftliche Software NEST dient der Simulation biologischer neuronaler Netze. NEST bildet Netzwerke von Nervenzellen mit all ihren Kontaktstellen, den Synapsen, ab und ist auch in der Künstlichen Intelligenz und Robotik relevant. Die Software läuft sowohl auf einem Laptop als auch auf den größten heute verfügbaren Supercomputern. NEST hat sich als leistungsfähiger Simulator in der Computational Neuroscience etabliert und wird als zentrale Simulationsmaschine der europäischen digitalen Infrastruktur EBRAINS genutzt.

NEST entstand Mitte der 1990er Jahre im Rahmen einer Diplomarbeit und wurde unter dem Namen SYNOD veröffentlicht. Zu dieser Zeit herrschte eine Kultur, in der die Entwicklung wissenschaftlicher Software nicht als Teil der wissenschaftlichen Arbeit angesehen und systematisch den unerfahrenen Mitarbeiterinnen und Mitarbeitern aufgetragen wurde. Oftmals endete mit dem Abschluss der Diplom- oder Doktorarbeit auch der Code, und nachfolgende Wissenschaftlerinnen und Wissenschaftler fingen wieder bei null an. Obwohl früher gegründet, war die Computational Neuroscience rückständig im Vergleich mit jüngeren Wissenschaftsbereichen wie der Systembiologie. Eric De Schutter vom Okinawa Institute of Science and Technology in Japan führte diesen Widerspruch auf die frühe Gründung zurück. Dadurch wurde die Wissenschaftspraxis geprägt als die Problematik noch nicht erkannt und eine geeignete Methodik noch nicht verfügbar war. Schon am Anfang des NEST-Projekts stand der Gedanke, die Entwicklung wissenschaftlicher Software aus den finsteren Zeiten herauszuführen und sie zu einem gleichberechtigten Teil der wissenschaftlichen Methode zu machen. Offiziell wurde diese Mission 2012 mit der Gründung der NEST Initiative als Verein.

Forschungssoftware unterliegt besonderen Anforderungen

Mit dem Wachsen des Projekts, siehe Bild 1, stellte sich heraus, dass Software-Entwicklung in der Wissenschaft besonderen Bedingungen unterliegt, und sich Erkenntnisse und Methoden aus dem auf die Industrie ausgerichteten Software-Engineering nur eingeschränkt übertragen lassen. Die Erstellung wissenschaftlicher Software ist Teil des Erkenntnisprozesses, einzelne Wissenschaftlerinnen und Wissenschaftler verbringen aber nur zwei bis fünf Jahre in einer Institution. Sie erfahren dabei Anerkennung für Publikationen in der Fachdomäne, aber nicht für Qualität und Nachhaltigkeit ihres Codes. Dies bestärkt sie darin, Software nur für ihren eigenen unmittelbaren Nutzen zu entwickeln. Investition und Wissen gehen verloren, sobald die Institution verlassen wird. Den Preis für diesen Mangel an Nachhaltigkeit zahlen Institution und Gesellschaft. Die Erforschung, wie gutes Research Software Engineering aussieht steht noch am Anfang.

Aus der Praxis der NEST-Entwicklung können folgende Schlüsse gezogen werden:

- **Iterativer Ansatz**
  Jede Generation von Forschenden baut auf den Ergebnissen der vorhergehenden auf. Wissenschaftliche Software muss dieses Prinzip unterstützen und zu jeder Zeit einsatzbereit sein. Ziel ist deshalb keine perfekte Implementation, sondern eine Architektur, die eine iterative Weiterentwicklung unterstützt, um weitere Aspekte der Natur nachbilden zu können, ohne dass es eine endgültige Funktionsbeschreibung gibt.

- **Kollaborative Strukturen**
  Wissenschaft wird heute in über Institutionen und Länder verteilten Teams betrieben. In einem wissenschaftlichen Code wird es auch vorkommen, dass verschiedene Teams unterschiedliche Fähigkeiten des Codes gleichzeitig weiterentwickeln. Die Robustheit der Software hängt deshalb von Entwicklungsprozessen und -infrastrukturen ab, wie zum Beispiel Versionskontrolle und automatisierten Tests, die gemeinschaftliches Arbeiten unterstützen.

- **Open Source**
  Die Beteiligung mehrerer Institutionen an einer Software hat den Vorteil, Code in einem Umfang und in einer Qualität entwickeln zu können, wie es allein unmöglich wäre. Die Sichtbarkeit fördert die Nachvollziehbarkeit und Reproduzierbarkeit wissenschaftlicher Resultate. Dabei sichert eine Lizensierung als Open Source den Kollaborationspartnern den Zugang zu einem Instrument, das kritisch für den Erfolg geworden ist. Die Nutzung eines bekannten Codes senkt die Hürde den Quellcode einer wissenschaftlichen Publikation offen zu legen.

- **Dokumentation**
  Wissenschaftlerinnen und Wissenschaftler sind meist keine ausgebildeten Entwickler, sollen aber in kurzer Zeit in der Lage sein, mit einer komplexen, gewachsenen Software zu forschen und diese weiterzuentwickeln. Sowohl Anwender- als auch Entwicklerdokumentation muss dieser Situation gerecht werden. Mechanismen zur Erstellung der Dokumentation aus dem Quellcode motivieren daran zu arbeiten. Aber eine im wissenschaftlichen Schreiben ausgebildete Person muss die Übersicht über die Architektur der Dokumentation behalten.

- **Schichtenarchitektur**
  Bibliotheken für graphische Benutzerschnittstellen und interpretierte Programmiersprachen, beispielsweise zur Simulationssteuerung verwendet, haben sich als kurzlebig erwiesen. Eine starke Verknüpfung mit solcher Fremdsoftware, ist nicht nur destabilisierend, sondern erschwert auch die Portierung von Code auf neue Systeme. Eine Organisation der Software in Schichten mit wenigen Interaktionspunkten erleichtert das Loslösen von einer überholten Fremdsoftware.

- **Gezielte Objektorientierung**
  Der Nutzen objektorientierter Programmierung für die Definition klarer Schnittstellen und damit der Unterstützung kollaborativer Entwicklung und Wartbarkeit ist unumstritten. Für einen wissenschaftlichen Code ist es jedoch entscheidend für Laufzeit und Speicherverbrauch, nicht einfach die Natur in Objekten abzubilden, sondern auch die für Hochleistungsrechner günstigen Datenstrukturen zu bedenken. Die Kunst besteht darin, zwischen beiden Welten effiziente und erklärbare Übergänge zu finden.

- **Ein Quellcode**
  Der wissenschaftliche Alltag bietet kaum die Kapazität zur Pflege mehrerer umfangreicher Codes. Zwangsläufig führt die kurze Lebensdauer von Hardware zur Vermeidung von Abhängigkeiten von plattformspezifischen Bibliotheken, es sei denn sie implementieren offene Standards. Und Anstrengungen in einen einzigen extrem skalierbaren Code zu investieren, der auf allen Systemen vom Laptop bis zum Supercomputer effizient läuft, rentiert sich. Eine neue Herausforderung ist die Verbreitung von Hardware-Beschleunigern wie Grafikprozessoren.

- **Agiler Ansatz**
  In der Forschung ändern sich die Anforderungen ständig, und naturgemäß ist das Verständnis über den Forschungsgegenstand begrenzt. Versuche, allgemeine Lösungen zu implementieren, die über die konkreten Bedürfnisse der Anwender hinaus gehen, aber für die Zukunft wichtig scheinen, scheitern oft an falsch eingeschätzten Anforderungen oder Leistungseinschränkungen. Nur agile Softwareentwicklung unter Einhaltung des Minimalprinzips, eine neue Funktion nur zu implementieren, wenn sie direkt eine Anwendung findet, ist mit Forschung vereinbar.

- **Kostenschätzung**
  Wie relevant eine neue Fähigkeit für die Fachdomäne ist, ist meist schwer zu beurteilen, und die Kosten können erheblich sein, besonders wenn tiefgreifende Änderungen erforderlich sind. Aufwand und Nutzen abwägen kann nur eine in Softwarearchitektur ausgebildete Person, die sowohl Code als auch Fachdomäne kennt. Um die Konsequenzen für Genauigkeit und Leistungsfähigkeit des Gesamtsystems und die Kosten einschätzen zu können ist es für das Research Software Engineering wichtig, den theoretischen Unterbau und die Begriffsklärung der Simulationstechnologie in einer Fachdomäne voranzubringen.

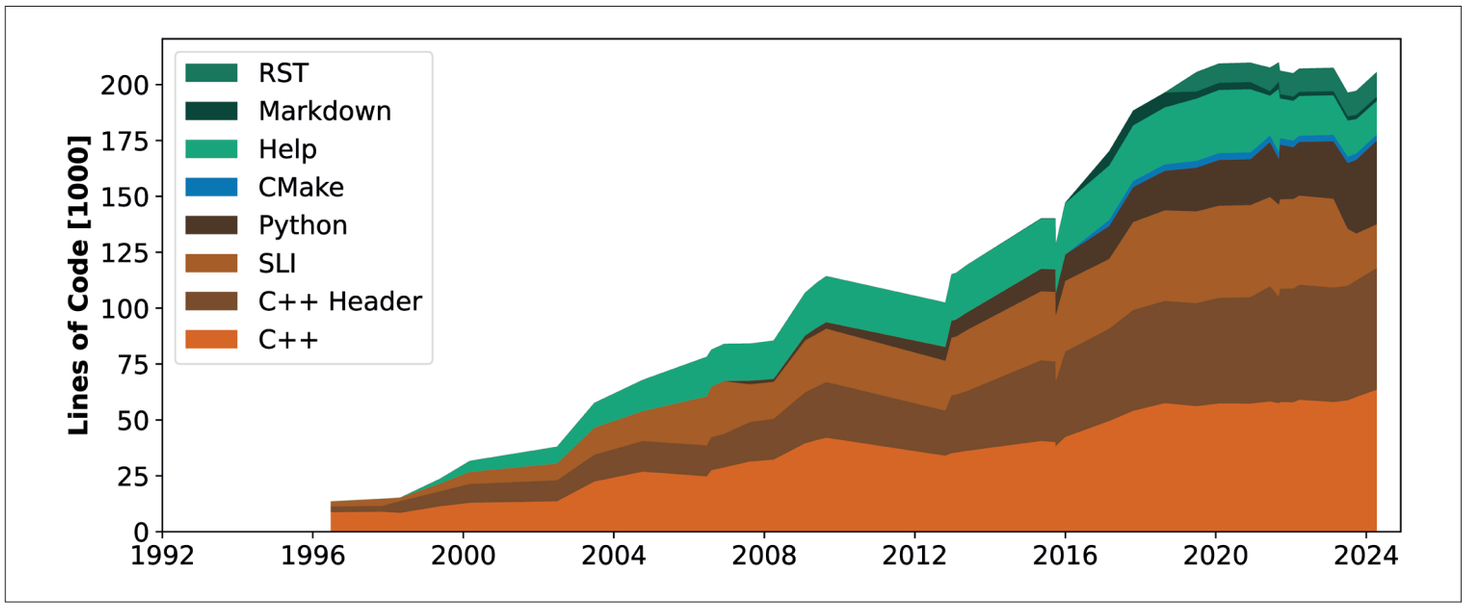


Bild 1: Anzahl der Code-Zeilen der NEST-Software (vertikal) als Funktion der Zeit (horizontal). Der Code hat Anteile mehrerer Programmiersprachen und enthält auch einen eigenen Interpreter für Simulationsskripte (SLI) sowie die Dokumentation (Legende). Daten liegen seit der Einführung eines Versionskontrollsystems vor (CVS, vergleiche Bild 2).

Nachhaltige Forschung braucht Infrastruktur auf allen Ebenen

Die wachsende Komplexität des Codes erforderte die Einführung neuer Technologien, siehe Bild 2. Dies gilt, auch um entsprechend der beschriebenen Erkenntnisse zum Research Software Engineering zu handeln. Diese Fortschritte sind vor dem Hintergrund der allgemeinen Entwicklung in der Computational Neuroscience zu sehen. Um die Jahrtausendwende wurde klar, dass die Neurowissenschaft sich in einer Software-Krise befand: die Softwareentwicklung machte kaum noch Fortschritte und die Reproduzierbarkeit von Ergebnissen nahm ab. In 2007 wurde schließlich die International Neuroinformatics Coordinating Facility (INCF) gegründet, und heute sind die größten Probleme überwunden. Nach intensiver Forschung über ein weiteres Jahrzehnt ist mit dem Abschluss des Europäischen Human Brain Projects in 2023 und dem Aufbau der Europäischen ICT Infrastruktur EBRAINS, das Problem eine Software-Infrastruktur für die Neurowissenschaft bereit zu halten, technisch und administrativ im Prinzip gelöst. Ob die vorgeschlagene Infrastruktur auch dauerhaft finanziert werden kann, ist eine offene Frage.

Forschungssoftware soll den Forschergeist wachsen lassen

Simulation hat sich auch in der Computational Neuroscience als dritte Säule der wissenschaftlichen Methode neben Experiment und Theorie etabliert. Ein wichtiger Schritt war dabei die formale Trennung konkreter Modellbeschreibungen neuronaler Netze von den allgemeinen Algorithmen zur Lösung der Gleichungen solcher Modelle. Trotz dieser Entwicklung ist der Wissenschaftsbereich weiterhin dominiert vom hypothesengetriebenen Ansatz. Immer schon gibt es auch den technologiegetriebenen Ansatz, bei dem ein neues Instrument, beispielsweise ein Teleskop, eine völlig neue Welt eröffnet, über die es vorher keine Hypothesen geben konnte. Die NEST Initiative möchte der Computational Neuroscience so ein Instrument zur Verfügung stellen. Sie hat den Anspruch, Untersuchungen komplexer Netzwerke auf der Größenskala des Gehirns mit einem höheren Erklärungswert für die Hirnfunktion zu ermöglichen. Die Arbeit soll dabei so einfach und schnell sein, dass bestehende Komplexitätsbarrieren überwunden werden. Gleichzeitig soll der Austausch von Modellbeschreibungen eine Kultur etablieren, in der Neurowissenschaftler auf den Arbeiten anderer aufbauen können, und bewährte Modelle als Komponenten größere Modelle verwendet werden können. Oft heißt es: „Auf meinem Laptop kann ich 10.000 Neuronen mit einem selbstgeschriebenen Python-Programm simulieren, mehr brauche ich nicht für meine Theorie“. Aber wie wahrscheinlich ist es, dass dieses Programm fehlerfrei und numerisch genau ist? Und, wenn es die Möglichkeit gibt, mehrere Millionen Neuronen zu simulieren, warum nicht die Gültigkeit einer Theorie an Netzwerken natürlicher Größe überprüfen?

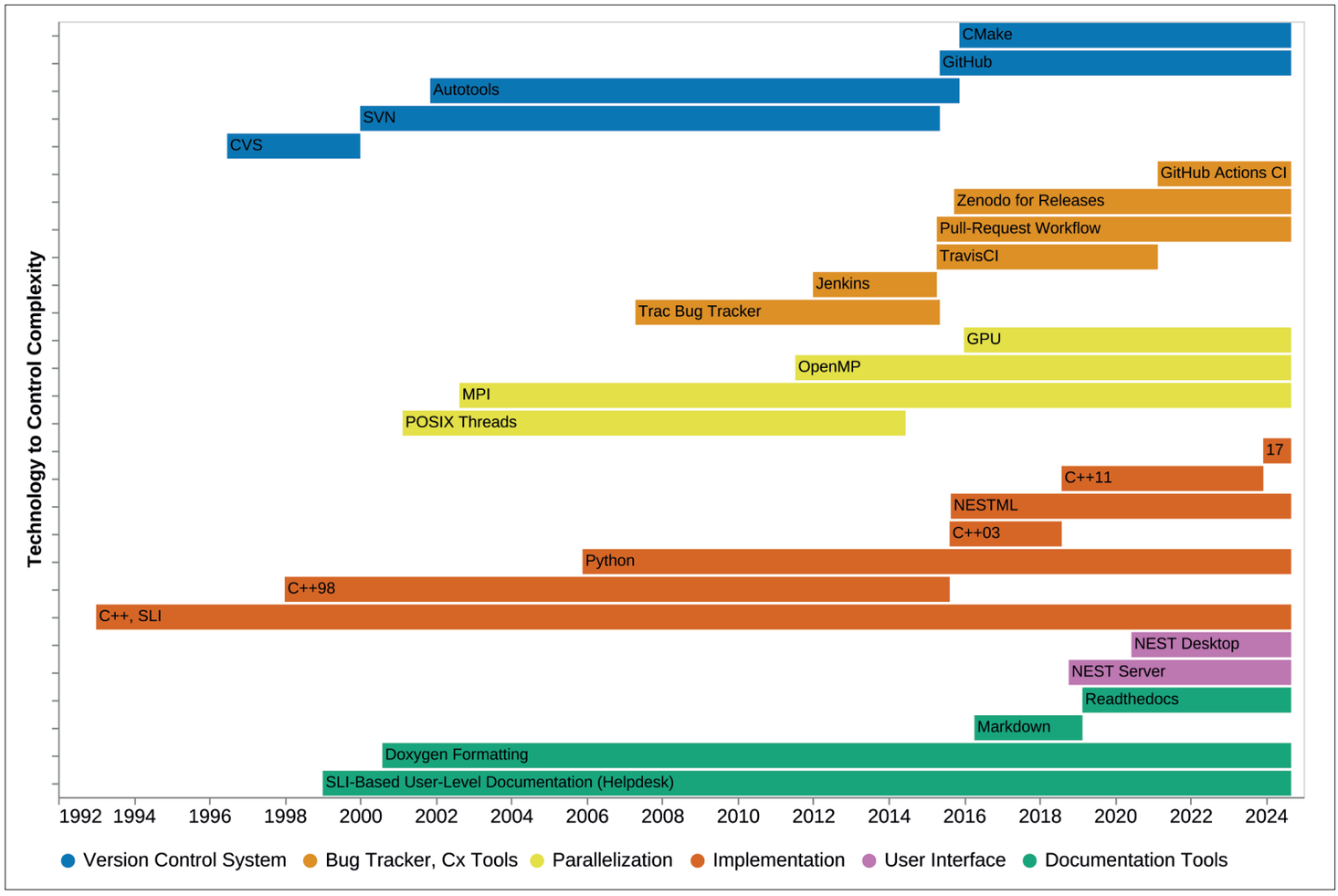


Bild 2: Einführung neuer Software-Technologien in das NEST-Projekt als Funktion der Zeit mit dem Ziel bei steigender Größe des Codes (vergleiche Bild 1) die Qualität und Leistungsfähigkeit gewährleisten zu können.

Research Software Engineering muss Teil des Wissenschaftsalltags sein

Research Software Engineering hat zwei Aspekte. Einerseits erzeugt oder pflegt heute jede forschende Person Software und sollte daher entsprechend ausgebildet werden. Andererseits braucht jeder wissenschaftliche Code eine zentrale Person, die die Gesamtarchitektur im Blick hat. Eine Forschungsgruppe, die Software als Infrastruktur betreibt, braucht eine Person, die mit tiefem Domänenwissen langfristig mit dem Projekt verbunden bleibt. Hauptaufgabe ist die iterative Verbesserung, um die Qualität und Wartbarkeit des Codes und seiner Architektur aufrecht zu erhalten. Neue Funktionen können Wissenschaftlerinnen und Wissenschaftler zielgenauer und aggressiver orientiert an den wirklichen Bedürfnissen entwickeln. Idealerweise kommt es dabei früh zu einer Interaktion. Dabei ist es wichtig, dass sich kein Konservatismus ausbreitet, sondern es oberstes Ziel bleibt, dass der Code wissenschaftliche Probleme löst.

Wissenschaftliche Software ist wissenschaftliche Infrastruktur

In der NEST-Entwicklung hat sich aus den besonderen Bedingungen der Forschung eine Praxis des Research Software Engineering herausgebildet. Über die Details der Methodik und Gewichtung einzelner Aspekte gibt es in der Community noch keine Einigkeit. Wie Research Software Engineering gelehrt werden kann, muss noch erarbeitet werden und was die optimale Methodik ist, muss weiter erforscht werden. Die RWTH hat sich aufgemacht, dieses neue Thema für sich und ihre Studierenden zu erschließen. Relevante wissenschaftliche Software hat alle Eigenschaften traditioneller Forschungsinfrastruktur aus Stahl und Beton: Sie muss über Jahrzehnte entwickelt und betrieben werden, es werden die höchsten Ansprüche an Qualität und Zuverlässigkeit gestellt und es bedarf eines Teams von Experten, die sich um Wartung und Koordination der Entwicklung kümmern. Wie bei anderen Infrastrukturen übernimmt ein Standort eine große Verantwortung, kann aber auch eine hohe Bekanntheit erreichen. Diese neue Sicht auf wissenschaftliche Software wird auch die Finanzierunglandschaft tiefgreifend umgestalten.

