Simulations- und Softwaretechnik

Statusbericht
2011–2015

Teil 1

DLR

Inhalt

Vorwort

Das Team der Simulations- und Softwaretechnik im September 2014

Die „Simulations- und Softwaretechnik" (SC) wurde 1998 gegründet und 2007 durch den DLR-Senat als wissenschaftlich-technische Einrichtung bestätigt. Im Dezember 2002 und Oktober 2010 stellten wir uns erfolgreich externen Überprüfungen. Der vorliegende Statusbericht bereitet die turnusmäßige Überprüfung im April 2015 vor.

Unsere Kernkompetenz ist die moderne Softwaretechnologie. Mit dieser Kompetenz ergänzt SC die mehr als 30 DLR-Institute und -Einrichtungen überwiegend ingenieurwissenschaftlicher Ausrichtung. In vielen gemeinsamen Projekten mit diesen Instituten übernehmen wir anspruchsvolle Software-Entwicklungsaufgaben. Im Auftrag des zentralen IT-Managements erfüllen wir darüber hinaus eine Querschnittsaufgabe im Software-Engineering. Wir beraten andere Institute, wie sie fortschrittliche Software-Engineering-Verfahren einsetzen können, und wir schulen ihre Mitarbeiter.

Softwaretechnologie wird für ingenieurwissenschaftliche Forschungsprojekte immer wichtiger. Auch im DLR wächst die Bedeutung eigener Forschung und Entwicklung auf diesem Gebiet. Wir forschen in eigenen Vorhaben und in Drittmittelprojekten mit Partnern aus Wissenschaft und Industrie. Unsere Mitarbeiterinnen und Mitarbeiter betreuen Dissertationen, Master- und Bachelorarbeiten und übernehmen Lehrveranstaltungen an Hochschulen. Unsere aktuellen Forschungsthemen decken ein breites Spektrum ab und spiegeln den vielfältigen Einsatz von Software im DLR wider. Wir kooperieren inzwischen mit den meisten DLR-Instituten.

Dieser erste, öffentliche, Teil des Statusberichts gibt einen Überblick über die Themen, die im Berichtszeitraum bearbeitet wurden, und präsentiert unsere Ergebnisse und Pläne. Er stellt den Bezug zu den Anwendungen in den Partnerinstituten her. Am Ende steht eine Übersicht unserer Publikationen und Aktivitäten in Ausbildung und Lehre.

Rolf Hempel

Überblick

Wir erforschen und entwickeln Softwaretechnologien, die für aktuelle oder zukünftige Anwendungen im DLR relevant sind. Unsere Vorlaufforschung ist hauptsächlich drittmittelfinanziert. Damit bauen wir Know-how auf und entwickeln grundlegende Softwareprodukte. Programmatisch finanzierte Verbundprojekte mit anderen DLR-Instituten bilden den Schwerpunkt unserer Arbeit. In diesen Projekten übernehmen wir anspruchsvolle Software-Aufgaben und ergänzen damit die ingenieurwissenschaftlichen Beiträge der Partnerinstitute.

Wir sind an den Standorten Köln-Porz und Braunschweig vertreten und unterhalten eine Außenstelle in Berlin.
SC ist in zwei Abteilungen gegliedert:

- Verteilte Systeme und Komponentensoftware (VSS, Leitung: Andreas Schreiber)
- Software für Raumfahrtsysteme und interaktive Visualisierung (SRV, Leitung: Dr. Andreas Gerndt)

Die Abteilungen umfassen jeweils mehrere Arbeitsgruppen (Abbildung 1).

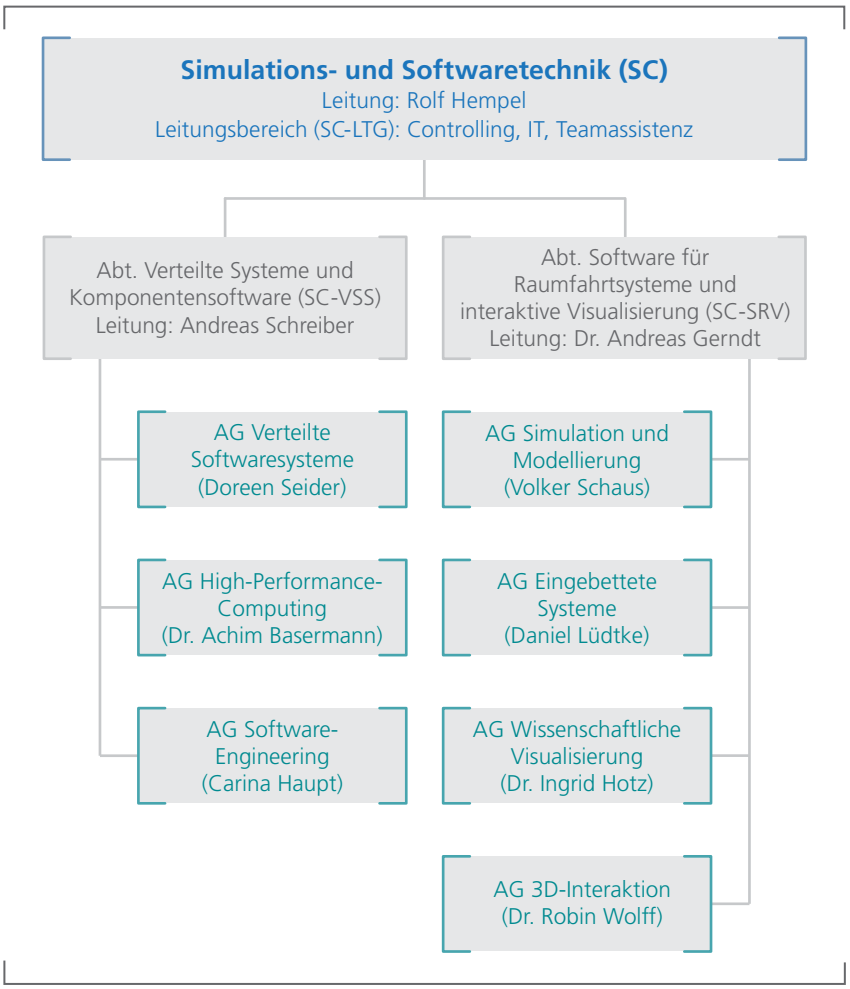


Abbildung 1: Organigramm der Einrichtung Simulations- und Softwaretechnik

Im Folgenden stellen wir die Themen der Arbeitsgruppen kurz vor.

Verteilte Softwaresysteme

Experten vieler Fachrichtungen müssen zusammenarbeiten, um komplexe technische Systeme zu analysieren oder zu bewerten. Es ist ein strategisches Ziel des DLR, diese „Systemfähigkeit" herzustellen. Bei der multidisziplinären Zusammenarbeit gibt es viele technische Hürden: Jedes Institut verwendet seine eigenen Simulationswerkzeuge, unterschiedliche Schnittstellen und Datenformate. Die Computer stehen an verschiedenen Standorten und haben unterschiedliche Betriebssysteme.

Unsere Software für die multidisziplinäre Zusammenarbeit ermöglicht es den Ingenieuren, diese Hürden zu überwinden. Dabei achten wir besonders darauf, dass die hochkomplexen Gesamtsysteme flexibel, wartungs- und nutzerfreundlich bleiben. Hierbei helfen uns neueste Softwaretechnologien und konsequentes Software-Engineering.

Abseits der klassischen Anwendungen aus Luft- und Raumfahrt entwickeln wir verteilte Softwaresysteme für Eignungsdiagnostik und Telemedizin. Fragen zum Anwenderverhalten und die Nutzung mobiler Hardware stehen hier im Vordergrund.

High-Performance-Computing (HPC)

Die numerische Simulation hat sich neben Theorie und Experiment als drittes Standbein der Forschung etabliert. Die wichtigste Anwendung im DLR ist die Simulation von Strömungsvorgängen. Sie stellt besonders hohe Anforderungen an Rechengeschwindigkeit und Datenmanagement und verursacht hohe Kosten. Daher werden große Anstrengungen unternommen, um die Ressourcen möglichst effizient zu nutzen.

Wir unterstützen dieses Ziel mit grundlegenden Beiträgen. Wir erforschen effiziente skalierbare Algorithmen und Datenstrukturen für die hochparallelen Rechner der Zukunft. Wir entwickeln Software, um große Simulationsrechnungen mit alternativen Hardwarekomponenten wie Graphik- oder Vielkernprozessoren zu optimieren und zu beschleunigen. Und wir sorgen für ein effizientes Management großer Datenmengen. In einer neuen Anwendung geht es auch darum, Weltraumrückstände zu katalogisieren.

Zusätzlich nutzen wir HPC-Techniken auch intern, etwa für die interaktive Visualisierung. Dabei müssen komplexe Berechnungen in kürzester Zeit ausgeführt werden. Das geht nur mit schnellen parallelen Algorithmen auf HPC-Systemen.

Software-Engineering

In den Instituten des DLR wird etwa ein Viertel der gesamten Arbeitszeit für Softwareentwicklungen aufgewendet. Entsprechend wichtig ist es, eine Methodik zu wählen, die zu maximaler Softwarequalität bei minimalem Ressourceneinsatz führt. Im Auftrag des zentralen IT-Managements verfolgen wir dieses Ziel mit modernen Software-Engineering-Verfahren. In unseren eigenen Forschungsprojekten erproben wir Entwicklungsprozesse und passen sie an die Erfordernisse des DLR an. Wir testen Software-Entwicklungswerkzeuge und entwickeln eigene Speziallösungen. Über das Software-Engineering-Netzwerk und in Schulungen verbreiten wir unsere Kenntnisse im ganzen DLR. Wir erweitern unsere Expertise im Software-Engineering durch eigene Forschungsbeiträge zu ausgesuchten Themen. Beispiele sind Provenance von Daten und Prozessen, Prozess-Visualisierung und Wissensmanagement.

Simulation und Modellierung

Anders als in der Automobilindustrie ist der Entwurfsprozess in Raumfahrtprojekten noch überwiegend dokumentenbasiert. Das macht den Prozess unflexibel und bei Designänderungen fehleranfällig. Die Zukunft gehört auch in der Raumfahrt dem „Model-based Systems Engineering“ (MBSE). Die Idee ist, den kompletten Entwurfsprozess auf ein zentrales digitales Modell des technischen Systems zu beziehen. Dabei geht man von einem groben Konzeptmodell aus, das im Laufe des Projekts immer weiter verfeinert wird. Gemeinsam mit den Ingenieuren aus anderen DLR-Instituten und Partnerinstitutionen wie der ESA möchten wir diese Idee verwirklichen. Wir erforschen die informationstechnischen Grundlagen und entwickeln Prototypen für Designwerkzeuge. Als Software-Rahmen dient uns dabei unser Produkt „Virtueller Satellit“. Dieser hat sich in frühen Designstudien bereits sehr bewährt. In der „Concurrent Engineering Facility“ des Bremer DLR-Instituts für Raumfahrtsysteme ist er inzwischen im Routineeinsatz.

Eingebettete Systeme

Onboard-Software für Satelliten und Raumsonden erledigt heute viele Funktionen, für die früher die Hardware zuständig war. Gleichzeitig werden die Systeme immer komplexer – und damit auch die Software. Zuverlässigkeit ist daher eine der wichtigsten Anforderungen. Die Arbeitsgruppe „Eingebettete Systeme“ erforscht neue Konzepte für Onboard-Softwaresysteme und setzt sie in Missionen ein. Begonnen haben wir mit der objektorientierten Programmierung von Lageregelungssoftware für DLR-Satellitenmissionen. In den vergangenen Jahren kamen die Themen Onboard-Navigation und „Command & Data Handling“ hinzu. Wir wenden neue Softwaretechniken an, um die Flexibilität und Zuverlässigkeit zu erhöhen – zum Beispiel die modellbasierte Softwareentwicklung. Wir entwickeln ein verteiltes Betriebssystem, das die vorhandenen Onboard-Rechner je nach Missionsphase intelligent den anstehenden Aufgaben zuweist.

Wissenschaftliche Visualisierung

Visualisierung hilft dabei, die immer größeren Ergebnisdatensätze aus Simulationsrechnungen und Experimenten zu interpretieren. Einfache Frontend-Rechner sind damit überfordert, Datensätze im Terabyte-Bereich aufzubereiten. Wir entwickeln eine skalierbare Lösung in Form einer Postprocessing-Pipeline. HPC-Systeme übernehmen dabei die rechenintensiven Teilaufgaben, eventuell sogar direkt in der laufenden Simulation. Wir erforschen neue Verfahren, um Merkmale aus den Datensätzen zu extrahieren. Beispiele sind die Tensor-Visualisierung und die Mustererkennung in Strömungsfeldern. Die Anwendungen im DLR und in Drittmittelprojekten sind vielfältig. Der „klassische“ Anwendungsfall ist die Strömungssimulation, andere Anwendungen sind die Simulation von Energieflüssen in Schienenfahrzeugen, ein städtisches Desaster-Managementsystem und die Auslegung mechanischer Bauteile.

3D-Interaktion

Die Technik der „virtuellen Realität“ erlaubt es dem Nutzer, mit den Objekten in einer computergenerierten Szene möglichst naturgetreu und intuitiv umzugehen. Der Szene liegt ein räumliches Datenmodell zugrunde. Erzeugt wurde es zum Beispiel aus Fotografien einer Planetenoberfläche oder den Komponenten eines Raumfahrzeugs im Orbit. Dabei gibt es zwei große Herausforderungen: die hochaufgelöste visuelle Darstellung in Echtzeit zu erreichen und ein haptisches Feedback bei der manuellen Interaktion mit den Objekten zu bekommen. Wir erforschen neue Techniken, die dies ermöglichen, und wenden sie in Projekten mit anderen DLR-Instituten und externen Partnern an. Die wichtigsten Einsatzgebiete sind derzeit die interaktive Visualisierung der Marsoberfläche und die simulierte Reparatur eines Satelliten.

Programmatische Einbindung und Kooperationen

Programmatische Einbindung

Wie die anderen Institute und Einrichtungen bezieht SC den größten Teil der internen Finanzierung von den Programmdirektionen. Derzeit sind wir mit Projektbeiträgen in den Forschungsbereichen Raumfahrt, Luftfahrt und Verkehr vertreten:

- **Raumfahrt:** Der größte Teil der Projekte ist der Programmlinie „Technik für Raumfahrtsysteme" zugeordnet. Hier werden technologische Grundlagen für die anderen Programmlinien entwickelt. Ferner beteiligen wir uns an den Programmlinien Raumtransport und Robotik. Das Bremer Institut für Raumfahrtsysteme ist unser wichtigster interner Partner.
- **Luftfahrt:** Im Forschungsgebiet „Aircraft Research" kooperieren wir insbesondere mit dem Braunschweiger Institut für Aerodynamik und Strömungstechnik, im Forschungsgebiet „Engine Research" mit dem Kölner Institut für Antriebstechnik. Ferner sind wir beteiligt an Projekten aus den Forschungsgebieten „ATM and Operation" und „Rotorcraft Research".
- **Verkehr:** Im Forschungsbereich Verkehr leisten wir Beiträge zu den Forschungsgebieten „Bodengebundene Fahrzeuge" und „Verkehrssystem". Das Institut für Flughafenwesen in Köln ist derzeit unser wichtigster Partner.
- **IT-Management:** Eine Besonderheit unserer internen Finanzierung sind die Projekte im Auftrag des IT-Managements. Hierunter fallen unsere Querschnittsarbeiten zum Software-Engineering sowie kleinere Softwareentwicklungen, die keinem DLR-Forschungsbereich zugeordnet werden können.

In den ersten Jahren war SC dem Vorstandsbereich Luftfahrt zugeordnet und wurde überwiegend von dort finanziert. Im Laufe der Zeit hat sich das Verhältnis deutlich in Richtung Raumfahrtforschung verschoben. Inzwischen machen die Raumfahrt-Ressourcen über die Hälfte unseres gesamten internen Budgets aus. Von der Programmdirektion Energie erhalten wir keine Finanzierung.

Kooperation mit DLR-Instituten

Dem breiten Themenspektrum unserer Softwareprojekte entsprechend kooperieren wir mit den meisten DLR-Instituten. Zu den Ausnahmen zählen die wenigen Institute, bei denen Softwareentwicklung keine nennenswerte Rolle spielt, sowie die Institute des Forschungsbereichs Energie.

Die wichtigsten Kooperationspartner unserer Arbeitsgruppen im DLR sind:

- **Verteilte Softwaresysteme:** Institute, bei denen die Systemkompetenz eine große Rolle spielt, sind hier die Hauptpartner. Auf Luftfahrtseite ist das die Einrichtung für Luftfahrtkonzepte in Hamburg, bei der Raumfahrt das Institut für Raumfahrtsysteme in Bremen. Zusätzlich sind jeweils viele andere Institute mit ihren spezifischen Anwendungen an den Integrationsprojekten beteiligt.
- **High-Performance-Computing:** Wir arbeiten mit allen DLR-Instituten zusammen, in denen die numerische Simulation eine nennenswerte Rolle spielt. Vor allem zwei Institute entwickeln eigene Software zur numerischen Strömungssimulation: das Institut für Aerodynamik und Strömungstechnik in Braunschweig und das Kölner Institut für Antriebstechnik. In beiden Fällen sind unsere Mitarbeiterinnen und Mitarbeiter in die Kern-Entwicklerteams integriert.
- **Simulation und Modellierung:** Das Institut für Raumfahrtsysteme betreibt eine „Concurrent Engineering Facility" (CEF) für Konzeptstudien und den frühen Entwurf von Raumfahrtsystemen. Diese Anlage wurde zunächst mit der Excel-basierten Software der ESA betrieben. Mit der Software „Virtueller Satellit" haben wir die Softwareausstattung der CEF modernisiert.
- **Eingebettete Systeme:** Das Berliner Institut für Optische Sensorsysteme und das Institut für Raumfahrtsysteme sind unsere wichtigsten Partner bei der Entwicklung von Lageregelungssoftware für Satelliten. In den vergangenen Jahren haben wir die Kooperation erweitert um die Themen „verteilte Onboard-Computersysteme" und „autonome optische Navigation".
- **Wissenschaftliche Visualisierung:** Die Zusammenarbeit mit dem Institut für Plantenforschung bei der 3D-Visualisierung von Planetenoberflächen begann mit einer Dissertation. Inzwischen ist daraus ein gemeinsames EU-Projekt unter unserer Leitung geworden.

- **Virtuelle Realität:** Hier steht die Anwendung „On-Orbit-Servicing“ im Vordergrund, also die ferngesteuerte Reparatur von Raumfahrtsystemen im Orbit. Wir kooperieren mit dem Institut für Robotik und Mechatronik in einem programmatisch finanzierten Projekt.

Noch größer wird die Liste der Partnerinstitute, wenn man die Zusammenarbeit im Software-Engineering-Netzwerk berücksichtigt. Mit derzeit 32 beteiligten Organisationseinheiten ist hier die Zusammenarbeit fast flächendeckend.

Nationale und internationale Kooperationen

Alle Arbeitsgruppen unterhalten externe Kooperationen zur Weiterentwicklung ihrer Technologien. Im Folgenden nennen wir nur die wichtigsten:

- **Verteilte Softwaresysteme:** Das Integrationsframework RCE haben wir ursprünglich in enger Kooperation mit der Flensburger Schiffbaugesellschaft und mit Fraunhofer SCAI entwickelt. In den letzten Jahren arbeiteten wir vornehmlich daran, RCE für DLR-Institute weiter zu entwickeln. Externe Kooperationen beginnen erst jetzt wieder, zum Beispiel mit Airbus Defense and Space, mit dem Maritime Research Institute Netherlands (MARIN) oder dem Korea Aerospace Research Institute (KARI) aus Südkorea.
- **High-Performance-Computing (HPC):** Wichtige Partner, mit denen wir Algorithmen zum Lösen dünnbesetzter Probleme entwickeln, sind die Friedrich-Alexander-Universität Erlangen-Nürnberg (Prof. Gerhard Wellein) und die Universität Groningen (Prof. Fred Wubs). In der Industrie ist Airbus Defense and Space ein Partner, mit dem wir unsere Geometrie-Bibliothek TiGL weiter entwickeln. Mit dem Argonne National Laboratory und der Firma Continuum Analytics aus Austin gestalten wir seit fünf Jahren eine Workshop-Reihe auf der Supercomputing-Konferenz zu Python im High-Performance-Computing.
- **Software-Engineering:** Für das DLR entwickeln und evaluieren wir Software-Engineering-Werkzeuge. Wichtige Partner dabei sind vor allem kleine, innovative Unternehmen. Mit TripleChecker (Nuno Brito) aus Darmstadt arbeiten wir an der automatischen Prüfung von Open-Source-Lizenzen. Mit Quantified Code (Andreas Dewes) aus Berlin und München kooperieren wir bei der Erkennung von Code-Schwächen (insbesondere Performance-Schwächen). Mit Torsion Analytics aus München arbeiten wir an der Migration von IDL nach Python.
- **Simulation und Modellierung:** Unser wichtigster Partner bei der Weiterentwicklung von Software für das Concurrent Engineering und Wissensmanagement ist ESA ESTEC (Massimo Bandecchi). Beim Thema modellbasierte Ansätze kooperieren wir eng mit ESA ESTEC (Joachim Fuchs), TU Chemnitz (Prof. Matthias Werner), MIT / Skoltech (Prof. Alessandro Golkar), und der University of Rome (Andrea D‘Ambrogio). Weitere Kontakte bestehen zu JPL/NASA, zur JAXA und zu chinesischen Universitäten und Forschungseinrichtungen.
- **Eingebettete Systeme:** Die Astro- und Feinwerktechnik Adlershof GmbH ist unser langjähriger Partner in der Entwicklung von Lageregelungssoftware für Satellitenmissionen. Im Verbundprojekt MAIUS nutzen wir die Kooperation insbesondere mit der Leibniz-Universität Hannover (Prof. Ernst Rasel) und dem Zentrum für angewandte Raumfahrttechnik und Mikrogravitation (ZARM) dazu, modellbasierte Ansätze auch für spätere Projektphasen zu entwickeln und zu evaluieren.
- **Wissenschaftliche Visualisierung:** Mit der TU Kaiserslautern (Prof. Hans Hagen) verbindet uns eine langjährige Kooperation in der Doktorandenausbildung. Mit dem Vertrag zur „Angeleiteten Forschung“ haben wir sie neu ausgerichtet. Beim Thema Tensorfeld-Visualisierung arbeiten wir eng mit den Universitäten Leipzig (Prof. Gerik Scheuermann) und Saarbrücken (Prof. Markus Stommel) zusammen. Weitere Kooperationen bestehen mit dem Konrad-Zuse-Institut in Berlin (Hans-Christian Hege) und der Universität Linköping (Prof. Anders Ynnermann).
- **3D-Interaktion:** Ein Kooperationsabkommen mit der RWTH Aachen (Prof. Torsten Kuhlen) regelt die Zusammenarbeit bei der Weiterentwicklung der ViSTA-Software für Anwendungen der virtuellen Realität (VR). Im EU-Projekt „CROSS DRIVE“ arbeiten wir mit der University Salford an kooperativen virtuellen Umgebungen (Prof. Dave Roberts, Prof. Terrence Fernando).

Forschungsergebnisse und Planungen

Dieser Hauptteil des Statusberichts gibt einen Überblick über unsere wissenschaftliche Arbeit. Zur besseren Übersicht gehen wir dabei nach den Themen der Arbeitsgruppen vor, nicht nach den zahlreichen Einzelprojekten.

Wir charakterisieren jeweils kurz das Thema, fassen den Stand der Technik zusammen und beschreiben dann die Projektarbeiten und Ergebnisse, die wir im Berichtszeitraum erzielt haben. Den Abschluss bildet jeweils ein Ausblick auf zukünftige Entwicklungen.

Verteilte Softwaresysteme

Es gibt Software, die ihre Aufgabe nur lösen kann, indem sie zusammen mit anderer Software auf anderen Computern ausgeführt wird und die Programme untereinander kommunizieren. Diese Softwareprogramme bilden zusammen ein verteiltes Softwaresystem. Die Aufgaben in der Forschung und Entwicklung sind komplex. Sie können oft nur mit einem verteilten Softwaresystem gelöst werden. Wir entwickeln verteilte Softwaresysteme für Anwendungsbereiche im DLR, für die es keine fertige Software gibt. Aktuell entwickeln wir Software für die multidisziplinäre Zusammenarbeit, für die Eignungsdiagnostik sowie für die Telemedizin.

Software für multidisziplinäre Zusammenarbeit

Wenig Betriebskosten, lange Lebensdauer, umweltfreundlich: Das erwarten Fluggesellschaften von neuen Flugzeugen. Das Ziel des DLR ist es, neue Flugzeugkonfigurationen wie Schulterdeckerflugzeuge oder welche mit einem so genannten Blended-Wing-Body am Computer zu entwerfen, zu analysieren und zu bewerten (Abbildung 2). Ein Flugzeug zu entwerfen erfordert Experten von verschiedenen Disziplinen wie Aerodynamik, Strukturmechanik oder Antriebstechnik (Abbildung 3). Um ein Flugzeug am Computer zu entwerfen, programmieren die Experten Entwurfs- und Analysewerkzeuge, die auf einen Teilaspekt des Entwurfs spezialisiert sind. Um ein gesamtes Flugzeug zu entwerfen, müssen die Werkzeuge verknüpft werden. Sie sind untereinander abhängig. Das muss beim Verknüpfen berücksichtigt werden.

Das weltweite Verkehrsnetz aus Autos, Zügen, Schiffen und Flugzeugen wird immer dichter. Das DLR erforscht den Verkehr: Was sind die Ursachen, Wirkungen und Wechselwirkungen? Das Ziel ist vorherzusagen, wie sich der Verkehr entwickeln wird. Dafür erstellt das DLR ein Modell, welches das weltweite Verkehrssystem simulieren soll. Experten verschiedener Disziplinen sind beteiligt: Geographen, Ökonomen oder Physiker. Jeder Experte bringt Einzelmodelle aus seiner Fachdisziplin ein. Daraus entsteht zum Beispiel ein Modell des Luftverkehrs. Die Einzelmodelle werden zu einem Gesamtmodell verknüpft. Die Einzelmodelle sind untereinander abhängig. Das muss – analog zur Luftfahrt – beim Verknüpfen berücksichtigt werden.

Die Beispiele zeigen: Multidisziplinäre Zusammenarbeit ist in der Forschung und Entwicklung sehr wichtig.

Abbildung 2: Simulation eines Flugzeugs mit einem Blended-Wing-Body. Der Rumpf geht fließend in den Flügel über.

Abbildung 3: Experten von verschiedenen Disziplinen wie Aerodynamik, Strukturmechanik oder Antriebstechnik entwerfen ein Flugzeug.