Autoren

Dr.rer.nat. Susanne Kunkel ist Abteilungsleiterin am Institut für Neuromorphic Software Ecosystems (PGI-15) am Forschungszentrum Jülich. Univ.-Prof. Dr.rer.nat. Markus Diesmann leitet das Lehr- und Forschungsgebiet Computational Neuroscience und ist Leiter des Instituts für Computational and Systems Neuroscience (IAS-6) am Forschungszentrum Jülich.

Alexander Kruschewsky, Camelia Oprea, Mark Schoberer, André Stollenwerk

Software-Toolchain als neues Werkzeug in der Medizin

Komplikationen frühzeitig erkennen

Digitalization in intensive care opens up new forms of therapy and enables early diagnostic support. In interdisciplinary cooperation, a research software was developed that enables data to be collected and analyzed in compliance with the data protection regulations in order to address scientific questions based on it. The intensive collaboration between medical doctors and computer scientists made it possible to create research software which, thanks to its aforementioned capabilities, enables both the early detection of complications and therapy support.

Lebensbedrohliche Gesundheitszustände werden auf Intensivstationen behandelt. Hier wird Medizintechnik in erheblich größerem Umfang eingesetzt als in der nicht-intensiven Krankenversorgung, etwa zur Überwachung und Visualisierung von Vitalparametern oder zur Durchführung von Beatmungstherapien. Während die von den Geräten gelieferten Daten lange Zeit ausschließlich für Diagnose, Therapie und Dokumentation eingesetzt wurden, begünstigt die Digitalisierung der Patientenversorgung die Zusammenführung und dauerhafte Speicherung dieser Daten[1]. Die resultierenden Datenbanken ermöglichen die retrospektive Analyse und insbesondere den Einsatz von Algorithmen und automatisierten Erkennungsverfahren. In interdisziplinärer Zusammenarbeit entwickeln Ärztinnen und Ärzte der Uniklinik RWTH Aachen gemeinsam mit Mitarbeitenden des Lehrstuhls Embedded Software (Informatik 11) Visualisierungs- und Annotationssoftware. Das medizinische Personal kann hier die Datensätze nicht nur einsehen, sondern auch um Kommentierungen erweitern. Daraus lassen sich neue physiologische Muster erkennen sowie Hypothesen erstellen, die nach der Identifikation algorithmisch ausgewertet wer-

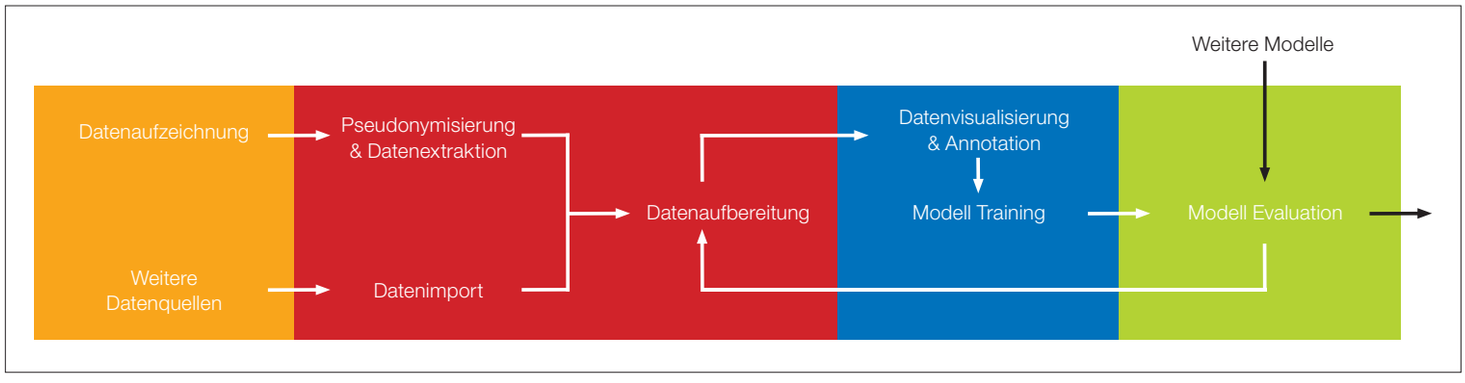


Bild 1: Prozessablauf einer nachhaltigen Software-Toolchain in der Datenwissenschaft: Zunächst müssen die gesammelten Daten in ein gemeinsames Format gebracht und die Datenqualität sichergestellt werden. Daraufhin erfolgt eine algorithmische Datenauswertung. Die entstehenden Modelle können mit Modellen aus der Literatur verglichen werden und iterativ zur Verbesserung der Toolchain beitragen.

Bild 2: Die Digitalisierung von intensivmedizinischer Überwachungs- und Diagnose-Technik liefert kontinuierlich einen großen Datenstrom.
Foto: Uniklinik RWTH Aachen-

den. Ziel ist unter anderem ein klinisches Entscheidungsunterstützungssystems, welches als einheitliches System das medizinische Fachpersonal bei dem beschriebenen Arbeitsfluss unterstützt. Schwerpunkte einer solchen Software sind in Bild 1 dargestellt.

Datenaufzeichnung und -extraktion

Die Medizingeräte auf einer Intensivstation geben in variierenden Frequenzen Werte aus, diese sind primäre Datenquelle. Zusätzlich dokumentiert das Pflegepersonal über Tablets klinische Beobachtungen, diagnostische oder therapeutische Maßnahmen. Diese werden mit den Primärdaten der Medizingeräte zusammengeführt und durch Vernetzung krankenhausintern in Datenbanken zusammengeführt. Schließlich erfolgt eine externe Auswertung jenseits der akuten Patientenbehandlung. Bei diesem Arbeitsschritt müssen die Datenschutzrichtlinie und die patienteninvididuelle Einwilligung beachtet werden.
Da die vorgabenkonforme Implementierung der skizzierten Aufzeichnungs- und Extraktionsprozesse aufwändig ist, wird in der Forschung häufig auf existente, schon veröffentlichte Datensätze zurückgegriffen. Diese können jedoch in unterschiedlichen Formaten vorliegen, was einen zusätzlichen Importprozess erforderlich macht. Dafür gibt es zwei Ansätze. Eine Möglichkeit besteht darin, die Daten direkt in ein gängiges medizinisches Datenmodell, beispielsweise OMOP (Observational Medical Outcomes Partnership), zu konvertieren[2]. Dies ermöglicht anschließend einen einheitlichen Importprozess für alle Datensätze. Der Anfangsaufwand ist hier größer, da die Datenbank, welche häufig mehrere Tausend Patienten umfasst, in das neue Format umzuwandeln ist. Alternativ können maßgeschneiderte Importprozesse für jeden Datensatz entwickelt und modular integriert werden. Zwar ist der anfängliche Aufwand geringer, dennoch ist dieser Weg des Datenimports meist aufwendiger.
Allgemein gilt es beim Importprozess zu beachten, dass verschiedene Datenquellen unterschiedliche Qualitätsniveaus aufweisen können[3]. Probleme wie fehlende Werte und Messfehler sind keine Seltenheit, sie entstehen beispielsweise durch verschobene Sensoren während pflegerischer Tätigkeiten am Patienten. Um diesem Problem entgegenzuwirken, werden Methoden zur Verbesserung der Datenqualität genutzt. Diese erkennen fehlerhafte Werte und können mitunter Datenlücken schließen. Hierbei ist die Unterstützung durch Ärzte essenziell. Die medizinische Fachexpertise wird bei der Identifizierung der genannten Stellen genutzt und die Qualität der Analysealgorithmen ist so sichergestellt.

Bild 3: Über die Sicherheit dieser kleinsten Patienten können außer Menschen und Monitoren auch Algorithmen wachen.
Foto: Uniklinik RWTH Aachen


WARNUNG !
Luftkanäle in der Deckplatte nicht
Niemals Tücher über die Luft
Niemals den Patienten
Verbrennungsgefahr

Datenannotation und visuelle Bereitstellung

Nach der Extraktion beziehungsweise dem Import der Daten können diese mittels eines Visualisierungstools bereitgestellt werden. Dabei sind das schnelle Laden der unterschiedlich aufgelösten Daten und die übersichtliche Visualisierung der einzelnen Datentypen erheblich. Bei den Datentypen unterscheidet man beispielsweise nach der Auflösung: Daten mit Frequenzen von 1-500 Hz, wie EKG oder Atemkurven sind in einem Graph dargestellt. Werte, die nur wenige Male an einem Tag erhoben werden, kommen in eine gemeinsame tabellarische Darstellung. Ein wertvolles Merkmal einer Software, die für die nachträgliche Analyse eingesetzt wird, ist das Anzeigen von Trenddaten in nutzerdefinierten Zeitrahmen. Dabei ist es für den Anwender wertvoll, wenn die zeitliche Auflösung fließend und intuitiv verändert werden kann und sich gleichzeitig numerische und graphische Daten sinnvoll zusammenführen lassen. Hierbei ist ein Datenmodell entscheidend, welches verschiedene Auflösungen ermöglicht und dabei das medizinische Fachwissen berücksichtigt, sodass entscheidende Merkmale beim Zusammenfassen nicht beziehungsweise möglichst spät verloren gehen. In der Medizin sind zudem allgemeine Informationen über den Patienten oft von hohem Nutzen. Ein schneller Herzschlag mag bei jungen Menschen normal sein, doch bei Älteren ist er eher ungewöhnlich. Alter, Geschlecht und Gewicht einer Person können das Risiko für bestimmte Krankheiten erhöhen oder verringern[4]. Diese Informationen helfen somit, in Situationen, in denen die Daten nicht eindeutig sind, die Genauigkeit der nachträglichen Analyse zu erhöhen und das Krankheitsbild besser zu verstehen.

Die nachträgliche Kommentierung der Daten wird durch eine Annotationsfunktion ermöglicht, die durch kontextsensitive Auswahlmöglichkeiten, Such- und Filterfunktionen optimiert ist.

Die Gesundheitsdaten eines einzelnen Patienten können unzählige Informationen enthalten. Es wäre somit äußerst mühsam und zeitaufwendig, jeden Datenpunkt auf Fehler oder Anzeichen von Krankheiten zu überprüfen. Durch die Eintönigkeit der Aufgabe ist diese zusätzlich sehr fehleranfällig. Daher wurden unterstützte Annotationen implementiert[5]. Bei dieser Methode wird ein Algorithmus trainiert, um selbstständig Datenpunkte zu erkennen. Die Ärztin beziehungsweise der Arzt muss dann nur noch bei den Datenpunkten eingreifen, bei denen der Algorithmus unsicher ist. Außerdem stellt eine Übersicht die Entscheidungen der Methode dar. So ist sichergestellt, dass Fehler vermieden werden. Diese innovative Technik unterstützt die medizinische Bewertung, indem sie die Effizienz steigert und die Genauigkeit erhöht, selbst in komplexen Datenmengen.

Datenauswertung anhand eines modularen Plugin-Systems

Durch Annotationen entstehen Datensätze die sich algorithmisch auswerten lassen, Physiologischen Mustern wird eine „Bezeichnung“ zugewiesen, dieser Zusammenhang zwischen Muster und „Bezeichnung“ kann von einem Algorithmus erlernt werden. Alternative Visualisierungen und weitere Berechnungen wie Minimal- oder Durchschnittswerte über angegebene Zeitabschnitte im Tool gehören ebenfalls zum Funktionsumfang.

An dieser Stelle berücksichtigt die Forschungssoftware ein Plugin-Paradigma, die einzelnen Features sind in separaten Modulen verpackt, deren Einsatz unabhängig voneinander erfolgt. Entscheidend ist eine einheitliche Schnittstelle für die Module, inklusive einer einheitlichen Bereitstellung der Daten. Nur die notwendigen Module werden dann heruntergeladen und dynamisch nach dem Start der Applikation eingesetzt. Dementsprechend ist eine effiziente Anpassung der Software auf sich verändernde beziehungsweise neu erkannte Bedarfe möglich.

Innerhalb einer Software können aber nicht nur die Algorithmen zur Auswertung modular ausgetauscht werden, sondern auch die genutzten Daten. In der Forschung entsteht dabei ein Problem: Es ist schwer, Ergebnisse von Algorithmen und Modellen aus der wissenschaftlichen Literatur miteinander zu vergleichen, da es keine einheitlichen

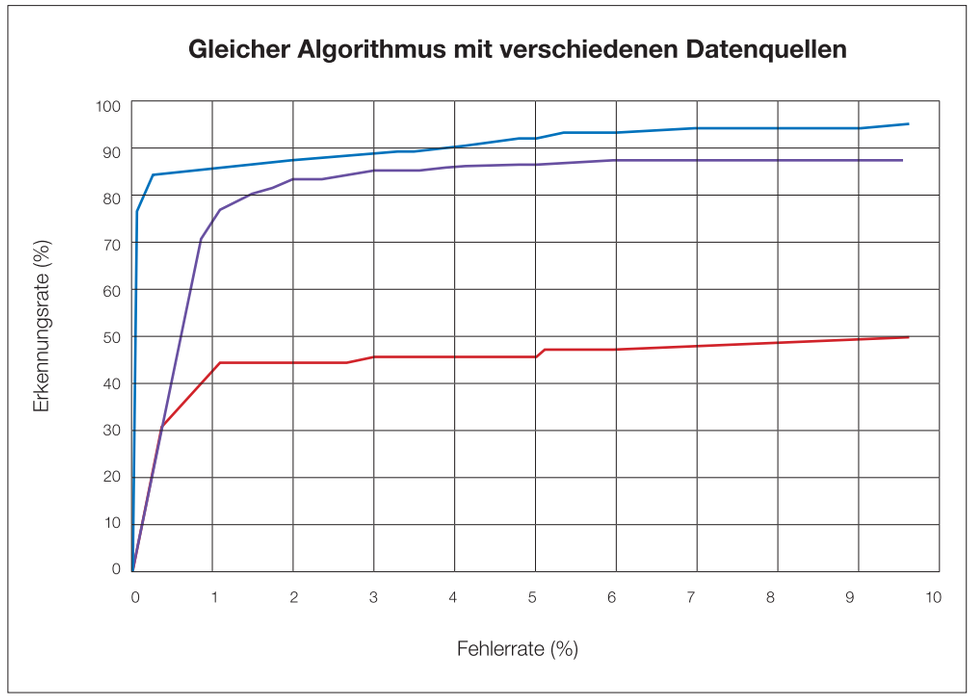


Bild 4: Der Algorithmus hat bei einer Evaluation mit verschiedenen Datenquellen erhebliche Schwankungen in der Performance.

‚Vergleichsdatensätze' gibt[6]. Die Auswirkungen der Vielfalt der eingesetzten Datensätzen werden in Bild 4 deutlich.
Der dargestellte Plot zeigt die Anwendung desselben Algorithmus auf zufällig ausgewählten Daten aus drei Datenbanken. Die Erkennungsrate ergibt erhebliche Unterschiede. Diese Erkenntnis führte dazu, dass ein modulares Benchmarking-System implementiert wurde. Damit können Daten aus verschiedenen Datenbanken parallel verwendet sowie Algorithmen und Modelle aus der Literatur auf denselben Daten getestet und vergliechen werden.
Die einheitlichen Schnittstellen der Algorithmen bieten noch weitere allgemeine Vorteile: Mehrere Nutzende arbeiten gleichzeitig an eigenen Algorithmen und Modellen. Darüber hinaus können die Datenverarbeitung und -annotation iterativ verbessert werden. Die Plugin-Architektur ermöglicht an dieser Stelle die nahtlose Integration von optimierten Methoden zur Fehlererkennung in den Daten. Dieser modulare Ansatz fördert die Zusammenarbeit und trägt dazu bei, die Qualität und Vergleichbarkeit der Forschungsergebnisse zu verbessern.

Frühgeborene mit Atemproblemen profitieren

Die zuvor beschriebene Toolchain wird seit 2017 erfolgreich mit verschiedenen Partnern, unter anderem in der Uniklinik RWTH Aachen eingesetzt. Die Methoden zur Erkennung von Krankheiten und Komplikationen konnte bereits erfolgreich vorangetrieben werden.
Im Schnitt arbeiten aktuell sieben Personen an der Weiterentwicklung der Software. Aufgrund ihrer Arbeit konnten Ansätze zur Erkennung von akutem Lungenversagen, basierend auf Sensordaten und Röntgenbildern, sowie zur Beatmung von Frühgeborenen auf der Intensivstation, untersucht und vergleichen werden. Ein Fokus liegt auf der Beatmung von Frühgeborenen. Das Beatmungsgerät misst Atemfluss- und Atemdruckkurven in einer Auflösung von 125Hz, hieraus wird in einem Modul der Forschungssoftware unter Berücksichtigung der Messungenauigkeiten das Atemvolumen abgeleitet. Außer der Bestimmung des Atem-Zeit-Volumens ist die Identifikation von einzelnen Atemzügen aus den Fluss- und Druckkurven relevant. Durch festgelegte Regeln werden die einzelnen Atemzüge getrennt und annotiert beziehungsweise zur weiteren Annotation durch das medizinische Personal vorbereitet. Ein Beispiel hierfür ist die Erkennung von Asynchronitäten zwischen dem Bedarf der Frühgeborenen und der von der Maschine gelieferten Beatmungsunterstützung. Die frühe Erkennung von Asynchronitäten kann den Behandlungsstress für die Frühgeborenen reduzieren. Mithilfe von manuell annotierten Atemzügen, wurde eine KI-basierte Methode entwickelt, die zur automatisierten Erkennung der Komplikation genutzt wird.
Da die Ethik in der Medizininformatik eine entscheidende Rolle spielt, wurden die Daten in Übereinstimmung mit ethischen Standards, nach Zustimmung durch die jeweiligen Ethikkommissionen verwendet[7] und unter anderem pseudonymisiert. Ein weiteres Anliegen ist die Vermeidung von Vorurteilen. Daher ist es erklärtes Ziel medizinische Entscheidungsunterstützungssysteme fair und gerecht zu gestalten. Vorurteile in den Daten und Algorithmen sollen daher erkannt und eliminiert werden, um sicherzustellen, dass die Modelle für alle Patientengruppen zuverlässig funktionieren beziehungsweise ausreichend ausdifferenziert sind.

Literatur

[1] Johnson, A., Bulgarelli, L., Pollard, T., Horng, S., Celi, L. A., Mark, R., MIMIC-IV, 2022, https://doi.org/10.13026/rrgf-xw32
[2] Maier, C., Lang, L., Storf, H., Vormstein, P., Bieber, R., Bernarding, J., Herrmann, T., Haverkamp, C., Horki, P., Laufer, J., Berger, F., Höning, G., Fritsch, H. W., J. Schüttler, Ganslandt, T. P. H. U., Sedlmayr, M., Towards Implementation of OMOP in a German University Hospital Consortium, Applied Clinical Informatics, Bd. 9, Nr. 1, pp. 54-61, 2018
[3] Wang, X., Wang, C., Time Series Data Cleaning: A Survey, IEEE Access, Bd. 8, pp. 1866 - 1881, 2019
[4] Kompaniyets, L., Goodman, A. B., Belay, B., Freedman, D. S., Sucosky, M. S., Lange, S. J., Gundlapalli, A. V., Boehmer, T. K., Blanck, H. M., Body mass Index and risk for COVID-19–Related hospitalization, intensive care unit admission, invasive mechanical ventilation, and death — United States, March–December 2020. Morbidity and Mortality Weekly Report, morbidity and mortality weekly report, Bd. 70, Nr. 10, p. 355–361, 2021
[5] Budd, S., Robinson, E. C., Kainz, B., A survey on active learning and human-in-the-loop deep learning for medical image analysis, Medical Image Analysis, Bd. 71, 2021
[6] Kelly, C. J., Karthikesalingam, A., Suleyman, M., Corrado, G., King, D., Key challenges for delivering clinical impact with artificial intelligence, BMC medicine, Nr. 17, pp. 1-9, 2019
[7] Reinhart, L., Strathmann, M., Ethische KI: Ein Leitfaden für verantwortungsvolle KI-Entwicklung, Heise Medien, 22 11 2023, https://heise.de/-9312403

Autoren

Alexander Kruschewsky, M.Sc., Camelia Oprea, M.Sc., und Dr.-Ing. André Stollenwerk sind wissenschaftliche Mitarbeitende am Lehrstuhl Embedded Software (Informatik 11). Privatdozent Dr.med. Mark Schoberer ist Oberarzt der Klinik für Kinder- und Jugendmedizin (Sektion Neonatologie und Pädiatrische Intensivmedizin) der Uniklinik RWTH Aachen.