Multidisziplinäre Zusammenarbeit
Einzelmodelle aus der Verkehrsforschung sind aus technischer Sicht identisch mit Entwurfs- und Analysewerkzeugen aus der Luftfahrtforschung. Es sind Computerprogramme, die Eingabedaten erwarten, etwas berechnen und Ausgabedaten erzeugen. So kann zum Beispiel ein Programm alle Flugrouten eines Jahres aus einer Datenbank lesen. Das Programm berechnet daraus die Schadstoffemissionen. Die Ergebnisse schreibt es in eine Datei. Nachfolgend werden sowohl die Einzelmodelle aus dem Verkehr als auch die Entwurfs- und Analysewerkzeuge aus der Luftfahrt als Programme bezeichnet.

Programme zu verknüpfen bedeutet bei den Anwendungsbeispielen aus der Luftfahrt und dem Verkehr, die Programme auszuführen sowie Eingabe- und Ausgabedaten auszutauschen. Die Reihenfolge, in der die Programme ausgeführt werden, ist sehr wichtig. Sie bildet die Abhängigkeiten zwischen den Programmen ab. So darf ein Programm erst ausgeführt werden, nachdem alle benötigten Eingabedaten von anderen Programmen als Ausgabedaten erzeugt wurden. Welche Eingabedaten von welchen Programmen benötigt werden, definieren die Experten.

Multidisziplinäre Teams sind oft auf mehrere Standorte verteilt. So sitzen im DLR die Experten der Aerodynamik in Braunschweig und die Antriebstechniker in Köln. Die Programme, die verknüpft werden, müssen meist an dem Ort ausgeführt werden, an dem sich die Experten befinden, die das Programm erstellt haben. Dafür gibt es entweder technische oder politische Gründe. Um Programme zu verknüpfen, müssen die Programme auf verschiedenen Computern an verschiedenen Standorten ausgeführt und die Daten zwischen ihnen ausgetauscht werden.

Wenn Experten verschiedener Disziplinen zusammenarbeiten, ergeben sich drei Arten von Herausforderungen:

- menschliche Herausforderungen
- fachliche Herausforderungen
- technische Herausforderungen

Menschliche Herausforderungen sind u.a. das Spannungsfeld zwischen disziplinspezifischen Zielen einerseits und das übergeordnete, interdisziplinäre Ziel andererseits. Menschliche Herausforderungen müssen auf Organisationsebene gelöst werden. Fachliche Herausforderungen betreffen u.a. die fachlichen Abhängigkeiten zwischen Disziplinen und damit Programmen. Fachliche Herausforderungen müssen von den Experten gelöst werden.

Technische Herausforderungen beziehen sich auf die technische Verknüpfung der Programme:

- Programme sind heterogen (bzgl. Programmiersprache, Betriebssystem, etc.)
- Programme laufen auf verschiedenen Computern
- Programme sind dynamisch (neue Version, neues Programm, Programm nur temporär verfügbar)
- Daten müssen verteilt, ausgetauscht und dem Nutzer zur Auswertung bereitgestellt werden

Die technischen Herausforderungen werden von so genannten Integrationsumgebungen gelöst.

Integrationsumgebungen
Verknüpfte Programme können manuell ausgeführt werden. Die Daten können über E-Mail oder Fileserver ausgetauscht werden. Dieser manuelle Ansatz hat Nachteile. Er ist fehleranfällig, für Dritte schwer nachzuvollziehen und aufwändig zu wiederholen. Eine Alternative für den manuellen Ansatz sind Integrationsumgebungen. In ihnen werden die Programme verschiedener Disziplinen eingebunden, miteinander verknüpft und in Abhängigkeit zueinander ausgeführt.

Wir haben 2006 in dem BMBF-Projekt SESIS damit begonnen, die verteilte Integrationsumgebung RCE (Remote Component Environment) zu entwickeln und für den frühen Entwurf von Schiffen einzusetzen. Wir haben zum einen die Anforderungen aus dem Schiffbau bei der Entwicklung berücksichtigt. Zum anderen haben wir unsere Erfahrungen mit Integrationsumgebungen im Flugzeug- und Automobilbau einfließen lassen. Entstanden ist mit RCE eine Software, die wir inzwischen für weitere Anwendungen über den Schiffbau hinaus einsetzen: Zunächst für den Vorentwurf von Flugzeugen und die Auslegung von Raumfahrzeugkomponenten, später unter anderem in der Verkehrsforschung sowie im hochgenauen Flugzeugentwurf.

Wir entwickeln RCE zusammen mit Partnern in Forschungsprojekten. RCE wird in den Projekten als Integrationsumgebung eingesetzt. In jedem Projekt wird es erweitert. Das Budget, das für RCE in allen Projekten zusammen verfügbar ist, bildet die Finanzierung von RCE. Beispiele für Projekte sind:

- **FrEACs (Future Enhanced Aircraft Configurations) und Digital-X:** DLR-Projekte, in denen Flugzeugkonfigurationen in verschiedenen Detailstufen analysiert, bewertet und optimiert werden. In FrEACs und Digitial-X sind jeweils über zehn Institute beteiligt. Es sind aktuell die beiden größten Projekte in der Luftfahrt im DLR, die sich mit dem multidisziplinären Flugzeugentwurf beschäftigen. Die Einrichtung Lufttransportsysteme in Hamburg leitet FrEACs. Digital-X wird vom Institut für Aerodynamik und Strömungstechnik in Braunschweig geleitet.
- **VEU (Verkehrsentwicklung und Umwelt):** DLR-Projekt, in dem das DLR erforscht, wie sich der Verkehr entwickelt und wie er sich auf die Umwelt auswirkt. In VEU sind acht Institute beteiligt. Das Institut für Verkehrsforschung in Berlin leitet VEU.
- **THERMAS:** DLR-Projekt, in dem Raumfahrzeugkomponenten ausgelegt werden (siehe Kapitel „Multidisziplinäre Optimierung"). In THERMAS sind vier Institute beteiligt. Das Institut für Aerodynamik und Strömungstechnik in Köln leitet das Projekt.

Verteilte Integrationsumgebung RCE
In unserer verteilten Integrationsumgebung RCE werden Programme integriert und zu einem so genannten Workflow verknüpft. Welche Daten ausgetauscht und wie die Programme ausgeführt werden, beschreibt der Nutzer mit der grafischen Nutzeroberfläche von RCE (Abbildung 4).

Startet der Nutzer einen Workflow, führt RCE die Programme an den verschiedenen Standorten aus und übermittelt die Daten zwischen den Programmen. Die grafische Nutzeroberfläche von RCE zeigt in Abbildung 5 einen Workflow, der gerade ausgeführt wird. Er optimiert die Gleitzahl des SpaceLiners (siehe Kapitel „Multidisziplinäre Optimierung").

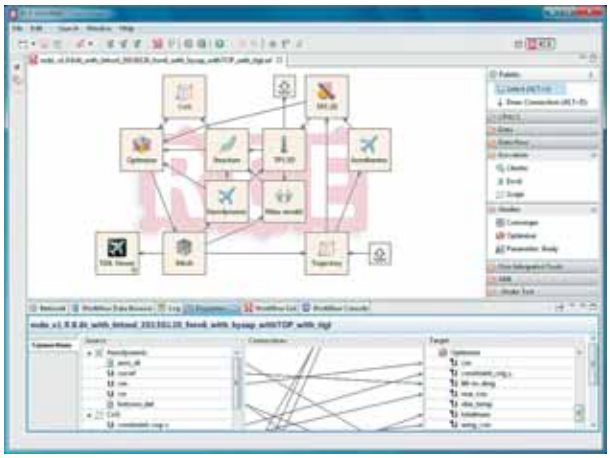


Abbildung 4: Die grafische Nutzeroberfläche von RCE mit dem Workflow-Editor. Die Quadrate sind die Programme, die Linien dazwischen definieren den Datenaustausch. Sie repräsentieren die Abhängigkeiten zwischen den Programmen.

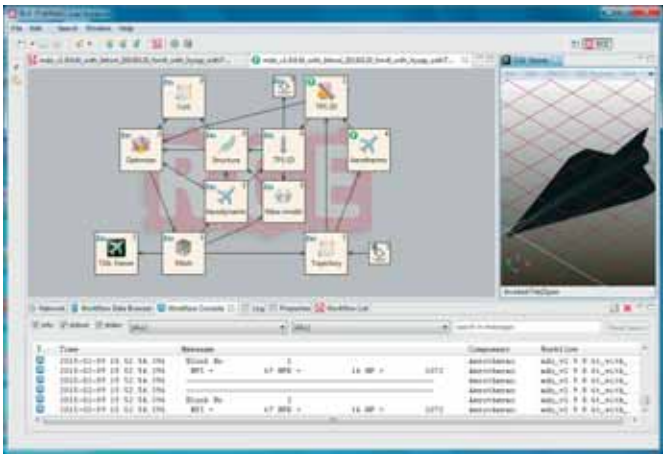


Abbildung 5: Die grafische Nutzeroberfläche von RCE mit einem Workflow, der gerade ausgeführt wird.

RCE ist eine verteilte Software. Instanzen von RCE können sich untereinander zu einem Netzwerk verbinden. Das Netzwerk ist ein dynamisches Peer-to-Peer-System. RCE-Instanzen können dynamisch hinzukommen und wieder verschwinden. Eine RCE-Instanz kann so konfiguriert werden, dass sie verschiedene Rollen im Netzwerk (gleichzeitig) einnehmen kann (Abbildung 6):

- **Compute-Instanz:** stellt Programme zur Verfügung
- **Nutzer-Instanz:** nutzt eigene Programme und Programme von anderen Compute-Instanzen (meist mit grafischer Nutzeroberfläche)
- **Relay-Instanz:** vermittelt zwischen Compute- und Nutzer-Instanzen

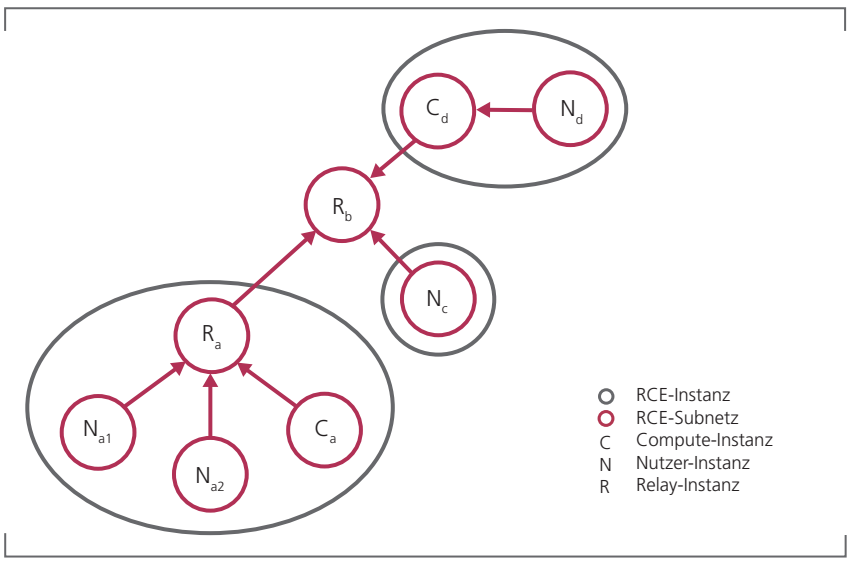


Abbildung 6: RCE-Netzwerk bestehend aus einzelnen Instanzen von RCE. Die Pfeile geben an, welche Instanzen direkt miteinander verbunden sind. Die Richtung der Pfeile zeigt, welche Instanz die Verbindung initialisiert hat.

RCE unterstützt die Zusammenarbeit in multidisziplinären Teams, indem es Workflows mit Programmen verteilt und automatisiert ausführt. Um wissenschaftliche Erkenntnisse aus einem Workflowlauf zu gewinnen, müssen die Ergebnisdaten nach der Ausführung ausgewertet werden. RCE unterstützt auch diese Phasen in der multidisziplinären Zusammenarbeit, da es über ein verteiltes Datenmanagement für Ergebnisdaten und über verteilte Datenstrukturen für flüchtige Daten verfügt. Flüchtige Daten sind Daten, die nicht gespeichert und zum Beispiel nur in der grafischen Nutzeroberfläche angezeigt werden. Experten im Team können Ergebnisdaten einsehen und laufende Workflows überwachen, sobald sie mit dem RCE-Netzwerk verbunden sind. Nutzer können zum Beispiel auch den zeitlichen Verlauf ihres Workflows in der grafischen Oberfläche von RCE sehen (Abbildung 7).

Abbildung 7: Zeitlicher Verlauf eines Workflows. Die Balken zeigen, wann ein Programm des Workflows wie lange ausgeführt wurde.

Wir setzen RCE kontinuierlich in neuen Anwendungsbereichen ein. In bestehenden Anwendungsbereichen entwickeln wir zusammen mit den Nutzern stetig neue Ideen, mit denen sie in ihren multidisziplinären Teams ihre komplexen Aufgaben noch besser lösen können. Wir entwickeln RCE kontinuierlich weiter, verändern bestehende Funktionalität und erweitern es um neue.

Zum Beispiel haben wir erkannt, dass die Zusammenarbeit in verteilten Teams besser funktioniert, wenn einzelne Teammitglieder ad-hoc zusammenarbeiten können. Auf technischer Ebene erfordert das in RCE ein Netzwerk, das sich dynamisch verändern kann. Zum einen müssen RCE-Instanzen sich verbinden können, ohne dass andere ihre RCE-Instanz neu starten müssen. Zum anderen müssen Programme von Teammitgliedern bereitgestellt werden können, ohne dass ein Administrator hinzugezogen oder eine RCE-Instanz neu gestartet werden muss.

Um das zu erreichen, haben wir die Netzwerkschicht von einem Client-Server-Konzept auf das dynamische Peer-to-Peer-System umgestellt. Alle RCE-Instanzen eines Netzwerks bekommen damit sofort alle Änderungen im RCE-Netzwerk mit. Änderungen sind zum Beispiel:

- RCE-Instanz verbunden oder getrennt
- Programme bereitgestellt, entfernt oder in einer neuen Version verfügbar

Außerdem haben wir RCE so erweitert, dass Nutzer ihre Programme über die grafische Nutzeroberfläche von RCE einfach integrieren und sofort verwenden können. Durch die beschriebenen Eigenschaften des Peer-to-Peer-Systems können andere Nutzer diese Programme sofort einsetzen – insofern diese Funktion aktiviert ist.

Vor allem in der frühen Phase der Zusammenarbeit entwickeln Experten ihre Programme noch ständig weiter. Es gibt häufig neue Versionen, und die Programme laufen in dieser Phase auch oft nur auf dem Computer des Experten. Nachdem wir RCE erweitert hatten, sind die technischen Hürden sehr niedrig, um ad-hoc zusammenzuarbeiten. Zum Beispiel kann ein Experte einem anderen Teammitglied die neueste Version seines Programms sofort verfügbar machen, auch wenn es nur lokal auf seinem Computer lauffähig ist.

Die numerische Optimierung ist ein weiteres Beispiel, das zeigt, dass sich neue Anforderungen an RCE ergeben. Zunächst wurde RCE eingesetzt, um komplexe Systeme wie beispielsweise Flugzeuge zu entwerfen und zu analysieren, später dann auch, um sie

zu optimieren. Daher haben wir RCE in den vergangenen Jahren um Optimierungsmethoden erweitert. Aktuell entwickeln wir eine Schnittstelle in RCE, mit der es Nutzern möglich ist, eigene Optimierungsmethoden in RCE einzubinden, ohne RCE selber erweitern zu müssen. Die Anforderung kam aus dem DLR-Projekt THERMAS. Einerseits setzen wir dort RCE ein. Andererseits entwickeln wir neue Optimierungsmethoden (siehe Kapitel „Multidisziplinäre Optimierung"). Diese möchten wir als eigenständige Softwareprogramme entwickeln. Gleichzeitig sollen sie aber einfach in RCE eingebunden werden können.

Softwaretechnologien im RCE
Mit der Entwicklung von RCE evaluieren wir ständig neue Softwaretechnologien, wenden sie bei Bedarf an und entwickeln gegebenenfalls neue Methoden. Da RCE eine verteilte Umgebung ist, tun wir das speziell auf den Gebieten der dynamischen Peer-to-Peer-Netzwerke, dem verteilten Datenmanagement und der verteilten Datenstrukturen. Weitere Beispiele sind Usability, automatisiertes Testen und Software-Engineering.

Usability bezeichnet die Gebrauchstauglichkeit von Software. Die Aufgaben, die mit RCE gelöst werden, sind komplex. Unser Anspruch ist, dass RCE dennoch leicht verständlich und intuitiv benutzbar ist. Wir wenden verschiedene Usability-Methoden an, um diesem Anspruch gerecht zu werden. So führen wir zum Beispiel Studien durch, bei denen wir Nutzer beobachten, wenn sie RCE verwenden. Diese Studien sind zeitintensiv. Wir erproben Methoden, die mit weniger Zeitaufwand Usability-Schwächen aufdecken. Bei einem unserer Ansätze werden Ereignisse protokolliert, die der Nutzer auslöst, wenn er mit RCE arbeitet - zum Beispiel, wenn er auf einen Button klickt. In diesem Ereignisprotokoll sucht ein Algorithmus nach Mustern. Die Muster werden vorher definiert. Ein Muster kann aus mehreren Ereignissen bestehen, die ausgelöst werden, wenn der Nutzer den Abrechen-Button eines Dialogs klickt. Die Herausforderung bei diesem Ansatz ist es, die Muster zu definieren. Sie müssen genau genug sein, so dass nicht zu viele Usability-Schwächen erkannt werden. Sie dürfen aber auch nicht zu speziell sein, so dass echte Schwächen unerkannt bleiben. Wir haben den Ansatz so allgemein gehalten, dass er auch für andere Software genutzt werden kann.

Software zu testen ist sehr wichtig. So viele Tests wie möglich müssen automatisiert sein. Das stellt sicher, dass sie regelmäßig und korrekt ausgeführt werden. Es ist aktuell nicht möglich, verteilte Softwareprogramme wie RCE einfach automatisiert zu testen. Daher entwickeln wir ein Framework, um Tests für beliebige, verteilte Software einfach zu schreiben und automatisiert ausführen zu können. Bei unserem Ansatz definieren Entwickler die Tests. Sie definieren die Testlogik und auf welchen Ressourcen ein Test ausgeführt werden soll. Wir entwickeln eine domänenspezifische Sprache (DSL), mit der die Tests beschrieben werden.

Das Framework stellt einige Herausforderungen, die wir bei der Entwicklung berücksichtigen:

- Die domänenspezifische Sprache muss so definiert werden, dass sie mächtig genug ist, um alle verteilten Szenarien abzudecken. Gleichzeitig muss sie speziell genug sein, so dass Entwickler effizient mit ihr umgehen können.
- Ressourcen müssen entsprechend der Testbeschreibung automatisiert erzeugt und wieder freigegeben werden.
- Die zu testende Software muss entsprechend der Testbeschreibung automatisiert installiert und konfiguriert werden.

RCE ist inzwischen zu einer komplexen Software herangewachsen. Je komplexer eine Software ist, desto größer ist die Gefahr, dass sie erodiert. Ist eine Software erodiert, kann sie nur noch schwer gewartet und kaum weiterentwickelt werden. Mit geeigneten Methoden aus dem Software-Engineering kann eine Software davor geschützt werden, zu erodieren. Die angewendeten Methoden müssen zur Software und zum Entwicklerteam passen. Verändert sich eines von beiden, müssen immer wieder auch die Methoden angepasst werden.

Um RCE für Entwickler verständlicher zu machen, möchten wir Methoden entwickeln, um verschiedene Aspekte von RCE zu visualisieren. Wir möchten zum Beispiel die Abhängigkeiten zwischen Komponenten visualisieren, um sicherzustellen, dass keine falschen existieren. Wir möchten auch visualisieren, wie häufig Komponenten wie stark geändert wurden, um potentiell erodierte Komponenten früh zu identifizieren. Erodierte Komponenten werden in der Regel oft, aber nur in geringem Maße geändert.

Wir werden die Methoden so generisch wie möglich halten, so dass sie auch auf andere Software angewandt werden können.

Zukunftsthemen
Multidisziplinäre Zusammenarbeit ist komplex. RCE unterstützt multidisziplinäre Zusammenarbeit, indem es die technische Komplexität minimiert. Wir haben das Minimum noch nicht erreicht und sehen für RCE noch viel Potential. Folgende drei Themen möchten wir unter anderem in Zukunft bearbeiten:

- Monitoring von Workflows
- Sicherheit in Multi-Team-Netzwerken
- Nachvollziehbarkeit von Ergebnissen

Workflows, die lange laufen, müssen früh überwacht werden. Je früher Experten Probleme erkennen, desto geringer sind die Kosten, wenn ein Workflow ohne valides Ergebnis abgebrochen werden muss. Wir möchten RCE um Monitoring-Funktionen erweitern, die den Nutzern besser helfen, technische und fachliche Probleme zu erkennen. Zum Beispiel soll RCE versuchen, Anomalien automatisiert zu erkennen, indem es aktuelle und frühere Workflow-Läufe miteinander vergleicht. Außerdem sollen Warnungen und Fehler den Nutzer aktiv erreichen. Unter anderem werden wir dafür eine App für Smartphones programmieren.

Multidisziplinäre Teams sind nicht immer disjunkt. Ein Experte kann in mehreren Teams arbeiten. In jedem Team könnte RCE eingesetzt werden. Oft dürfen die Mitglieder eines Teams die Daten und Programme des anderen Teams nicht sehen bzw. nutzen. Um die technische Komplexität zu minimieren, sollte der Experte dennoch mit beiden Teams über eine RCE-Instanz zusammenarbeiten können. RCE soll dafür sorgen, dass die Daten und Programmen über Teamgrenzen hinweg so sichtbar sind, wie erlaubt. Wir entwickeln ein Sicherheitskonzept für RCE, das dies möglich macht. Eine Herausforderung ist dabei das Peer-to-Peer-System. Eine RCE-Instanz kann auch Vermittler zwischen zwei anderen RCE-Instanzen sein. Zum Beispiel wenn zwischen ihnen keine direkte Verbindung besteht. Eine andere Herausforderung ist, Identitäten im gesamten Netzwerk vor Angriffen zu schützen. Es muss bei jeder Aktion im Netzwerk zweifelsfrei festgestellt werden können, wer die Aktion ursprünglich ausgelöst hat. Nur dann ist es sinnvoll zu prüfen, ob derjenige die Rechte für die Aktion besitzt.

Hinter jedem Workflow stecken fachliche Fragen. Um die Fragen zu beantworten, werten Experten die Ergebnisdaten aus, die ein Workflow generiert. Oft müssen sie nachvollziehen, wie die Daten entstanden sind. Erst dann können sie verstehen, wie Ursachen und Wirkungen zusammenhängen. Das ist grundlegend, um komplexe Systeme zu verstehen und damit gezielt zu entwerfen.
In RCE sind alle Informationen vorhanden, die beschreiben, wie Daten entstanden sind. Wir möchten diese Informationen in RCE aufzeichnen. Mit Provenance-Technologie möchten wir aus den Informationen Wissen generieren (siehe Kapitel „Provenance von Prozessen“). Mit dem Wissen soll RCE Fragen beantworten können:

- Was hat das Programm A in Workflow X an der Flugzeugkonfiguration geändert?
- Welche Programme waren in welcher Version an Workflow Y beteiligt?
- Welche Eingangsparameter hatten Wert Z bei der Verkehrsentwicklung beeinflusst?

Wir möchten die Anwendungsbereiche von RCE erweitern. Ab 2015 werden wir RCE im DLR-Projekt X-TRAS für den Entwurf von Launchern in der Raumfahrt weiterentwickeln und einsetzen. X-TRAS wird vom Institut für Raumfahrtantriebe in Lampoldshausen geleitet. Weiterhin möchten wir mit internationalen Partnern zusammenarbeiten. Ab 2015 werden wir RCE in AGILE und IDEalisM einsetzen und weiterentwickeln. AGILE und IDEalisM sind zwei EU-Projekte mit über 15 Partnern aus der europäischen Luftfahrtindustrie und -forschung, die von der DLR-Einrichtung Lufttransportsysteme in Hamburg geleitet werden.

RCE ist quelloffen. Das heißt, der Quellcode ist frei verfügbar. Zusätzlich stellen wir RCE als installierbare Software bereit. Wir möchten eine Open-Source-Community um das quelloffene RCE herum aufzubauen. Die Community soll sowohl aus Nutzern als auch aus Entwicklern bestehen. Unsere Vision ist, RCE gemeinsam mit Partnern wie Universitäten, Forschungseinrichtungen und Firmen zu entwickeln und einzusetzen, um nationale und internationale Forschung in verteilten, multidisziplinären Teams zu unterstützen. Ein Schlüssel zu dieser Vision ist, dass RCE quelloffen ist. Nur dann können sich sowohl Nutzer als auch andere Entwickler und andere Organisationen mit RCE so identifizieren, dass sie es gemeinsam mit uns weiterentwickeln möchten.

Software für Eignungsdiagnostik

Piloten, Astronauten und Fluglotsen haben komplexe Aufgaben und hohe Verantwortung. Passieren ihnen Fehler, sind die Kosten sehr hoch. Um das Risiko von menschlichen Fehlern zu minimieren, müssen Personen sorgfältig ausgewählt werden, die diese Berufe ausüben. Das DLR entwickelt das psychologische Testverfahren GAP (Group Assessment of Performance and Behaviour), um Menschen innerhalb von Gruppen möglichst zuverlässig zu beurteilen.

GAP basiert auf Rollenspielen, welche die Gruppenteilnehmer individuell und simultan am Computer durchführen. Es ist ein softwarebasiertes Testverfahren. Es ist so ausgelegt, dass es nur mit einer Software durchgeführt werden kann.

Wir entwickeln gemeinsam mit dem Institut für Luft- und Raumfahrtmedizin des DLR die Software. Die Psychologen des Instituts entwickeln die psychologischen Methoden. Wir entwickeln gemeinsam Ideen, wie wir mit Softwaretechnologie diese als Tests realisieren können und setzen diese um. Darüber hinaus evaluieren und entwickeln wir neue Technologien, um den Psychologen Möglichkeiten aufzuzeigen, ihre Methoden umzusetzen.

Testsoftware GAP
Abbildung 8 zeigt den Raum am Institut für Luft- und Raumfahrtmedizin des DLR, in dem mit GAP Tests zur Auswahl von Piloten durchgeführt werden. Es sind die Plätze der Bewerber – im folgenden Kandidaten genannt – und die Plätze der so genannten Beobachter zu sehen. Die Beobachter sind hier Psychologen und Piloten. An jedem Platz läuft eine Instanz der Software. Zusätzlich läuft außerhalb des Raumes auf einem Computer eine weitere Instanz. Darüber sind alle anderen Instanzen miteinander verbunden. Sie bilden zusammen das GAP-Netzwerk (Abbildung 9).