Marc S. Boxberg, Nino Menzel, Florian Wagner

Geophysik lässt tief blicken

Abbildung von Strukturen und Prozessen im Untergrund mit pyGIMLi

A better understanding of the world below our feet is key to answer many pressing topics of our time, such as the localization of groundwater, geothermal and lithium resources, safe and long-term storage of radioactive waste and carbon dioxide, as well as reliable monitoring of these storage facilities. The scientists at the Teaching and Research Unit Geophysical Imaging and Monitoring are developing geophysical measurement methods and software solutions to image structures and processes in the subsurface at high spatial and temporal resolution. By means of a combination of different geophysical methods, subsurface structures and properties can be imaged more clearly and with less ambiguity in their interpretation. The open-source software pyGIMLi is an essential tool for this purpose. It enables the numerical solution of equations that describe the physics of various geophysical measurement methods as well as the estimation of subsurface structure and physical properties from data measured at the Earth's surface. It also provides capabilities to visualize and interpret the generated images and is widely used in geophysical research and education.

Wo befinden sich Grundwasser-, Erdwärme- und Lithiumressourcen? Wo lassen sich radioaktive Abfälle und Kohlenstoffdioxid langfristig sicher speichern und wie können diese Speicher verlässlich überwacht werden? Wie wirken sich Extremwetterereignisse auf die Hangstabilität in Bergregionen aus? Zahlreiche drängende Fragen unserer Zeit erfordern ein verbessertes Verständnis des Untergrunds. Die Wissenschaftlerinnen und Wissenschaftler des im Februar 2023 gegründeten Lehr- und Forschungsgebiets Geophysikalische Bildgebung und Prozessbeobachtung entwickeln dazu geophysikalische Messverfahren und Softwarelösungen, um Strukturen und Prozesse im Untergrund mit hoher räumlicher und zeitlicher Auflösung abzubilden.

Einblick ohne Eingriff

Die Geophysik benutzt und entwickelt dabei Methoden, die in ähnlicher Form aus der Medizin bekannt sind. Bis zur Entdeckung von Wilhelm Conrad Röntgen im Jahr 1895 und den darauffolgenden Entwicklungen in der medizinischen Tomographie musste man den menschlichen Körper chirurgisch öffnen, um ihn zu untersuchen. Analog dazu erlaubt auch

Bild 1: Professor Florian Wagner mit Studierenden des Masterstudiengangs „Applied Geosciences“ bei der Qualitätskontrolle während einer geoelektrischen Messung.
Foto: Peter Winandy

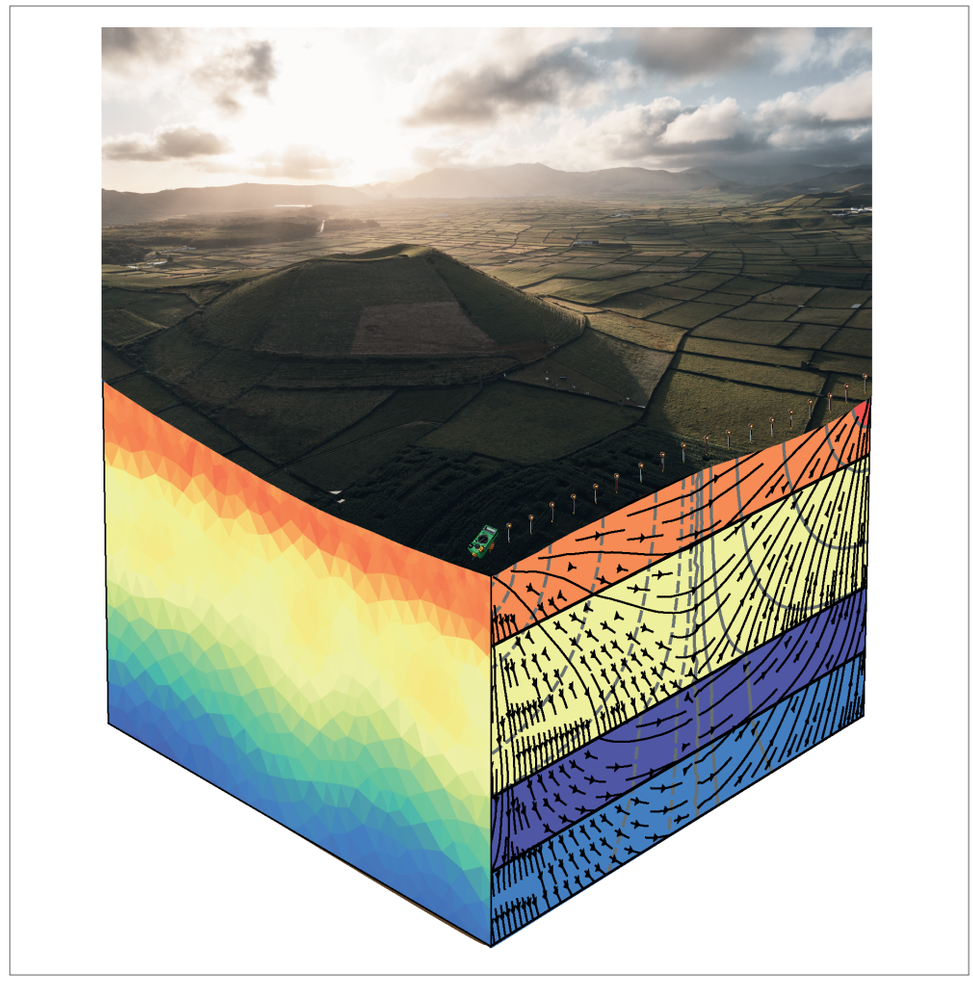


Bild 2: Veranschaulichung der geoelektrischen Untergrundabbildung. Unten rechts: Vereinfachtes geologisches Schichtmodell durch das ein elektrischer Strom fließt (schwarze Linien) und ein daraus resultierendes elektrisches Potentialfeld (graue Linien), welches an der Erdoberfläche mit Elektroden gemessen wird. Aus diesen Messungen lässt sich ein Abbild des Untergrunds rekonstruieren (unten links).

die geophysikalische Tomographie Einblicke in den Untergrund ohne invasive Eingriffe wie kostenintensive Bohrungen. Dabei werden beispielsweise elektrische Ströme gezielt an der Erdoberfläche angeregt, siehe Bild 2. Deren Ausbreitung im Untergrund ist abhängig von den elektrischen Eigenschaften der Gesteinsschichten, insbesondere von den in Poren und Klüften des Gesteins befindlichen Flüssigkeiten und Gasen. Sobald bei der Beobachtung klar wird, wie der Untergrund auf die angeregten Felder reagiert, in diesem Fall durch die Messung resultierender elektrischer Spannungen an der Erdoberfläche, kann die räumliche Verteilung physikalischer Gesteinseigenschaften abgebildet werden. Durch Wiederholung solcher Messungen zu späteren Zeitpunkten lassen sich auch Veränderungen im Untergrund, beispielsweise aufgrund von Fließ- und Transportprozessen in durchlässigen Gesteinsschichten, detektieren und lokalisieren. Ein Beispiel ist die in Bild 3 gezeigte Zunahme des elektrischen Widerstands aufgrund der Ausbreitung von Kohlenstoffdioxid im Speichergestein am ehemaligen Pilotstandort für CO_2-Speicherung in Ketzin, Brandenburg[1].

Viel Physik hilft viel

Ein häufig auftretendes Problem bei geophysikalischen Untersuchungen besteht darin, dass die gewonnenen Abbildungen Unschärfe aufweisen und diese in der Regel mit der Tiefe zunimmt. Darüber hinaus ist die Interpretation der Abbildungen oftmals mehrdeutig. Ist der Porenraum des Gesteins mit Luft oder Eis gefüllt? Beides würde sich im Vergleich zu einer Füllung mit Wasser in einer geringeren elektrischen Leitfähigkeit widerspiegeln. Ein effektiver Ansatz, um geophysikalische Abbildungen zu verbessern und Mehrdeutigkeiten bei deren Interpretation zu reduzieren, besteht darin, verschiedene geophysikalische Messverfahren in einer gemeinsamen Auswertung zu kombinieren[2]. In Ergänzung zu geoelektrischen Verfahren können seismische Wellen angeregt und die dadurch erzeugten Bodenbewegungen aufgezeichnet werden. Da sich seismische Wellen in Eis deutlich schneller ausbreiten als in Luft, kann in dem zuvor genannten Beispiel durch die Kombination der beiden Methoden die Abschätzung von Eis-, Wasser- und Luftanteilen im Porenraum des Gesteins verbessert werden[3]. Dies ist unter anderem für die Abschätzung von Georisiken und die zukünftige Wasserversorgung in alpinen Regionen im Hinblick auf steigende Temperaturen relevant.

Geophysik braucht Software

Der Einsatz von Computern hat die klassische Röntgentomographie in der Medizin revolutioniert und zu der leistungsfähigeren Computertomographie geführt. Auch die dreidimensionale Abbildung des Untergrunds

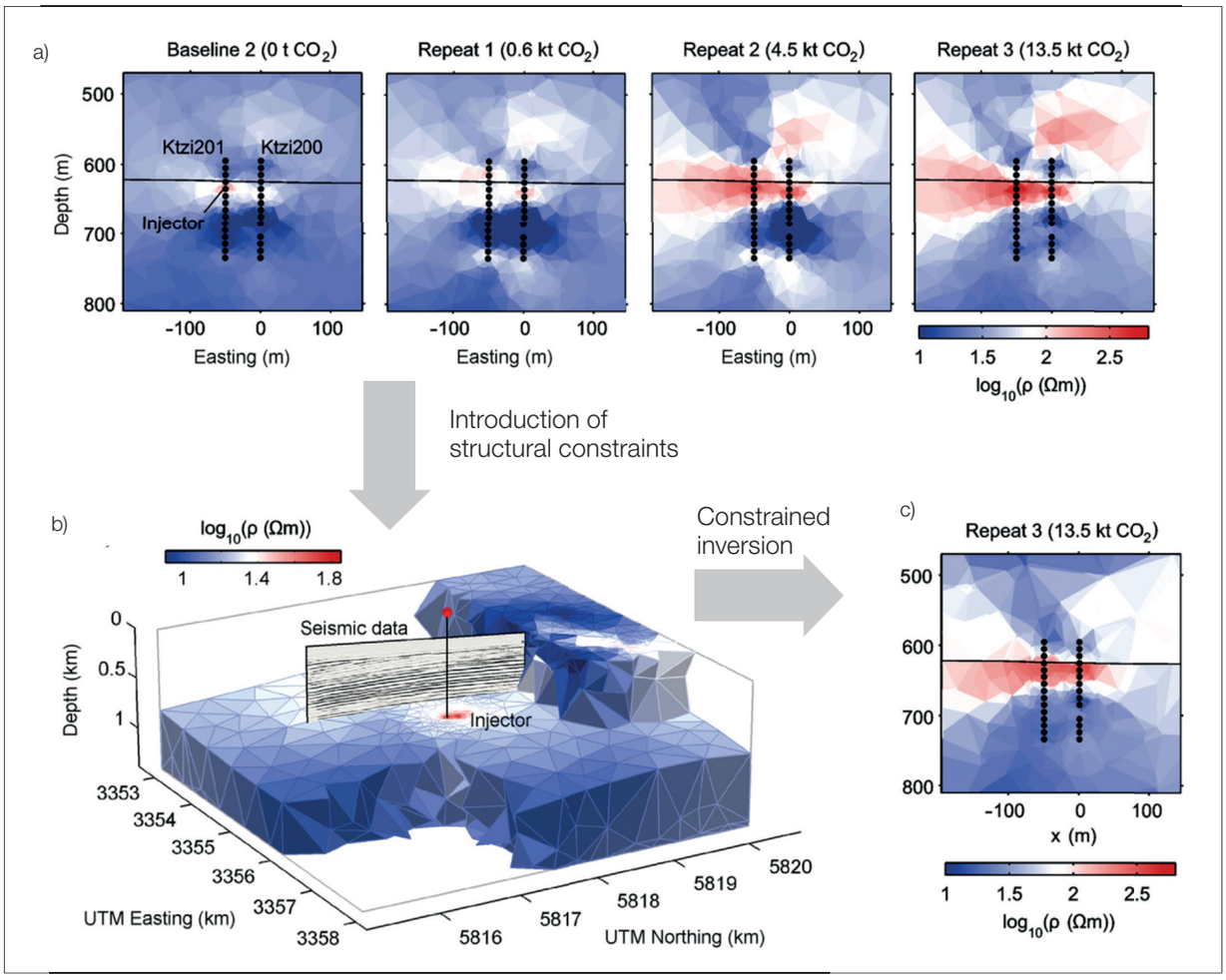


Bild 3: Geophysikalisch bestimmte Änderung des elektrischen Widerstands im CO_2-Speicher am ehemaligen Pilotstandort in Ketzin, Brandenburg.[1]

mit verschiedenen geophysikalischen Verfahren wäre ohne leistungsfähige Computer und vielseitige Softwarelösungen nicht denkbar. Denn es bedarf dazu unter anderem (1) der numerischen Lösung mathematischer Gleichungen, welche die zugrundeliegenden physikalischen Prozesse wie den Stromfluss oder die Ausbreitung von Wellen in porösen Medien beschreiben[4], (2) Inversionsalgorithmen, um aus den gemessenen Daten Untergrundmodelle abzuschätzen, welche die erhobenen Daten im Rahmen ihrer Messfehler erklären können und (3) Werkzeugen zur Visualisierung und Interpretation der erzeugten Abbildungen. Diese Arbeitsschritte ermöglicht die Forschungssoftware pyGIMLi[5].

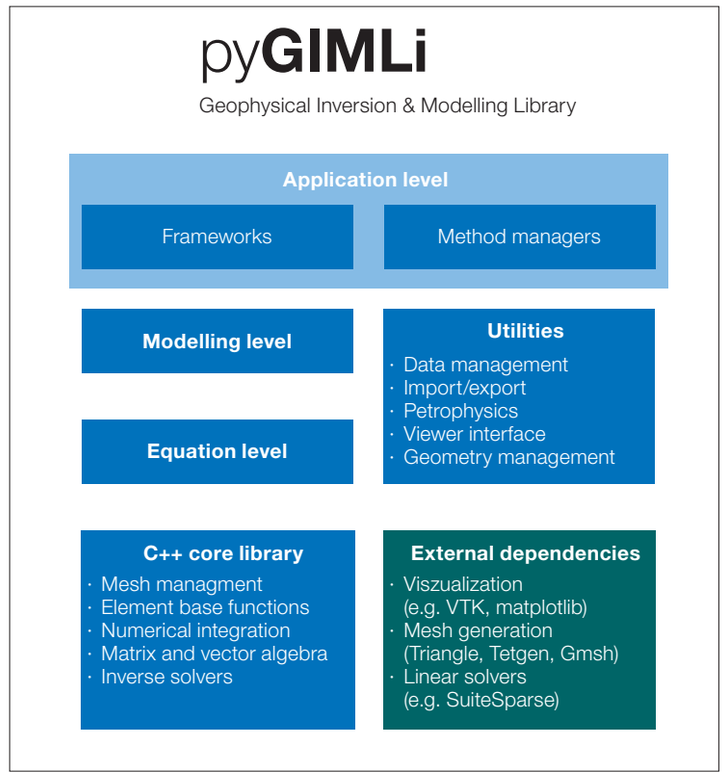


Bild 4: Struktur und Inhalt des Softwarepakets pyGIMLi[5], das am Lehr- und Forschungsgebiet Geophysikalische Bildgebung und Prozessbeobachtung mitentwickelt wird.

Bild 5: Studierende des Bachelorstudiengangs „Angewandte Geowissenschaften“ bei einer magnetischen Messung im Rahmen einer Geländeübung im April 2024.
Foto: Peter Winandy

pyGIMLi – Eine quelloffene Bibliothek zur Modellierung und Inversion

pyGIMLi erlaubt die Auswertung und Kombination zahlreicher geophysikalischer Messdaten. Zeit- und speicherintensive Prozesse, wie das Lösen großer Gleichungssysteme, werden aus Effizienzgründen in einer in C++ geschriebenen Kernbibliothek abgedeckt. Darauf fußt eine Python-Bibliothek, die den Nutzerinnen und Nutzern über entsprechende Schnittstellen die gesamte Funktionalität der Kernbibliothek in Python zur Verfügung stellt und diese um weitere Funktionalität, beispielsweise zur Datenverarbeitung und -visualisierung, ergänzt. Da Python im Gegensatz zu C++ zur Laufzeit interpretiert wird und keine vorherige Kompilierung benötigt, lassen sich geophysikalische Arbeitsabläufe schnell und einfach umsetzen.

Die Software ist in drei Zugangsebenen gegliedert, siehe Bild 4. Die meisten Nutzerinnen und Nutzer arbeiten auf der Anwendungsebene (Application level), die sogenannte Methodenmanager für alle gängigen geophysikalischen Verfahren bereitstellt. Dies sind Klassen, die alle Arbeitsschritte vom Einlesen und Prozessieren der Messdaten bis hin zur Erstellung und Visualisierung eines Untergrundmodells durchführen können. Fortgeschrittene können auf der Modellierungsebene (Modelling level) Funktionen nutzen, die verschiedene geophysikalische Prozesse modellieren und so eigene Auswerteverfahren, beispielsweise zur Kombination von mehreren Methoden, entwickeln. Erfahrene Nutzerinnen und Nutzer können auf der Gleichungsebene (Equation level) verschiedene partielle Differentialgleichungen auf beliebigen Geometrien lösen, was nicht

Literatur

[1] Bergmann, P., Diersch, M., Götz, J., Ivandic, M., Ivanova, A., Juhlin, C., Kummerow, J., Liebscher, A., Lüth, S., Meekes, S., Norden, B., Schmidt-Hattenberger, C., Wagner, F. M., Zhang, F., Geophysical Monitoring of CO2 Injection at Ketzin, Germany, In Geophysical Monitoring for Geologic Carbon Storage (pp. 403–438), 2022, Wiley. https://doi.org/10.1002/9781119156871.ch23

[2] Wagner, F. M., Uhlemann, S., An overview of multimethod imaging approaches in environmental geophysics. Advances in Geophysics, Vol. 44., 2021, https://doi.org/10.1016/bs.agph.2021.06.001

[3] Wagner, F. M., Mollaret, C., Günther, T., Kemna, A., Hauck, C., Quantitative imaging of water, ice and air in permafrost systems through petrophysical joint inversion of seismic refraction and electrical resistivity data. Geophysical Journal International, Vol. 219(3), 2019, https://doi.org/10.1093/gji/ggz402

[4] Boxberg, M. S., Prévost, J. H., Tromp, J., Wave Propagation in Porous Media Saturated with Two Fluids, Transport in Porous Media, Vol. 107(1)., 2015, https://doi.org/10.1007/s11242-014-0424-2

[5] Rücker, C., Günther, T., Wagner, F. M., pyGIMLi: An open-source library for modelling and inversion in geophysics, Computers & Geosciences, Vol. 109., 2017, https://doi.org/10.1016/j.cageo.2017.07.011

nur für die Geophysik nützlich sein kann. Die Software ist für alle gängigen Plattformen frei verfügbar und unter www.pygimli.org dokumentiert. Sie wird daher auch an vielen Standorten in der geophysikalischen Hochschullehre eingesetzt. Seit der Publikation der Version 1.0 im Jahr 2017[5] wurde die Software über 25.000-mal heruntergeladen und von einer breiten und internationalen Nutzergemeinschaft in rund 100 wissenschaftlichen Publikationen verwendet. Der Quellcode ist offen und unterstützt so Forschende dabei, reproduzierbare Wissenschaft zu betreiben und damit den Weg von gemessenen Daten bis hin zu den gewonnenen Untergrundmodellen nachvollziehbar zu machen. Dies dient nicht nur der Transparenz, welche insbesondere bei der Endlagersuche von großer Bedeutung ist, sondern beschleunigt auch den wissenschaftlichen Fortschritt.

www.pygimli.org

Autoren

Dr.rer.nat. Marc S. Boxberg ist wissenschaftlicher Mitarbeiter und stellvertretender Leiter des Lehr- und Forschungsgebiets Geophysikalische Bildgebung und Prozessbeobachtung.

Nino Menzel, M.Sc. ist wissenschaftlicher Mitarbeiter am Lehr- und Forschungsgebiet Geophysikalische Bildgebung und Prozessbeobachtung.

Univ.-Prof. Dr.sc. Florian Wagner ist Leiter des Lehr- und Forschungsgebiets Geophysikalische Bildgebung und Prozessbeobachtung.

Stefan Blügel, Gregor Michalicek, Daniel Wortmann

Dichtefunktionaltheorie auf dem Weg zum Exascale-Computing

Entwicklung des FLEUR Community Codes für sukzessive Generationen von Supercomputern

The continuous development of simulation software in materials physics and science is an essential requirement for technological progress in many fields relevant to enabling the transformation, safety, health and prosperity of our societies. In particular, sustaining the functionality provided by software over long periods of time while at the same time adopting new functionalities and adapting to new computer architectures, is a key challenge in research software engineering. Using the example of the FLEUR code, developed at Forschungszentrum Jülich, we discuss some aspects of this task and highlight the specific redesign needed to make it ready for use on the Exascale computer JUPITER to be installed at Forschungszentrum Jülich.