Abbildung 8: Raum im Institut für Luft- und Raumfahrtmedizin des DLR, in dem mit GAP Tests zur Auswahl von Piloten durchgeführt werden.

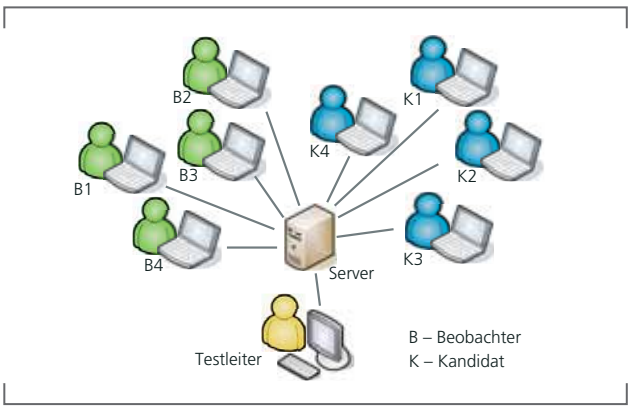
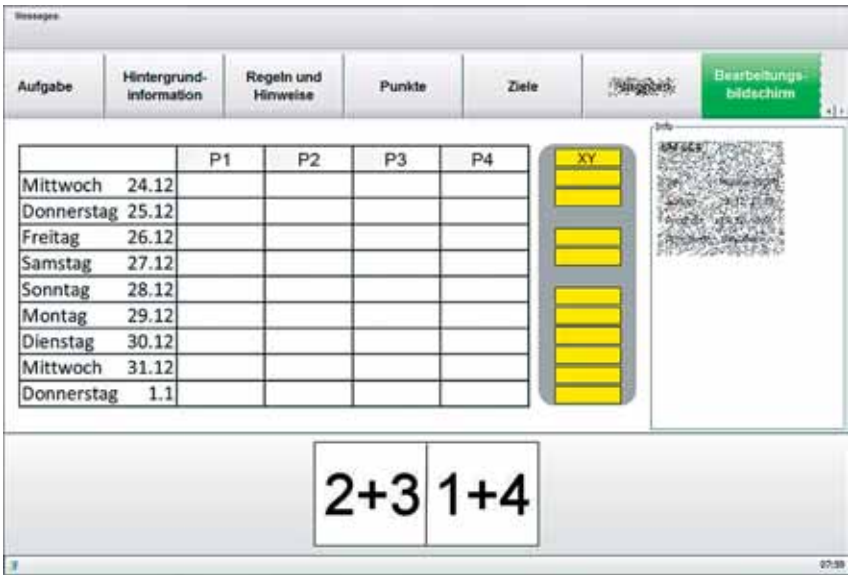


Abbildung 9: GAP-Netzwerk bestehend aus verschiedenen Instanzen von GAP: Kandidaten-, Beobachter-, Testleiter- und Server-Instanzen.

Die Kandidaten spielen gemeinsam ein Rollenspiel. Hinter jedem Rollenspiel steckt ein Szenario mit mehreren Aufgaben. Eine Aufgabe kann zum Beispiel sein, dass die Kandidaten Flugbegleiter sind und den Flugplan der nächsten Woche erstellen sollen. Jedem Kandidaten wird eine bestimmte Rolle zugewiesen – zum Beispiel, am Wochenende keine Flüge haben zu wollen. Es gibt Punkte für die Gruppe und für Einzelpersonen, wenn bestimmte Kriterien erfüllt sind: Keiner macht Überstunden, Pausenzeiten werden eingehalten oder der eigene Wunsch ist erfüllt.

Während die Kandidaten die Aufgabe bearbeiten, sehen sie ein Spielfeld. Die Spielfelder können untereinander vollständig synchronisiert, teilweise synchronisiert oder gar nicht synchronisiert sein. Bei der Aufgabe, einen Flugplan zu erstellen, haben die Kandidaten eine Tabelle und verschiebbare Felder, die ihre Flüge darstellen (Abbildung 10). In diesem Fall sind die Spielfelder nicht synchronisiert. Die Kandidaten sehen nicht die Tabellen der anderen Kandidaten. Sie müssen aber eine gemeinsame Lösung erstellen, die durch die Software geprüft und bewertet wird.

Abbildung 10: Grafische Nutzeroberfläche von GAP aus Sicht eines Kandidaten.



Um die Aufgabe lösen zu können, müssen die Kandidaten miteinander reden. Die Software misst den Redeanteil jedes Kandidaten. Zusätzlich müssen die Kandidaten neben dem Rollenspiel weitere Aufgaben individuell absolvieren, beispielsweise kleine Rechenaufgaben lösen. Am Ende des Tests bewerten sich die Kandidaten selbst und gegenseitig.

Die Beobachter sehen die Spielfelder aller Kandidaten sowie die Ergebnisse und Punkte der Einzel- und Gruppenaufgaben. Sie bewerten die Kandidaten auf Basis von Verhaltensankern. Verhaltensanker können sein: „Gibt Feedback", „Ist freundlich" oder „Unterbricht Andere". Verhaltensanker sind mit so genannten Kompetenzbereichen verknüpft.

Das Testverfahren GAP ist nicht mit einem einzigen Rollenspiel verbunden. Rollenspiele werden weiterentwickelt oder neu entworfen. Wichtig ist, dass Psychologen Rollenspiele in die Software einbinden können, ohne dass wir die Software selbst verändern müssen. Wir haben dafür einen Szenarien-Editor entwickelt. Psychologen entwerfen das Spielfeld, definieren das Punktesystem und legen Verhaltensanker sowie Kompetenzbereiche fest (Abbildung 11).

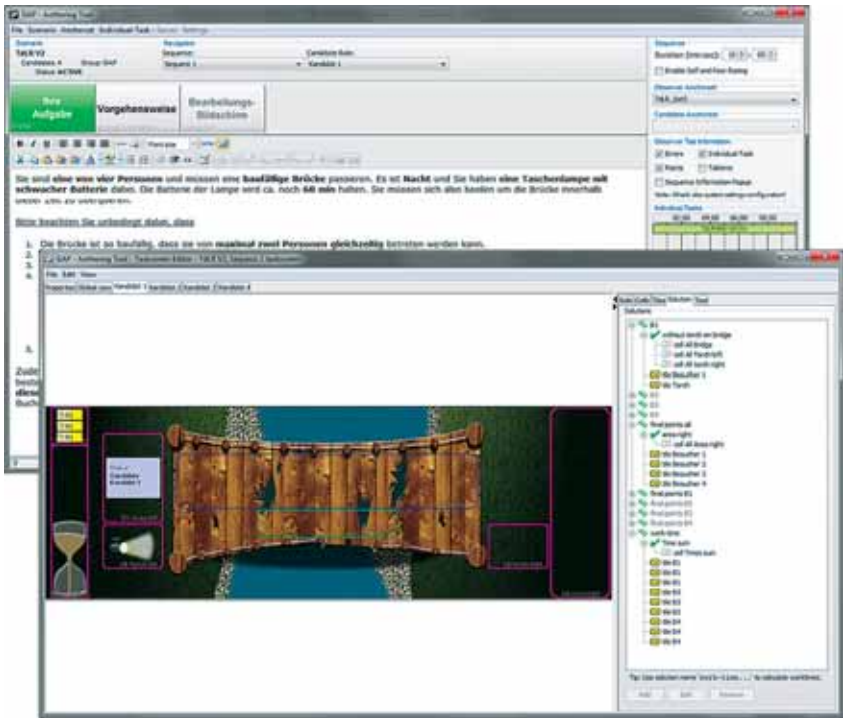


Abbildung 11: Grafische Nutzeroberfläche des Szenarien-Editors von GAP. Es wird gerade ein Szenario entworfen, bei dem eine Brücke unter bestimmten Randbedingungen gemeinsam im Dunkeln überquert werden muss.

GAP ist ein verteiltes Softwaresystem, bei dem es sehr wichtig ist, dass alle beteiligten Instanzen stets synchronisiert sind:

- Kandidaten können Spielfelder von anderen Kandidaten sehen oder sie teilen sich ein gemeinsames Spielfeld. Verändert ein Kandidat sein Spielfeld, muss das sofort für die anderen Kandidaten sichtbar sein. Verzögerungen lenken die Kandidaten vom Rollenspiel ab. Das vermindert die Qualität der Testergebnisse.
- Auch für die Beobachter müssen die Änderungen auf Spielfeldern sofort sichtbar sein. Sie bewerten die Kandidaten während des gesamten Rollenspiels. Dabei wird nicht nur der Verhaltensanker, sondern auch der Zeitpunkt gespeichert, an dem sie das Verhalten beobachtet haben. Das ist wichtig, wenn sie nachvollziehen möchten, welches Verhalten zu dem Anker geführt hat. Das ist besonders dann interessant, wenn ein Verhalten bei zwei Beobachtern zu zwei unterschiedlichen Ankern geführt hat.

Mit GAP werden Piloten und Astronauten ausgewählt, die in ihrem Beruf eine hohe Verantwortung tragen. Verfälschte Testergebnisse können schwerwiegende Folgen haben. Eine fehlerhafte Synchronisierung in GAP kann die Testergebnisse beeinflussen. Die Herausforderung bei der Entwicklung von GAP ist sicherzustellen, dass das verteilte System stets synchronisiert ist.

Zukunftsthemen
Wir möchten den Anwendungsbereich von GAP erweitern. Wir haben die Software bereits für medizinische Studien eingesetzt, bei denen Probanden eine bestimmte Zeit lang von der Außenwelt isoliert leben. Die Probanden haben mit GAP Aufgaben bearbeitet, ohne Kontakt zu den Beobachtern zu haben. Darüber hinaus möchten wir mit anderen Instituten des DLR – wie dem Institut für Luft- und Raumfahrtmedizin oder der Einrichtung Lufttransportsysteme – GAP so erweitern, dass damit multidisziplinäre Teams zusammengestellt werden können. Wie in Kapitel „Multidisziplinäre Zusammenarbeit" beschrieben, lösen wir mit RCE die technischen Herausforderungen, wenn multidisziplinäre Teams zusammenarbeiten. Mit GAP möchten wir auch die menschlichen Herausforderungen angehen.

Wir möchten außerdem mit Softwaretechnologie helfen, das Testverfahren GAP zu erweitern und zu verbessern. Eine Idee ist, die Kandidaten per Video aufzuzeichnen. GAP soll die Aufzeichnungen automatisiert auswerten und anhand von Stimme, Mimik und Gestik das Verhalten eines Kandidaten bestimmen. Zum Beispiel kann eine zittrige Stimme darauf hinweisen, dass ein Kandidat nervös ist. Wenn das Zittern nur sehr leicht war, kann es sein, dass es den Beobachtern während des Tests nicht aufgefallen ist. In dem Fall würde GAP neue Informationen bereitstellen, die in die Auswertung miteinfließen und die Qualität der Testergebnisse verbessern können.

Software für Telemedizin

Medizinische Betreuung über räumliche Distanz heißt Telemedizin. Für Telemedizin gibt es verschiedene Einsatzgebiete:

- Wenn Astronauten im Weltraum auf der internationalen Raumstation ISS erkranken, müssen sie von der Erde aus medizinisch versorgt werden.
- In ländlichen Regionen in Deutschland fehlen niedergelassene Ärzte. Der Weg zum nächsten Arzt ist dort oft lang. Vor allem ältere Menschen können lange Wege nur noch schwer bewältigen. Sie werden daher in ihrem Zuhause von Ärzten aus der Ferne medizinisch betreut (Abbildung 12).
- Chronisch kranke Menschen sollten medizinisch sehr eng betreut werden. Wenn Menschen mit Bluthochdruck oder Diabetes regelmäßig ihre Blutzucker- oder Blutdruckwerte aufzeichnen, können Ärzte Therapien erstellen, die auf den Verlauf der Erkrankung genau zugeschnitten sind.

Wir entwickeln gemeinsam mit dem Institut für Luft- und Raumfahrtmedizin des DLR Ideen, wie wir mit Softwaretechnologie Patienten, Ärzte und Therapieeinrichtungen in der Telemedizin unterstützen können und setzen diese um. Wir erstellen zum Beispiel Software, mit der medizinische Messwerte einfach und sicher zwischen Patienten und Ärzten ausgetauscht werden können.

Abbildung 12: Ältere Frau, die eigenständig ihren Blutdruck misst und die Werte über ihr Smartphone an ihren Arzt übermittelt.

Telemedizin-App Plug & Care Connector
In der Telemedizin müssen Ärzte, Patienten und Therapieeinrichtungen medizinische Daten elektronisch austauschen. Medizinische Daten sind Messwerte, Bilddaten oder Befunde. Es gibt viele verschiedene Messgeräte von unterschiedlichen Herstellern, um medizinische Werte zu messen. Ärzte und Therapieeinrichtungen haben außerdem unterschiedliche Systeme, um Patientendaten zu verwalten. Um Daten elektronisch auszutauschen, müssen die Messgeräte mit den Systemen kommunizieren können. Hersteller von Messgeräten und Systemen achten in der Regel nicht darauf, dass ihre Produkte mit denen anderer Hersteller kompatibel sind. Daher ist es meist nur möglich, Daten auszutauschen, wenn Messgerät und System von einem Hersteller sind. Das ist unflexibel und schränkt Patienten, Ärzte und Therapieeinrichtungen bei der Auswahl der passenden Geräte und Systeme ein.

Wir haben eine Software entwickelt, die Daten von beliebigen Messgeräten empfangen und an beliebige Systeme weiterleiten kann. Die Software heißt Plug & Care Connector und ist eine App für Smartphones. Für jedes Messgerät und jedes System wird eine Schnittstelle implementiert. Wir haben den Plug & Care Connector als Framework entwickelt, indem die Schnittstellen als Plug-ins integriert werden können (Abbildung 13).

Vor allem die Schnittstellen von medizinischen Messgeräten sind proprietär und oft wenig dokumentiert. Es war eine Herausforderung, die Schnittstellen auf dieser Basis robust und sicher zu implementieren. Falsche Messwerte dürfen in der Medizin nicht auftreten.

Bis heute haben wir viele Blutdruck- und Blutzuckermessgeräte, elektronische Waagen und EKG-Geräte angebunden.

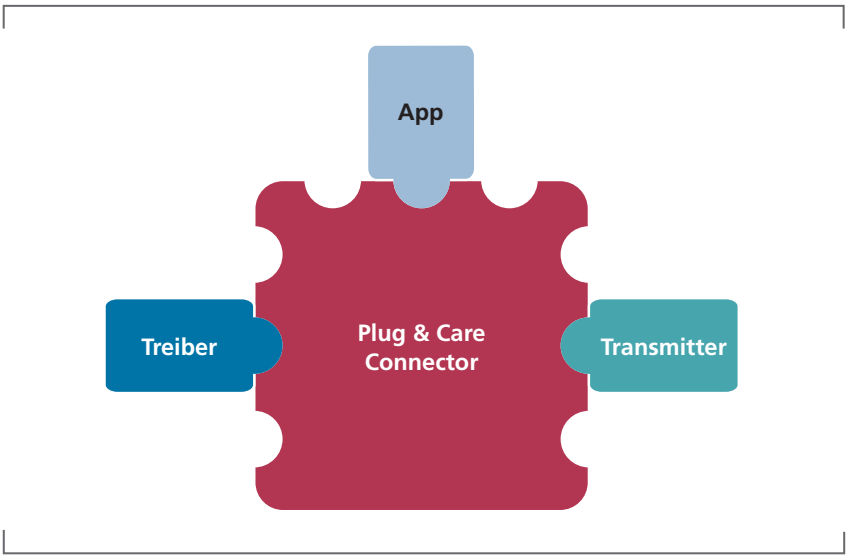


Abbildung 13: Die App Plug & Care Connector kann Daten von beliebigen Messgeräten empfangen und an beliebige Systeme weiterleiten. Für Messgeräte werden Treiber-Plugins und für die Systeme Transmitter-Plugins eingebunden. Zusätzlich können lokale Apps auf die Daten zugreifen.

Zukunftsthemen
„Fit for Duty“ – Piloten oder Fahrer von LKWs und Fernreisebussen müssen unbedingt flug- bzw. fahrtauglich sein. Ist ein Pilot oder Fernfahrer zu müde, nimmt unter anderem seine Reaktionszeit ab. Dann sind Menschen in Gefahr.

Zusammen mit dem Institut für Luft- und Raumfahrtmedizin des DLR möchten wir eine App entwickeln, mit der ein Mensch seine Reaktionszeit messen kann. Mit einem Algorithmus möchten wir die Daten auswerten und die Müdigkeit eines Menschen ableiten. Der Algorithmus soll sowohl die aktuelle Müdigkeit bestimmen als auch die Müdigkeit für einige Stunden vorhersagen. Piloten und Fernfahrer sollen vor Flug- bzw. Fahrtantritt die App nutzen und damit feststellen, ob sie tauglich sind, einen mehrstündigen Flug bzw. eine Fahrt anzutreten. Wir möchten die Softwaretechnologien aus der Telemedizin verwenden und erweitern, um die Tauglichkeit verschiedener Berufsgruppen zu bewerten.

Wie auch in der Telemedizin sollen die Apps von Menschen verwendet werden, die unter Umständen mit Smartphones nicht vertraut sind. Die Apps müssen daher einfach zu bedienen sein. Idealerweise macht es den Menschen sogar Spaß, die Apps zu benutzen. Dadurch würden sie die Apps schneller akzeptieren und auch längerfristig nutzen. Wir möchten uns daher auf die Usability der Apps konzentrieren. Um die Daten auszuwerten, wollen wir Big-Data-Verfahren und Machine-Learning-Algorithmen einsetzen. Die Daten, die ausgetauscht werden, sind meist sensibel, da sie personenbezogen und medizinisch sind. Wir möchten die Provenienz der Daten aufzeichnen, um ihre Wege nachzuvollziehen. Das soll dabei helfen, Datenschutzverletzungen zu erkennen und schafft Vertrauen bei den Menschen, die die Apps nutzen sollen.

High-Performance-Computing

High-Performance-Computing oder Hochleistungsrechnen (HPC) ist ein Bereich des computergestützten Rechnens. Er umfasst alle Rechenarbeiten, die eine hohe Rechenleistung oder Speicherkapazität erfordern. HPC braucht Rechnerarchitekturen, die Daten und Operationen parallel verarbeiten. Damit Anwendungen diese Architekturen nutzen können, müssen sie parallel programmiert sein. Realisiert wird dies z. B. mit OpenMP (Open Multi-Processing) und MPI (Message Passing Interface).

Das DLR benötigt HPC für große numerische Anwendungen zur Lösung von CFD-Problemen (Computational Fluid Dynamics). Beispiele sind der TAU-Code des Instituts für Aerodynamik und Strömungstechnik oder der TRACE-Code des Instituts für Antriebstechnik. HPC ermöglicht in diesen Anwendungen kurze Simulationszeiten für die Design-Optimierung, z. B. von Flugzeugen. Heutige Arbeitsplatzrechner sind mit hochparallelen Architekturen ausgestattet, z. B. Mehrkern-Prozessoren mit Beschleunigungseinheiten. Daher werden auch hier HPC-Techniken eingesetzt, um diese Architekturen effizient zu nutzen. DLR-Mitarbeiter können dadurch auf Arbeitsplatzrechnern z. B. Hubschrauber simulieren und Bahnen von Weltraumrückständen berechnen.

Für HPC-Anwendungen im DLR und in externen Projekten liefern wir Algorithmen und Softwaretechnik. Wir entwickeln skalierbare Algorithmen und Datenstrukturen, passen Software für moderne Rechnerarchitekturen an und behandeln effizient große Datenmengen.

Algorithmen und Datenstrukturen

Nur wenige Algorithmen erlauben es, hochparallele Architekturen zu nutzen. Für das DLR wichtige Algorithmen sind z. B. Verfahren, um große Gleichungssysteme in CFD-Simulationen zu lösen. Für die effiziente Ausführung auf Parallelrechnern müssen diese Verfahren so asynchron wie möglich ablaufen und so wenig wie möglich zwischen parallelen Prozessen kommunizieren. Optimierungsalgorithmen sind für das DLR eine weitere wichtige Klasse von Verfahren. Anwendung finden sie z. B. in der Design-Optimierung von Flugzeugen oder Raumfahrzeugen. Um hohe Performanz auf modernen Rechnerarchitekturen zu erreichen, müssen klassische Algorithmen häufig deutlich verändert oder neu entwickelt werden.

Effiziente Algorithmen brauchen effiziente, verteilte Datenstrukturen. Beispiele sind kompakte Speicherschemata für dünnbesetzte Matrizen. Dünnbesetzte Matrizen treten in den meisten diskreten Verfahren auf und enthalten nur wenige Nicht-Null-Elemente. Die Datenstruktur muss ermöglichen, nur diese verteilt auf dem Parallelrechner zu speichern und effizient für parallele Rechenoperationen zu verwenden.

Wir erstellen hocheffiziente numerische Bibliotheken und Werkzeuge. Sie können in DLR-Anwendungen und externen Projekten aus Strömungstechnik, Materialphysik oder Thermalmanagement genutzt werden. Algorithmisch entwickeln wir parallele Gleichungs- und Eigenwertlöser, Optimierungsverfahren, spezielle Strömungslöser und Verfahren zur Geometriedarstellung. Um diese Verfahren performant auf modernen Rechnerarchitekturen ausführen zu können, entwerfen und implementieren wir effiziente Datenstrukturen.

Parallele Lösung großer Gleichungssysteme
Die großen strömungsmechanischen Codes des DLR sind TRACE für Innenströmungen im Triebwerk (Abbildung 14) und TAU für Außenströmungen. Sie erfordern die hochparallele Lösung großer Gleichungssysteme mit dünnbesetzter Matrix. Um diese Gleichungssysteme effizient auf Hochleistungsrechnern mit Mehrkernprozessoren zu lösen, müssen die Matrixdaten geeignet auf die Prozessoren des Hochleistungsrechners verteilt werden.

Im BMBF-Projekt „Hocheffiziente Implementierung von CFD-Codes für HPC-Many-Core-Architekturen“ (HICFD) entwickelten wir die parallele Bibliothekssoftware DSC-HPC, um die in TAU- oder TRACE-Simulationen auftretenden linearen Gleichungssysteme zu

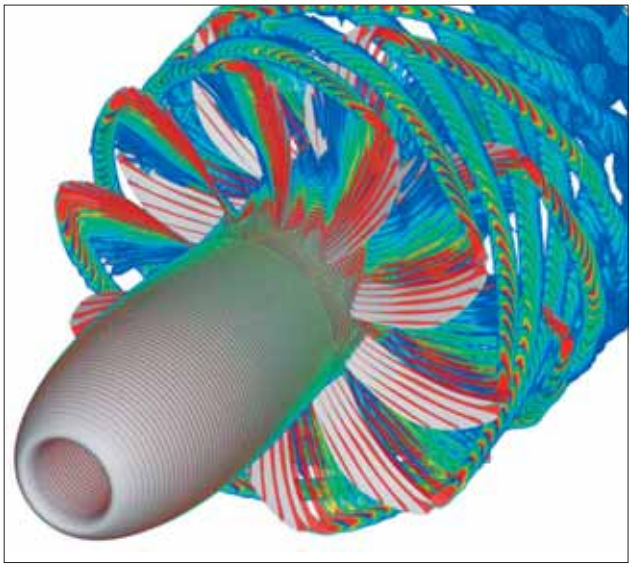


Abbildung 14: Ergebnis einer Simulation mit dem CFD-Code TRACE (Bild: Institut für Antriebstechnik).

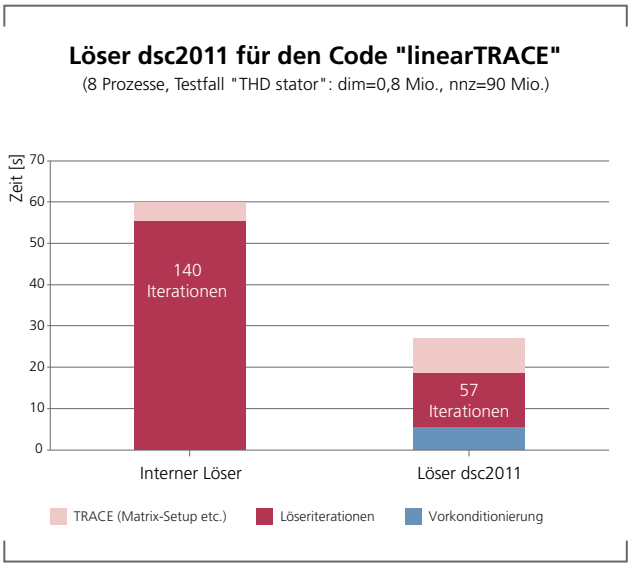


Abbildung 15: Ausführungszeiten von linearTRACE mit ursprünglichem Löser und DSC-Löser auf 8 Kernen eines Intel-XEON-E5520-Systems für ein typisches TRACE-Problem.

lösen. Zur optimierten Datenverteilung implementierten wir ein Block-Partitionierungswerkzeug basierend auf Graphalgorithmen. Unsere iterativen Gleichungslöser mit skalierbarer „Distributed Schur Complement"-Vorkonditionierung (DSC) führten zu einer deutlich schnelleren TRACE-Simulation (Abbildung 15).

Skalierbare Eigenwertlöser für dünnbesetzte Matrixprobleme

Graphen und topologische Isolatoren sind neue Materialien, die sich gut eignen, um innovative Computerchips zu entwickeln. Die Simulation dieser Materialien erfordert die numerische Lösung der Schrödinger-Gleichung. Diese beschreibt die zeitliche Entwicklung von Quantensystemen. Als aufwendigste Operationen treten dabei große Eigenwertprobleme mit dünnbesetzter Matrix auf.

Im DFG-Projekt „Equipping Sparse Solvers for Exascale" (ESSEX) entwickeln wir eine hocheffiziente Software-Bibliothek zur numerischen Lösung von großen dünnbesetzten Eigenwertproblemen auf heutigen und zukünftigen Höchstleistungsrechnern mit hunderttausenden oder Millionen von Rechenkernen. Neben Graphen-Design sind weitere Anwendungen für die entstehende Software denkbar, die für das DLR unmittelbar relevant sind: Materialphysik, Strukturmechanik und Stabilitätsanalyse von Strömungen.

Das Ziel unserer Entwicklungen ist, die Spitzenleistung eines Hochleistungsrechners möglichst auszunutzen. Wir nutzen dazu optimierte Block-Operationen für Multicore-CPUs und Grafikkarten. Kommunikation und Rechnung überlappen sich dabei. Dadurch werden Energie- und Zeitbedarf einer Simulation so gering wie möglich gehalten. Um diese Techniken anwenden zu können, haben wir Algorithmen umformuliert und mathematisch neu analysiert. Ziel ist die effiziente Nutzung der größten verfügbaren Rechner, wie etwa SuperMUC am Leibniz-Rechenzentrum München oder Piz Daint am Swiss National Supercomputing Centre in Lugano (Abbildung 16).

Während der Entwicklung setzten wir im Projekt ein systematisches Testkonzept um. Unser Continous-Integration-Server überprüft regelmäßig nicht nur den eigenen Anteil an der Software, sondern auch die Bausteine anderer Projektpartner. Eine durchdachte Software-Architektur sorgt dafür, dass die Bibliothek langfristig eingesetzt werden kann.

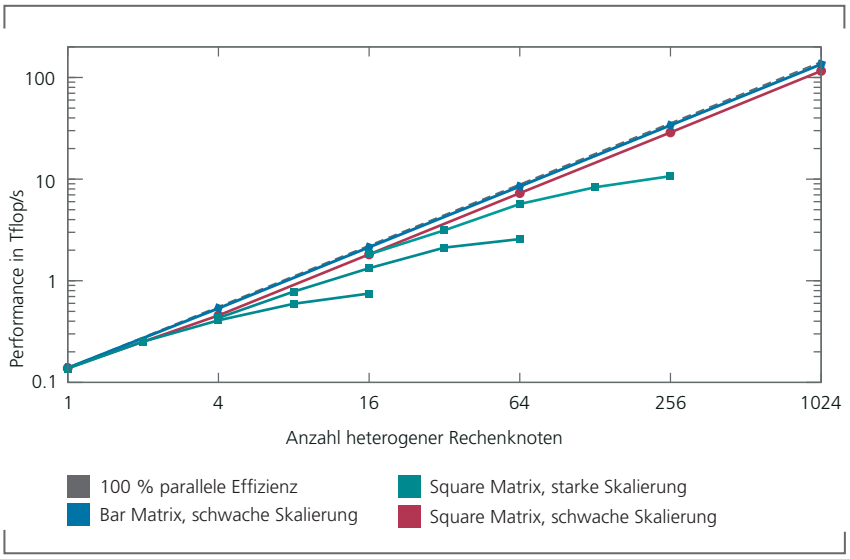


Abbildung 16: Skalierungstests auf Piz Daint, einem schweizer CPU-GPU-Cluster (aktuell Platz 6 der „Top 500"-Liste). Der Chebyshev-basierte Algorithmus kann zur Berechnung der Eigenwertdichte des Hamilton-Operators verwendet werden. Die entscheidende Grundoperation ist das Block-Matrix-Vektor-Produkt. Anwendung: topologische Isolatoren (Bild: DFG-Projekt ESSEX, Ernst-Moritz-Arndt-Universität Greifswald und Friedrich-Alexander-Universität Erlangen-Nürnberg).

Multidisziplinäre Optimierung

Multidisziplinäre Optimierungsverfahren werden im DLR in der Design-Optimierung von Flugzeugen oder Raumfahrzeugen eingesetzt.

Eine wichtige Anwendung sind neuartige Thermalschutzsysteme für den Wiedereintritt von Raumfahrzeugen in die Erdatmosphäre. Thermalschutzsysteme optimieren wir im DLR-Projekt THERMAS, das das Institut für Aerodynamik und Strömungstechnik in Köln leitet. Das Ziel der Optimierung ist, Gewicht zu reduzieren. Ein Simulationssystem für Thermalschutz muss viele Disziplinen berücksichtigen: Geometrie, Aerodynamik, Struktur, Subsystemmassen und Thermalmanagement. Da die Simulationsverfahren der Disziplinen in der Regel keine Ableitungen zur Verfügung stellen, müssen ableitungsfreie Optimierungsverfahren verwendet werden.

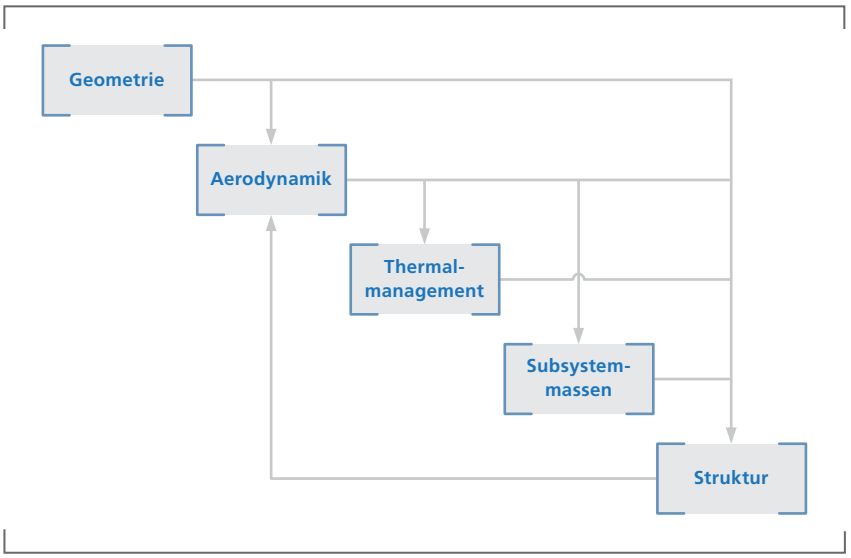


Abbildung 17: Am Gesamtsystem eines Raumfahrzeugs beteiligte Disziplinen.

Wir haben ein Optimierungswerkzeug für das Design von Thermalschutzsystemen entwickelt. Es verwendet Verfahren zur multidisziplinären Optimierung (MDO). Als Referenzkonfiguration zum Test der Verfahren wählten wir den DLR-SpaceLiner, das Konzept eines Hyperschall-Passagierflugzeugs. MDO löst Rückkopplungen zwischen zwei oder mehreren Disziplinen auf (Abbildung 17). Wir haben dazu den multidisziplinären Ansatz sIDF (sequential Individual Design Feasible, Abbildung 18) gewählt. Zur Lösung des MDO-Problems entwickelten wir einen effizienten Zwei-Schritt-SQP-Optimierer. Dieser erwies sich in punkto Zuverlässigkeit und Performanz als sehr konkurrenzfähig gegenüber bekannten Optimierer-Bibliotheken wie Dakota.

Softwaretechnisch setzten wir das Optimierungswerkzeug in unserer Integrationsumgebung RCE (siehe Abschnitt „Verteilte Integrationsumgebung RCE“) um. Die Ingenieur-Tools sind als miteinander verbundene Boxen dargestellt. Das Datenmanagement von RCE verschickt über diese Verbindungen einzelne Werte oder Dateien von einem Tool zum nächsten. Um den aktuellen Optimierungsstand ständig visualisieren zu können, integrierten wir unser Visualisierungstool TiGL-Viewer in RCE (siehe Abschnitt „Eine Software-Bibliothek zur Geometrie-Darstellung“ und Abbildung 5 im Abschnitt „Verteilte Integrationsumgebung RCE“).

Der Workflow des Optimierungswerkzeugs verwendet ein numerisches DLR-Simulationstool zur passiven Kühlung der Außenhaut mit neuartigen Faserverbundstoffen. Diese leiten die Wärme von heißeren zu kühleren Fahrzeugregionen. Das optimierte Design des SpaceLiners mit Transpirationskühlung spart gegenüber der Ausgangskonfiguration eine Tonne Gewicht ein (Abbildung 19).

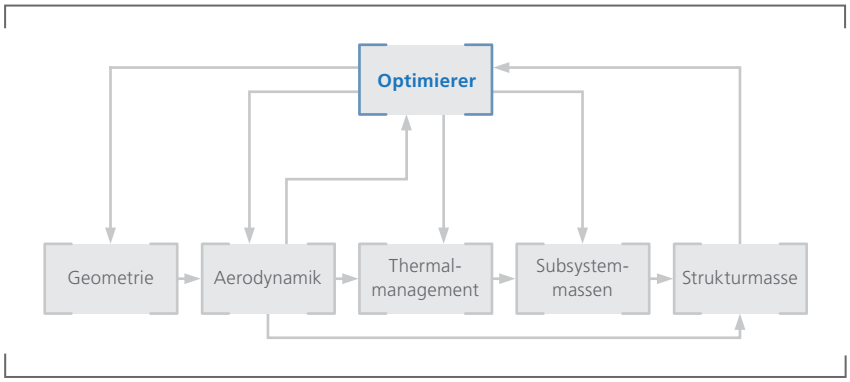
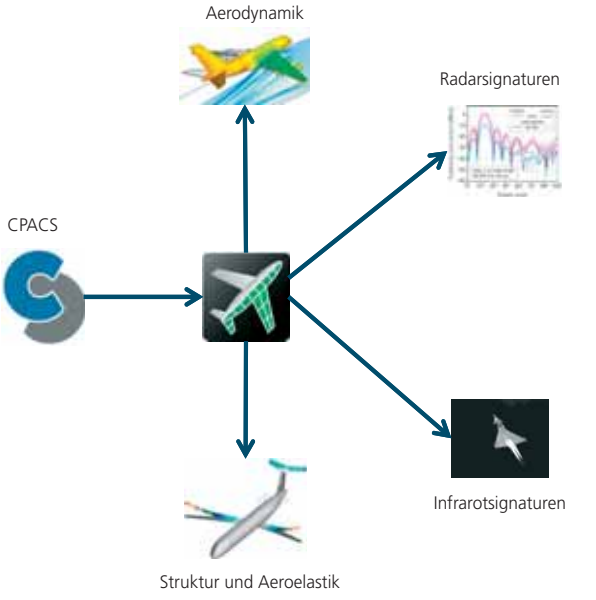


Abbildung 18: Schema des verwendeten multidisziplinären Ansatzes sIDF (sequential Individual Design Feasible).

Abbildung 19: Ausgangskonfiguration versus optimierte Konfiguration des SpaceLiners.

Abbildung 20: TiGL wird in verschiedenartigen Simulationstools verwendet.

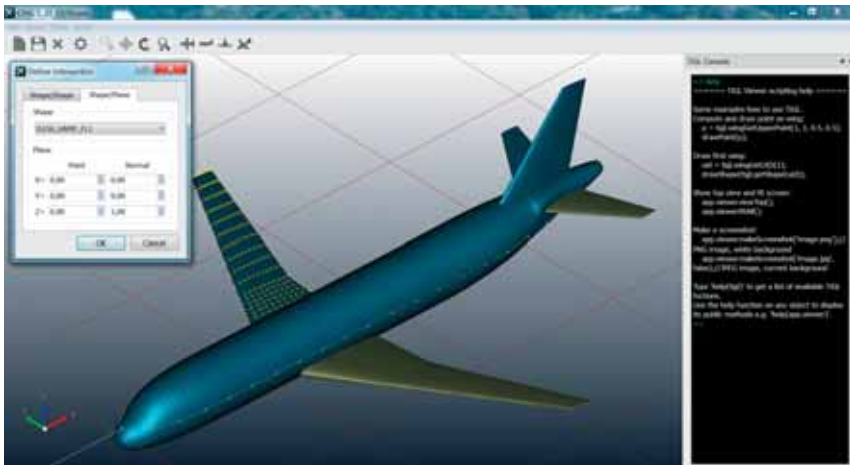


Eine Software-Bibliothek zur Geometrie-Darstellung
Geometrie-Bibliotheken erzeugen Objekte aus mathematischen Beschreibungen. Anwendungen findet man in der Spiele-Industrie oder auch im frühen Entwurf von Fahrzeugen.

TiGL (**Ti**GL **G**eometry **L**ibrary) ist eine Software, um dreidimensionale Geometrien von Flugzeugen zu erstellen. Grundlage von TiGL ist das CPACS-Datenformat, das die Form eines Flugzeugs parametrisch beschreibt. Viele DLR-Institute verwenden TiGL für Flugzeugdesign und Flugzeugvorentwurf in verschiedenartigen Simulationstools (Abbildung 20). Derzeit verwendet das DLR-Projekt FrEACs die TiGL-Bibliothek zur Bewertung von Flugzeugkonzepten. Die DLR-Projekte FaUSST und Mephisto verwenden TiGL, um unbemannte Flugzeuge zu entwickeln. Im DLR-Projekt Digital-X unterstützt TiGL automatisierte Netzgenerierung.

TiGL besteht aus einer Softwarebibliothek für mehrere Programmiersprachen und dem TiGL-Viewer, einem auf OpenGL basierten Programm zur Visualisierung der erstellten Geometrien (Abbildung 21).

Abbildung 21: TiGL-Viewer für Windows und als Android App.



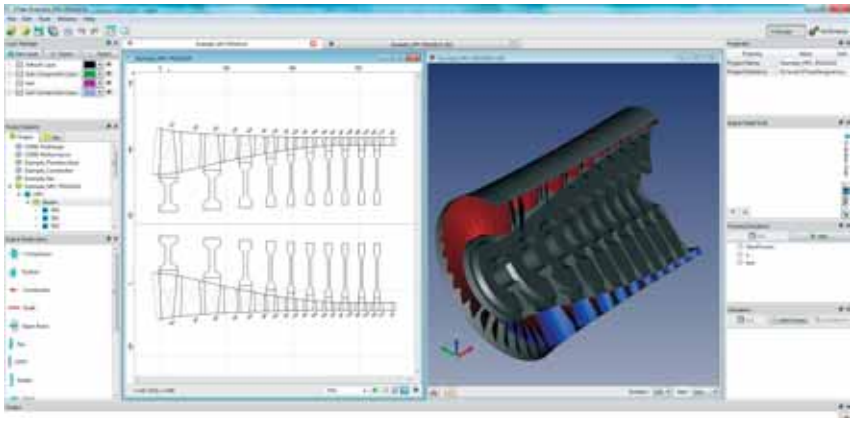


Abbildung 22: Design eines Hochdruckverdichters mit dem Tool GTlab (Institut für Antriebstechnik). Zu sehen ist die von uns integrierte 3D-Visualisierung.

Die TiGL-Bibliothek erlaubt es auch, die Geometrie für CAD-Programme oder Netzgeneratoren bereitzustellen, z. B. um Punkte auf der Flugzeugoberfläche zu berechnen, um Schnittlinien des Flugzeugs mit Ebenen zu berechnen oder um den maximalen Abflugwinkel beim Flugzeugstart zu bestimmen.

In aktuellen DLR-Projekten entwickeln wir TiGL für den Detailentwurf weiter und verwenden TiGL als Werkzeug für die automatisierte Netzgenerierung. Für höherwertige Geometriemodellierung haben wir TiGL bereits erweitert und die von TiGL exportierten Formate angepasst, damit sie in Netzgeneratoren verwendet werden können.

Diese Erfahrungen konnten wir auch für die Weiterentwicklung von GTlab nutzen. GTlab ist eine Anwendung für das Design von Flugzeugtriebwerken und Gasturbinen des Instituts für Antriebstechnik (Abbildung 22). Wir integrierten einen CAD-Kernel in GTlab und erreichten damit dreierlei:

- dreidimensionale Modellierung
- eine intuitive Visualisierung
- ein genaues Abschätzen physikalischer Eigenschaften jedes einzelnen Bauteils

Zukunftsthemen
Wir wollen unsere Eigenwertlöser-Bibliothek erweitern um extremen Parallelismus, um die Behandlung nicht-hermitescher Probleme und um effiziente Vorkonditionierer. Ferner werden wir hochparallele Multi-Level-Verfahren zur Lösung von Gleichungssystemen aus CFD-Anwendungen untersuchen. Ein weiteres Zukunftsthema ist die Entwicklung neuer hierarchischer Verfahren für die Wirbelverfolgung. Diese Verfahren können z. B. Hubschrauber-Simulationen deutlich beschleunigen.

Innovative Quantencomputer ermöglichen völlig neue Methoden in der mathematischen Optimierung. Wir planen Experimente zu neuen Optimierungsverfahren auf Quantencomputern in Zusammenarbeit mit der Firma D-Wave, die adiabatische Quantencomputer vertreibt. Simulator-Software von D-Wave steht uns bereits zur Verfügung. Zum Beispiel im Triebwerksdesign ließe sich die Designparameter-Optimierung auf Quantencomputern mit einer Simulation auf herkömmlichen HPC-Systemen verbinden.

Für TiGL planen wir, Modellierungsmöglichkeiten für Triebwerke in die Bibliothek zu integrieren. Ferner werden wir bei der TiGL-Entwicklung enger mit Airbus Defense and Space zusammenarbeiten. Wir wollen TiGL quelloffen weiterentwickeln und eine Open-Source-Community um TiGL aufbauen.

Parallelisierung für moderne Rechnerarchitekturen

Die Bandbreite moderner Rechnerarchitekturen ist weit. Sie reicht von extrem parallelen Systemen mit Hundertausenden von Rechenkernen, Beschleunigereinheiten wie GPUs (Graphics Processing Unit) und Intel Xeon Phi bis zu ersten Quantencomputer-Entwicklungen. Diese Systeme erfordern neue Software-Entwicklungskonzepte, z. B. beim Software-Test, bei der fortlaufenden Integration und der Performanz-Analyse. Um performante, wartbare Software für diese Systeme zu entwickeln, müssen geeignete Sprachkonstrukte, z. B. Direktiven, und Werkzeuge zur parallelen Programmierung eingesetzt werden.

In DLR-internen und -externen Projekten entwickeln wir innovative Software für moderne Rechnerarchitekturen mit modernen Software-Engineering-Methoden.

Software-Entwicklung für Exascale-Systeme
Künftige Exascale-Systeme erlauben die Ausführung von 10^{18} Flops (Floating Point Operations per Second). Sie verfügen voraussichtlich über einige Hundert Millionen Rechenkerne. Heterogene Rechenknoten aus Mehrkern-CPUs mit Beschleunigereinheiten wie z. B. GPUs oder Intel Xeon Phi sind durch ein schnelles Netzwerk verbunden. Derartige Systeme stellen Software-Entwickler vor große Herausforderungen, da sich nur wenige Algorithmen für extreme Parallelität eignen und Anwendungssoftware in der Regel inkrementell an neue Technologien angepasst wird. Exascale-Parallelität erfordert fundamentale algorithmische Änderungen und neue Konzepte für Ausfallsicherheit, Programmiermodelle und Frameworks zur Modellierung und Simulation. Um die Performanz eines Exascale-Systems nutzen zu können, muss die Software möglichst gut auf die Eigenschaften der heterogenen Rechenknoten abgestimmt sein. Zusätzlich muss die Software effiziente Kommunikation zwischen den Rechenknoten gewährleisten. Hierzu eignen sich hybride Programmierkonzepte, die unterschiedliche Parallelisierungsmethoden auf den Rechenknoten und zwischen den Rechenknoten verwenden.

Um die Performanz extrem paralleler Systeme nutzen zu können, verfolgen wir das hybride Programmierkonzept „MPI + X“ (Abbildung 23). Zwischen den Rechenknoten eines solchen Systems werden Nachrichten über MPI (Message Passing Interface) ausgetauscht. Innerhalb des Rechenknotens wird Parallelität über verschiedene Sprachkonstrukte genutzt: OpenMP (Open Multi-Processing), OpenACC (Open Accelerators), OpenCL (Open Computing Language) oder CUDA (Compute Unified Device Architecture). Sie eignen sich zur parallelen Programmierung von Mehrkern-CPUs oder Beschleunigereinheiten.

Ferner untersuchen wir die effiziente Nutzung von PGAS-Sprachen (Partitioned Global Address Space), um parallele Software zu entwickeln. Beispiele sind das Global-Arrays-Toolkit und Co-Array-Fortran. PGAS-Sprachen ermöglichen Lese- und Schreib-Operationen im verteilten Speicher eines Rechnersystems – ähnlich wie das Lesen und Beschreiben des lokalen Speichers eines Prozessors. Auch der neueste MPI-Standard unterstützt diese Operationen effizient.

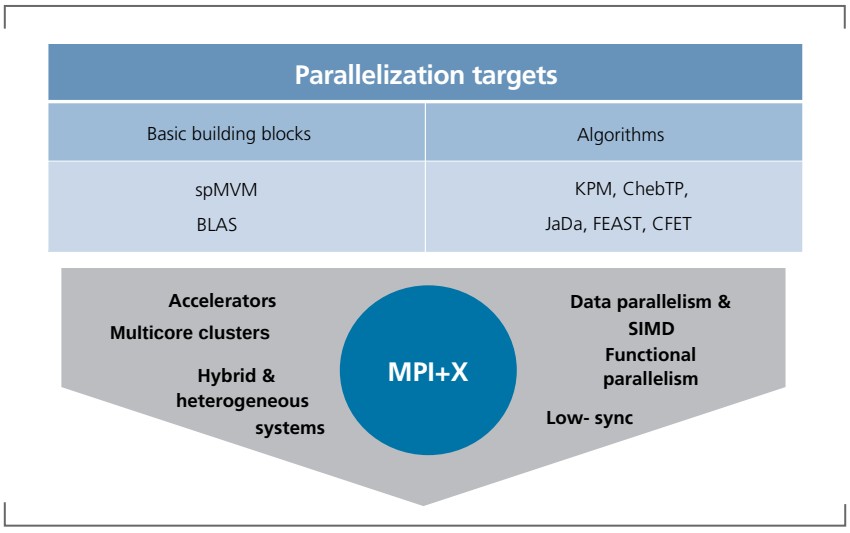


Abbildung 23: Software-Entwicklung mit dem Konzept „MPI + X“ im DFG-Projekt ESSEX (Bild: DFG-Projekt ESSEX, Friedrich-Alexander-Universität Erlangen-Nürnberg, RRZE).

Software zur gleichzeitigen Nutzung von Mehrkern-CPUs und GPUs
Um Systeme aus Mehrkern-CPUs und GPUs effizient zu programmieren, eignet sich das Konzept MPI+OpenACC. MPI wird verwendet, um Mehrkern-CPUs parallel zu programmieren, OpenACC zur gleichzeitigen Nutzung massiv-paralleler GPUs. OpenACC basiert wie OpenMP auf Direktiven und ist daher in bestehende Codes leicht zu integrieren.

Wir verwendeten MPI+OpenACC, um den Hubschrauber-Simulationscode Freewake des DLR-Instituts für Flugsystemtechnik auf GPUs zu portieren. Freewake simuliert die Strömung im Nachlauf eines Helikopter-Rotors (Abbildung 24). Der Code war bereits mit MPI parallelisiert.

Um möglichst kurze Rechenzeiten auf Workstations zu erzielen, verteilten wir die Rechnung von Freewake auf Mehrkern-CPU und GPU. Dies halbierte die Simulationszeit. Darüber hinaus erweiterten wir Freewake für die Simulation von Windkraftanlagen um eine benutzerfreundlichere Konfiguration und um eine neue Visualisierung der Ausgabedaten (Abbildung 25).

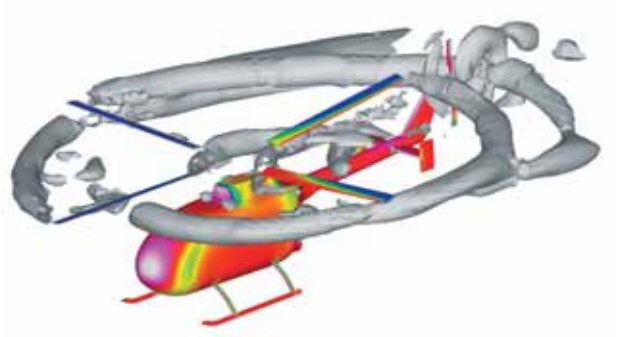


Abbildung 24: Grafische Darstellung der Strömung an den Rotoren eines Helikopters (Bild: Institut für Flugsystemtechnik).

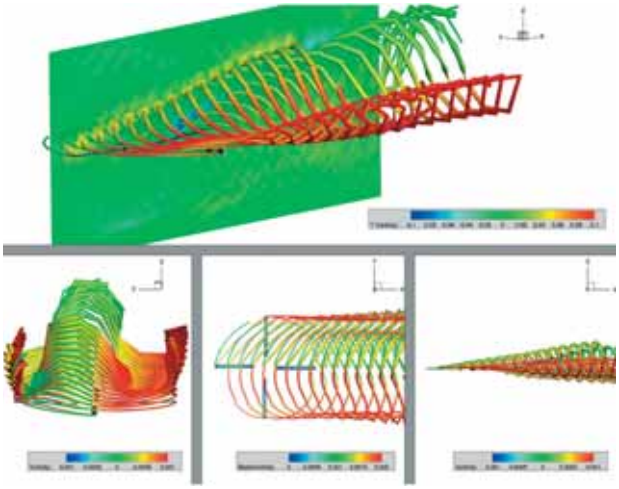


Abbildung 25: Visualisierung der mit Freewake berechneten Wirbel im Nachlauf eines Rotors.

Python-Software für Mehrkern-CPUs und GPUs
Die Programmiersprache Python wird zunehmend beliebter, um schnell kompakte Prototypen zu entwickeln. HPC-Anwendungen nutzen sie in der Regel nicht für numerische Berechnungen. Der Grund ist die geringere Performanz im Vergleich zu hardwarenahen Programmiersprachen wie C oder Fortran. Die Programmentwicklung mit hardwarenahen Programmiersprachen ist jedoch sehr zeitaufwendig.

Für Mehrkern-CPUs und GPUs untersuchten wir, ob Python eine einfache und effiziente Programmierung dieser Systeme ermöglicht. Dazu testeten wir die Performanz von Python-Implementierungen mit unterschiedlichen Zusatzbibliotheken und verglichen sie mit Implementierungen in C oder Fortran. Als Benchmark wählten wir einen Ausschnitt des Hubschrauber-Simulationscodes Freewake. Dieser Benchmark lag in optimierten Fortran-Versionen für die Ausführung auf Mehrkern-CPUs und GPUs vor. In Python implementierten wir mit unterschiedlichen Zusatzbibliotheken kompakte Versionen für diese Systeme.

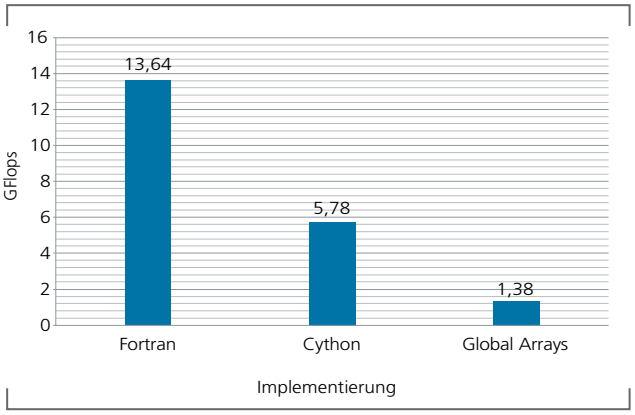


Abbildung 26: Multi-Core-Performanz-Tests auf Intel Xeon E5645 mit 6 Kernen.