Computersimulationen sind eine unverzichtbare Methodik in der modernen Wissenschaft geworden. Eine besondere Stellung nehmen dabei solche Simulationen ein, die nicht nur der Analyse, der Charakterisierung oder dem Verständnis des zu beschreibenden Systems dienen, sondern die auch die Vorhersage von gegebenenfalls nützlichen Eigenschaften unter bestimmten Bedingungen gestatten. Im Bereich der Physik der Materialen sind dies vor allem die sogenannten ab-initio Simulationen, die nur auf der Grundlage der fundamentalen Gesetzmäßigkeiten der Quantenmechanik und ohne die Verwendung von empirischen Modellen komplexe Materialeigenschaften beschreiben. Diese Simulationen ermöglichen so die gezielte Suche nach Materialen für gewünschte Funktionalitäten und Anwendungen. Die erfolgreichste Methode hierbei ist die sogenannte Dichtefunktionaltheorie. Mit dieser relative jungen Methodik kann die elektronische Struktur, also die fundamentale Wechselwirkung der vielen Elektronen eines Materials beschrieben

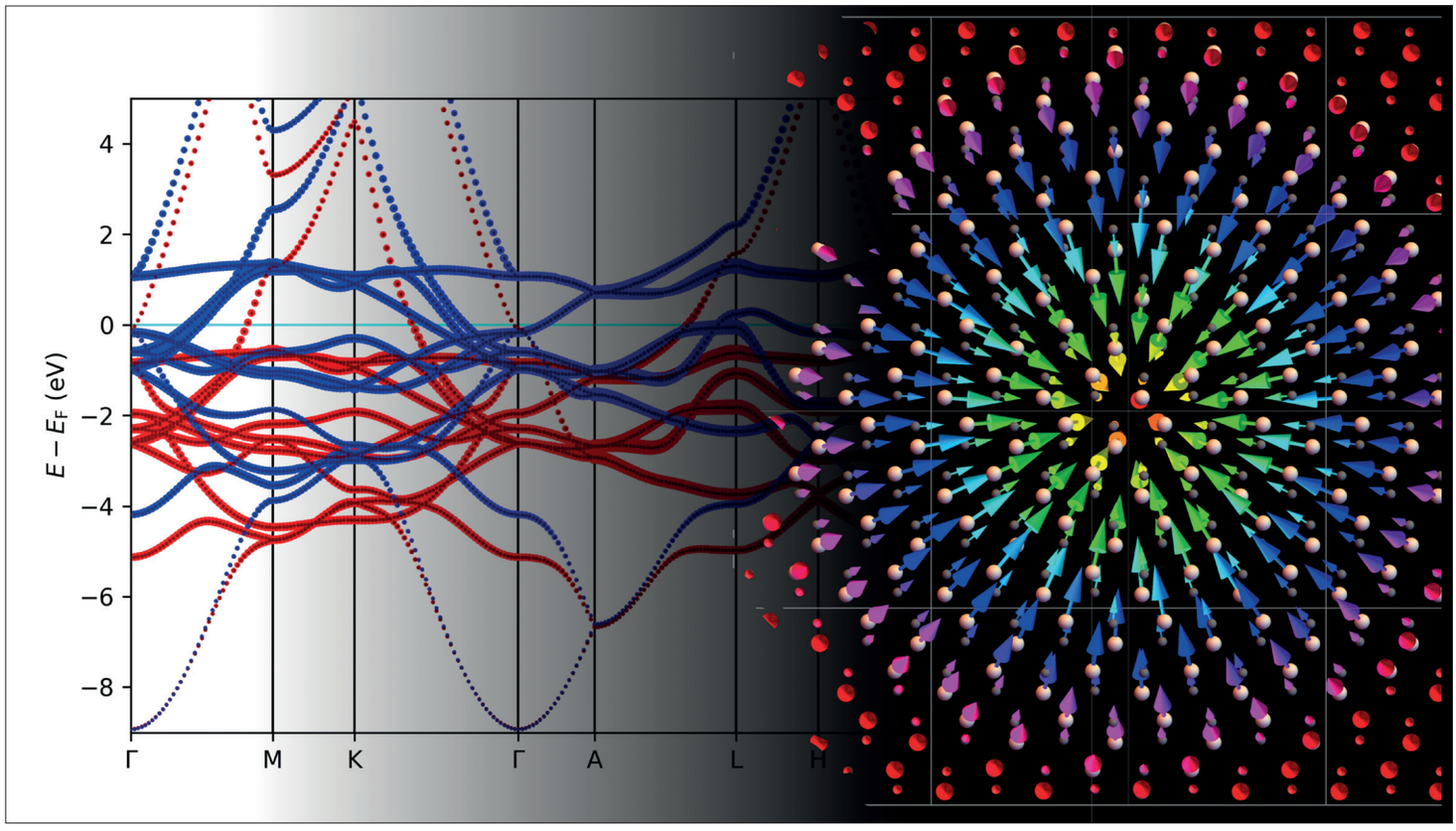


Bild 1: Links: Darstellung der simulierten elektronischen Struktur eines magnetischen Materials. Rechts: Komplexe magnetische Nanostruktur, ein Skyrmion, mit speziellen topologischen Eigenschaften wie sie auf Supercomputern und dem kommenden Exascale Rechner mit FLEUR simuliert werden können.

werden. Auf dieser Grundlage lassen sich dann sowohl makroskopische Materialeigenschaften, etwa Härte, elastisches Verhalten oder Leitfähigkeit und Farbe berechnen, oder auch komplexere Quanteneffekte in funktionalen und exakt definierten Nanostrukturen, beispielsweise kleinste magnetische Objekte oder Grenzflächen simulieren[1]. Solche Quanteneffekte sind die Grundlage neuer technologischer Entwicklungen. Hierzu gehören Quantenmaterialien, Quantencomputer oder energieeffizientere Elektronik. Materialien, die entsprechende Effekte zeigen, sind die Bausteine zukünftiger Assistenzsysteme, Internet- oder Energieinfrastruktur.

Dichtefunktionaltheorie wird in vielfältiger Weise in Software oder, präziser gesagt, in Software-Infrastrukturen umgesetzt. Die Nutzung von Linearized Augmented Plane Waves, kurz LAPWs, zur Darstellung der quantenmechanischen Wellenfunktionen ermöglicht es, mit wenigen und zudem kontrollierbaren Näherungen bei der Implementierung auszukommen. Der quelloffene Code FLEUR[2] gehört auf dieser Basis zu einem der präzisesten Simulationsprogramme für Grundzustandseigenschaften von dreidimensionalen Kristallen, zweidimensionalen Filmen und Oberflächen[3]. FLEUR wird insbesondere für die Forschung an komplexen, nichtkollinearen magnetischen Strukturen, sowie der Auswirkungen der Spin-Bahn-Wechselwirkung auf Material- und Transporteigenschaften genutzt. Zum Einsatz kommt der Code auf vielfältiger Computerhardware von Notebooks bis hin zu Supercomputern.

Nachhaltigkeit in der Softwareentwicklung

Eine der Herausforderungen wissenschaftlicher Softwareentwicklung ist die Notwendigkeit, das entstehende Programm nachhaltig zu gestalten. Die Entwicklung von FLEUR erfolgt seit über 30 Jahren in Fortran und basierte zunächst auf einem Vorgängerprogramm. Solch eine lange Nutzungs- und Entwicklungszeit ist in einer Welt ständiger Innovationen im Bereich der Softwareentwicklung ungewöhnlich. Der Hauptgrund hierfür liegt in der Komplexität der verwendeten Methoden, die jede Weiterentwicklung zu einer eigenständigen Forschungsarbeit auf dem Niveau einer Doktor- oder Masterarbeit macht. Es ist nicht möglich, ein derartig komplexes Simulationsprogramm schnell neuzuschreiben oder massiv zu verändern. Gleichzeitig ist es aber notwendig, mit den massiven Änderungen des Kontextes, in dem Software existiert, schrittzuhalten. Über einen Entwicklungszeitraum von mehreren Jahrzehnten ändern sich beispielsweise die Computerarchitekturen, es kommen neue Programmierparadigmen auf und die Anforderungen der Nutzerinnen und Nutzer an die Software wandeln sich. Nachhaltige Softwareentwicklung muss derartige Einflüsse aufnehmen und so umsetzen, dass die Funktionalität der Software erhalten bleibt, neue Fragestellungen behandelt und die Weiterentwicklungen der Hardware genutzt werden können.

So ist die typische Architektur der schnellsten verfügbaren Rechner, der Supercomputer, wie sie sowohl an der RWTH als auch am Forschungszentrum Jülich betrieben werden, einem stetigen Wandel ausgesetzt. Während vor über 30 Jahren diese Rechner ihre Rechenleistung noch wenigen extrem leistungsfähigen Prozessoren verdankten, wurden schon in den 1990er-Jahren Rechner eingeführt, die hunderte solcher Prozessoren nutzen. Diese konzeptionelle Änderung erfordert eine massive Anpassung der Software, denn zur Nutzung solcher Computer mit vielen Recheneinheiten ist es nötig, das zu lösende Problem in entsprechend viele Unterprobleme zu zerlegen. Diese sogenannte Parallelisierung erfordert also viele Aufgaben, die gleichzeitig und möglichst unabhängig voneinander bearbeitet werden können. Hier erforderte die nachhaltige Weiterentwicklung von FLEUR nicht nur eine entsprechende Anpassung, sondern auch die Wahl entsprechender Software-Bibliotheken und die Verwendung möglichst offener Standards der parallelen Programmierung. Einerseits kann so von der rasanten Hardwareentwicklung profitiert werden, andererseits aber ergibt sich keine Abhängigkeit von einer bestimmten, üblicherweise schnell überholten, Hardwarearchitektur.

Heute stehen die Supercomputer an der Schwelle zum sogenannten Exascale-Computing, diese Rechner können 10^{18} Rechenoperationen pro Sekunde ausführen. Dies wird in einem Zusammenschluss von mehreren tausend Rechenknoten erreicht, bei dem jeder mehrere, ebenfalls hochparallele Grafikprozessoren bereitstellt. Im Vergleich zu vorherigen Supercomputern ist insbesondere die Einbeziehung der Grafikprozessoren eine fundamentale Neuerung, die die Parallelität auf den einzelnen Rechenknoten nochmals auf ein anderes Niveau bringt. Während konventionelle Prozessoren in Supercomputern einige zehn Rechenkerne enthalten, sind in Grafikprozessoren über tausend parallel verfügbare Recheneinheiten vorhanden.

Die Nutzung der Parallelität auf den verschiedenen Stufen der Supercomputerhardware erfordert eine mehrstufige, hybride Parallelisierung der Software, die auf die jeweilige Hardware abzielt. Eine Parallelisierung über viele Rechenknoten wird in der Software zum Beispiel anders erreicht als eine Parallelisierung über mehrere Rechenkerne auf einem Prozessor oder eine Parallelisierung auf einem Grafikprozessor. Insbesondere in Bezug auf Grafikprozessoren gibt es bei der Entwicklung unterschiedliche Möglichkeiten, um die Software von der Hardware profitieren zu lassen. So gibt es explizite Programmiersprachen, die einen hochkontrollierten Zugang zur Grafikhardware ermöglichen. Diese sind allerdings herstellerspezifisch und somit würde es eine Duplizierung von Programmcode erfordern, um die Software auf unterschiedliche Architekturen zu portieren. Eine andere und nachhaltigere Möglichkeit stellt die Verankerung von Zusatzinformationen im Quellcode der Software dar, die von den Compilern genutzt werden können, um direkt ein ausführbares Programm für die jeweils verfügbare Grafikhardware zu erstellen. In der Entwicklung von FLEUR wurde dieser nachhaltigere Weg sowohl für die Nutzung von Grafikprozessoren, als auch für die Nutzung mehrerer Rechenkerne in konventionellen Prozessoren gewählt[4].

Mit der sich entwickelnden Computerhardware und den mitziehenden Entwicklungen in der Software ist es einerseits möglich, komplexere, früher unerreichbare wissenschaftliche Simulationsfragestellungen zu bewältigen, andererseits lassen sich im Rahmen von Hochdurchsatzstrategien große Parameterräume von ganzen Materialfamilien abtasten, beispielsweise um eine Datenbasis für inverses Materialdesign aufzubauen. Für eine solche multidimensionale Nutzung ergeben sich vielfältige Herausforderungen, die durch entsprechende Softwareentwicklung angegangen werden müssen. So erfordert das Hochdurchsatzschema eine enorme Resilienz der Simulationswerkzeuge bezüglich großer Parameterräume der Rechnung sowie die Organisation von großen Datenströmen. Große Datenmengen werden zur Eingabe genutzt und es entstehen auch große Datenmengen als Ausgabe des Programms. Eine Verarbeitung dieser ist im Gesamten nur automatisiert machbar und erfordert statistische Methoden und den Einsatz künstlicher Intelligenz. Daher wurde in FLEUR die Ein- und Ausgabe der Daten mittels eines formalisierten, erweiterbaren Dateiformats nachhaltig maschinenlesbar umgestaltet, zudem wurde eine Anbindung des Programms an das Automatisierungsframework AiiDA implementiert[5]. Sowohl die Arbeit an wenigen, aber sehr komplexen Simulationsaufgaben, als auch die Arbeit an vielen kleinen, verhindert durch den Arbeits-

aufwand das explizite manuelle Einstellen der Parametrisierung der Rechnungen. Ein Augenmerk wurde deshalb darauf gerichtet, ein automatisiertes Rezept zum Einstellen dieser Parametrisierungen zu entwickeln, um stabil ein genaues Simulationsergebnis zu erhalten[6].

Zusammenfassend erfordert die langfristige Entwicklung einer komplexen Software immer wieder Anpassungen aufgrund der sich wandelnden Realität bezüglich Computerhardware, Softwareentwicklungsmethodik und Nutzung. Diese Anpassungen müssen nachhaltig gestaltet sein. In der FLEUR Entwicklung wird der Nachhaltigkeitsaspekt fokussiert angegangen. Ein Großteil der Entwicklungsarbeit zielt allerdings nur indirekt auf wissenschaftliche Fragestellungen ab und ist somit keine Aufgabe für Master- oder Doktorarbeiten, daher kommen dedizierte wissenschaftliche Softwareentwickler zum Einsatz.

Literatur

[1] Heinze, S., von Bergmann, K., Menzel, M., et al., Spontaneous atomic-scale magnetic skyrmion lattice in two dimensions, Nature Phys 7, 713, (2011), https://doi.org/10.1038/nphys2045

[2] The FLEUR project: https://www.flapw.de; Wortmann, D., Michalicek, G., et al., https://doi.org/10.5281/zenodo.7576163

[3] Lejaeghere, K., Bihlmayer, G., Björkman T., et al., Reproducibility in density functional theory calculations of solids. Science 351, aad3000, (2016), https://doi.org/10.1126/science.aad3000

[4] Redies, M., Michalicek, G., Bouaziz, J., et al., Fast All-Electron Hybrid Functionals and Their Application to Rare-Earth Iron Garnets. Front. Mater. 9 (2022), https://doi.org/10.3389/fmats.2022.851458

[5] Automated workflows for computational science: https://www.aiida.net ; Pizzi, G., et al. Comp. Mat. Sci. 111, 218-230 (2016), https://doi.org/10.1016/j.commatsci.2015.09.013

[6] Bosoni, E., Beal, L., Bercx, M., et al., How to verify the precision of density-functional-theory implementations via reproducible and universal workflows. Nat. Rev. Phys. 6, 45 (2024), https://doi.org/10.1038/s42254-023-00655-3

Autoren

Dr.rer.nat. Gregor Michalicek und Dr.rer.nat. Daniel Wortmann sind wissenschaftliche Mitarbeiter am Peter Grünberg Institut im Forschungszentrum Jülich.

Univ.-Prof. Dr.rer.nat Stefan Blügel ist Inhaber des Lehrstuhls für Theoretische Physik sowie Leiter des Peter Grünberg Instituts im Forschungszentrum Jülich.

Pooja Babu, Charl Linssen, Abigail Morrison, Johanna Senk, José Villamar

NESTML und die Simulation pulsgekoppelter neuronaler Netze mit NEST GPU

Domänenspezifische Modellierungssprache begünstigt Hardware-Beschleunigung

The complex nature of the brain has led to a rich landscape of computational models in neuroscience, ranging from detailed models of individual neurons and synapses to large network models of simplified, interconnected cells. The diversity of models entails a diversity of simulation approaches and, consequently, a variety of simulation tools with different foci have been established.
The NEural Simulation Tool (NEST), for example, has been in continuous development for over 25 years. In addition to the complexity of the models themselves, the range of hardware used to execute them is highly diverse, ranging from laptops to supercomputers and general-purpose compute architectures to FPGAs (Field Programmable Gate Arrays) and biologically inspired neuromorphic hardware.
When all these aspects are considered, a multi-dimensional challenge arises for computational neuroscientists who have to deal with different model definitions, simulation backends, and hardware platforms.
To solve this, the NESTML toolchain provides an accessible, yet powerful, modeling language to define neuron and synapse models with varying levels of detail while remaining agnostic of both simulation backend and hardware platform. Model definitions are then used to automatically generate code specific to the targeted simulation backends and hardware platforms for optimum performance.

Das Gehirn ist ein äußerst komplexes System. Strukturelle Phänomene umfassen räumliche Skalen von einigen zehn Nanometern bis zu einigen Dezimetern und dynamische Phänomene ereignen sich auf Skalen von Bruchteilen von Millisekunden bis hin zu Jahren. Folglich müssen Modellierungswerkzeuge für die Neurowissenschaften flexibel sein und es den Wissenschaftlerinnen und Wissenschaftlern ermöglichen, ohne viel Aufwand Modelle mit einer völlig neuen oder von früheren Modellen angepassten Dynamik zu definieren. Im Bereich pulsgekoppelter neuronaler Netze sind die Nervenzellen, die sogenannten Neuronen, und ihre Verbindungspunkte, die Synapsen, diejenigen Komponenten, die bei der Modellierung am häufigsten verändert werden. Darüber hinaus ist die Simulation des Gehirns sehr rechenintensiv: Eine Simulation auf Zellebene von nur einem Kubikmillimeter menschlicher Hirnrinde umfasst etwa 100.000 Neuronen und eine Milliarde Synapsen. Die gewünschte Flexibilität darf also nicht auf Kosten der Recheneffizienz gehen. Es wird ein Ansatz benötigt, der beiden Anforderungen gerecht wird.

Eine Standardsprache für Neuronen- und Synapsenmodelle

Für diese Herausforderung wird die domänenspezifische Sprache NESTML[1] entwickelt. Sie ermöglicht es Forschenden ihre Modelle ohne eine Ausbildung in Softwareentwicklung zu beschreiben. Aus der Beschreibung in NESTML wird automatisch ausführbarer Code erzeugt: So können die Modelle sofort in Simulationen verwendet werden. Dies gilt selbst auf Hochleistungsrechnern. Durch die Beseitigung der Einstiegshürde von Programmierkenntnissen wird die Computersimulation als Forschungsinstrument einem viel größeren Kreis von Neurowissenschaftlerinnen und Neurowissenschaftlern zugänglich gemacht. NESTML hat eine an die Mathematik angelehnte Syntax, wie sie die Wissenschaftlerinnen und Wissenschaftler gewohnt sind, und verwendet Kategorien und Begriffe aus der Neurowissenschaft. Die Sprache erlaubt die direkte Eingabe von gewöhnlichen Differentialgleichungen und Faltungen, die Beschreibung von Zufallsprozessen, aber auch die Formulierung von ereignisgetriebenen Veränderungen der Zustandsvariablen der Modellkomponenten. Statt einer rein deklarativen Syntax sind auch Anweisungen im Stil der iterativen Programmierung möglich, die in einem regelmäßigen Zeitraster oder durch ein beliebiges Ereignis ausgelöst werden können. NESTML ist also auf dynamische Systeme ausgelegt, die wie das Gehirn eine kontinuierliche Dynamik mit diskreten Ereignissen kombinieren. Die Designprinzipien basieren auf MontiCore[2], einem Programmiergerüst für domänenspezifische Sprachen, das am Lehrstuhl für Software Engineering (Informatik 3) entwickelt wird.
Bei Modellbeschreibungen gibt es viele Grenzfälle zu berücksichtigen, so dass eine präzise

Bild 1: Neuronen- und Synapsenmodelle, die in der Standardsprache NESTML ausgedrückt sind, können mithilfe eines Code-Generierungsansatzes auf verschiedenen Rechnerarchitekturen simuliert werden: zum Beispiel auf CPUs oder GPUs über den Simulator NEST oder sogar auf der neuromorphen Hardware SpiNNaker.
Foto: Peter Winandy

und eindeutige Semantik aller Sprachkonstrukte wichtig ist. Aus diesem Grund ist es sinnvoll, auch bei scheinbar einfachen Modellen standardisierte Beschreibungsformate zu verwenden. Simulation sollte ein zuverlässiges Forschungsinstrument sein, das die Reproduzierbarkeit wissenschaftlicher Ergebnisse gewährleistet. Wiederholte Simulationen desselben Modells müssen bei Verwendung derselben Simulationsplattform identische Ergebnisse liefern. Die Verwendung von deterministischen Pseudo-Zufallszahlengeneratoren ermöglicht stochastisches Verhalten und erlaubt gleichzeitig eine perfekte Replizierbarkeit. Im Gegensatz dazu kann die Simulation desselben Modells auf einer anderen Plattform oder unter Verwendung eines anderen numerischen Gleichungslösers oder einer anderen Simulationsauflösung die Ergebnisse verändern, insbesondere bei Netzwerkmodellen mit chaotischer oder instabiler Dynamik. Kleine numerische Unterschiede können selbst auf statistisch sehr robuste Messwerte gravierende Auswirkungen haben. Jüngste Arbeiten zum Vergleich numerischer Ergebnisse verschiedener Simulationswerkzeuge unterstreichen die Notwendigkeit einer sorgfältigen Zusammenstellung an Tests. NESTML enthält weit über hundert hierarchisch organisierter Tests, die überprüfen, ob durch geeignete Fehlermeldungen eine konsistente Modellbeschreibung eingefordert wird und ob aus abstrakten Modelldefinitionen der entsprechende Code erzeugt wird. Numerische Ergebnisse detaillierter Simulationsläufe werden zudem mit bekannten Referenzwerten abgeglichen.