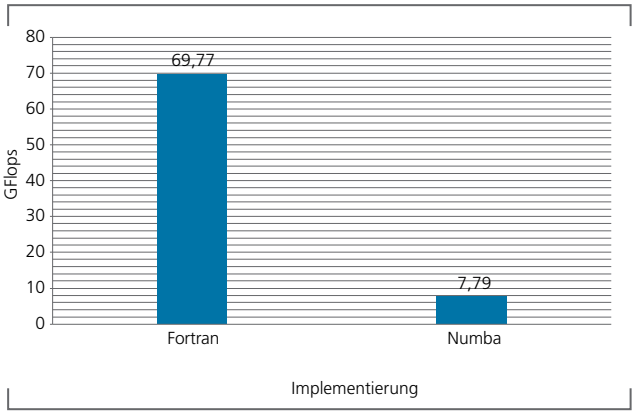


Abbildung 27: GPU-Performanz-Tests auf Nvidia Tesla C2075 mit 448 CUDA-Kernen.

Wir erreichten durch Implementierungen mit Python-Syntax auf Mehrkern-CPUs und GPUs ungefähr ein Zehntel der Performanz der Fortran-Implementierungen. Für Mehrkern-CPUs haben wir die Zusatzbibliothek „Global Arrays" und auf GPUs die Zusatzbibliothek „Numba" verwendet (Abbildungen 26 und 27). Eine höhere Performanz erreichte eine hybride Implementierung aus C und Python (Cython). Diese erzielte auf Multi-Core-Systemen ungefähr die Hälfte der Performanz von Fortran (Abbildung 26).

Die parallele Programmierung mit Python bietet auf Mehrkern-CPUs und GPUs einen klaren Produktivitätsvorteil gegenüber hardwarenahen Programmiersprachen; eine ähnliche Performanz wie bei hardwarenahen Programmiersprachen erreichen jedoch nur hybride Implementierungen mit C-ähnlicher Syntax.

Software-Test in hochparallelen Frameworks

Software-Tests in hochparallelen Frameworks sind ein aktuelles Forschungsthema. Die parallele Verarbeitung führt gegenüber der sequentiellen zu zusätzlichen Fehlerquellen wie Deadlocks, Race-Conditions und Kommunikationsfehlern. Software-Tests für parallele Programme müssen diese Fehler berücksichtigen und die Performanz des parallelen Codes prüfen.

Im BMBF-Projekt HPC-FliS entwickelten wir ein Testkonzept für ein hochparalleles Software-Framework. Das Framework ermöglicht die Lösung inverser Streuprobleme auf strukturierten Netzen. Um die Leistungsfähigkeit des Frameworks prüfen zu können, stellte die Siemens AG Daten aus Experimenten zur Verfügung. Wir verglichen Simulationsergebnisse des Frameworks mit Versuchsdaten aus einer Radialspaltmessung an einer Gasturbine und aus einer zerstörungsfreien Materialprüfung mit Ultraschall (Abbildung 28). Wir automatisierten Unit-Tests, Integrationstests und Systemtests des hochparallelen Frameworks mit dem Ziel der fortlaufenden Integration und Qualitätssicherung. Die Tests überprüften nicht nur die Funktionalität des Frameworks, sondern auch dessen Performanz. Unser Testbericht bestätigte das Konzept des hochparallelen Frameworks. Er ist Voraussetzung, um das Framework zertifizieren zu lassen.

Abbildung 28: Konzept der zerstörungsfreien Materialprüfung mit Ultraschall (Quelle: BMBF-Projekt HPC-FLiS, SIEMENS AG, Corporate Technology).

Zukunftsthemen
Um parallele Software effizienter entwickeln und warten zu können, werden wir HPC-Werkzeuge für Performanz-Analyse, -Vorhersage und paralleles Debugging evaluieren. Geeignete Werkzeuge werden wir in unseren Software-Engineering-Prozess integrieren. Um die Expertise in Entwicklungsmethoden für parallele Software im gesamten DLR zu erhöhen, wollen wir Schulungen zum Thema „Parallele Programmierung mit Einsatz von HPC-Werkzeugen" anbieten. Sie sollen die bestehenden Schulungen der Software-Engineering-Gruppe ergänzen. In der Community der „European Exascale Software Initiative" planen wir, Konzepte zur statischen Programmanalyse und zum Software-Test für extrem parallele Systeme weiterzuentwickeln.

Darüber hinaus planen wir, Programmierkonzepte für Quantencomputer zu untersuchen. Quantencomputer sind erheblich schneller als klassische Computer, z. B. in der Datenbanksuche mit dem Grover-Algorithmus oder in der Faktorisierung großer Zahlen mit dem Shor-Algorithmus. Der kanadischen Firma D-Wave ist es gelungen, einen adiabatischen Quantencomputer zu bauen, der unter anderem von NASA Ames und Google zu Forschungszwecken genutzt wird. Diese Maschine ist hauptsächlich für die Lösung von Optimierungsproblemen ausgelegt. Mit einer Simulator-Software für D-Wave-Quantencomputer werden wir mit der Formulierung von Optimierungsproblemen experimentieren. Wir wollen evaluieren, inwieweit das DLR von Quantencomputern profitieren kann.

Effiziente Behandlung großer Datenmengen

Simulationen auf HPC-Systemen erfordern häufig große Mengen an Eingabedaten und erzeugen große Mengen an Ausgabedaten. Der Schlüssel zu einer guten Lastbalance ist eine günstige Verteilung der Eingabedaten auf die Rechenknoten eines HPC-Systems durch geeignete Partitionierungsmethoden. Die verteilte Analyse der Ausgabedaten erfordert eine andere Datenpartitionierung als die Simulation – zum Beispiel, weil sich die Analyse nur auf einen Teil des simulierten Modells erstreckt. Um in diesem Fall Lastbalance zu erreichen, müssen die Daten für die Analyse entweder neu verteilt werden, oder die Ausgangsverteilung muss die Anforderungen von sowohl Simulation als auch Analyse berücksichtigen.

Die Kopplung verschiedener Simulationscodes erfordert einen effizienten Austausch großer Datenmengen zwischen den einzelnen Codes. Ein Beispiel aus dem DLR ist die Kopplung von CFD-, Strukturmechanik- und Flugmechanik-Codes.

Daten aus Simulationen, Experimenten und von Sensoren werden häufig in Datenbanken verwaltet. Sie müssen sowohl einen effizienten Zugriff auf die Daten und ihre Historie ermöglichen als auch eine effiziente Analyse der Daten. Ein Beispiel aus dem DLR ist ein verteiltes Datenbanksystem mit Bahndaten von Weltraumrückständen. Diese werden verwendet, um z. B. Kollisionen mit Satelliten vorherzusagen.

In DLR-internen und -externen Projekten entwickeln wir Methoden und Werkzeuge zur Datenpartitionierung, zur effizienten Analyse großer Ergebnisdatenmengen aus Simulationen sowie zum performanten Datenmanagement in gekoppelten Simulationen und in Datenbanken.

Daten-Partitionierung für Exascale-Systeme
Wettervorhersage, Molekulardynamik, physiologische Modellierung, Fusionsenergie und klassische Strömungssimulation sind gesellschaftlich wichtige Themen. Sie erfordern die Simulation auf den größten verfügbaren Supercomputern. Neue Werkzeuge müssen entwickelt werden, um Simulationssoftware auf künftigen Exascale-Systemen effizient ausführen zu können und um Lastbalance zwischen Millionen paralleler Prozesse zu erreichen. Solche Werkzeuge müssen neuste Partitionierungsalgorithmen verwenden und verschiedene Simulationsphasen berücksichtigen, z. B. Rechnung und Visualisierung.

Im europäischen Projekt CRESTA (Collaborative Research into Exascale Systemware, Tools and Applications) entwickelten wir gemeinsam mit der Abteilung SRV Pre- und Post-Processing-Werkzeuge, um Exascale-Anwendungen und -Nutzer zu unterstützen. Wir erstellten Pre-Processing-Werkzeuge, während SRV für skalierbare Post-Processing-Lösungen verantwortlich war.

Um eine möglichst gute Lastverteilung durch Partitionierung großer Modelldaten zu erreichen, entwickelten wir die Pre-Processing-Schnittstelle „PPStee" zur Steuerung von Exascale-Simulationen (Abbildung 29a). PPStee unterstützt moderne Partitionierungsbibliotheken und berücksichtigt verschiedene Simulationsphasen. Als Beispiel-Anwendung haben wir im Projekt eine Aneurysma-Geome-

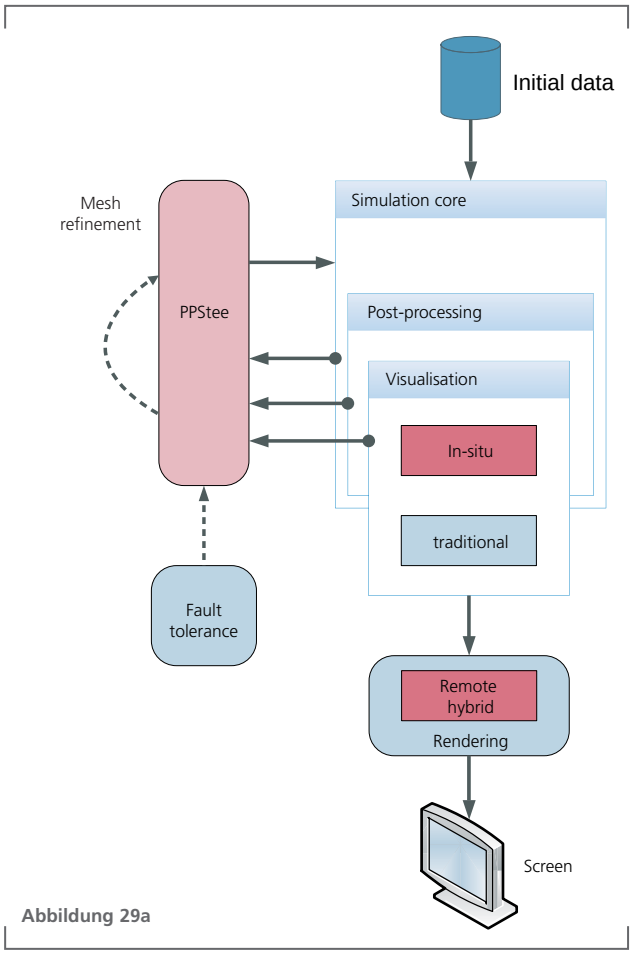


Abbildung 29a

Abbildung 29: Lastverteilungswerkzeug PPStee und mit PPStee partitionierte Aneurysma-Geometrie. (Quelle Aneurysma-Geometrie: EU-Projekt CRESTA, University College London).

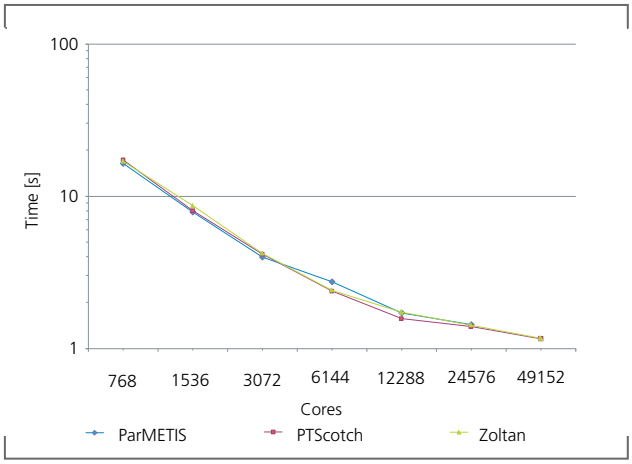


Abbildung 30: CFD-Simulationszeiten auf ARCHER unter Verwendung verschiedener PPStee-Partitionierer.

trie für eine hochparallele Blutfluss-Simulation mit PPStee partitioniert (Abbildung 29). Simulationen für diese großen Geometrien haben wir auf dem britischen Supercomputer ARCHER in Edinburgh unter Verwendung verschiedener PPStee-Partitionierer durchgeführt (Abbildung 30). Archer ist ein Cray XC30 Petascale-System mit 118.080 Rechenkernen.

Block-Splitting für blockstrukturierte Netze

Die Verteilung blockstrukturierter Netze erfordert spezielle Partitionierungsmethoden. Diese Netze bestehen in der Regel aus unterschiedlich großen Blöcken. Eine parallele Simulation, die auf einer Verteilung dieser Blöcke basiert, hat eine schlechte Lastbalance und ist daher häufig ineffizient. Um eine günstige Lastbalance zur erreichen, müssen die Blöcke zerteilt werden.

Um blockstrukturierte Netze des Strömungslösers TRACE automatisch zu partitionieren, entwickelten wir das Werkzeug Cujamara. Die Zerteilung der Blöcke muss geometrische Restriktionen einhalten, die durch TRACE vorgegeben sind. So dürfen Ein- und Austrittsebenen der Luft in die Geometrie nicht zerteilt werden.

Die Resultate der Partitionierung sind vielversprechend. Es werden Lastbalancen von mehr als 95 % erreicht. Wegen der Restriktionen durch TRACE verspricht weitere Partitionierung ab einem bestimmten Teilungsgrad keinen Vorteil mehr. In diesem Fall schlägt Cujamara vor, welche Prozesszahl für die Strömungssimulation sinnvoll ist. In Abbildung 31, rechts, ist die Partitionierung durch die Ein- und Austrittsebenen links und rechts begrenzt.

Abbildung 31: Partitionierung einer realistischen Triebwerksgeometrie mit Cujamara.

Datenmanagement für gekoppelte HPC-Simulationen

Um Simulationen aus dem Flugzeug-Design realitätsnäher zu machen, werden Aerodynamik und weitere physikalische Disziplinen wie Strukturmechanik oder Flugdynamik häufig gekoppelt simuliert. Auf einem HPC-System erfordert dieses Szenario einen effizienten Austausch großer Datenmengen zwischen den Simulationscodes. Da jede Einzelsimulation auf dem HPC-System in der Regel parallel abläuft, ist performantes, verteiltes Datenmanagement für die gesamte, gekoppelte Simulation unabdingbar.

Im DLR und bei Airbus unterstützt das FlowSimulator-Werkzeug parallele, gekoppelte Simulation (Abbildung 32). Der FlowSimulator ist eine Sammlung von Softwarepaketen für die numerische Simulation von Luftfahrzeugen. Diese Softwarepakete nutzen den FlowSimulator-Data-Manager (FSDM) als einheitliche Schnittstelle zum Austauschen der Daten. Die Bibliothek FSDM stellt Datenstrukturen sowie Methoden zur Datenverteilung und zum Datenaustausch zwischen verschiedenen Simulationswerkzeugen bereit (Abbildung 33). FSDM stellt Lastausgleich durch geeignete Daten-Partitionierung auch auf einer hohen Anzahl an parallelen Rechenknoten sicher.

Im DLR-Projekt Digital-X verbessern wir das Datenmanagement von FSDM durch neue Partitionierungsmethoden sowie Routinen zum Datenaustausch und integrieren neue Simulationskomponenten. Durch unsere umfangreichen Erfahrungen im Software-Engineering für HPC-Systeme sichern wir die Software-Qualität von FSDM.

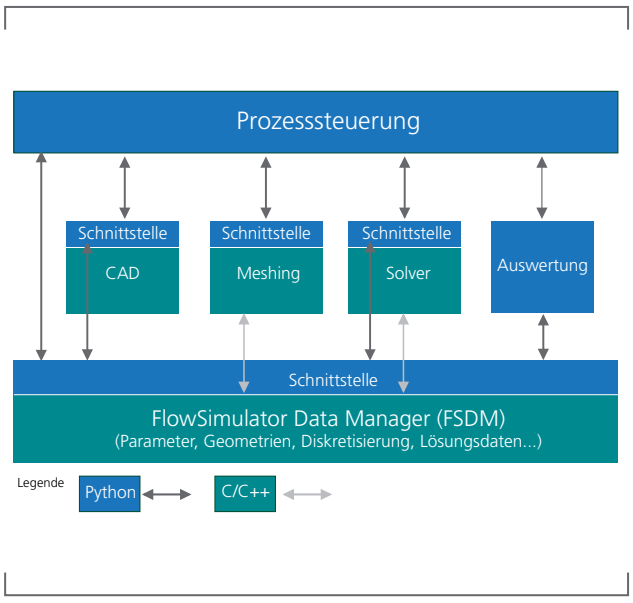


Abbildung 32: Systemarchitektur des FlowSimulators mit dem FlowSimulator-DataManager (FSDM) als grundlegende Bibliothek und Schnittstelle.

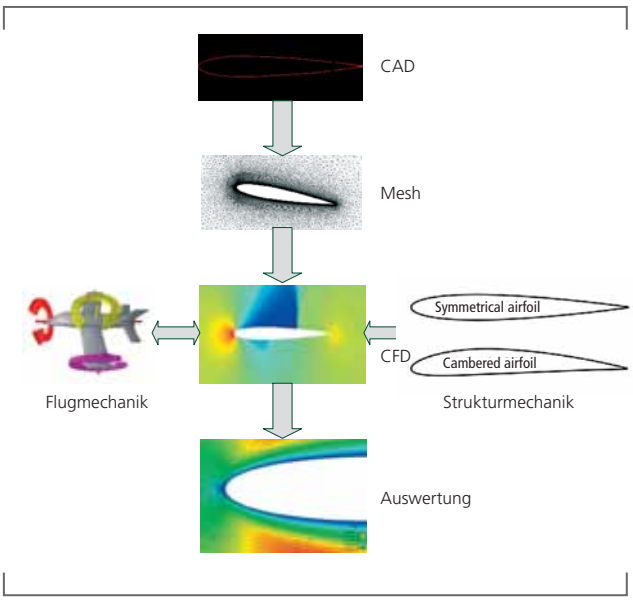


Abbildung 33: Beispiel für die Prozessierungskette einer gekoppelten CFD-Simulation.

Ein Datenbank-System für die Analyse von Weltraumrückständen
Die Anzahl an Weltraumrückständen in erdnahen Orbits ist hoch und steigt weiter (Abbildung 34). Selbst kleine Weltraumrückstände gefährden Raumfahrzeuge, Satelliten und Weltraum-Missionen (Abbildung 35). Um Kollisionen vorherzusagen, müssen Weltraumrückstände unbedingt erfasst und ihre Bahndaten in Datenbanken effizient verwaltet werden.

Zusammen mit der DLR-Einrichtung Raumflugbetrieb und Astronautentraining entwickeln wir den „Backend Catalog for Relational Debris Information" (BACARDI). Die BACARDI-Datenbank enthält Bahndaten von aktiven und inaktiven künstlichen Himmelskörpern wie Satelliten und Weltraumrückständen. Die BACARDI-Software unterstützt die Analyse dieser Bahndaten, um Kollisionen vorherzusagen. BACARDI bildet den Kern von SmartNET, einem Netzwerk, das zur Detektion von künstlichen Himmelskörpern auf Sensor-Daten von optischen, Radar- und Laser-Teleskopen zugreift.

Flugdynamik- und Weltraumlageexperten entwerfen und implementieren Algorithmen zur Auswertung der Sensor-Daten. Im DLR-Projekt „Optische Methoden für Space-Debris-Analysen" unterstützen wir sie dabei, diese Algorithmen auf Mehrkern- und GPU-Systemen zu implementieren. Ferner stellen wir ein Software-Rahmenwerk zur Verfügung, damit die Auswertungsprozesse die vorhandenen Ressourcen so effizient wie möglich nutzen. Dazu setzen wir auf moderne Techniken, um eine performante, skalierbare Architektur bereitzustellen. Mit dem Ausbau von SmartNET erwarten wir einen deutlichen Anstieg der auswertbaren Daten. Voraussichtlich werden wir ca. eine Million Objekte im BACARDI-System verwalten.

Abbildung 34: Künstlerische Darstellung der bekannten Objekte > 10cm. Objekte aus dem Space-Track.net-Katalog vom JSpOC (Bild: DLR).

Zukunftsthemen
Um parallele Simulationen noch effizienter zu machen, entwickeln wir unsere Werkzeuge für paralleles Datenmanagement weiter. Wir werden neue verteilte Datenstrukturen in den FlowSimulator integrieren, um spezielle Netz-Typen und -Formate behandeln zu können. Wir erweitern unser Cujamara-Werkzeug, um unstrukturierte Blöcke zerlegen zu können. Dies erfordert neue Block-Partitionierungsstrategien.

Für extrem parallele Systeme entwickeln wir neue skalierbare Methoden zur Datenanalyse und -visualisierung. Diese Methoden sollen z. B. die effiziente Analyse großer Datenmengen aus der Fernerkundung ermöglichen.

Ferner werden wir ein Werkzeug zur Datenanalyse und -visualisierung für experimentelle Daten und Simulationsdaten aus dem Triebwerksbau entwickeln. Dieses Werkzeug wird auf einer Datenbank basieren, in der Daten aus verteilten Experimenten und Simulationen gespeichert werden.

Unsere BACARDI-Software zur Analyse von Weltraumrückständen soll nach ihrer Weiterentwicklung ca. eine Million Objekte effizient verwalten und analysieren können. Ferner werden wir Schnittstellen zu weiteren Sensoren und Datenbanken integrieren. Neben dem Rahmenwerk werden wir eine intuitive Benutzeroberfläche entwickeln, um Kunden und Partnern die einfache Nutzung zu ermöglichen. Da wir die Kooperation mit Raumfahrtagenturen unterschiedlicher Länder anstreben, implementieren wir Authentifikations- und Autorisierungsmechanismen für den gesicherten Zugriff.

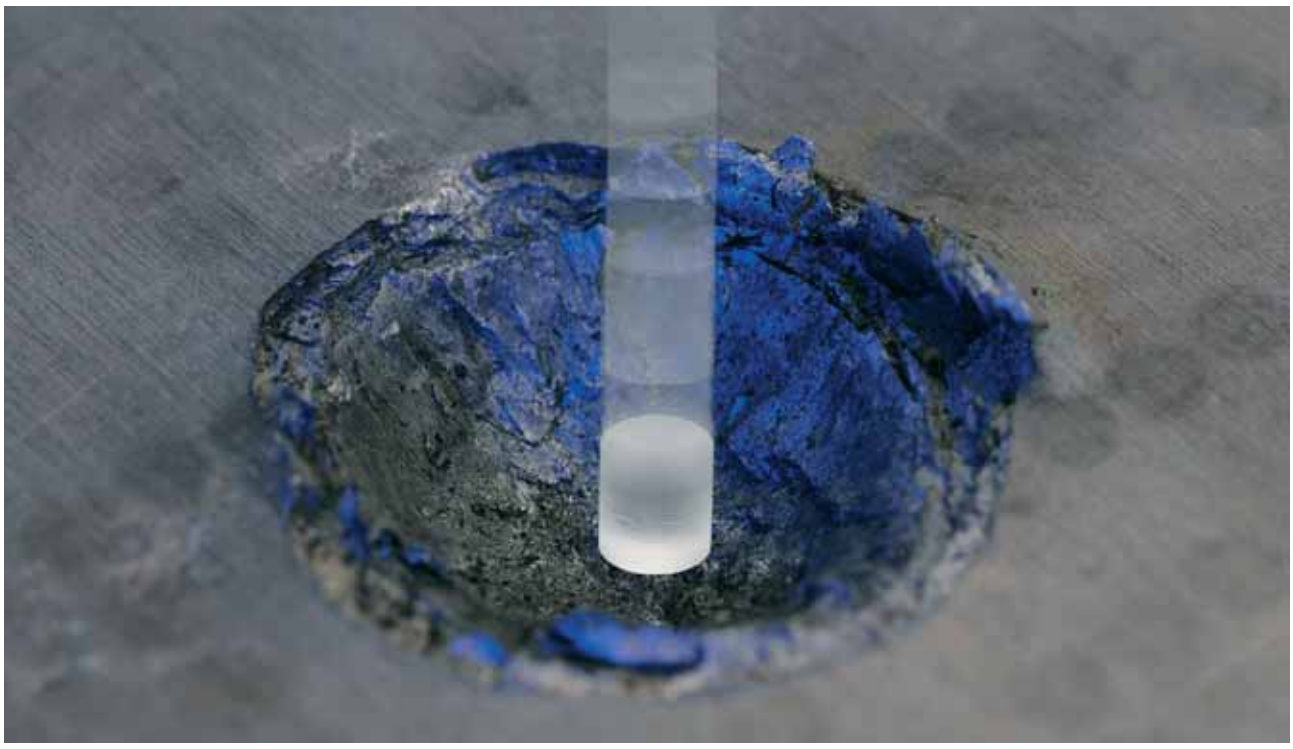


Abbildung 35: Wucht des Aufpralls von Weltraumschrott.

Software-Engineering

Etwa ein Drittel aller Mitarbeiter im DLR entwickelt Software für ingenieurwissenschaftliche Anwendungen oder ist daran beteiligt. Dies sind ca. 2.500 Menschen.

Die Bandbreite der Softwareprojekte ist groß. Typische Beispiele für Projekte:

- Eingebettete Softwaresysteme für Flugzeuge, Satelliten oder Raumfahrzeuge
- Unterstützungssoftware, beispielsweise zum Datenmanagement oder zur Prozessunterstützung
- Software mit hohen Effizienzanforderungen, beispielsweise HPC-Software zur Strömungssimulation

Diese Bandbreite birgt zahlreiche Herausforderungen für die Softwareentwicklung und das Software-Engineering im DLR. Neben der thematischen Vielfalt sind es insbesondere folgende Merkmale:

- **Projektumfang:** Vom Proof-Of-Concept bis zum Großprojekt ist alles vertreten.
- **Projektdauer:** Softwareprojekte im DLR können nur wenige Wochen, aber auch Jahrzehnte dauern.
- **Format:** Diplomarbeit, Softwareeinsatz im laufenden Betrieb oder Software als Gegenstand der Forschung. Jedes Format hat seinen eigenen Anspruch an die Entwickler.
- **Teamgröße:** Im DLR wird Software häufig durch „Einzelkämpfer" entwickelt. Teams von 50 Mitarbeitern sind aber auch vertreten.
- **Verteiltes Entwickeln:** Im DLR arbeiten Wissenschaftler aus verschiedenen Instituten und Einrichtungen zusammen. Diese befinden sich oft an unterschiedlichen Standorten.
- **Mitarbeiterfluktuation:** Praktikanten, Diplomanden und Doktoranden sind nur kurzfristig im DLR tätig. Mitarbeiter wechseln häufig in andere Institute oder verlassen das DLR.
- **Regularien:** Es kann erforderlich sein, extern vorgegebene Regularien und Standards einzuhalten (z. B.: Anwendung der European Cooperation for Space Standardization (ECSS) in Raumfahrtprojekten).

Während der Laufzeit eines Softwareprojekts verändern sich die Ausprägungen dieser Merkmale, da häufig neue Fragen in den Fokus rücken. So kann sich ein Proof-Of-Concept aus einer Diplomarbeit zu einem großen Softwareprojekt wandeln, das ein mehrköpfiges Team über Jahre vorantreibt. Es ist daher notwendig, den Software-Engineering-Prozess kontinuierlich zu reflektieren.

Ohne Techniken des Software-Engineering ist es nicht möglich, hochqualitative Software zu entwickeln. Daher ist Software-Engineering ein wichtiges Querschnittsthema im DLR und in unserer Einrichtung.