Die Reproduzierbarkeit von Ergebnissen setzt voraus, dass die ursprüngliche Software – einschließlich verwendeter Bibliotheken und anderer Abhängigkeiten – und gegebenenfalls die ursprüngliche Modellbeschreibungen in Form einer Spezifikation oder eines wissenschaftlichen Artikels in natürlicher Sprache verfügbar sind. NESTML vereinfacht die Reproduzierbarkeit durch die formale Trennung der Beschreibung eines konkreten Modells von allgemeinem Simulationscode.

Effiziente großskalige Simulationen

Definitionen von Neuronen- und Synapsenmodellen sollen generisch sein, das heißt unabhängig von der Simulationsplattform, formuliert werden können. Umgekehrt sollen auch Simulationsplattformen generisch sein, also eine breite Palette von Modellen unterstützen. Die am häufigsten verwendeten Simulationsplattformen unterscheiden sich jedoch nicht nur in ihrem neurowissenschaftlichen Fokus, sondern auch in ihrer Schnittstelle zur Definition ausführbarer Modellbeschreibungen sowie ihrer Soft- und Hardware. Der Simulator NEST[3] eignet sich gleichermaßen für die Simulation kleiner Netzwerke mit einigen tausend Neuronen auf einer Laptop CPU (Central Processing Unit), als auch für große Netzwerke mit Millionen von Neuronen parallel auf vielen CPUs eines hochmodernen Supercomputers wie den Maschinen des Jülich Supercomputing Centres am Forschungszentrum Jülich. Es ist wichtig, großskalige neuronale Netzwerkmodelle mit realistischen Dichten von Neuronen und Synapsen simulieren zu können, da herunterskalierte

Bild 2: Das NEST-Team diskutiert darüber, wie man Benutzerfreundlichkeit, Flexibilität und Recheneffizienz kombinieren kann.
Foto: Peter Winandy

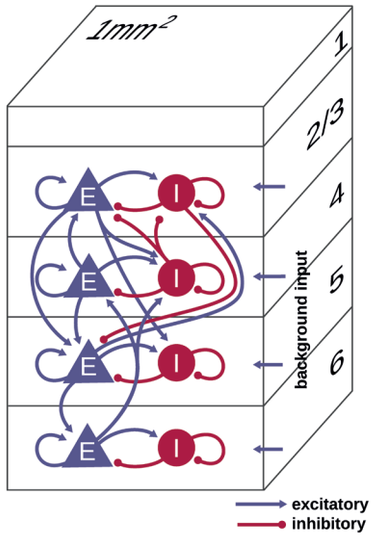

Chari
Jasper
Guido
Catherine Schöfmann
Pooja
Jose Villamar
Melissa Lober
Markus Diesmann
nest::desktop
nest::simulate()
nest::simulate()
nest::gpu
nest::ml

nest::ml

```
neuron iaf_psc_exp:

    state:
        r integer = 0    # Counts number of tick during the refractory period
        V_m mV = E_L     # Membrane potential

    equations:
        kernel I_kernel_inh = exp(-t / tau_syn_inh)
        kernel I_kernel_exc = exp(-t / tau_syn_exc)
        inline I_syn pA = convolve(I_kernel_exc, exc_spikes) * pA -
                          convolve(I_kernel_inh, inh_spikes) * pA
                          (I_syn + I_e + I_stim) / C_m


    # Capacitance of the membrane
    # Membrane time constant
    # Time constant of inhibitory synaptic
```

nest::gpu

```
import nest
import numpy as np
…
for i in np.arange(num_pops):
    population = nest.Create("iaf_psc_exp", num_neurons[i])
    pops.append(population)
…
for i, target_pop in enumerate(pops):
    for j, source_pop in enumerate(pops):
        if num_synapses[i][j] >= 0.0:
            conn_dict = {
                "rule": "fixed_total_number",
                "N": num_synapses[i][j]}
            …
            nest.Connect(
                source_pop, target_pop,
                conn_spec=conn_dict, syn_spec=syn_dict)
```



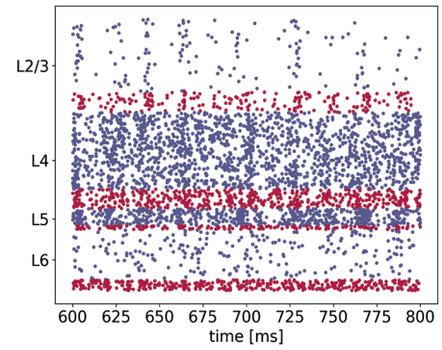


Bild 3: Modellierung und Simulation des kortikalen Netzwerks. Illustration des Netzwerkmodells[6], Bild eines Hochleistungsrechners des Forschungszentrums Jülich und neuronale Aktivität als Simulationsergebnis. Entsprechender NESTML- und PyNEST-Programmcode im Hintergrund.

Netzwerke keine realistische Aktivität wiedergeben können. Wegen der benötigten Systemressourcen übersteigt der Energiebedarf für die Simulation von Gehirnaktivitäten den von natürlichen Gehirnen um mehrere Größenordnungen. Um gleichzeitig den Ressourcenverbrauch zu senken und schneller zu Simulationsergebnissen zu gelangen, wird eine möglichst effiziente Ausnutzung der Maschinen angestrebt. Die meisten der Top-500-Computersysteme und alle kommenden Exascale-Maschinen verwenden neben CPUs auch GPUs (Graphics Processing Units). Daher arbeitet die NEST-Entwicklergemeinschaft an einer Version mit für GPUs optimierten Algorithmen[4]. Ziel ist, dass die Nutzer in naher Zukunft über eine einheitliche Python-Benutzeroberfläche sowohl CPUs als auch GPUs ohne Änderung der Modellbeschreibung nutzen können.

Code-Generierung für verschiedene Rechnerarchitekturen

Der für die Ausführung auf CPUs und GPUs erforderliche Code ist unterschiedlich, was die Gefahr von Doppelarbeit, höheren Wartungskosten und abweichendem Verhalten auf verschiedenen Architekturen birgt. Diese Risiken können durch die Erweiterung des Anwendungsbereichs von NESTML deutlich reduziert werden. Die NESTML-Toolchain analysiert und prüft die Korrektheit des Modells, bevor sie automatisch Code für eine bestimmte Simulationsplattform generiert. Der Code wird in verschiedenen Sprachen erzeugt, die auf unterschiedliche Plattformen und APIs, sogenannten Schnittstellen zur Programmierung von Anwendungen, abzielen: Für die Version des NEST-Simulators, die auf CPU-basierter Hochleistungsrechnerhardware läuft, wird beispielsweise C++-Code generiert; für die Version des NEST-Simulators, die auf GPUs läuft, muss CUDA-C++-Code generiert werden. Der generierte Code wird dann kompiliert oder dynamisch in die Simulationsumgebung geladen, in der die neu erstellten Modelle instanziiert werden und die Simulation ausgeführt wird. Mit der Unterstützung der GPU-Code-Generierung in NESTML können die bestehenden Modelle, die in der NESTML-Sprachsyntax geschrieben sind – derzeit eine Datenbank mit über zwei Dutzend Neuronenmodellen und synaptischen Plastizitätsregeln – auch für GPU-basierte Simulationen verwendet werden. Dies reduziert den Aufwand für die Neurowissenschaftlerinnen und Neurowissenschaftler und hilft Simulationen durchzuführen, ohne bestehende Modelle an neue Plattformen anpassen zu müssen. Die Erweiterung von NESTML für diese neue Simulationsplattform wird in Zusammenarbeit des NESTML-Teams mit den Entwicklern der GPU-fähigen Version von NEST durchgeführt.

Neben der oben beschriebenen Simulation auf CPUs und GPUs kann derselbe Code-Generierungsansatz auch auf exotischeren Plattformen angewandt werden, zum Beispiel auf biologisch inspirierter neuromorpher Hardware, wie dem SpiNNaker-Projekt der Universitäten Manchester und Dresden oder dem FPGA-basierten neuroAIx-Cluster der RWTH. Solche Systeme sind vielversprechend, haben aber auch ihre Grenzen. Für die Entwicklung dienen die NEST-Simulatio-

nen als Referenz für Simulationsergebnisse und Energieverbrauch[5]. Dafür wird ein neurowissenschaftlich relevantes Modell als gemeinsamer Test auf allen Plattformen ausgeführt. Eine spezielle Hardware ist nur sinnvoll, wenn sie in irgendeiner Eigenschaft im Sinne einer Validierung überlegen ist.

Zusätzlich zur Auswahl einer Simulationsplattform können während der Code-Generierung optional mehrere Optimierungen vorgenommen werden, um die Simulationsleistung (Laufzeit oder Speicherverbrauch) zu verbessern. Zum Beispiel werden abhängig von den Modellgleichungen und der Wahl der Parameter mehrere numerische Gleichungslöser getestet, und der optimale Löser für ein bestimmtes dynamisches System wird automatisch ausgewählt. NESTML kombiniert eine Modellierungssprache und eine Code-Generierungsfunktion und bietet daher den Vorteil, dass beides parallel entwickelt werden kann. Die Vereinheitlichung von Modelldefinitionen durch eine spezielle Modellierungssprache hilft der Forschung, indem sie Programmierbarrieren beseitigt und die Interoperabilität zwischen verschiedenen Soft- und Hardwareplattformen durch Code-Generierung ermöglicht. Das gegenseitige Verständnis eines Modells fördert die Zusammenarbeit zwischen Wissenschaftlerinnen beziehungsweise Wissenschaftlern und ihren Forschungsbereichen, was den Forschungsprozess entscheidend beschleunigt. Durch die Etablierung eines benutzerfreundlichen Standards und die Ausweitung auf neue Plattformen ermöglicht NESTML etablierten Nutzerinnen und Nutzern, ihre Arbeit zukunftssicher zu gestalten, und gewährt neuen Nutzerinnen und Nutzern einen schnellen Zugriff auf die umfangreiche Datenbank mit bestehenden Modellen. Mit dem Aufkommen neuromorpher Hardware und der zunehmenden Konzentration von Hochleistungsrechenzentren auf Graphikprozessoren entwickelt sich die Simulationstechnologie stetig weiter, um sich an die verschiedenen Plattformen anzupassen und diese optimal zu nutzen.

Literatur

[1] NESTML-Dokumentation: https://nestml.readthedocs.io

[2] Hölldobler, K., Kautz, O., Rumpe, B., MontiCore Language Workbench and Library Handbook: Edition 2021. Aachener Informatik-Berichte, Software Engineering, Band 48, ISBN 978-3-8440-8010-0, Shaker Verlag, 2021

[3] NEST-Website: https://www.nest-simulator.org

[4] Golosio, B., Villamar, J., Tiddia, G., Pastorelli, E., Stapmanns, J., Fanti, V., Paolucci, P. S., Morrison, A., Senk, J., Runtime Construction of Large-Scale Spiking Neuronal Network Models on GPU Devices. Applied Sciences. 13(17):9598. https://doi.org/10.3390/app13179598, 2023

[5] van Albada, S. J., Rowley, A. G., Senk, J., Hopkins, M., Schmidt, M., Stokes, A. B., Lester, D. R., Diesmann, M., Furber, S. B., Performance Comparison of the Digital Neuromorphic Hardware SpiNNaker and the Neural Network Simulation Software NEST for a Full-Scale Cortical Microcircuit Model. Frontiers in Neuroscience. 12:291. doi: 10.3389/fnins.2018.00291, 2018

[6] Potjans, T. C., Diesmann, M., The Cell-Type Specific Cortical Microcircuit: Relating Structure and Activity in a Full-Scale Spiking Network Model, Cerebral Cortex. 24(3)785–806. doi:10.1093/cercor/bhs35, 2014

Autoren

Pooja Babu und Charl Linssen sind wissenschaftliche Mitarbeitende am Simulation and Data Laboratory des Jülich Supercomputing Centres am Forschungszentrum Jülich.

Univ.-Prof. Dr.rer.nat. Abigail Morrison leitet das Lehr- und Forschungsgebiet Neural Computation und ist Leiterin der Gruppe Computation in Neural Circuits am Institut für Computational and Systems Neuroscience (IAS-6) am Forschungszentrum Jülich.

Dr.rer.nat. Johanna Senk ist Lecturer an der University of Sussex in Großbritannien und Leiterin des Teams Future Simulation Architectures am Institut für Computational and Systems Neuroscience (IAS-6) am Forschungszentrum Jülich.

José Villamar ist wissenschaftlicher Mitarbeiter am Institut für Computational and Systems Neuroscience (IAS-6) am Forschungszentrum Jülich.

Uwe Naumann, Jens Deussen, Markus Towara

Differenzierbare Forschungssoftware

We argue that differentiability of research software must become a fundamental requirement in Research Software Engineering. Error analysis and -control, calibration of free parameters, optimization of the simulated (continuous) real-world scenarios as well as reduction of simulation cost in the context of future-proof high-performance computing rely on methods that are based on first and possibly higher derivatives of the simulation. The crucial co-design of mathematical and software models benefits tremendously from the availability of derivative information. Algorithmic differentiation is usually the method of choice for computing derivatives both accurately and efficiently. Its systematic inclusion into the Research Software Engineering process is expected to facilitate the development of more robust and sustainable research software. Ultimately, novel differentiable model design patterns and programming languages need to be developed. The Jülich Aachen Research Alliance provides an ideal ecosystem for making valuable contributions to this field.

Auf mathematischer Modellbildung basierende numerische Simulation ist eine etablierte Methode für vertieftes Verständnis relevanter Probleme in den Natur-, Lebens- und Ingenieurwissenschaften. Entsprechende Softwarelösungen ermöglichen die quantitative Evaluation der Modelle mithilfe von Computern. Die zugrundeliegende numerische Simulationssoftware ist typischerweise mathematisch und algorithmisch anspruchsvoll, rechenintensiv, sowie approximativ und parametrisiert. Daraus folgt eine Reihe von Anforderungen sowohl an die mathematische Modellierung als auch an den (idealerweise simultan ablaufenden) Entwicklungsprozess für entsprechende Forschungssoftware, welche in der etablierten Softwareentwicklung eine meist weniger zentrale Rolle einnehmen. „Wie stark ändert sich das Resultat meiner Simulation bei Variation der Werte bestimmter freier (Eingabe-)Parameter?“ „Wie sensitiv ist diese Änderung bezüglich Variation von Werten potenziell anderer Parameter?“ Analoge Fragen stellen sich Anwenderinnen und Anwender numerischer Simulationssoftware

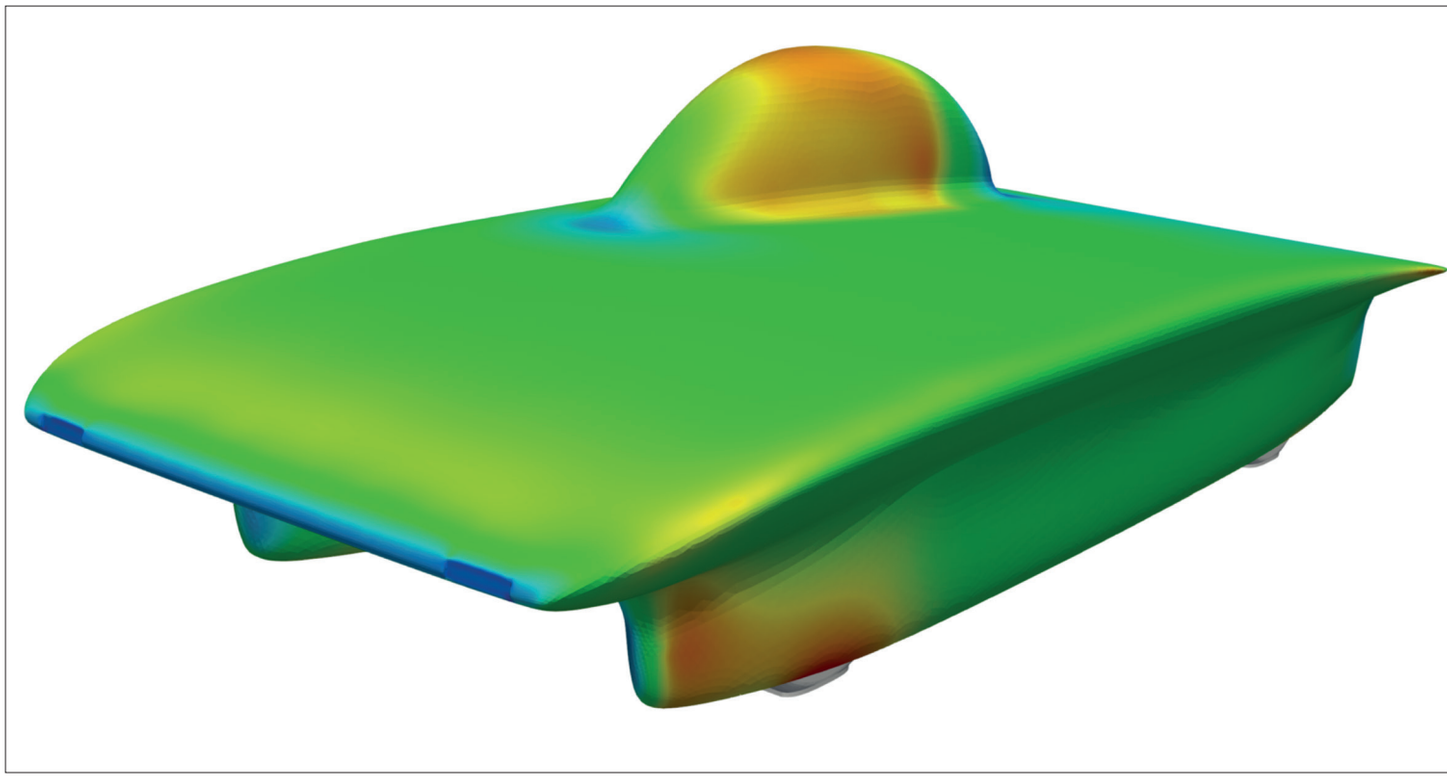


Bild 1: Sensitivität des zu minimierenden Strömungswiderstands eines solarbetriebenen Sonnenwagen 1, www.sonnenwagen.org. Blau dargestellt ist die Expansion, rot die Kompression. Die Resultate wurden von Lennart Moltrecht im Rahmen seiner Masterarbeit mithilfe der algorithmisch adjungierten Version der Strömungssimulation OpenFOAM generiert[3]. Diese wurde am Lehr- und Forschungsgebiet Software und Werkzeuge für Computational Engineering entwickelt und ist als Open-Source-Software verfügbar.

regelmäßig. Zahlreiche numerische Methoden basieren auf der Verfügbarkeit dieser Informationen. Letztere übersetzt sich in Ableitungen erster, zweiter und gegebenenfalls auch höherer Ordnung der simulierten Größen bezüglich der Parameter.
Dieser Artikel stellt Differenzierbarkeit und die Berechnung entsprechender Ableitungen numerischer Simulationssoftware als Anforderung im Research Software Engineering in den Mittelpunkt. Der Fokus liegt dabei auf kontinuierlichen – in Gegensatz zu diskreten – Simulationen. Die zentralen Aussagen gelten sowohl für mechanistische Modelle als auch für rein datengetriebene Ansätze des maschinellen Lernens inklusive hybrider Szenarien, in welchen beide Methoden zu einer Gesamtlösung kombiniert werden. Auf nichtfunktionaler Ebene ermöglichen Ableitungen Fehleranalyse und -kontrolle, Kalibrierung sowie robustere Softwaretests. Die in vielen Fällen essenzielle Transition von reiner Simulation der zugrundeliegenden Systeme hin zu deren (ableitungsbasierter) Optimierung ist oft zentraler Bestandteil der funktionalen Anforderungen. Die Grenze zwischen Funktionalität und Nichtfunktionalität von Differenzierbarkeit als Anforderung verschwimmt zusehends im Kontext digitaler Zwillinge, welche unter anderem Fehlerkontrolle in Echtzeit zum Ziel haben können. In jedem dieser Fälle muss die numerische Simulation differenziert werden. Dafür unterscheidet man grob zwischen zwei Ansätzen. Beim symbolischen Differenzieren werden Ableitungen des mathematischen Modells analytisch hergeleitet. Zumeist handelt es sich dabei um eine anspruchsvolle manuelle Tätigkeit. Die Lösung des differenzierten Modells muss anschließend numerisch approximiert und somit in Form von Forschungssoftware implementiert werden. Die numerische Approximation der Ableitung entspricht meist nicht der korrekten Ableitung der numerischen Approximation des Originalmodells, was eine Reihe von Komplikationen bei der Verwendung dieser Werte im Rahmen von numerischen Methoden nach sich ziehen kann.