Wir haben einen hohen Wissens- und Erfahrungsstand, welcher kontinuierlich wächst – indem wir in anspruchsvollen

Definition: Software-Engineering
„Zielorientierte Bereitstellung und systematische Verwendung von Prinzipien, Methoden und Werkzeugen für die arbeitsteilige, ingenieurmäßige Entwicklung und Anwendung von umfangreichen Softwaresystemen." [Helmut Balzert: Lehrbuch der Software-Technik. Bd.1. Software-Entwicklung. Spektrum Akademischer Verlag, Heidelberg 1996, 1998, 2001, ISBN 3-8274-0480-0.]

Softwareprojekten mitarbeiten, neue Methoden und Werkzeuge evaluieren sowie aktuelle Software-Engineering-Schwerpunkte erforschen. Dieses Wissen fließt nicht nur in unsere eigenen Softwareprojekte ein, sondern wird auch ins DLR getragen.

Wir treiben das Thema Software-Engineering im DLR auf dreierlei Weise voran:

- Wir stellen Wissen zum Thema Software-Engineering bereit und vermitteln es. Das machen wir durch das Software-Engineering-Netzwerk des DLR, durch Schulungen sowie durch ein Wiki (SoftwareEngineering.Wiki).
- Wir beraten und unterstützen Institute und Projekte, indem wir Werkzeuge und Services bereitstellen sowie in Projekten mitarbeiten.
- Wir forschen in ausgewählten Gebieten und entwickeln anspruchsvolle Individualsoftware.

Im Folgenden stellen wir unsere Arbeiten in diesen drei Bereichen vor.

Software-Engineering-Wissen bereitstellen und vermitteln

Es ist sehr anspruchsvoll, komplexe und qualitativ hochwertige Software zu entwickeln, wenn sich die Randbedingungen kontinuierlich verändern. Fachkenntnisse aus dem Anwendungsbereich der Software und aus dem Bereich Software-Engineering sind unerlässlich, um Projekte erfolgreich durchzuführen. Den meisten Mitarbeitern des DLR fehlen Kenntnisse aus dem Software-Engineering-Bereich. Sie sind Physiker, Mediziner, Ingenieure oder Geologen und haben sich häufig ihre IT-Kenntnisse selbst angeeignet. Wir wollen diese Mitarbeiter des DLR unterstützen. Das übergeordnete Ziel ist, die Qualität und Effektivität der Softwareentwicklung im DLR nachhaltig zu verbessern.

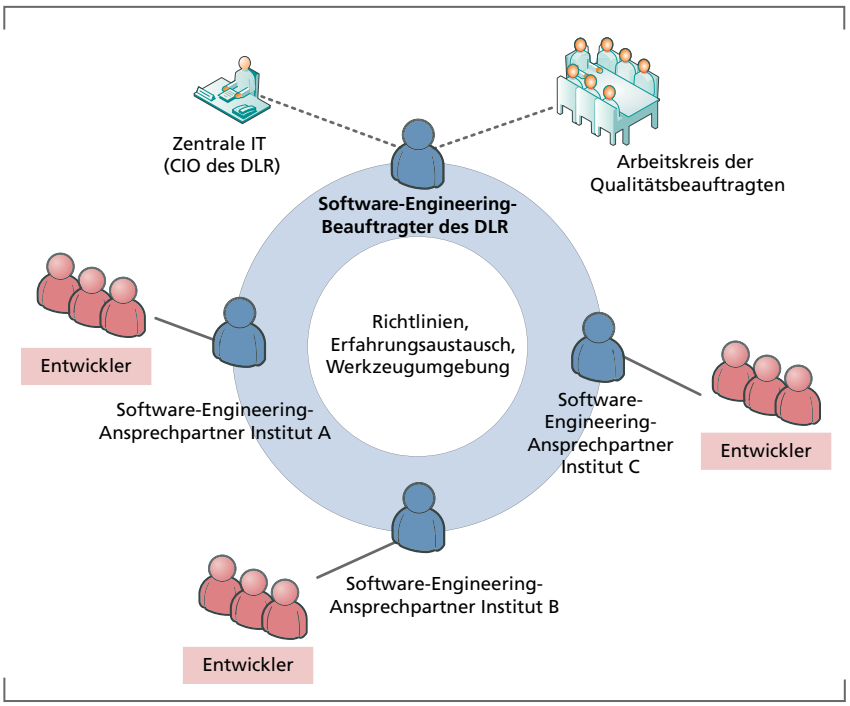


Abbildung 36: Aufbau des Software-Engineering-Netzwerks.

Wir möchten die Institute und Einrichtungen des DLR direkt beteiligen. Das Software-Engineering-Netzwerk bietet dafür einen direkten Kommunikationskanal und fördert den DLR-weiten Austausch. Zudem erlaubt es, konzeptionelle Hilfe durch uns zu erhalten. Als Informationsbereich betreiben wir das SoftwareEngineering.Wiki. Es ermöglicht, Software-Engineering-Wissen und –Erfahrungen aufzufinden und zu teilen. Schließlich unterstützen wir durch individuelle und praxisorientierte Schulungen die Institute bei der Einführung von Software-Engineering-Methoden und -Werkzeugen.

Das Software-Engineering-Netzwerk des DLR
Das Software-Engineering-Netzwerk ist das zentrale Austauschforum für die Institute und Einrichtungen des DLR zum Thema Software-Engineering. Es bündelt das Software-Engineering-Expertenwissen im DLR. Das Netzwerk verfolgt eine Kultur des Gebens und Nehmens: Zum einen kann ein Institut auf das Know-How des Netzwerks zurückgreifen. Zum anderen bringt es seine Erfahrung ein. Das Netzwerk erarbeitet Richtlinien für die Softwareentwicklung im DLR und treibt die Entwicklung der Werkzeugumgebung für DLR-Software-Entwickler voran. Wir werden das Software-Engineering-Netzwerk auch in Zukunft federführend unterstützen.

Wir haben das Software-Engineering-Netzwerk bereits Ende 2005 etabliert, unterstützt durch den Chief Information Officer (CIO) des DLR und in Zusammenarbeit mit der Einrichtung Qualitäts- und Produktsicherung. Das Netzwerk ist als offizielles DLR-Gremium durch die „Rahmenrichtlinie Software-Engineering“ legitimiert. Die Richtlinie legt zudem verbindliche Regeln zur Softwareentwicklung im DLR fest. Die Richtlinienkompetenz liegt beim CIO. Dieser delegiert die inhaltliche Weiterentwicklung der Richtlinie an SC; geknüpft an die Rolle des Software-Engineering-Beauftragten.

Das Software-Engineering-Netzwerk setzt sich aus den Vertretern der Software entwickelnden Institute und Einrichtungen zusammen, den so genannten Software-Engineering-Ansprechpartnern (Abbildung 36). Derzeit beteiligen sich schon über 30 Software-Engineering-Ansprechpartner aktiv im Netzwerk. Damit sind ca. 80 Prozent aller Institute und Einrichtungen des DLR vertreten. Dies zeigt den hohen Stellenwert des Themas im DLR.

Wir leiten das Netzwerk und stellen den Software-Engineering-Beauftragten. Zu seinen wesentlichen Aufgaben gehört, die Rahmenrichtlinie zusammen mit den Instituten zu pflegen und weiterzuentwickeln. Er nimmt Anforderungen der Institute und Einrichtungen auf, z. B. Werkzeug- oder Schulungsbedarf, und koordiniert deren Umsetzung. Weiterhin beruft er regelmäßig Treffen und Telefonkonferenzen ein.

Die wichtigste Aufgabe eines Software-Engineering-Ansprechpartners besteht darin, die Softwareentwicklung im jeweiligen Institut zu organisieren. Dafür muss er Vorgaben für das typische Vorgehen bei der Softwareentwicklung erarbeiten. Diese schreiben beispielsweise den Einsatz von Methoden und Werkzeugen vor. Schließlich unterstützt der Ansprechpartner die Mitarbeiter seines Instituts beim Umsetzen dieser Vorgaben. Die Anforderungen, die sich daraus ergeben, thematisieren und vertreten die Software-Engineering-Ansprechpartner im gesamten Netzwerk.

Das SoftwareEngineering.Wiki
Das SoftwareEngineering.Wiki ist die zentrale Anlaufstelle, um sich über Software-Engineering im DLR zu informieren. Alle Mitarbeiter des DLR können das Wiki bearbeiten und pflegen. Insbesondere nutzen wir das SoftwareEngineering.Wiki als zentrales Werkzeug, um Software-Engineering-Wissen im DLR bereitzustellen. So werden Schulungsthemen zusätzlich durch eine Informationsseite im Wiki aufbereitet. Wir moderieren das Wiki kontinuierlich, damit die Strukturen erhalten bleiben.

Das Wiki stellt grundlegendes Methodenwissen bereit und sammelt Erfahrungsberichte. Dadurch können sich Mitarbeiter einen kompakten Überblick zu Software-Engineering-Themen verschaffen. Im Wiki können sie auf einfache Weise Fragen stellen und mit erfahrenen DLR-Kollegen in Kontakt treten. Das SoftwareEngineering.Wiki fördert den Austausch zwischen Softwareentwicklern im DLR.

Derzeit bietet das Wiki Informationen zu folgenden Themen:

- **Frage-Antwort-Bereich:** Dort können konkrete Fragen zu verschiedenen Themen der Softwareentwicklung gestellt werden.
- **Software-Engineering-Themen:** Dieser Bereich liefert einen kompakten Überblick zu Themen wie Anforderungsmanagement, Software-Konfigurationsmanagement oder Software-Tests.

- **Best Practices:** Erfahrungsberichte und konkrete How-To-Berichte werden dort geteilt. Zudem finden sich umfangreiche Empfehlungen für verschiedene Programmiersprachen.
- **Projekthandbuch:** Dieser Bereich liefert praktische Empfehlungen, wie die Entwicklung von wissenschaftlicher Software zu organisieren ist. Dazu zählen die verschiedenen Phasen des Software-Lebenszyklus sowie der Werkzeugeinsatz.
- **Veranstaltungen:** Ankündigungen wichtiger DLR-interner Software-Engineering-Veranstaltungen werden dort veröffentlicht.

Zudem verfügt das Wiki über einen Blog. Er ist der zentrale Kommunikationskanal für Ankündigungen jeder Art, Hinweise zu Veranstaltungen, spannende Technologien oder neue Inhalte im Wiki. Mitarbeiter können den Blog abonnieren und sich über aktuelle Neuigkeiten informieren lassen.

Schulungen zu Software-Engineering-Themen
Wir bieten praxisorientierte und individuelle Schulungen und Seminare zu verschiedenen Software-Engineering-Themen an. Derzeit gibt es Veranstaltungen zu Anforderungsermittlung, Software-Architektur, Software-Konfigurationsmanagement, Software-Test sowie Open-Source-Recht. Dadurch ergänzen wir das Weiterbildungsangebot des DLR. Den konkreten Bedarf ermitteln wir zusammen mit den Software-Engineering-Ansprechpartnern im Software-Engineering-Netzwerk. Zudem trägt die Abteilung Personal- und Organisationsentwicklung konkrete Anfragen aus den Instituten und Einrichtungen an uns heran. Da es ständig neue Anforderungen und neue Technologien gibt, passen wir den Inhalt der Schulungen entsprechend an oder entwickeln zusätzliche Schulungen. Hervorzuheben ist die Software-Engineering-Grundlagenschulung. Diese findet jährlich an den Standorten Braunschweig, Köln und Oberpfaffenhofen statt und ist Teil des offiziellen DLR-Bildungsprogramms (Abbildung 37). In dem zweitätigen Workshop erleben die Teilnehmer, wie ein professionell organisiertes Softwareprojekt abläuft – von der Initiierung bis zur Veröffentlichung der ersten Produktivversion. Sie lernen den Entwicklungsprozess strukturiert, nachvollziehbar und praxisnah zu gestalten sowie mit gezieltem Werkzeugeinsatz zu unterstützen. Insbesondere bietet der Workshop viel Raum für individuelle Fragen und praktische Übungen mit den am DLR vorhandenen Werkzeugen.

Abbildung 37: Software-Engineering-Grundlagenschulung im DLR-Standort Köln-Porz.

Unterstützung beim Software-Engineering

Aufgrund der Vielzahl und Bandbreite unserer eigenen Softwareprojekte verfügen wir über umfassende Erfahrungen, um Entwicklungsprozesse samt Werkzeugunterstützung in Forschungsprojekten zu etablieren. Zudem erproben wir innovative Software-Engineering-Werkzeuge und machen sie für das DLR zugänglich.

Mitarbeit von SC in DLR-Projekten
In Softwareprojekten ist es wichtig, frühzeitig die Regeln der Zusammenarbeit zwischen allen Beteiligten festzulegen und im Verlauf zu optimieren. Diese Regeln manifestieren sich im Entwicklungsprozess. Beispielsweise legt der Entwicklungsprozess fest, welche Schritte erforderlich sind, um eine neue Funktionalität in die Software zu integrieren. Softwareprojekte sind oft komplex. Damit sie handhabbar bleiben, ist es wichtig, professionelle Softwareentwicklungswerkzeuge zu nutzen und wiederkehrende Aufgaben so weit wie möglich zu automatisieren. Weiterhin müssen alle Beteiligten die Abläufe und eingesetzten Werkzeuge beherrschen. Mit unseren Erfahrungen helfen wir regelmäßig, Entwicklungsprozesse in Projekten kontinuierlich zu optimieren.

Bei einer Beratung geht es vor allem darum, Prozesse oder grundlegende Software-Engineering-Werkzeuge in einem DLR-Institut einzuführen – zum Beispiel ein Versionsverwaltungssystem inklusive entsprechendem Arbeitsprozess aufzusetzen, die Werkzeugumgebung zu erweitern oder Prozesse unter Berücksichtigung von Normen wie ECSS zu optimieren. Oft laufen diese Beratungen über mehrere Monate vor Ort in dem jeweiligen Institut. Als Beispiele sind die Beratung des Instituts für Technische Thermodynamik (2012) und des Raumfahrtkontrollzentrums (2014) zu nennen.

Wenn wir in Projekten direkt mitarbeiten, übernehmen wir als Projektpartner neben den Software-Engineering-Aufgaben auch konzeptionelle und Entwicklungstätigkeiten. Dies ist zum Beispiel in den Verkehrsprojekten VABENE, MOBiNET und Instant Mobility, dem Luftfahrtprojekt Digital-X oder dem Raumfahrtprojekt ADAM der Fall. Diese Projekte profitieren durch unsere Kernkompetenz im Bereich Software-Engineering: die Definition und Umsetzung eines Entwicklungsprozesses inklusive der verbindlichen Regelungen und Werkzeugunterstützung. Zusätzlich übernehmen wir in diesen Projekten weitere Aufgaben:

- wir konzipieren die Software (MOBiNET, Instant Mobility);
- wir entwickeln Teile der Software selbst, zum Beispiel die graphische Benutzerschnittstelle (ADAM);
- wir bringen spezielles Wissen ein, wie zum Beispiel zu verteilten Systemen (MOBiNET, Instant Mobility).

Solche Aufgaben verankern uns stärker im Projekt und erleichtern es, den Entwicklungsprozess zu etablieren.

Bereitstellen von Software-Engineering-Werkzeugen und Services im DLR
Auf Basis unserer Erfahrungen aus der Projektarbeit erproben wir innovative Software-Engineering-Werkzeuge und machen sie für das DLR nutzbar. Dies erfolgt in Projekten, aber auch in Form eigenständiger Vorhaben. Die Ergebnisse bringen wir in das Software-Engineering-Netzwerk ein und gestalten somit aktiv die Werkzeugumgebung für DLR-Entwickler mit. Zudem unterstützen wir konzeptionell bei der Überführung von Werkzeugen in den Produktivbetrieb. Wir sind somit die inhaltliche Schnittstelle zur zentralen Organisationseinheit Informations- und Kommunikationstechnik (IT) und dem IT-Dienstleister des DLR. Dies ist auch zukünftig eine wesentliche Aufgabe. Hierzu einige aktuelle Beispiele:

- Subversion ist das Standard-Versionskontrollsystem im DLR. Git ist ein weiteres Versionskontrollsystem, das sich im Open-Source-Umfeld zunehmend durchsetzt. Wir evaluieren derzeit verschiedene Git-Hosting-Plattformen, um zukünftig Git als Alternative zu etablieren.
- Im DLR steht das Werkzeug Mantis zur Verfügung, um Anforderungen und Fehlerberichte zu verwalten. Seine Benutzerführung ist nicht intuitiv, und es lässt sich nicht gut an eigene Projektgegebenheiten anpassen. Daher soll es abgelöst werden. Wir konzipieren derzeit in Abstimmung mit dem Software-Engineering-Netzwerk und IT einen professionellen Ersatz.
- Das Prinzip der „kontinuierlichen Integration“ (Continuous Integration) hilft, frühzeitig Integrationsfehler bei der Softwareentwicklung zu finden. Dieser Ansatz hat sich in der Softwareentwicklungs-Community durchgesetzt und wird auch bei uns angewandt. Dazu wird ein Integrations-Server benötigt. Dieser prüft regelmäßig die aktuellen Änderungen in der Gesamtheit, stellt die Ergebnisse übersichtlich dar und informiert das Projektteam. Wir arbeiten daran, einen Integrations-Server im DLR zu etablieren. Bei Bedarf soll er in DLR-Projekten über den IT-Dienstleister vorkonfiguriert bereitstehen.

- Wir entwickeln ein Werkzeug, das die Zugriffsrechteverwaltung für den DLR-Subversion-Service stark vereinfacht. Zuvor pflegte ausschließlich der Subversion-Administrator die Zugriffsrechte. Mittlerweile können die jeweiligen Projektverantwortlichen selbstständig mit dem DLR-Standardwerkzeug (CoMet) Zugriffsrechte festlegen. Das Werkzeug stellt die Synchronisation der Systeme sicher.

Neben diesen Werkzeugen betreiben wir ein Software-Testlabor für interne und externe Projekte. Der Schwerpunkt liegt dabei auf den Phasen System- und Abnahmetest. Dadurch können wir im Labor auch Software testen, an deren Entwicklung wir nicht beteiligt waren. In der Praxis führen wir Softwaretests aber eher im Rahmen einer direkten Institutsberatung oder eines Projekts durch.

Erforschung und Entwicklung von Software zu DLR relevanten IT-Themen

Wir erforschen für das DLR relevante Bereiche der Informatik, z. B. Wissensmanagement, Visualisierung von Daten und Provenance von Prozessen. Zu diesen Themen entwickeln wir auch spezielle Software für das DLR.

Wissensmanagement
Wissensmanagement in Projekten bedeutet, vorhandenes Wissen – also Informationen, Daten und Fähigkeiten – zu organisieren und zugreifbar zu machen. Zu den Aufgaben des Wissensmanagements gehört, Dokumente abzulegen und zu suchen, sowie Informationen und Wissen effizient zwischen den Projektbeteiligten auszutauschen. Wissensportale dienen dabei als zentrale Plattform im Wissensmanagement und können als eine Art Wegweiser durch die Informationen eines Unternehmens angesehen werden.

Stand der Technik ist, vorhandene und neu entstehende Dokumente semantisch zu beschreiben, in einem Datenmanagementsystem abzulegen und durchsuchbar zu machen. Die Dokumente und Daten werden klassifiziert und mit Metadaten versehen. Weiterhin können externe Datenquellen angebunden werden, zum Beispiel Literaturdatenbanken (DLR elib u.ä.). Sämtliche Daten werden indiziert, um eine Volltextsuche nach Inhalten zu ermöglichen. Für die Suche werden effiziente, moderne Bibliotheken eingesetzt, zum Beispiel Apache Solr oder ElasticSearch. Entscheidend ist heutzutage eine flexible Filterung der Suchergebnisse, zum Beispiel anhand von Klassifikationen, Herkunftsquellen oder Datenformaten.

Wir entwickeln hierfür zwei Werkzeuge: DataFinder und KnowledgeFinder. DataFinder reichert Daten mit Metadaten an und legt sie strukturiert ab. KnowledgeFinder dient dazu, Daten effizient wiederaufzufinden und zu erforschen (Abbildung 38).

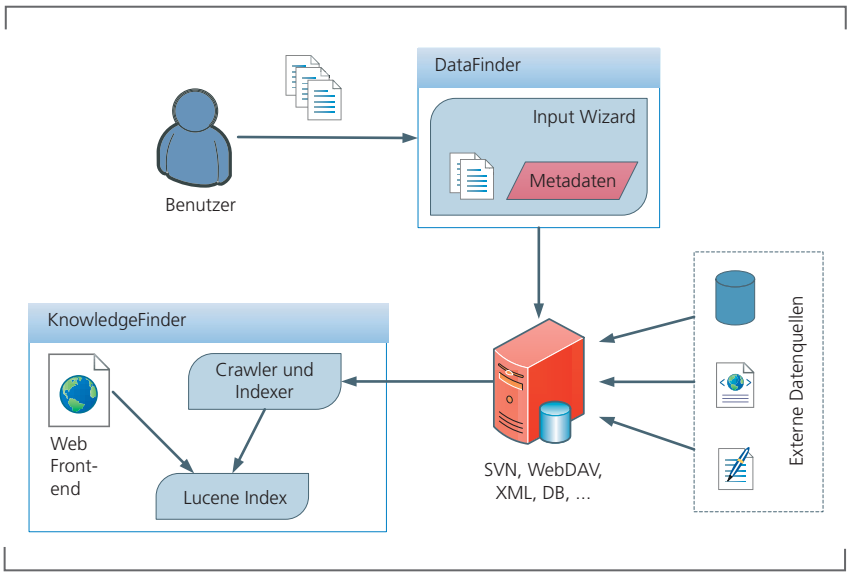


Abbildung 38: Dokumentenworkflow im Wissensmanagement mit DataFinder und KnowledgeFinder.

Das Institut für Flughafenwesen und Luftverkehr (FW) nutzt in dem Projekt MonitorPortal beide Werkzeuge, um Verkehrsdaten zu verwalten und anzubieten. KnowledgeFinder kommt außerdem im Projekt STRADA zum Einsatz. STRADA möchte die existierenden Suchportale der Clearingstelle „Clearing House of Transport and Mobility“ und von STRADA aggregieren, um eine zentrale Anlaufstelle für alle Daten rund um das Thema Verkehr bereitzustellen. Zukünftig werden weitere Datenquellen des DLR sowie externer Partner eingebunden. Basierend auf KnowledgeFinder möchten wir weitere Wissensportale für andere Forschungsbereiche aufbauen.

Visualisierung von Daten
Daten zu visualisieren ist eine komplexe Aufgabe. Sie reicht von der Darstellung von Messpunkten in einem Graphen bis zur 3D-Darstellung von Orbiteraufnahmen, die einen Flug über den Mars erlaubt.

Wir beschäftigen uns in beiden Abteilungen mit verschiedenen Aspekten dieses Themas. In diesem Abschnitt geht es um den Teilbereich der Data Exploration. Unter Data Exploration versteht man, Daten aus verschiedenen Quellen und in verschiedenen Formaten zu analysieren und zu erforschen.

Im Projekt STRADA liegt genau diese Situation vor. STRADA kombiniert die Dokumente der Clearingstelle und des Instituts FW in einem Portal. Dort können Nutzer die Dokumente mithilfe von Filtern und einer Volltextsuche durchsuchen. Kennt der Wissenschaftler die Art, die Struktur und den Umfang der zugrunde liegenden Dokumente, reichen diese Suchmöglichkeiten aus. Arbeitet er sich aber in ein neues Thema ein oder sind ihm die Dokumente vollkommen unbekannt, hilft es, die Dokumentstruktur visuell darzustellen (Abbildung 39). So lässt sich auf einen Blick erfassen, welche Themen, Regionen oder Zeiträume in den Dokumenten verknüpft sind oder welche Dokumente sich ähnlich sind. Der Wissenschaftler kann seine Frage auf Basis der erhaltenen Ergebnisse anpassen, indem er Filter setzt und den Graphen dadurch verändert. Im nächsten Schritt sollen Benutzerstudien helfen, die Visualisierung zu optimieren.

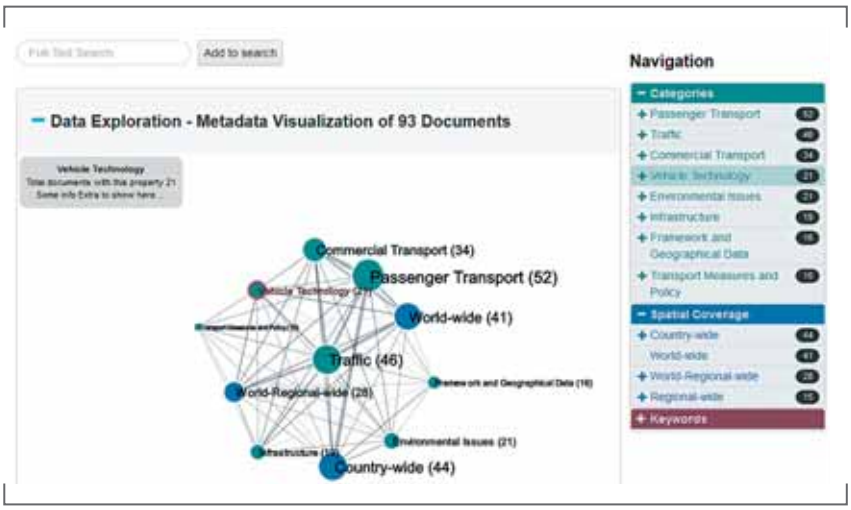


Abbildung 39: Graph aus der Visualisierung von STRADA.

Provenance von Prozessen
Provenance-Daten beschreiben die Menschen, Einrichtungen, Informationen und Aktivitäten, welche Daten oder Dinge erzeugen. Mit Provenance-Daten werden Prozesse nachvollzogen oder deren Einhalten überprüft (Compliance). Wir beschäftigten uns seit längerem mit der Provenance von Prozessen. Beispiele sind multidisziplinäre Simulations-Workflows, die Datenverwaltung von Simulations- und Experimentaldaten (elektronisches Laborbuch) oder die Langzeitarchivierung von Veröffentlichungen inklusive der zugehörigen Daten.

Ein aktuelles Anwendungsgebiet ist die Provenance von Prozessen aus der Softwareentwicklung. Die Hauptmotive sind, höhere Softwarequalität zu erreichen und Wiederverwendungsmöglichkeiten zu verbessern. Die sehr komplexen Prozesse in der Softwareentwicklung mit vielen Interaktionen zwischen Entwicklern und Werkzeugen sind dabei eine Herausforderung.

Um die Provenance zu erfassen, werden die Prozesse durch ein Provenance-Modell beschrieben. Dafür gibt es das Provenance-Datenmodell PROV. Dieses wurde 2014 durch das W3C standardisiert. PROV modelliert Prozesse als Graph mit Entities, Activities und Agents als Knoten sowie deren Verbindungen und Abhängigkeiten als Kanten. In der Softwareentwicklung sind Entwickler die Agents. Aufgaben, Revisionen oder Releases entsprechen Entities. Activities sind Vorgänge wie das Anlegen von Aufgaben, das Programmieren, das Bauen der Anwendung oder das Übertragen der Änderung in das Versionskontrollsystem (Abbildung 40). Während der Entwicklung der Software zeichnet man sämtliche Aktionen und deren Abhängigkeiten auf. Dabei entsteht ein gerichteter azyklischer Graph, der den zeitlichen Ablauf des Prozesses beschreibt. Eine Graph-Datenbank (Neo4J) speichert diese Informationen.

Durch Abfragen mit einer Graph Query Language kann man die Provenance analysieren. Dadurch lassen sich komplexe Fragen über die Entstehungshistorie der Software beantworten. Die Fragen lassen sich in drei Aspekte aufteilen:

- **Erkennen von Fehlern.** Mit den Provenance-Daten sollen automatisch Ursachen für fehlgeschlagene Tests oder Bugs gefunden werden. Zum Beispiel: *„Welche Anforderung führt zu den meisten Fehlern im fertigen Software-Release?"*
- **Statistische Analysen.** Durch aggregierte Provenance-Daten sollen Aussagen über Komplexität und Qualität ermittelt werden. Zum Beispiel: *„Wie viele Aufgaben (Issues) hat Entwickler X für das Release Y implementiert?"*
- **Bewerten der Entwickler.** Einzelne Entwickler sollen auf Grundlage der Provenance-Daten bewertet werden. Durch erkannte individuelle Stärken und Schwächen können Projektleiter den Entwicklern besser geeignete Aufgaben übertragen. Zum Beispiel: *„Welcher Entwickler ist am aktivsten beim Schreiben von Dokumentation?"*

In der Zukunft möchten wir verstärkt daran arbeiten, Provenance-Daten automatisch zu analysieren und zu visualisieren. Die Herausforderung ist dabei, dass die Provenance-Daten von langlaufenden Prozessen sehr groß werden.

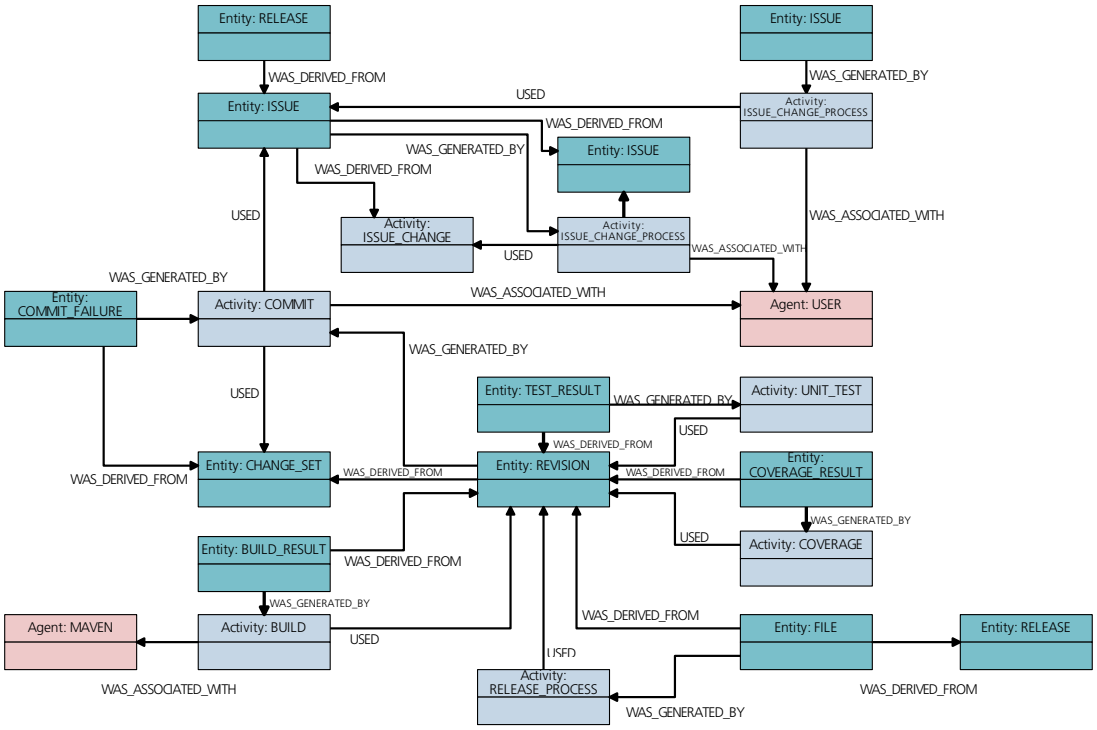


Abbildung 40: Provenance-Modell eines Softwareentwicklungs-Prozesses in PROV-Notation.

Simulation und Modellierung

Die Digitalisierung der Arbeitswelt macht vor der Raumfahrt nicht halt. Sie birgt großes Innovationspotential und bietet neue Möglichkeiten. Die Gruppe „Simulation und Modellierung" arbeitet im thematischen Überlappungsbereich zwischen IT und raumfahrttechnischen Ingenieurdisziplinen. Ihre Aufgabe ist es, moderne Entwurfsmethoden mit den Experten zu erarbeiten und Spezialsoftware zu entwickeln. Die zwei wichtigsten Themen der Gruppe sind modellbasiertes Systems Engineering und Concurrent Engineering. Diese Themen stellen die heute eingesetzten Vorgehensweisen, Prozesse, Organisationsformen und IT-Lösungen auf den Prüfstand. Der Fokus der Arbeitsgruppe liegt auf den IT-relevanten Aspekten: Ziel ist der vollständige, digitale Produktentwurf, ein virtueller Satellit. Gelingen soll dies, indem Entwurfsprozesse über den gesamten Lebenszyklus mit formalen Methoden abgebildet werden.

Modellbasiertes Systems Engineering

Modellbasiertes Systems Engineering gilt als Schlüsseltechnologie für die interdisziplinäre und integrierte Produktentwicklung. Genau wie beim klassischen Systems Engineering hat auch das modellbasierte Systems Engineering das Ziel, eine möglichst ausgewogene Systemlösung für ein Entwurfsproblem zu liefern. Dabei sollen nach Möglichkeit alle Bedürfnisse der Stakeholder berücksichtigt werden. Dies erfordert eine genaue Beschreibung der Anforderungen. Anhand derer lässt sich später der Entwurf verifizieren.

Mit dem klassischen dokumentenbasierten Ansatz können die immer komplexer werdenden Projekte nicht mehr bearbeitet werden. Der modellbasierte Ansatz dagegen basiert auf digitalen Systemmodellen, die maschinell verarbeitet werden können. Zur Beschreibung des Systemmodells kommen Modellierungssprachen wie SysML oder domänenspezifische Sprachen zum Einsatz. Sie helfen bei der Komplexitätsbewältigung und ermöglichen den Informationsaustausch zwischen den beteiligten Disziplinen. Modellbasierte Verfahren kombinieren also die Ziele des Systems Engineering mit den Möglichkeiten moderner Computerunterstützung. Dadurch entstehen ganz neue Optionen für die Entwurfsbewertung und Systemanalyse, zum Beispiel die multidisziplinäre Simulation und Optimierung, oder die formale Verifikation.

Concurrent Engineering

„Concurrent Engineering" bezeichnet ein strukturiertes Vorgehen im Produktentwurf. Es basiert auf der Parallelisierung der einzelnen Aufgaben. Das bedeutet, dass die verschiedenen Mitglieder des Designteams gleichzeitig am Entwurf arbeiten und sich in kurzen Iterationen austauschen. Dies fördert Zusammenarbeit, Vertrauen und Informationsaustausch zwischen den verschiedenen Experten im Team. Das Ergebnis ist ein hohes Maß an Transparenz. Ingenieure können nun Designparameter, die Einfluss auf verschiedene Subsysteme besitzen, parallel betrachten. Dies führt zu einem breiten Konsens von Entwurfsentscheidungen. Die Methode fördert eine ganzheitliche Betrachtung des Entwurfs auf Systemlevel über den gesamten Lebenszyklus (Systems Thinking). Sie wird hauptsächlich in frühen Entwurfsphasen eingesetzt.

Das DLR unterhält zwei Zentren, an denen parallel und gleichzeitig entworfen wird. An beiden Standorten wird Software von SC eingesetzt:

- Die Concurrent Engineering Facility am Institut für Raumfahrtsysteme in Bremen: Hier wird die Software Virtueller Satellit verwendet, die von uns in der Arbeitsgruppe „Simulation und Modellierung" entwickelt wird.
- Das Integrated Design Laboratory am Institut für Lufttransportsysteme in Hamburg: Hier kommt die von der Arbeitsgruppe „Verteilte Softwaresysteme" entworfene Software RCE zum Einsatz.

Herausforderungen und Ziele

Entwurf von Raumfahrtmissionen
Raumfahrzeuge sind komplexe Systeme. Sie bestehen aus zahlreichen Instrumenten und Teilsystemen, zum Beispiel dem Lageregelungssystem oder dem Kommunikationssystem für die Verbindung mit der Erde. Am Ende zählt allerdings nur eins: Das perfekte Zusammenspiel aller Komponenten für eine erfolgreiche Mission. Um genau das sicherzustellen, nutzt man die Methoden des Systems Engineerings.

Die Simulation von Entwürfen spielt dabei eine wichtige Rolle. In der Raumfahrt wird seit jeher das Verhalten von Teilsystemen oder des Gesamtsystems simuliert. Damit wird verifiziert, dass die Anforderungen an das System erfüllt werden. Vor der Simulation steht die Modellbildung. Das bedeutet, dass das Verhalten idealisiert abgebildet wird. Also muss auch die Modellbildung verifiziert werden. Modellbildung und Simulation gehören zusammen, sie sind ein iteratives System.

Simulation der Subsysteme
Die Computersimulation ist fester Bestandteil der Analyse von Raumfahrzeugentwürfen. Alle Einzeldisziplinen haben Werkzeugketten entwickelt und für ihre jeweilige Domäne als Standard etabliert. Mit diesen Werkzeugen wird zunächst ein systemspezifisches Modell erstellt. Dann wird das dazugehörige Verhalten simuliert, das anschließend gegenüber nötigen Auslegungsfällen oder Anforderungen bewertet werden muss. Entspricht der Entwurf nicht den Anforderungen, muss er überarbeitet werden.

Eine Herausforderung dabei: Die Subsysteme stehen in Wechselwirkung miteinander. Die Einzeldisziplinen können daher nicht isoliert betrachtet werden. Mal ist die Kopplung stärker, mal schwächer. Aus Systemsicht ist es daher wichtig zu wissen, welche Kopplungen es gibt und wie stark die Auswirkungen auf die anderen Disziplinen sind. Eine typische Frage ist beispielsweise, ob die Struktur verstärkt werden muss, wenn ein größerer Antrieb verbaut werden soll. Solche Systemabhängigkeiten zu identifizieren und abzubilden ist eine zentrale Aufgabe im Systems Engineering.

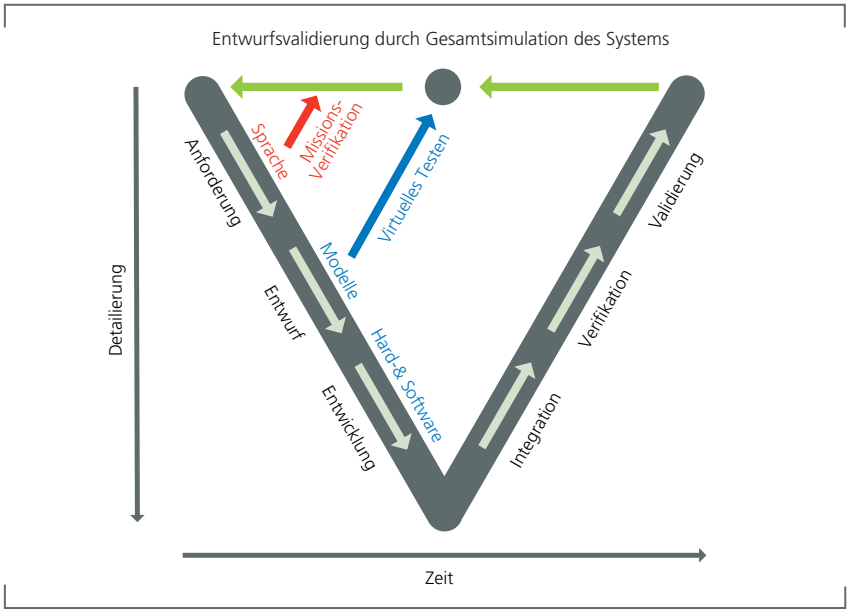


Abbildung 41: Das V-Modell des System Engineering.

Die Simulation des Gesamtsystems ist eine Möglichkeit, diese gegenseitigen Abhängigkeiten zu berücksichtigen. Üblicherweise wird als Entwurfsprozess im Systems Engineering das so genannte V-Modell eingesetzt. Dieses gibt ein sequentielles Vorgehen vor. Der Weg der Gesamtsystemsimulation ist eine Abkürzung im V-Modell (Abbildung 41 verdeutlicht diesen Schritt in blau). Damit erhält man eine permanente Entwurfsvalidierung über den gesamten Entwicklungszyklus.

Sobald Modelle von Teilsystemen vorhanden sind, kann man sie zu einer Gesamtsimulation verbinden. Für eine gemeinsame Analyse ist es zwingend notwendig, dass der Datenaustausch zwischen den Disziplinen reibungslos funktioniert. Dadurch kann untersucht werden, welchen Effekt eine Änderung in einem Modell auf die anderen Modelle hat. Wir verwenden für die gesamtheitliche Betrachtung ein zentrales Systemmodell. Hierin werden die zu verwendenden Schnittstellen definiert, so dass die Modelle zusammenpassen. Zur einfacheren Integration werden die Modelle der Einzeldisziplinen vom zentralen Modell abgeleitet. Dies ermöglicht es, Lösungen für folgende zwei Forschungsbereiche zu entwickeln:

- Wir arbeiten an der automatischen Generierung von Modellen und der benötigten Artefakte. Dabei handelt es sich zum Beispiel um einen Zwischenbericht. Einmal entwickelt, arbeitet die automatische Erzeugung fehlerfrei und effizient.
- Die Modelltransformation bindet bestehende Alt- oder Fremdmodelle ins Systemmodell ein.

Kostenfestlegung
Während der Missionsplanung treten Fehler auf. Das lässt sich nicht vermeiden, und sie müssen korrigiert werden. Je später ein Fehler bemerkt wird, desto schwieriger ist es, ihn zu beheben und desto höher sind die Kosten. Deswegen versucht man im Systems Engineering, möglichst früh so viel wie möglich über Verhalten und Abhängigkeiten des Systems herauszufinden. Man nennt das auch „Front Loading“ im Entwurf. Je mehr Zusammenhänge früh verstanden und analysiert sind, umso geringer ist die Gefahr, einen Fehler erst spät im Entwicklungszyklus zu bemerken. Das bedeutet auch: Die Gesamtkosten eines Projekts werden maßgeblich zu Beginn bestimmt. Bis zu 80 % der Kosten werden in den frühen Entwicklungs- und Verifikationsphasen festgelegt (Abbildung 42).

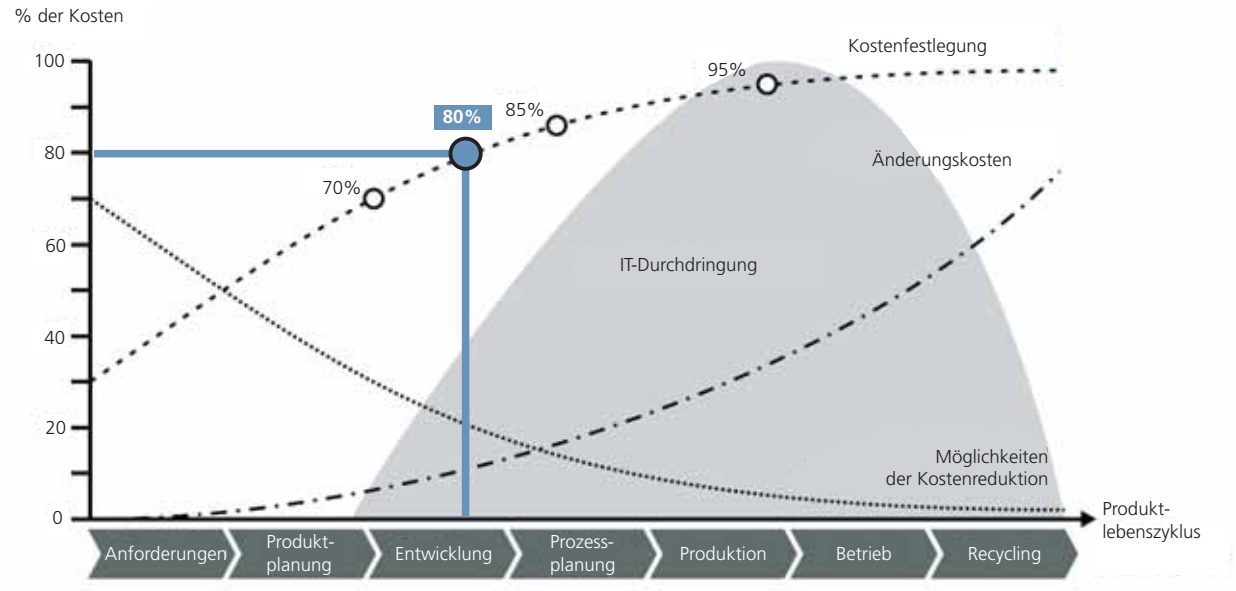


Abbildung 42: Kostenfestlegung und Änderungskosten in der Produktentwicklung; annotiert aus Eigner, M.: Product Lifecycle Management, Springer-Verlag, 2009.

Phasendurchgängigkeit
Der integrierte Produktentwurf zielt auf eine Durchgängigkeit über die Projektphasen hinweg. Momentan bestehen noch starke Brüche an den Übergängen von einer zur nächsten Phase. Information aus früheren Phasen lassen sich nur ungenügend bis gar nicht als Input für die nächste Phase verwenden. Auch hierfür bieten modellbasierte Ansätze vielversprechende Lösungen.

Die Brüche zwischen den Entwicklungsphasen kommen zum einen dadurch zustande, dass die IT-Werkzeuge nicht die benötigten Konvertierungen für einen reibungslosen Datenfluss bereitstellen. Zum anderen liegt es an den Anforderungen an die Modelle, die in den einzelnen Phasen sehr unterschiedlich sind. Nötig wäre eine Modellierungsrichtlinie, nach der im Projekt entwickelt wird. Dazu gehört eine Modellverwaltung, die es ermöglicht, Modelle wiederzuverwenden und zu verknüpfen. Für einen sinnvollen Entwurf sollten die Modelle den passenden Detailierungsgrad für die jeweilige Phase erfüllen. Zu Beginn können Modelle noch grob sein, es wird mit größeren Toleranzen gearbeitet. Später werden die Analysen dann immer genauer, die Modelle präziser und Toleranzen geringer.

Aktivitäten und Ergebnisse

In der Arbeitsgruppe „Simulation und Modellierung“ gibt es ein großes, übergeordnetes Softwareprojekt: „Virtueller Satellit“. Es ist der Rahmen für alle Aktivitäten der Arbeitsgruppe.

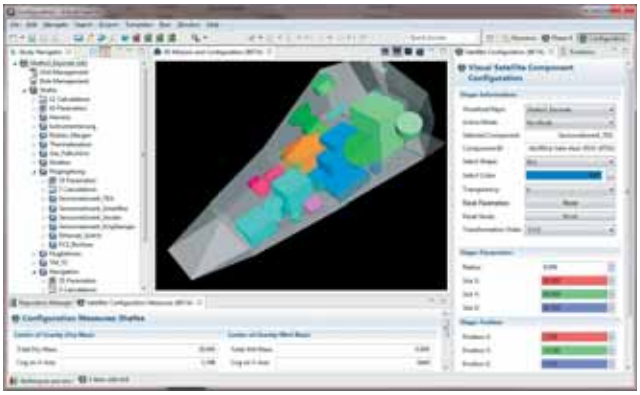
Der wichtigste Erfolg war die Einführung der Software in der Concurrent Engineering Facility (CEF) des Instituts für Raumfahrtsysteme im Dezember 2011 (Abbildung 43). Es zeigte sich schon in den ersten Studien, dass der „Virtuelle Satellit“ alle grundlegenden Voraussetzungen für verteiltes Zusammenarbeiten an einem gemeinsamen Entwurfsmodell unterstützt. Seitdem wurde die Software regelmäßig verwendet, insgesamt bisher in mehr als zehn Entwurfsstudien. Die Akzeptanz der Software im Team der CEF ist sehr hoch. Die enge Zusammenarbeit zwischen Anwendern und Entwicklern führt ständig zu neuen Ideen. Schrittweise werden neue Funktionen und Forschungsergebnisse zum Virtuellen Satelliten hinzugefügt.

Die agile Softwareentwicklung und die modulare Architektur sorgen für hohe Qualität, die für den produktiven Einsatz unerlässlich ist. Auf der anderen Seite ermöglicht es diese Arbeitsweise auch Masterstudenten und Doktoranden, Ideen schnell zu implementieren und auszuprobieren.

Kooperative Visualisierung
Im „Virtuellen Satelliten“ kommt ein zentrales Systemmodell zum Einsatz. Ein Vorteil zeigt sich an der integrierten Visualisierungskomponente für die Konfiguration des Raumfahrzeugs. Darin werden die eingebebenen Werte direkt visualisiert. Das wiederum fördert deutlich das gemeinsame Verständnis des Gesamtentwurfs. Die Ansicht sorgt für Diskussion und fördert dadurch die Zusammenarbeit. Fehler oder Missverständnisse lassen sich schneller identifizieren und bereinigen.

Abbildung 43: Die Software „Virtueller Satellit“ im Einsatz in der Concurrent Engineering Facility.

Abbildung 44: Interaktive 3D-Visualisierung in der Software „Virtueller Satellit".



Bislang war im Concurrent Engineering eine Person für die Konfiguration verantwortlich. Das Besondere an dem neuen Ansatz ist, dass nun das ganze Team zugleich an der Konfiguration des Raumfahrzeugs arbeitet. Gleichzeitig eingegebene Veränderungen, z. B. der Größe oder der Position von Komponenten, lassen sich direkt graphisch sichtbar machen (Abbildung 44). Jeder Teilnehmer der Studie ist selbst für die Positionierung seiner Bauteile verantwortlich. Der Konfigurationsingenieur unterstützt die Experten lediglich. Konflikte oder Varianten lassen sich nun viel besser besprechen und iterieren. Dies geschieht in einer Diskussionsrunde während der Concurrent Engineering Session.

Das Visualisierungsmodell ersetzt keine CAD-Zeichnung und ist bewusst einfach gehalten. Die Körperformen beschränken sich auf geometrische Primitive wie Quader, Kugeln oder Zylinder. Zusätzlich kann man die Farbe und Transparenz einstellen. Diese Werte werden in der Parameterstudie abgespeichert.

Ein Systemmodell, das angereichert ist mit Information über Position und Orientierung der Komponenten, hat weitere Vorteile:

- Es kann an Virtual-Reality-Anlagen und mobile Endgeräte angebunden werden, z. B. an Tablets und Smartphones. So schaffen wir Anknüpfungspunkte zu den anderen Arbeitsgruppen bei SC.
- Es generiert sich automatisch ein 3D-Geometriemodell in einer CAD-Software (vgl. Abbildung 45). Das reduziert den Arbeitsaufwand für den Konfigurationsingenieur deutlich. Das CAD-Modell dient als Basis für die Detailkonstruktion.
- Wichtige Parameter werden automatisch berechnet, z. B. Schwerpunkt oder Massenträgheit. Das ist etwa für die Auslegung des Regelungssystems oder der Struktur relevant.
- Jeder Stakeholder kann die aktuelle Konfiguration zu jeder Zeit selbst anschauen und in die für ihn gewünschte Position drehen. Das erhöht auch das Verständnis der Entscheidungsträger. Beschlüsse und Änderungswünsche lassen sich leichter kommunizieren.

Diagramme

Das zentrale Systemmodell enthält sämtliche Informationen aller Disziplinen. Dadurch lassen sich Abhängigkeiten analysieren, Automatisierungen sinnvoll nutzen und Verifikationen durchführen. Allerdings wird die Datenmenge mit ihren Beziehungen schnell unübersichtlich. Ein Beispiel: Die Concurrent-Engineering-Studie zum Wiedereintrittsexperiment Shefex 3 hatte nach einer Woche 105 Komponenten definiert. Jede Komponente enthielt ca. 50 Parameter und 10 Berechnungsformeln. Um diese Fülle an Information übersichtlich und strukturiert darzustellen, haben wir verschiedene Perspektiven auf die Daten entwickelt.

Modellierungssprachen wie SysML oder UML definieren verschiedene Diagrammtypen. Damit lassen sich Zusammenhänge gut abbilden. Die entsprechenden Modellierungswerkzeuge bieten graphische Editoren, mit denen der Benutzer ein technisches System graphisch beschreiben kann. Wir haben uns daran orientiert und drei Diagrammtypen in die Software „Virtueller Satellit" integriert:

- **Interface Diagram –** Es zeigt die Abhängigkeiten der beteiligten Disziplinen untereinander. Pro Komponente werden die Eingangs- und Ausgabeparameter dargestellt. Pfeile zeigen die Verbindungen zu anderen Disziplinen auf. Das Diagramm offenbart, wer von der Folge einer Änderung betroffen sein kann und auf welchen Parametern der eigene Entwurf basiert.

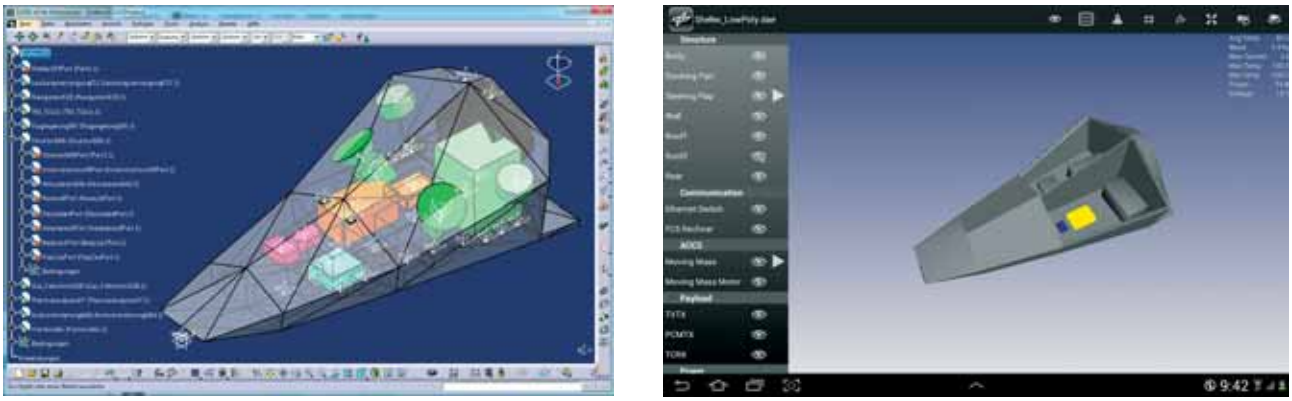


Abbildung 45: Ergebnis des automatisch generierten CAD-Modells in der Software Catia (links), Visualisierung des Modell-Exports auf einem Tablet (rechts).