Algorithmisches Differenzieren, kurz AD,[1,2] umgeht das oben skizzierte Problem durch Differenzieren des numerischen Simulationsprogramms mittels Kombination bekannter Ableitungen für dessen Elementarfunktionen gemäß der Kettenregel. Dieser Prozess kann zu großen Teilen automatisiert werden und ist somit auch auf sehr komplexe Simulationssoftware anwendbar. Die unter Ausnutzung von Assoziativität der Kettenregel generierten adjungierten Programme stellen Verallgemeinerungen der im Rahmen des maschinellen Lernens essenziellen backpropagation dar. Somit lassen sich erste Ableitungen skalarer Zielgrößen bezüglich einer potenziell sehr großen Anzahl freier Parameter effizient berechnen. Formoptimierung mittels numerischer Strömungsmechanik zählt zu den zahlreichen Anwendungen für adjungierte Methoden, siehe Bild 1.

Im Folgenden soll Differenzierbarkeit im Kontext der zuvor genannten fundamentalen Eigenschaften numerischer Simulationssoftware positioniert werden. Es ergeben sich sowohl Möglichkeiten als auch Herausforderungen für das Research Software Engineering.

Numerische Simulationssoftware ist mathematisch und algorithmisch anspruchsvoll. Sie basiert meist auf Dekaden von Forschung und Entwicklung ganzer Wissenschaftsbereiche. Ein signifikanter Teil menschlicher Expertise ist in ihr „verborgen". Das zentrale Ziel von Research Software Engineering muss daher die Sicherung nachhaltiger Verfügbarkeit dieses „Schatzes" inklusive seiner zukünftigen Erweiterungen sein. Dazu gehört nicht zuletzt auch der möglichst effektive und effiziente Einsatz der „Ressource Mensch" mittels weitestgehender Formalisierung und Optimierung des Entwicklungsprozesses. Jedoch stellen sich bereits die Übersetzung der zugrundeliegenden mathematischen Modelle in numerische Methoden und entsprechende Softwareentwürfe, algorithmische Details, Test- und Evolutionsstrategien als höchst anspruchsvoll dar. Eine rigorose Anforderungsanalyse ist ohne fundiertes Verständnis des modellierten Sachverhalts, der Mathematik inklusive assoziierter Numerik, der Simulationsumgebung bestehend aus Systemhardware und -software, sowie von Methoden der Softwareentwicklung bis hin zur eigentlichen Programmierung nicht möglich. Die Option einer hinreichenden Konzentration dieser Expertise in Individuen darf bezweifelt werden. Konsequenzen für die interdisziplinäre Lehre sollten abgeleitet werden. Zudem stellen sich weitergehende Herausforderungen an die Validierung von Konsistenz des mathematischen Modells und der zugehörigen numerischen Simulationssoftware sowie der Korrektheit letzterer. Zusätzlich zu den Resultaten der Simulation können hier Werte von Ableitungen als weitergehende Evidenz sehr hilfreich sein. Ableitungen können und sollten entsprechende Softwaretest- und kontinuierliche Integrationsstrategien erweitern. Mit jeder Ableitungsordnung wächst potenziell die Robustheit der Validierung. Zu diesem Zweck muss die numerische Simulationssoftware ausreichend oft differenzierbar sein sowie differenziert werden.

Numerische Simulationssoftware ist rechenintensiv

Ein signifikanter Teil der Entwicklungsarbeit fließt traditionell in die Minimierung der durch numerische Simulationen benötigten Ressourcen. Oft agieren diese Simulationen am Rande der Leistungsfähigkeit der jeweils aktuell verfügbaren Computerinfrastruktur mit dem Ziel vertretbarer Laufzeiten bei zulässigem Speicherbedarf. Effektive Vektorisierung, Parallelisierung und/oder Beschleunigung gehören zu den fundamentalen Voraussetzungen für eine effektive Nutzung des modernen IT-Ökosystems. Letzteres unterliegt einer Entwicklungsdynamik, welche die zentrale Rolle von Nachhaltigkeit beim Entwurf und der Umsetzung von numerischen Simulationen sowie von deren Ableitungen nochmals unterstreicht. Seit einigen Jahren und nicht zuletzt getrieben durch ressourcenhungrige Anwendungen des maschinellen Lernens rückt auch die Minimierung des Energiebedarfs immer mehr in den Fokus der Aufmerksamkeit. Zahlreiche Methoden zur Reduktion der Komplexität numerischer Simulationen bei idealerweise nur geringfügigen Abstrichen in deren Aussagekraft basieren auf Ableitungen der zu simulierenden Größen bezüglich einer meist sehr großen Anzahl an Zwischenwerten. Vernachlässigbare Sensitivität über Teilen des Definitionsbereiches erlaubt Modellreduktion mittels Spezialisierung der Originalsoftware. Wenig signifikante Variablen werden zu Konstanten. Etablierte Datenflussanalysen entfernen infolgedessen nicht mehr benötige Teilrechnungen. Die nu-

merische Simulationssoftware muss dafür nicht nur differenziert werden. Es muss auch eine Globalisierung der Ableitungsinformation ermöglicht werden. Intervallarithmetik oder komplexere Relaxationen kommen dafür zum Einsatz.

Numerische Simulationssoftware ist approximativ und parametrisiert

Dem britischen Statistiker George Box wird die auch heute noch in weiten Teilen gültige Aussage „Alle Modelle sind falsch, aber einige sind nützlich“ zugeschrieben. „Alle numerischen Simulationen sind falsch“ folgt unmittelbar. Die „Nützlichkeit einiger“ ist jedoch kein Automatismus. Gerade vor dem Hintergrund der Lösung potenziell schlecht konditionierter Probleme mittels möglicherweise nicht uneingeschränkt stabiler numerischer Algorithmen stellt sich die Frage nach dem „Wie falsch?“ Zahlreiche Verfahren zur numerischen Fehleranalyse und -kontrolle basieren wiederum auf Ableitungen der simulierten Größen bezüglich der zumeist mit substanziellen Unsicherheiten belegten Eingabedaten und -parameter. Relaxation dieser Ableitungen erlaubt die meist höchst wünschenswerte Globalisierung der zunächst nur lokal gültigen Werte. Adaptivität in numerischen Verfahren beruht ebenfalls auf oft hochdimensionalen und damit notwendigerweise adjungierten Ableitungen der jeweiligen Zielgrößen bezüglich Variablen aus deren Definitionsbereichen. Diese resultiert zum Beispiel in lokaler Verfeinerung der Auflösung von Rechengittern oder in der Konzentration der Datengenerierung auf besonders sensitive Teile des Definitionsbereiches beim Training von Surrogatmodellen mit Methoden des maschinellen Lernens. Letzteres ist ein Spezialfall der Kalibrierung – einer potenziell enormen Anzahl – freier Parameter von Forschungssoftware. In jedem dieser Szenarien muss zu diesem Zweck (potenziell adjungiert sowie relaxiert) differenziert werden.

Die Entwicklung differenzierbarer mathematischer Modelle sowie deren Implementierung in Form von differenzierbarer Forschungssoftware stellt ein höchst anspruchsvolles interdisziplinäres Thema an der Schnittstelle von Mathematik, Informatik und einer Vielzahl von Anwendungsgebieten dar. Tendenziell wird diese Tatsache im Rahmen von Research Software Engineering zu neuen differenzierbaren Modellierungstechniken und entsprechenden Softwareentwurfsmustern[5] für die konsistente simultane Entwicklung von mathematischen, numerischen beziehungsweise algorithmischen und Softwaremodellen führen müssen. Ultimativ werden neue massiv parallele und differenzierbare Programmiersprachen und Softwarebibliotheken benötigt. Die Jülich Aachen Research Alliance, ein Verbund der RWTH mit dem Forschungszentrum Jülich, bietet die ideale Umgebung für entsprechende substanzielle Beiträge.

Literatur

[1] Griewank, A., Walther, A., Evaluating Derivatives. Principles and Techniques of Algorithmic Differentiation. Society for Industrial and Applied Mathematics, 2008
[2] Naumann, U., The Art of Differentiating Computer Programs. An Introduction to Algorithmic Differentiation. Society for Industrial and Applied Mathematics, 2012
[3] Towara, M., Naumann, U., A discrete adjoint model for OpenFOAM. Procedia Computer Science 18, 429-438, 2013
[4] Towara, M., Schanen, M., Naumann, U., MPI-parallel discrete adjoint OpenFOAM. Procedia Computer Science 51, 19-28, 2015
[5] Naumann, U., Adjoint code design patterns. ACM Transactions on Mathematical Software 45(3), 1-32, 2019

Autoren

Univ.-Prof. Dr.rer.nat. Uwe Naumann leitet das Lehr- und Forschungsgebiet Software und Werkzeuge für Computational Engineering. Dr.rer.nat. Jens Deussen und Dr.rer.nat. Markus Towara sind wissenschaftliche Mitarbeiter am Lehr- und Forschungsgebiet Software und Werkzeuge für Computational Engineering.

Gabriele Gramelsberger, David Heyen, Dawid Kasprowicz, Frederik Kerksieck, Markus Pantsar, Thomas Venator, Daniel Wenz

Wissenschaftstheoretische Reflexionen zu Research Software Engineering

Konzeptuelle Analyse von Forschungssoftware

At the RWTH Computational Science Studies Lab (CSS-Lab), philosophers of science and technology explore the transformation of science into computational sciences. In particular, the CSS Lab is concerned with how research software can be conceptually analyzed from a philosophical perspective. Against this backdrop, we are developing software tools that enable easier access to software projects.

Wissenschaftstheorie setzt sich mit der Methoden- und Konzeptentwicklung in der Wissenschaft auseinander, analysiert die Theorienbildung und den Erkenntnisfortschritt in den verschiedenen Disziplinen sowie die Verfahren des Forschens. Mit dem Wandel der Wissenschaft in digitale Wissenschaft verändert sich auch das Themenfeld der Wissenschaftstheorie, denn der zunehmende Einsatz von Computern ab den 1970er Jahren transformiert die wissenschaftlichen Methoden der Erkenntnisgenerierung[1].
Zu den klassischen Methoden des Experiments, der Beobachtung und der Theoriebildung, treten nun die Computermodellierung und -simulation, die massive Datenerhebung (Big Data) sowie Datenanalytik hinzu. Seit einiger Zeit kommen Methoden der Künstlichen Intelligenz, insbesondere des Maschinellen Lernens (ML), zum Einsatz. Was all diese neuen Methoden vereint, ist, dass sie softwarebasiert sind. In einem Bericht des US-amerikanischen President's Information Technology Advisory Committee hieß es bereits 1999: „Software is the new physical infrastructure of [...] scientific and technical research." Wissenschaftlerinnen und Wissenschaftler müssen heutzutage programmieren oder mit Software-Programmen und -Tools

avanciert umgehen können, doch wissenschaftliches Programmieren ist zumeist kein zentrales Studienfach für natur- und technikwissenschaftliche Disziplinen. Dies gilt umso mehr für Philosophen und Philosophinnen. In der digitalen Wissenschaft bedeutet es jedoch, sich ausschließlich auf die Entwicklung und Programmierung von Simulationsmodellen, von Algorithmen für die Datenanalyse und -evaluation verlassen zu müssen. Dies wird oft als „Service“ missverstanden und auch in wissenschaftlichen Forschungsanträgen nicht goutiert. Sich professionell mit Research Software Engineering zu befassen, ist daher überfällig.

Mit GICAT Softwareprojekte strukturell analysieren

Am Computational Science Studies Lab des Lehrstuhls für Wissenschaftstheorie und Technikphilosophie wird seit 2018 erforscht, wie wissenschaftliche Software zugänglich und analysierbar sein kann. Im Mittelpunkt steht die Genealogie, also die Entwicklung von Softwareprojekten wie Klimamodelle, Analysesoftware astronomischer Daten oder geologische Visualisierungsmodelle über Jahre[2]. Zum einen, um die konzeptuelle Entwicklung, die in die Software implementiert ist, nachvollziehen zu können. Zum anderen, um den Einfluss der Programmierung auf die wissenschaftlichen Konzepte selbst zu verstehen. Denn die unterschiedlichen Softwareprojekte der Wissenschaften sind in verschiedenen Programmiersprachen und Frameworks verfasst: von Fortran über C++, Python bis hin zu TensorFlow. Um Zugang zu dieser, mittlerweile sehr unübersichtlichen Landschaft an Programmiersprachen, Tools, Libraries, oder Plug-ins für Laien zu erhalten, wird seit 2020 die Analysesoftware GICAT (General Isomorphic Code Analysis Tool) entwickelt, die Mitte 2024 open source verfügbar sein wird. GICAT soll die unterschiedlichen, wissenschaftlichen Softwareprojekte strukturell analysieren[3]. Dabei geht es darum, wie bestimmte wissenschaftliche Konzepte in Software umgesetzt werden, welche Parametrisierungen verwendet wurden oder wie die Softwarearchitektur mit der zugrundeliegenden, wissenschaftlichen Theorie korrespondiert. In einem Forschungsprojekt mit dem Lehrstuhl für Software Engineering (Informatik 3) wird aktuell untersucht, ob sich wissenschaftlicher von technischem Code unterscheiden lässt. Gelingt es, hier ein Unterscheidungskriterium zu finden, würde dies die Analyse erleichtern. Es würde aber auch im Research Software Engineering das Qualitätsmanagement einfacher gestalten.

Da Software die neue Infrastruktur der Forschung ist, wird in immer mehr wissenschaftlichen Communities über Formen des Code Review diskutiert, dieses gilt zunehmend als qualitativer Nachweis für wissenschaftliches Arbeiten. Zeitschriften müssen folglich über die Ausarbeitung sogenannter „Code Policies“ und ihre Verbindlichkeit nachdenken[4]. Git-Repositories dienen nicht nur als eine zusätzliche Quelle für das Peer-Review, sondern müssen bestimmten Coding-Standards entsprechen, damit die Forschungsergebnisse veröffentlicht werden können. Durch diese Spannung aus normativen Ansprüchen an wissenschaftliche Praktiken und den programmiertechnischen Herausforderungen, eben diese Ansprüche in die wissenschaftliche Praxis zu integrieren, entsteht im Research Software Engineering ein Wissen, das auch für die Wissenschaftstheorie zentrale Bedeutung hat. Wird es je nach Disziplin spezifische Verfahren der Codierung von computergestützten Modellen geben, oder werden sich Metamodelle aus dem Research Software Engineering durchsetzen, die das Prädikat „lesbarer Code“ vergeben? Wird es Disziplinen geben, die sich solcher Tendenzen entziehen können, da sich ihre Modelle allein durch erfolgreiche Anwendbarkeit legitimieren? Und nicht zuletzt: Welche Computermodelle und -simulationen werden nicht zugänglich sein, da die Software urhe-

berrechtlich geschützt ist, wie im Falle der Pharma- oder Automobilindustrie?[5]
Es ist anzunehmen, dass diese Fragen nicht nur die Praktiken der wissenschaftlichen Forschung, sondern auch die der Lehre nachhaltig verändern werden. Bis heute sind Publikationen wie „A Primer on Scientific Programming with Python“ eine Seltenheit – ein Lehrbuch, das von Hans Petter Langtangen, einem norwegischen Experten für Research Software Engineering, verfasst und inzwischen mehrfach wieder aufgelegt wurde. Wenn also die Infrastruktur der Forschung nicht mehr ausschließlich im Labor, sondern in zunehmendem Maße in der Software liegt, dann muss die Wissenschaftstheorie hier sowohl die Forschungs- als auch die Lehrpraktiken in den Fokus nehmen. Dabei dienen Tools wie GICAT auch dazu herauszufinden, welche externe Wissensquellen in die Ausführungen der Software einbezogen wurden, beispielsweise durch Libraries oder anderweitige Packages. So wird heute kaum eine programmierende Wissenschaftlerin oder ein programmierender Wissenschaftler die Eulerschen Gleichungen für Rotationsmatrizen selbst berechnen, sondern sich einer Library bedienen, die solche Kalkulationen nach den gewünschten Koordinaten vornimmt. Mit einem entsprechenden Filter ermöglicht GICAT das Abrufen von Libraries sowie den genauen Zeitraum ihrer Verwendung. Zukünftig wird es zu den Herausforderungen der Wissenschaftstheorie gehören, sowohl die Transition von analogen in digitale Modelle anhand ihrer Software nachzuzeichnen als auch die Infrastruktur aufzuzeigen, aus der die jeweiligen Forschungspraktiken ihr Wissen beziehen. Es handelt sich also um interne und externe Komponenten der Wissensproduktion, die selbst neuer Forschungstools bedürfen, um jenseits der Bildschirmoberfläche analysierbar zu bleiben.

Grundlegend philosophische Fragen der Softwarekultur

Ein weiterer Schwerpunkt des Computational Science Studies Labs ist die Erforschung des Einsatzes automatisierter Beweisverfahren. Eines der ersten Systeme Künstlicher Intelligenz ist der 1956 entwickelte Logic Theorist, ein Programm zur Führung logisch-mathematischer Beweise. Solche heute unter dem Namen „Theorem Prover“ geführten Programme können als eine der ältesten Formen des Research Software Engineering bezeichnet werden. Denn neben der Anwendung in der mathematischen Forschung werden Theorem Prover als Mittel zur Verifikation von Soft- und Hardware eingesetzt. Analog etwa zur Abfrage, ob eine bestimmte mathematische Struktur eine bestimmte Eigenschaft aufweist, kann durch sie überprüft werden, ob ein Programm oder eine Komponente der Hardware die gewünschten Spezifikationen erfüllt. Dieses Nachweisen ist im Bereich der Soft- und Hardwareentwicklung zumeist unbedenklich. Es kann jedoch im Kontext der auf Wissen und Verstehen ausgerichteten Disziplinen problematisch werden[6]. Exemplarisch ist der Einsatz des automatisierten Beweisens in der Mathematik selbst. Gilt die Mathematik gemeinhin als die Disziplin, bei der (in den meisten Fällen) Beweisen und Verstehen zusammenfallen, so scheint sich hier mit dem Einsatz zum Teil sehr komplexer oder auch genuin opaker Software etwas zu verschieben: Es kann etwas bewiesen werden, ohne dass es verstanden wird.
Neben diesen potenziell problematischen Auswirkungen automatisierter Beweisverfahren werden am Computational Science Studies Lab auch die konstruktiven und transformativen Aspekte ihres Einsatzes in den Wissenschaften eingeordnet: Handelt es sich nur um ein neues Handwerkszeug, oder wird die jeweilige Disziplin in ihrer generellen Aus-

richtung dadurch verändert? Werden durch den Einsatz von Verfahren der automatisierten Beweisführung lediglich alte Bereiche besser handhabbar, oder erschließen sie der jeweiligen Disziplin neue Bereiche? In diesem Zusammenhang sind auch die klassischen Probleme bezüglich der Reproduzierbarkeit computergestützter Forschung zu beachten. So ist etwa der Code für einen durch einen Theorem Prover geführten Beweis oft an die Versionsnummer des Programms gebunden. Damit ist die Reproduzierbarkeit des Beweises neben der oft je versionsspezifischen technischen Umsetzung der eingesetzten Verfahren auch von der jeweiligen Notation abhängig. Demgegenüber steht das Bestreben, große Archive an formalisierten Beweisen anzulegen (etwa Mizar) und der Allgemeinheit zur Verfügung zu stellen. So kann hier von einer Art Data-Mining bezüglich bestehender Beweise gesprochen werden. Neue, bisher nicht ‚sichtbare' Beweise und Informationen können durch Techniken der Datenanalyse aus einem Stock an bereits geführten Beweisen extrahiert werden. Auch solche Vorgehensweisen, die sich nicht durch die klassischen Kategorien der Wissenschaftstheorie erfassen lassen, werden am Computational Science Studies Lab analysiert.

https://www.css-lab.rwth-aachen.de

Literatur

[1] Hocquet, A., Wieber, F., Gramelsberger, G., et al., Software in science is ubiquitous yet overlooked. In: Nature Computational Science 4(6), 2024

[2] Schüttler, L., Kasprowicz, D., Gramelsberger, G., Computational Science Studies. A Tool-Based Methodology for Studying Code. In: Getzinger, G., Jahrbacher, M., Hrsg., Critical Issues in Science, Technology, and Society Studies, Conference Proceedings STS Conference Graz 2019, 385-401, Graz: Verlag der Technischen Universität Graz, 2019

[3] Gramelsberger, G., Wenz, D., Kasprowicz, D., Understanding and Analyzing Science's Algorithmic Regimes: a Primer in Computational Science Code Studies. In: Jarke, J., Prietl, B., Egbert, S., Boeva, Y., Heuer, H., Arnold, M., Hrsg., Algorithmic Regimes, 57-78. Amsterdam: Amsterdam University Press, 2024

[4] Thimbley, H., Improving Science That Uses Code. In: The Computer Journal, 1-24, 2023doi.org/10.1093/comjnl/bxad067

[5] Zum Verhältnis der Computational Chemistry und der Entwicklung von Software-Packages für Pharmaunternehmen: Hocquet, A., Wieber, F., "Only the Initiates Will Have the Secrets Revealed": Computational Chemists and the Openness of Scientific Software. In: IEEE Annals of the History of Computing, 39(4), 40-58. 2017, doi.org/10.1109/MAHC.2018.1221048

[6] Wenz, D., Künstliche Intelligenz in mathematischen Beweisen und das Problem der Erklärbarkeit. In: Strasser, A., Sohst, W., Stapelfeldt, R., Stepec, K., Hrsg., Künstliche Intelligenz - Die große Verheißung, 145-168. Berlin: Xenomoi,2021.

Autoren

Univ.-Prof. Dr.phil. Gabriele Gramelsberger ist Inhaberin des Lehrstuhls für Wissenschaftstheorie und Technikphilosophie.
Gastprofessor Dr. Markus Pantsar, Dr.phil. Daniel Wenz, Dr. Dawid Kasprowicz, Ph.D., Thomas Venator, M.Sc., sind wissenschaftliche Mitarbeiter am Lehrstuhl für Wissenschaftstheorie und Technikphilosophie.
Frederik Kerksieck und David Heyen sind studentische Hilfskräfte aus der Informatik, die die Tools softwaretechnisch umsetzen.

Nico Jansen, Bernhard Rumpe

Kompositionelle Sprachentwicklung mit der Language Workbench MontiCore

Wiederverwendbarkeit im Software Engineering

MontiCore is a state-of-the-art language workbench for designing and implementing domain-specific languages. It is an open-source research project, continuously developed since 2004, fostering open collaboration and innovation.
MontiCore's research is focused on generation technology and the provision of generative practices. The language workbench is a pioneer in language composition and comes with a large library of reusable language components. MontiCore's development and application is a successful research software engineering endeavor with numerous offsprings in both academia and industry. MontiCore has a lasting relevance in software language engineering, documented in multiple publications and dissertations, and plays a fundamental role in innovating advanced technologies and development practices.

Software hat einen großen Einfluss in Industrie, Gesellschaft und Forschung. Viele Produkte, Produktionsanlagen oder auch Forschungsunterfangen werden stark durch Software getrieben oder unterstützt. Dies erfordert effiziente und qualitativ hochwertige Softwarelösungen. Explizite Modellierungssprachen, wie die Unified Modeling Language, kurz UML, erlauben Architektur und Verhalten von Software zu beschreiben. Domänenspezifische Sprachen, DSLs für Englisch Domain-Specific Languages, ermöglichen es Expertinnen und Experten aus den jeweiligen Disziplinen, Lösungen innerhalb ihrer Expertise zu modellieren, indem Modellierungssprachen präzise auf die Terminologie der Domänen zugeschnitten werden.
Die reduzierten, aber problemangepassten Modellierungstechniken erlauben die automatisierte Verarbeitung der Modelle in Form von hochwertigen Analysen sowie die Synthese vollständiger Softwaresysteme weit jenseits des puren Ausführens von Simulationen. Modellierungssprachen schließen damit die Lücke zwischen Problem- und Lösungsdomäne. Dies bedeutet, dass Menschen ohne weitreichende Programmierkenntnisse die Kontrolle über die Softwareentwicklung übernehmen können. Die Erstellung und Bereitstellung von DSLs ist daher von großer Bedeutung in Forschung und Industrie. Modellierungssprachen sind aber selbst Forschungsgegenstand, weshalb Forschergruppen einschließlich des Lehrstuhls für Software Engineering (Informatik 3) an der Language Workbench MontiCore[1] gleichzeitig als Forschungssoftware und als Werkzeug zur Erstellung anderer Forschungssoftware arbeiten. MontiCore ist damit primär ein Meta-Werkzeug, das es ermöglicht, schnell Software-Werkzeuge zu erstellen, die ihrerseits die eigentliche Softwareentwicklung unterstützen.
Die Workbench erlaubt es komplexe, textuelle DSLs aus Bausteinen sowie darauf aufbauende Analyse- und Synthese-Werkzeuge zu erzeugen. Damit dient MontiCore als Basis für laufende Forschung in der modellgetriebenen Softwareentwicklung und unterstützt eine kompositionelle Entwicklung von DSLs für agile Projekte. Das Prinzip der Modellbibliothek wurde systematisiert; angeboten wird auch eine Bibliothek an Kernsprachen, die für die Zusammenstellungen eigener, domänenspezifischer Modellierungssprachen geeignet sind. Beispiele hierfür sind etwa NESTML[2] für biologische neuronale Netze, SpesML[3] für das Systems Engineering, Statecharts für die Möglichkeit explizit Unterspezifikation und

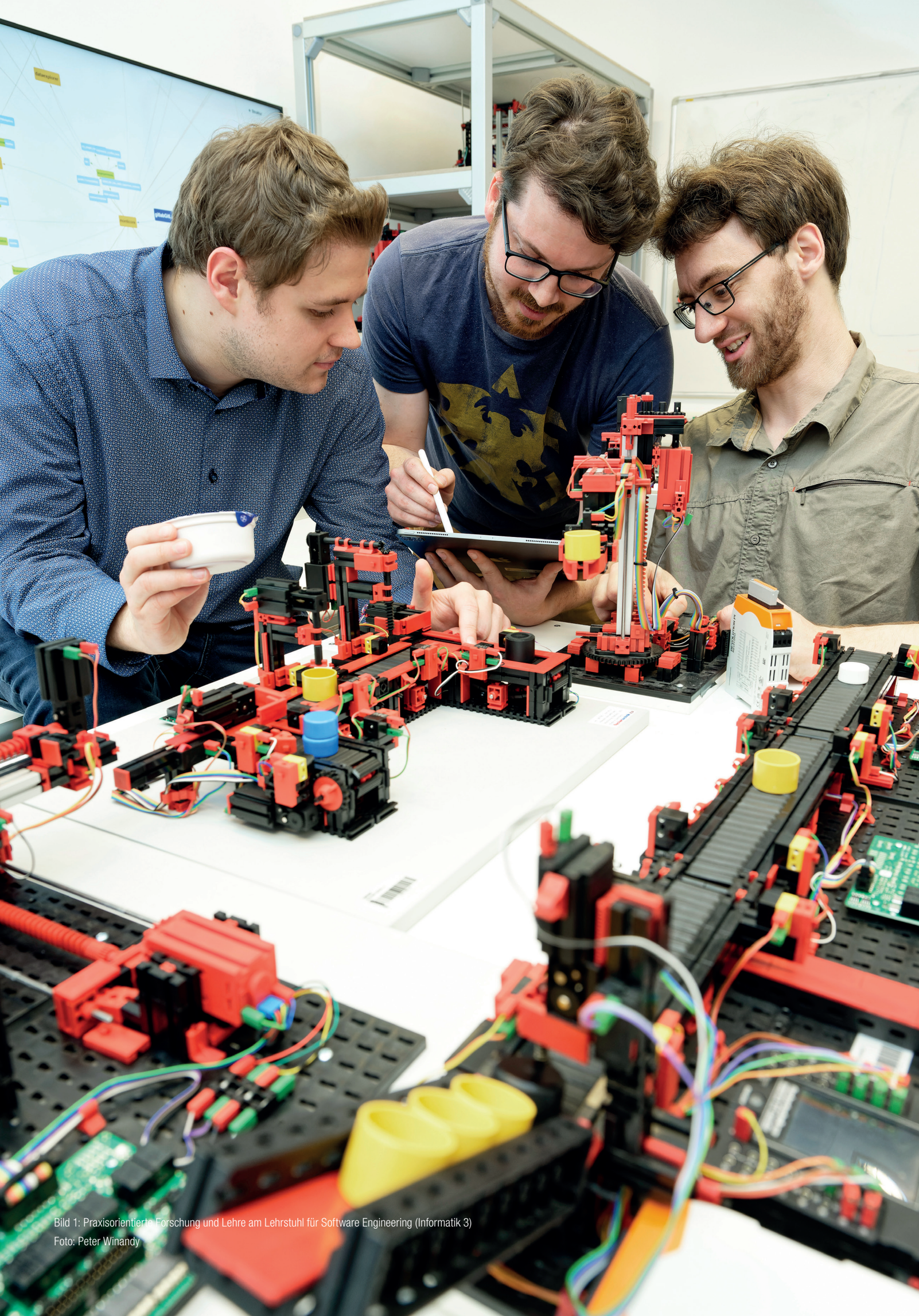


Bild 1: Praxisorientierte Forschung und Lehre am Lehrstuhl für Software Engineering (Informatik 3)
Foto: Peter Winandy

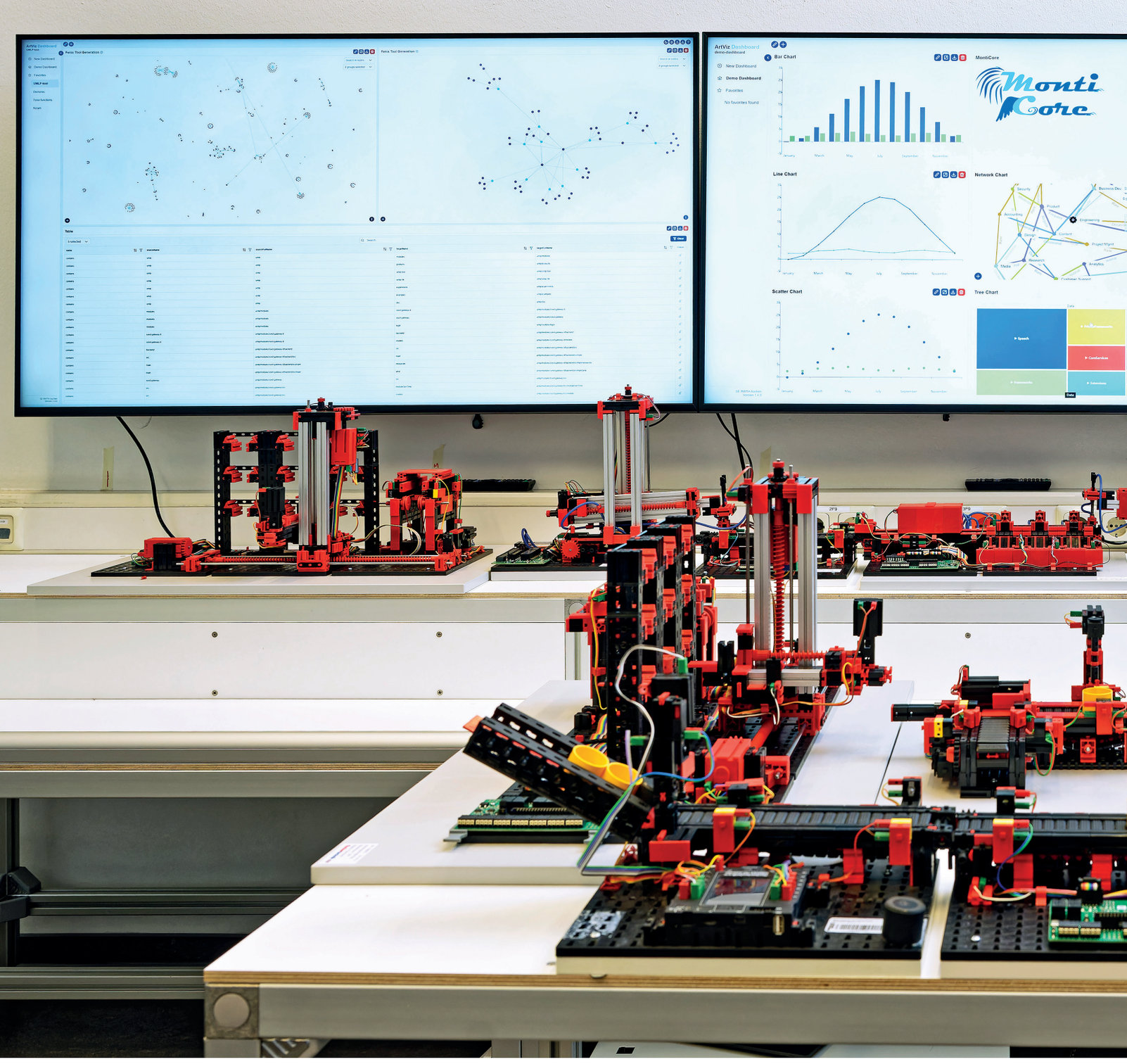
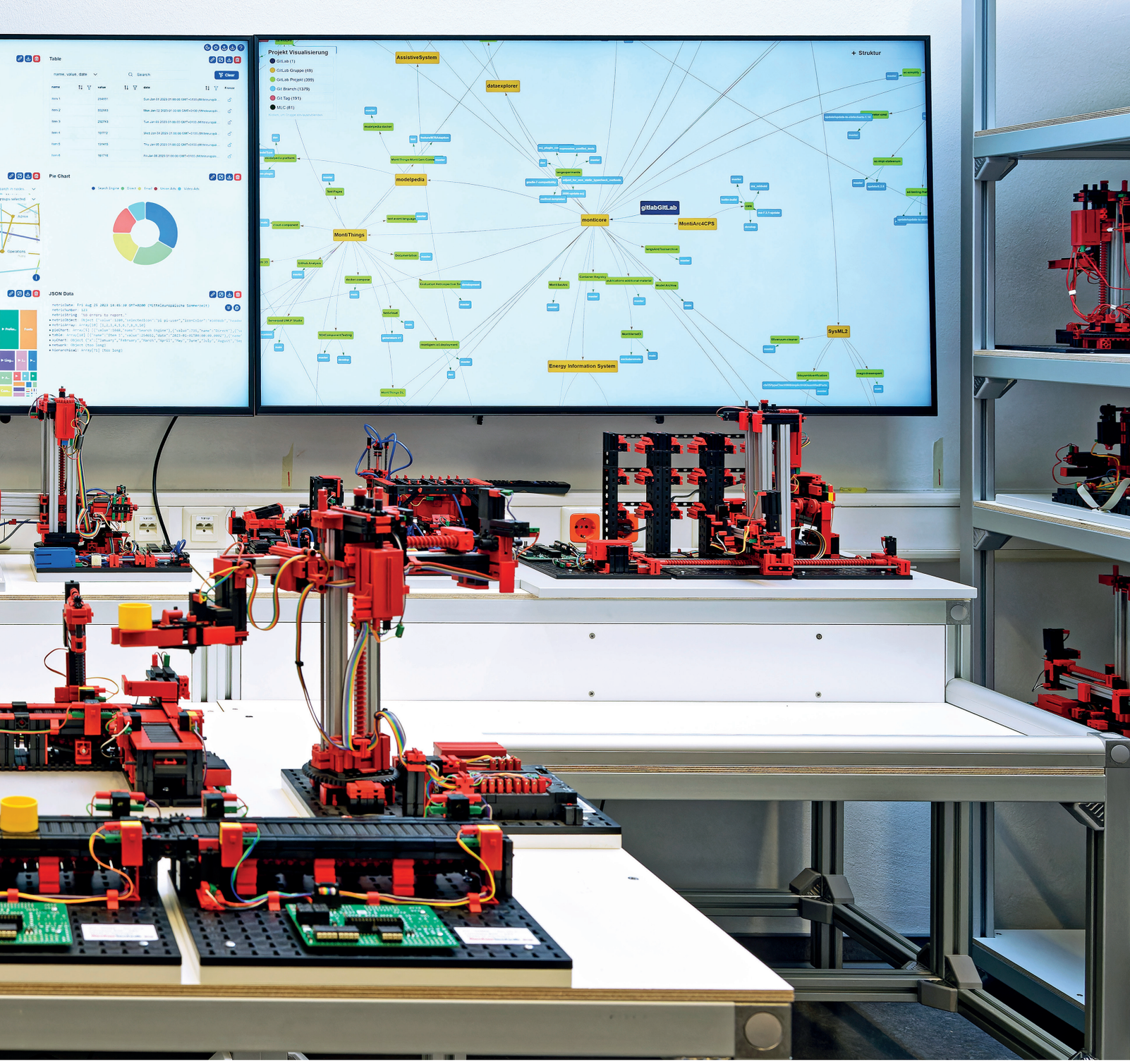


Bild 2: Modellgetriebene Analyse und Simulation einer Demofabrik
Foto: Peter Winandy

Nichtdeterminismus in die Modellierungstechniken einzubringen, Ontologie-Sprachen in zahlreichen Varianten oder Modellierungstechniken für digitale Zwillinge[4].
Die entwickelten Konzepte und Sprachen verbessern den Software-Engineering-Prozess nachhaltig. Sie erleichtern die Erstellung hochwertiger, individueller DSLs. Die Modelle dieser Sprache können wiederverwendbaren, domänenspezifischen Analysen unterzogen, aber auch als Zwischenschritt zwischen dem wissenschaftlich erklärenden Papier und dem finalen Code genutzt werden und unterstützen so eine effiziente, evolutionäre Weiterentwicklung. So lassen sich zum Beispiel Statecharts darauf vergleichen, ob eine Verfeinerungsbeziehung in Bezug auf das Verhalten herrscht und damit die Substituierbarkeit möglich ist. Auch können mit semantikbasierten Differenzverfahren Varianten einer Ontologie hinsichtlich ihrer unterschiedlichen Objektstrukturen geprüft werden.
Für die technische Umsetzung basiert MontiCore auf Grammatiken zur Sprachbeschreibung und synthetisiert daraus Infrastruktur für die Modellverarbeitung. Dazu gehören ein Modelloader, Infrastruktur für Wohlgeformtheitsregeln (Kontextbedingungen) und das Management von Symboltabellen[5]. Basierend auf der Modellierungssprache werden außerdem eine zugehörige Augmentation-Sprache (Tagging) und eine Transformationssprache synthetisiert, die die Modellnutzung und Übersetzung deutlich vereinfachen. Weitere Technologien bieten die Einbettung in Entwicklungsumgebungen, eine flexibel anpassbare Template-basierte Code-Generierung, und eine Übersetzung in Model Checking- und Verifikationsumgebungen.
MontiCore zeichnet sich durch die adäquate Anwendung bewährter Entwurfsmuster der Softwaretechnik aus, erweitert diese und hat darüber hinaus eigene Muster für die Entwicklung von DSLs realisiert[6]. So wird beispielsweise die effiziente Traversierung von Objektstrukturen mithilfe des generierten Visitor Patterns und mittels Template-Hook und RealThis Pattern der kompositionelle



Aufbau von Datenstrukturen und Analyse-/Synthese-Algorithmen ermöglicht. Dies ist eine Kernkonstruktion zur nachhaltigen und flexiblen Wiederverwendung von DSL-Komponenten in verschiedenen Sprachen und damit Werkzeugen. Maßgeschneiderte Werkzeuge können so schnell und effizient entstehen.

Wiederverwendbarkeit auf Sprachebene

Wiederverwendbarkeit ist im Software Engineering von entscheidender Bedeutung, da sie die Effizienz von Entwicklungsprozessen, sowie die Qualität und Wartbarkeit der Software erheblich verbessert. MontiCore ist ein Pionier auf dem Gebiet der Sprach- und der Werkzeugkomposition und stellt vielfältige Mechanismen zur Erweiterung, Einbettung und Aggregation von DSLs zur Verfügung, um Wiederverwendbarkeit auf Sprachebene zu ermöglichen. Mehrere Mechanismen umfassen sowohl eine technische Umsetzung als auch eine methodische Beschreibung, welche dabei unterstützen, neue DSLs aus existierenden Komponenten zu entwerfen. Eine aggregierende Kopplung von Modellen (auch verschiedener Sprachen) wird über die Technologie der Symboltabellen realisiert, welche Elemente zwischen Modellen nutzbar machen und so eine Gesamtbeschreibung von Systemen ermöglichen. Eine Variante ist die konservative Erweiterung einer Sprache, bei der die Modelle der ursprünglichen DSL gültig bleiben sowie modular definierte Analyse- und Synthese-Algorithmen ebenfalls weiter eingesetzt werden können. Das unterstützt die Agilität der Spracherweiterung passend zu den ergänzten Anforderungen. Die Einbettung von Sprachkomponenten ermöglicht deren Wiederverwendung und Integration in einer Gesamtsprache. Ein Beispiel ist die Einbettung verschiedener Sprachkomponenten für mehrere Arten von Expressions, Statements, Typdefinitionen oder Literalen in eine Hostsprache. Dadurch wird der Aufbau von Sprachbibliotheken für die modulare Komposition erst möglich wodurch DSLs gemeinsame Sprachkonzepte teilen und so den

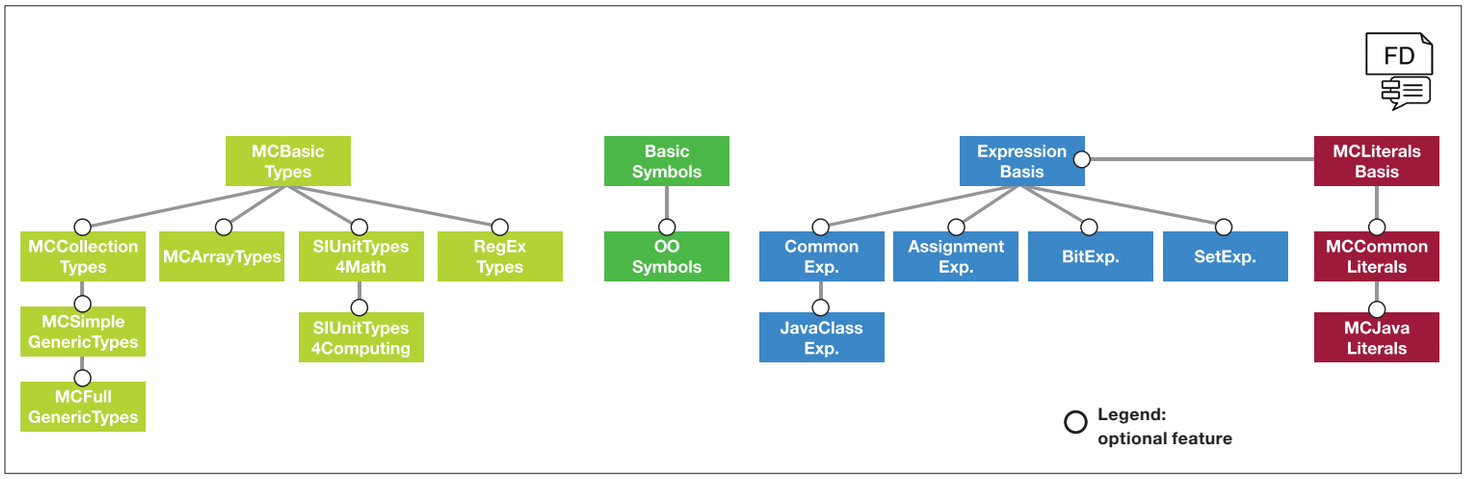


Bild 3: Bibliothek von wiederverwendbaren Sprachkomponenten in MontiCore

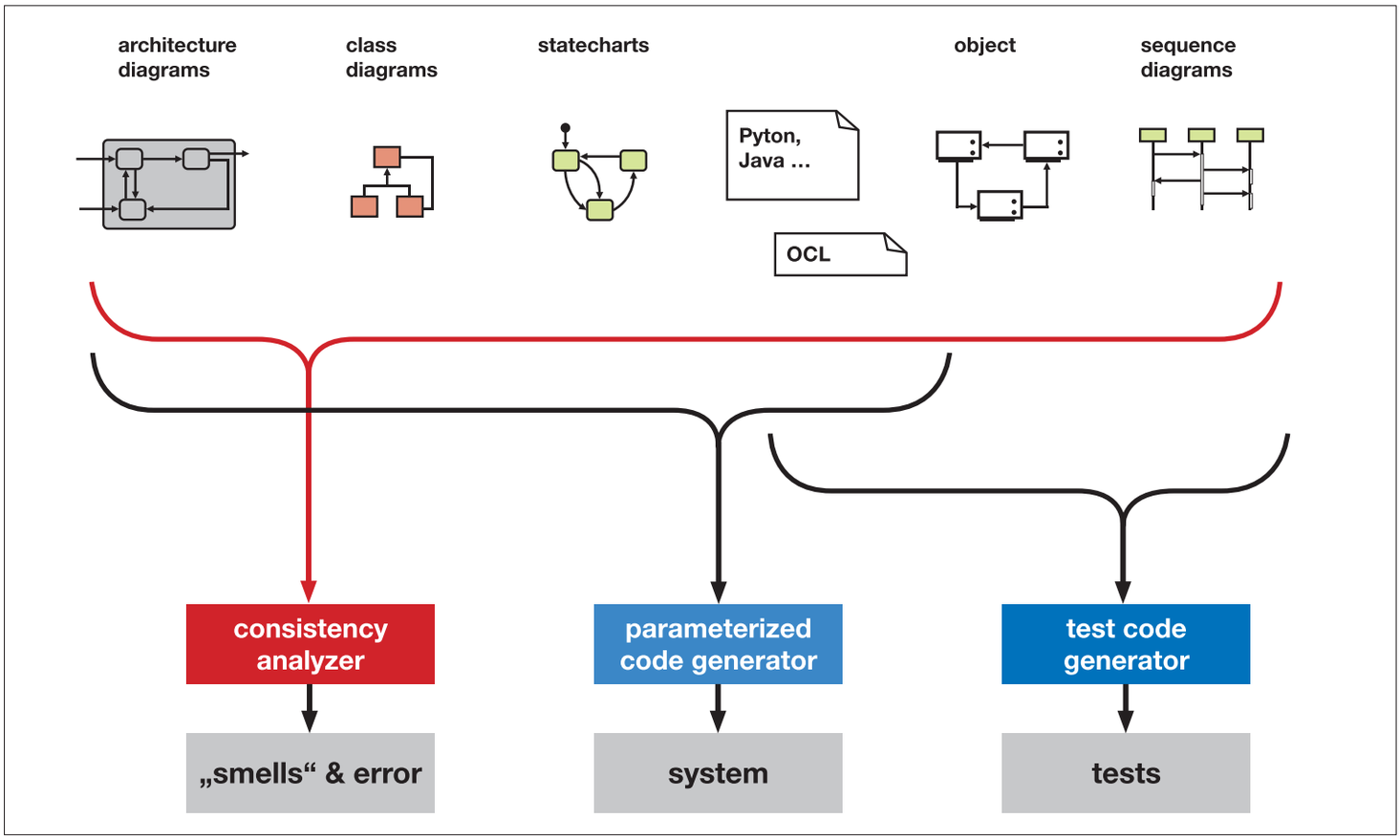


Bild 4: Bibliothek von Software-Modellierungssprachen und ihre Verwendung

Nutzern schnellere Einarbeitung und besseres Wiedererkennen ermöglichen. MontiCore bietet dafür eine weitreichende Bibliothek mit gebräuchlichen Sprachkomponenten[7].
Bild 3 zeigt den modularen Aufbau dieser Sprachbibliothek. Die verschiedenen Komponenten repräsentieren Ausbaustufen dedizierter Sprachkonzepte. Je nach Anwendungsfall können beim Design einer DSL beispielsweise nur einfache Datentypen (z.B. int, String) oder auch physikalische Typen, also SI-Units (z.B. 3m/s, 220VA) typsicher in eine Sprache eingebracht werden, sodass sich die integrierte Werkzeuginfrastruktur automatisch um die Korrektheit der Konversionen kümmert. Darauf aufbauend hat MontiCore eine weitere Bibliothek an Sprachkomponenten von Modellierungstechniken für Automaten, Sequenzdiagramme, Klassendiagramme, Ontologien oder Prozessmodelle, die den agilen, modellbasierten Entwicklungsprozess unterstützen.
Die Language Workbench wird als Forschungssoftware seit 2004 kontinuierlich entwickelt und hat im Bereich Software Language Engineering viele neue Konzepte hervorgebracht, welche industrielle Frameworks nachhaltig inspirierten. MontiCore hat sich als stabiles und erweiterbares Projekt erwiesen und gleichzeitig den Grundstein für viele weitere Forschungsvorhaben gelegt. Die Workbench ist öffentlich auf GitHub verfügbar, was die Zusammenarbeit mit Industrie und Forschung vorantreibt und Innovation fördert. Durch kontinuierliche Integration und automatisierte Tests, die die Konsistenz und

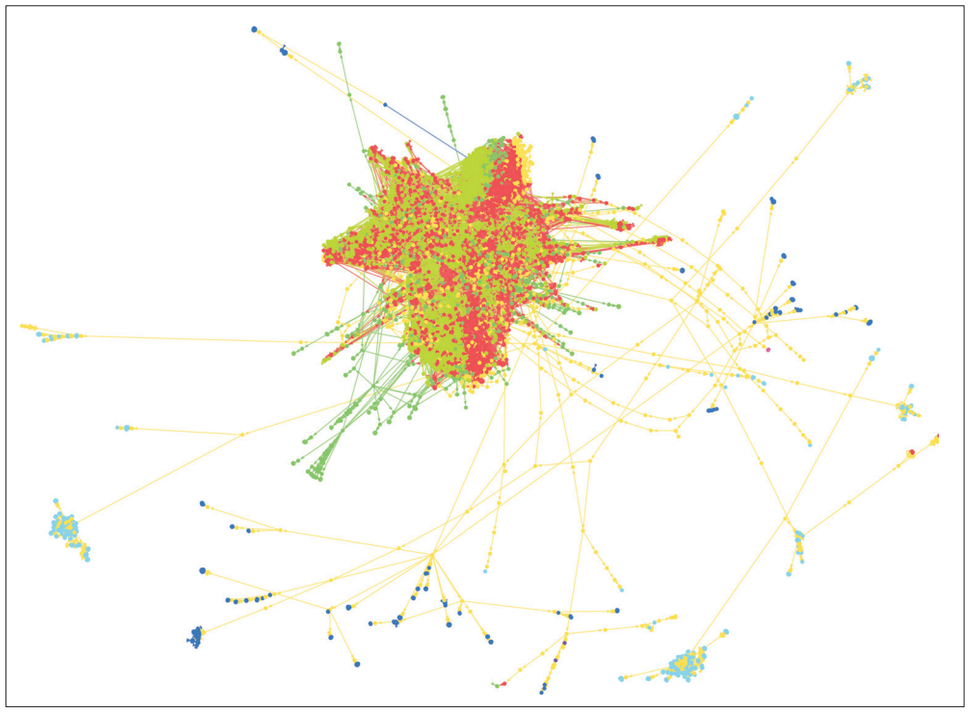


Bild 5: Abhängigkeitsgraph zusammenhängender Artefakte im Software-Engineering-Prozess

Zuverlässigkeit der Software prüfen, wird fortlaufend eine hohe Qualität sichergestellt. Insgesamt waren an der Weiterentwicklung über 40 Personen in mittlerweile 100 Personenjahren beteiligt.

MontiCore als Meta-Werkzeug

Als Meta-Werkzeug hat MontiCore zahlreiche Ableger, also konkrete Werkzeuge für Aufgaben in unterschiedlichen technologischen Bereichen. Eine Architekturbeschreibungssprache[8] erlaubt die Kontrolle von Softwarearchitekturen speziell für die automatisierte Synthese komplexer cyberphysischer Systeme, Infrastrukturen im Bereich Internet of Things und auch Digitaler Zwillinge. Assistenzsysteme im privaten Umfeld oder im Produktionsbereich[9], Robotersteuerungen, autonome Fahrinfrastrukturen, Verbesserung der Mensch-Maschine-Interaktion durch abstrakte Nutzerführungsmodelle, eine erste Modellierungssprache für Modelbasiertes Machine Learning und vieles mehr wurden so realisiert. Modellbasierte Artefaktanalysen helfen in Software- und Systems-Engineering-Projekten die Projektlandschaft zu verstehen, Abhängigkeiten zu erkennen, Konflikte aufzudecken und die Prozessperformance zu evaluieren[10]. Bild 5 zeigt einen Abhängigkeitsgraphen von Entwicklungsartefakten und verdeutlicht, wo sogenannte Spagetti-Klumpen die Modularität zerstören. MontiCore ist in die internationale Forschungskollaboration GEMOC Initiative integriert, um Software Language Engineering nachhaltig voranzubringen und so die modellbasierte Software- und Systementwicklung weiter in die Praxis zu bringen. Zahlreiche Publikationen belegen die Relevanz von MontiCore in Software Language Engineering, die aber auch durch industrielle Verwendungen nachgewiesen ist. Neuere Forschungsaktivitäten dienen dem Ausbau der Language Workbench als Infrastruktur für Forschungssoftware anderer Domänen, um zum Beispiel Forschungssoftware präziser mit den abstrakten, oft mathematischen Modellen in Physik, Chemie, oder anderen Fachbereichen zu verzahnen, Datenmengen und ihre Meta-Daten effektiv zu managen und gleichzeitig stringente Nachvollziehbarkeit der Forschungsergebnisse und Weiterentwicklungen der Software sicherzustellen.

www.se-rwth.de/research/

Autoren

Nico Jansen, M.Sc., ist wissenschaftlicher Mitarbeiter am Lehrstuhl für Software Engineering (Informatik 3).
Univ.-Prof. Dr.rer.nat. Bernhard Rumpe ist Inhaber des Lehrstuhls für Software Engineering (Informatik 3).

Literatur

[1] Hölldobler, K., Kautz, O., Rumpe, B., MontiCore Language Workbench and Library Handbook: Edition 2021, Aachener Informatik-Berichte, Software Engineering, Band 48, ISBN 978-3-8440-8010-0, Shaker Verlag, May 2021

[2] Plotnikov, D., Blundell, I., Ippen, T., Eppler, J. M., Morrison, A., Rumpe, B., NESTML: a modeling language for spiking neurons, in: Modellierung 2016 Conference, Volume 254, pp. 93-108, LNI, Bonner Köllen Verlag, Mar 2016

[3] Gupta, R., Jansen, N., Regnat, N., Rumpe, B., Implementation of the SpesML Workbench in MagicDraw, in: Modellierung 2022 Satellite Events, pp. 61-76, Gesellschaft für Informatik, Jun 2022

[4] Dalibor, M, Heithoff, M., Michael, J., Netz, L., Pfeiffer, J., Rumpe, B., Varga, S., Wortmann, A., Generating Customized Low-Code Development Platforms for Digital Twins, in: Journal of Computer Languages (COLA), Volume 70, Art. 101117, Elsevier, Jun 2022

[5] Butting, A., Michael, J., Rumpe, B.,Language Composition via Kind-Typed Symbol Tables, Journal of Object Technology (JOT), October 2022

[6] Drux, F., Jansen, N., Rumpe, B., A Catalog of Design Patterns for Compositional Language Engineering, Journal of Object Technology (JOT), October 2022

[7] Butting, A., Eikermann, R., Hölldobler, K., Jansen, N., Rumpe, B., Wortmann, A., A Library of Literals, Expressions, Types, and Statements for Compositional Language Design, in: Journal of Object Technology (JOT), October 2020

[8] Haber, A., Ringert, J. O., Rumpe, B., MontiArc - Architectural Modeling of Interactive Distributed and Cyber-Physical Systems, RWTH Aachen, AIB-2012-03, Technical Report, Feb 2012

[9] Michael, J., A Vision Towards Generated Assistive Systems for Supporting Human Interactions in Production, in: Modellierung 2022 Satellite Events, pp. 150-153, Gesellschaft für Informatik e.V., Jul 2022

[10] Greifenberg, T., Hillemacher, S., Hölldobler, K., Applied Artifact-Based Analysis for Architecture Consistency Checking,in: Ernst Denert Award for Software Engineering 2019, pp. 61-85, Springer, Dec 2020

Impressum

Herausgegeben im Auftrag des Rektors
der RWTH Aachen
Dezernat 3.0 – Presse und Kommunikation
Templergraben 55
52056 Aachen
Telefon +49 241 80 - 93687
pressestelle@rwth-aachen.de
www.rwth-aachen.de

Titelbild: Qualitative Untersuchung des Strömungsverhaltens eines Flügels in Hochauftriebskonfiguration mithilfe einer Rauchlanze im Unterschall-Windkanal
Peter Winandy, Aachen

Fotografie:
Peter Winandy, Aachen

Gestaltung:
Kerstin Lünenschloß, Aachen

Druck:
image Druck + MEDIEN GmbH, Aachen

ISSN-Nummer 0179-079X

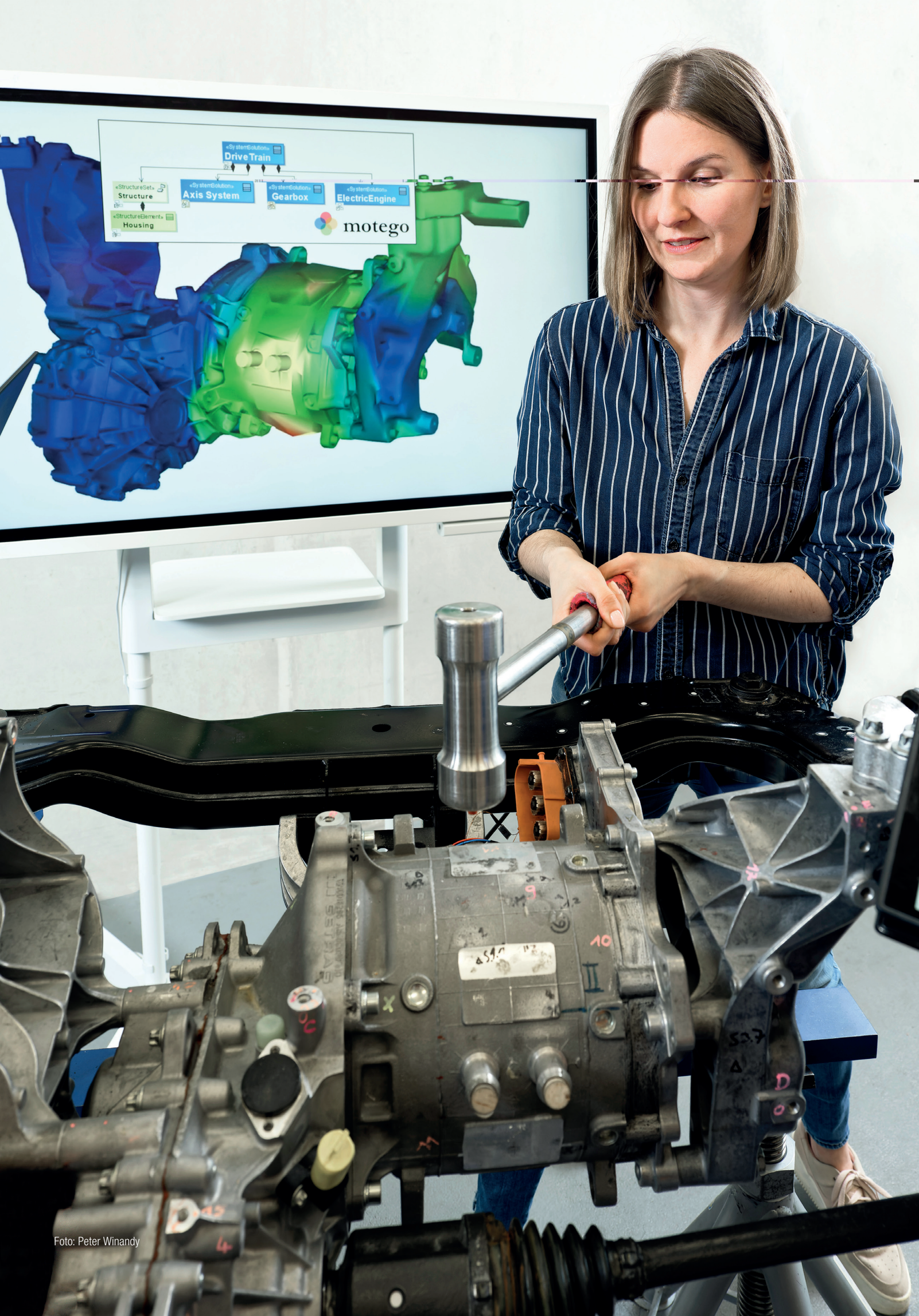


Foto: Peter Winandy

Thinking the Future
Zukunft denken