- **Internal Block Diagram –** Es zeigt den internen Aufbau einer Komponente. Zum Beispiel erkennt man, welche Parameter durch welche Berechnungen miteinander verknüpft sind.
- **Functional Block Diagram –** Es stellt die funktionalen Beziehungen der Subsysteme dar. Diese Ansicht gibt eine bessere Übersicht auf Systemebene.

Die Diagramme können für jede Komponente erstellt und mit der Studie abgespeichert werden. In Abbildung 46 sieht man ein Schnittstellen-Diagramm eines Antriebsystems (engl.: Propulsion). Die Anbindung der Diagramme an das Systemmodell ist bidirektional. Der Benutzer kann die aus dem Systemmodell erzeugten Diagramme direkt bearbeiten. Die Änderungen fließen automatisch in das Systemmodell zurück. Diese Aktualisierung funktioniert auch im umgekehrten Fall. Ändert sich das Systemmodell im Hintergrund, dann passt sich das Diagramm an.

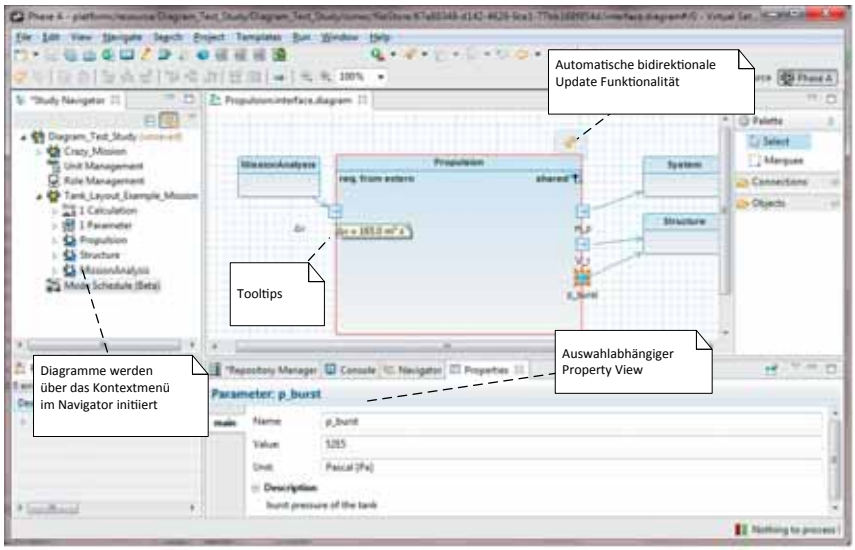


Abbildung 46: Der Screenshot zeigt ein Schnittstellen-Diagramm. So lassen sich Abhängigkeiten im Entwurf darstellen. Die Auslegung des Antriebssystems basiert im Beispiel auf dem Ergebnis der Missionsanalyse und liefert Werte an die Disziplinen System und Struktur.

Die Integration der Diagramme in den „Virtuellen Satelliten" bietet den Benutzern weitere Vorteile. Zum Beispiel erhält man kontextsensitive Informationen beim Überfahren des Mauszeigers (Tooltipps) oder eine erweiterte Eigenschaftenansicht (PropertyView) beim Klicken der rechten Maustaste. Dies ermöglicht eine schnelle Analyse der dargestellten Systemkomponenten.

Formale Verifikation
Mit Concurrent Engineering schafft man im Raumfahrzeugentwurf kurze Designzyklen. Diese Arbeitsweise ähnelt modernen Verfahren des Software-Engineering, beispielsweise Scrum und agilen Methoden. Es gibt jedoch ein Element, mit dem sich der klassische Produktentwurf schwer tut: die automatische Verifikation.

Im Software-Engineering nutzt man hierzu Test-Suiten, die automatisch ausgeführt werden können. Sie geben direkt Feedback, ob das gewünschte Verhalten eintritt oder nicht. Die Tests decken verschiedene Ebenen ab. Alles kann automatisch getestet werden, von atomaren Unit-Tests auf Methodenebene bis zu Integrations- und Akzeptanztests für das Verifizieren der Module der Gesamt-Software.

Ähnliches wünscht man sich auch im Raumfahrzeugentwurf. Gerade zu Beginn von Raumfahrtprojekten treten viele Änderungen auf, die schnell bewertet werden müssen. Für den System-Ingenieur ist es mitunter schwierig, die Konsequenzen dieser Änderungen abzusehen. Diese frühe Verifikation ist in V-Modell in Abbildung 41 rot gekennzeichnet. Man kann diesen Schritt als Mini-V bezeichnen. Er prüft schon mit den ersten groben Modellen, ob die Missionsziele erreicht werden.

Der erste Schritt zu einer automatischen Verifikation lautet: Anforderungen und Missionsziele formalisieren. Nur dann kann der Computer sie verarbeiten und mit den Simulationsergebnissen vergleichen. Das Beispiel verdeutlicht die Umwandlung einer solchen Anforderung:

- In natürlicher Sprache: „Die Mission soll in einem Jahr 10 Gigabyte an wissenschaftlichen Daten zur Erde senden."
- Als formaler Ausdruck:

```
Describe Operational Goals
     Parameter DownlinkData Above 10000.0;
     Parameter MissionTime Below 365.0;
End
```

Der zweite Schritt: das Verhalten des Raumfahrzeugs als Zustandsmaschine beschreiben. Hierzu werden aus dem Systemmodell geeignete Betriebszustände abgeleitet, die Zustandsvariablen definiert und die Transitionen zwischen den Zuständen beschrieben. Alle Informationen werden in einem Verifikationsmodell hinterlegt.

Abbildung 47 gibt einen Überblick über die Teile des Verifikationsverfahrens im „Virtuellen Satelliten". Vom Systemmodell wird das Verifikationsmodell abgeleitet. Der Benutzer kann es bequem über einen Texteditor bearbeiten. Der Texteditor unterstützt eine spezielle domänenspezifische Sprache, die extra für die Missionsdefinition entwickelt wurde (siehe Infobox). Das erleichtert die Arbeit der Ingenieure.

Das Verifikationsmodell kann mit Simulationsmodellen gekoppelt werden, um z. B. den Orbit des Satelliten zu simulieren. Außerdem kann der Suchalgorithmus für den Verifikationsschritt definiert und dessen Parameter eingestellt werden. Während der Verifikation

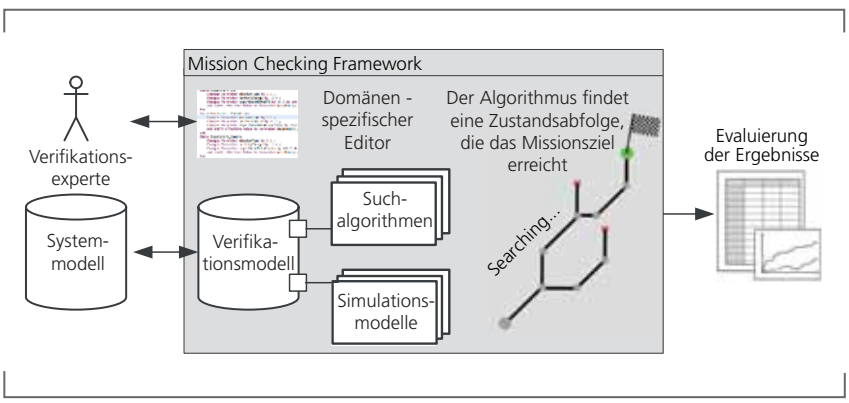


Abbildung 47: Mission Checking Framework zur automatischen Verifikation von Raumfahrtmissionen.

versucht der Suchalgorithmus, eine Sequenz von Zustandsänderungen zu finden, die die Missionsziele erreicht. Wird eine solche Sequenz gefunden, erfüllt der Entwurf die Anforderungen. Ansonsten muss entweder der Entwurf überarbeitet werden, oder die Missionsziele müssen angepasst und gelockert werden.

Eine domänenspezifische Sprache ist eine formale Sprache, die besonders für ein bestimmtes Problemfeld (die sogenannte Domäne) entworfen und implementiert wird. Beim Entwurf einer solchen Sprache sollte man beachten, dass sie die Probleme der Domäne darstellen kann und nichts zulässt, was außerhalb der Domäne liegt. Dadurch ist sie durch einen Domänenspezialisten ohne besonderes Zusatzwissen bedienbar.

Ein solches Verfahren gibt ein schnelles Feedback, ob die Mission so überhaupt durchgeführt werden kann. Es lässt sich prinzipiell kontinuierlich, also nach jeder Modifikation des Entwurfsmodells, verwenden. Daher bietet es sich an, diese Methode als automatische Funktionalität in das Concurrent Engineering zu integrieren.

Die formale Verifikation ist recht neu, hat aber eine große Tragweite. Sie verwendet zahlreiche Informatikkonzepte (Zustandsautomaten, Suchalgorithmen, domänenspezifische Sprachen) und versucht, sie in den klassischen Produktentwurf einzubringen. Die Domäne „Verifikation" ist ein spezielles Fachgebiet. Daher wurde sie als neue Disziplin in das Concurrent Engineering eingefügt. Die Funktionalität des Mission Checking Framework wurde in die Software „Virtuellen Satelliten" integriert.

Modell-Bibliothek

Wendet man den modellgetriebenen Ansatz konsequent an, bekommt man mit der Zeit eine Vielzahl von Modellen. Sie enthalten das Entwurfswissen, seien es Annahmen, Einschränkungen, analytische Beziehungen in Tabellenkalkulationen oder das für eine Finite-Element-Analyse aufbereitete Geometriemodell. Dieses Wissen sollte möglichst nicht verloren gehen. Deshalb haben mehrere DLR-Instituten das interne DLR-Vorhaben „SimMoLib" (Simulation Model Library) initiiert. Hauptziel des Vorhabens „SimMoLib" ist, dieses Wissen zu konservieren und für zukünftige Missionen und Projekte zur Verfügung zu stellen.

Die Aktivitäten können grob in zwei Bereiche unterteilt werden. Der erste und nächstliegende Aspekt ist die Bereitstellung einer Software, mit der nach vorhandenen Modellen gesucht und neue Modelle mit anderen geteilt werden können. Der zweite Aspekt beschäftigt sich allgemein mit Wiederverwendung: Wann ist die Verwendung eines gespeicherten Modells sinnvoll, und wie integriert man es am besten in eine laufende Studie? Wenn ein Ingenieur ein Modell wiederverwenden will, muss er es nicht nur schnell finden, sondern auch verstehen können. Nur dann kann er beurteilen, ob es für seinen Anwendungsfall eingesetzt werden kann. Vertrauen spielt dabei eine wichtige Rolle.

Im Projekt wurde eine Client-Server-Infrastruktur entwickelt, die nun im ganzen DLR verfügbar ist. Über eine Website kann man schnell und einfach nach Modellen suchen und sie ggf. herunterladen (Abbildung 48). Der Web-Service besteht aus einem einfachen Suchfeld, das an die Suchmaschine von Google erinnert, einer Ergebnisliste und Filtern zum Eingrenzen der Suchergebnisse.

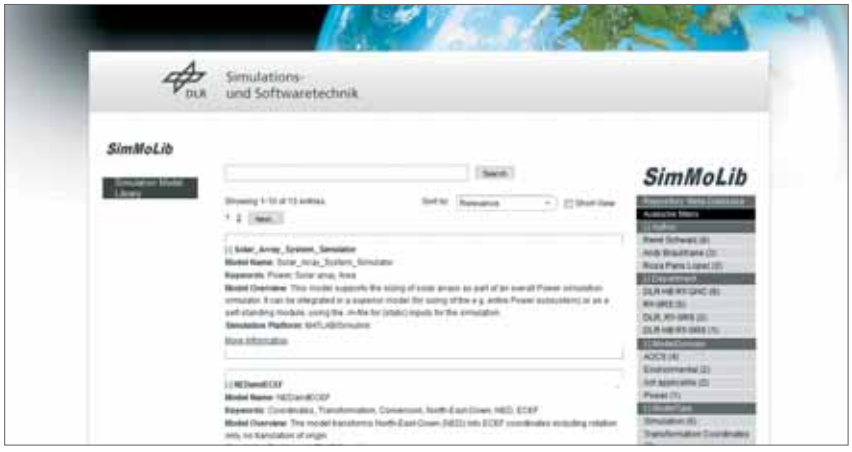
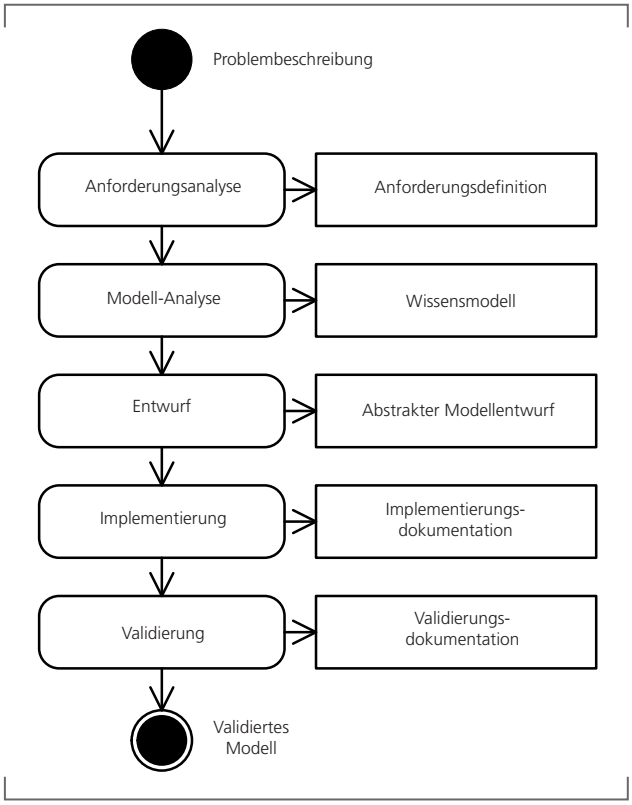


Abbildung 48: Server von SimMoLib zur Modellsuche.

Mit den Projektpartnern wurden Modellierungsrichtlinien entwickelt, mit denen eine hohe Qualität der Modelle erreicht werden kann. Nur Modelle, die nach diesen Richtlinien entwickelt wurden, können in die Bibliothek eingepflegt werden. Zu den Voraussetzungen gehören ein gewisser Anteil an Dokumentation, Schlüsselworte sowie Testfälle. Gerade die Testfälle sind wichtig, weil sie die beabsichtige Verwendung oder den Geltungsbereich eines Modells deutlich machen.

Als Teil der Richtlinien wurde ein Modellierungsprozess erarbeitet, der fünf Stufen in der Modellentwicklung festhält (Abbildung 49). In einem neuen Projekt passen die in der Datenbank gespeicherten lauffähigen Modelle häufig nicht direkt. In diesem Fall würde man auf das in den Modellen gespeicherte Wissen bis zum abstrakten Modellentwurf (Stufe 3) zurückgreifen. Die Implementierung (Stufe 4) würde dann entsprechend der Laufzeitumgebung angepasst.

Abbildung 49: Modellierungsprozess.



Die Inkompatibilität liegt häufig daran, dass im neuen Projekt eine andere Programmiersprache verwendet wird oder eine neuere Version der Simulationsumgebung vorgegeben ist. Bei MATLAB/Simulink beispielsweise gibt es jährlich mehrere neue Releases.

Umso wichtiger ist es, dass eine abstrakte Modellrepräsentation und das Wissensmodell aus den Stufen zwei und drei vorhanden sind. Möchte man ein Modell wiederverwenden, kann man an diesen Stellen ansetzen und muss nur die Implementierung in einer anderen Programmiersprache vornehmen. Dieser Schritt ist relativ einfach, wenn man die Modell-Architektur schon hat.

Der Client, um neue Modelle in die Bibliothek zu stellen und freizugeben, ist in der Software „Virtueller Satellit“ enthalten. Dadurch kann man die Bibliothek auch direkt fürs Concurrent Engineering nutzen. Modelle aus früheren Studien stehen zur Verfügung und können leicht in den aktuellen Entwurf eingebunden werden.

Fazit und Ausblick

Das Systems Engineering ist ein zentraler Bestandteil in ingenieurwissenschaftlichen Entwicklungsprojekten. Für das DLR ist es von fachbereichsübergreifender Bedeutung. Es fehlen aber häufig durchgängige und ganzheitliche Software-Lösungen, um Entwicklungsprozesse zu unterstützen. Antworten hierzu finden sich in modellbasierten Ansätzen. Sie werden das zukünftige Arbeiten prägen. Zudem ist die Community in diesem Forschungsbereich stark und wächst weiter – auch das ist ein wichtiger Faktor für die Erfolgsaussichten des Model-based Systems Engineering. Wir sehen uns verantwortlich, aktuelle MBSE-Entwicklungen in die DLR-Projekte hineinzutragen.

Die Arbeitsgruppe „Simulation und Modellierung" konnte sich in den vergangenen Jahren im Systems Engineering etablieren. Dass wir erfolgreich eine Software in der Concurrent Engineering Facility eingeführt haben, wurde über das DLR hinaus wahrgenommen. So war SC u. a. Mitorganisator der in diesem Bereich richtungsweisenden ESA-Konferenz „Systems and Concurrent Engineering for Space Application" (SECESA), die im Herbst 2014 in Stuttgart stattfand.

In den folgenden Jahren wird die Software „Virtueller Satellit" weiterhin die Basis für die Arbeitsgruppe sein. Auf europäischer Ebene wurden kürzlich Standards für eine Systemmodellierung im Concurrent Engineering definiert. Mit dem Projekt „Open Concurrent Design Tools" (OCDT) führt die ESA momentan eine zentrale Datenbank für ihre Studien ein. Für unsere Software ist es wichtig, diesen Standard und die Anbindung an den zentralen Server zu unterstützen. Die ESA versucht zudem, diese Werkzeuge für System-of-Systems-Studien anzuwenden. Wir sind momentan an einem ESA-Projekt beteiligt, das die Parameter und Schnittstellen für System-of-Systems-Studien bereitstellt. Für uns ist das die ideale Gelegenheit, die ESA-Aktivitäten besser kennenzulernen. In Zukunft werden wir solche Studien dann auch mit unseren eigenen Produkten unterstützen.

Die ESA-Projekte „Virtual Spacecraft Design" und COMPASS bergen spannende Themen für die späteren Entwicklungsphasen und die damit verknüpften Verifikationsansätze. Diese Projekte sind für uns maßgebend, sowohl für die Durchgängigkeit im Entwurf als auch für die Entwurfsbewertung.

Modellbasiertes Systems Engineering kann das künftige Arbeiten revolutionieren. Der Übergang von Dokumenten zu Modellen bedeutet einen Paradigmenwechsel, der alle Bereiche erfasst. Der Durchbruch und Erfolg dieser Technologie ist jedoch nur mit Softwarewerkzeugen möglich. Daher wird das Thema für die Arbeitsgruppe „Simulation und Modellierung" immer relevanter und mit Sicherheit das Leitmotiv der Zukunft darstellen.

Eingebettete Systeme

Eingebettete Computer-Systeme spielen in der Raumfahrt eine wichtige Rolle. Dies betrifft alle beteiligten Segmente: von Bodensystemen über die Trägerrakete bis hin zum Raumfahrzeug. In vielen Ingenieursbereichen – nicht nur in der Raumfahrt – zeichnet sich ab, dass immer mehr Funktionen von Software anstatt von spezialisierter Hardware erbracht wird. Missionen hängen somit immer mehr von zuverlässiger Software ab. Dies erhöht die Anforderungen an die Software-Entwicklung deutlich. Die Entwicklung zuverlässiger und leistungsfähiger Software erfordert ein höchstes Maß an aktuellem Software-Engineering-Wissen. Als wissenschaftlich-technische Einrichtung mit einem Informatik-Hintergrund wendet SC aktuelle Entwicklungen in der Informatik und dem Software-Engineering für die Raumfahrt an. Die Arbeitsgruppe „Eingebettete Systeme" in der Abteilung „Software für Raumfahrtsysteme und interaktive Visualisierung" legt ihren Forschungs- und Arbeitsschwerpunkt auf die Entwicklung zuverlässiger Flug-Software für unbemannte Raumfahrtsysteme. Hauptsächlich unterstützen wir Raumfahrtmissionen des DLR, entwickeln aber auch im Auftrag externer Partner und beraten sie.

Herausforderungen der Software-Entwicklung für Raumfahrtsysteme

Die Herausforderungen bei der Entwicklung eingebetteter Systemen für das Raumsegment einer Raumfahrt-Mission sind in vielen Fällen vergleichbar mit anderen Domänen. Es gibt jedoch einige Besonderheiten in der Raumfahrt.

Wiederverwendung und Komplexität
Raumfahrzeuge sind sehr oft Einzelanfertigungen, insbesondere im Forschungsbereich. Dies erschwert die Wiederverwendung von Software. Durch die begrenzte Anzahl der Projekte ist außerdem der Drang zur Standardisierung gering.

Erfahrungen zeigen, dass die Flug-Software oft für jede Raumfahrtmission weitestgehend neu entwickelt wird. Erklärungen hierfür sind vielfältig:

- Die langen Projektlaufzeiten haben zur Folge, dass Software-Entwickler, also Wissensträger, nicht unbedingt auch in einem nachfolgenden Projekt zur Verfügung stehen.
- Die Software wird für das jeweilige System hochoptimiert, so dass sich Teile aus der Software nur schwer herauslösen lassen und wiederverwendet werden können.
- Es wird aufgrund der begrenzten Rechen-Ressourcen an Bord nur mit geringer Abstraktion gearbeitet. Das macht die Herauslösung funktionaler Komponenten sehr schwierig. Nur langsam setzt sich objektorientiertes Design durch – ein weitverbreitetes Programmierparadigma, das die Kapselung von Funktionalitäten fördert.

In den vergangenen Jahren wurde die Wiederverwendbarkeit von Software-Komponenten zunehmend als Ziel in die Systementwicklung mit aufgenommen. Dies wird beispielsweise erreicht durch

- objektorientierte Programmierparadigmen;
- wiederverwendbare Frameworks;
- Programmbibliotheken, die nicht nur für die aktuelle Mission optimiert sind;
- modellbasierte Techniken zur Code-Generierung.

Die Wiederverwendbarkeit wird durch die zunehmende Komplexität der Software immer wichtiger. Die Missionen werden immer aufwändiger und erfordern dadurch deutlich mehr Software-Funktionalität. Es ist also in vielen Fällen gar nicht mehr möglich, die komplette Flug-Software für eine Mission neu zu entwickeln.

Zuverlässigkeit
Zuverlässige Systeme sind enorm wichtig. Das ist nicht nur in der Raumfahrt der Fall. Eine Besonderheit ist hier jedoch, dass man einen Satelliten nur selten warten kann. Software-Updates sind zwar Alltag, auch in der Raumfahrt. Allerdings muss das System für ein Software-Update noch funktionieren.

Die Umweltbedingungen, in denen Raumfahrzeuge operieren, erhöhen die Anforderungen an Hardware und Software. Höhenstrahlung, das Vakuum und die extremen Temperaturunterschiede im Weltall begünstigen Fehler und Alterung der Hardware.

Ein fehlertolerantes Software-Design muss dafür sorgen, dass Störungen abgefangen werden. Fehlertoleranz wird unter anderem durch FDIR-Maßnahmen (Fault Detection, Isolation, and Recovery) erreicht. Dazu gehören beispielsweise Redundanzen in der Software, Konsistenz-Prüfungen, Fehlerkorrekturen etc.

Validierung und Verifikation spielen während der Software-Entwicklung eine entscheidende Rolle, um eine hohe Zuverlässigkeit zu erreichen. Dazu gehören neben ausgiebigen Tests auch Methoden der formalen Software-Verifikation.

Die folgenden Abschnitte geben einen Überblick über die wichtigsten Forschungsthemen und Aktivitäten der Arbeitsgruppe Eingebettete Systeme.

Software-Entwicklung für Flug-Systeme

Die Arbeitsgruppe beschäftigt sich hauptsächlich mit der Software-Entwicklung für unterschiedliche Flugsysteme. SC ist an Projekten beteiligt, die folgende Bereiche abdecken:

- Lage- und Bahnregelung,
- Navigationssysteme,
- Command & Data Handling,
- Nutzlastsysteme.

Darüber hinaus entwickeln wir Hardware-Treiber sowie Kommunikationsprotokolle, die in vielen der genannten Bereiche benötigt werden.

Lage- und Bahnregelung
Im Bereich der Lage- und Bahnregelung (Attitude and Orbit Control System – AOCS) haben wir im Berichtszeitraum insbesondere an den Satelliten-Bussen TET-1, BIROS und Eu:CROPIS gearbeitet (Abbildung 50). SC ist bei diesen Projekten für die Softwaretechnik (mit-)verantwortlich.

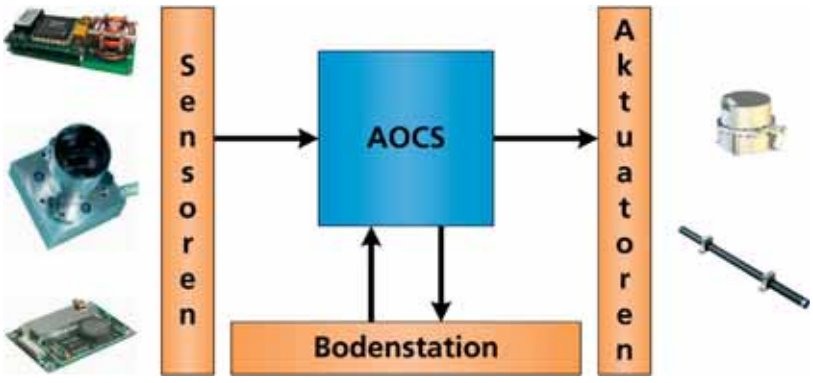


Abbildung 50: Einbettung der Lage- und Bahnregelung zwischen Sensoren, Aktuatoren und der Kommandierung durch die Bodenstation.

Abbildung 51: Künstlerische Darstellung des TET-1 im Orbit.

Abbildung 52: CAD-Entwurf des BIROS-Satelliten.

Dazu gehören:

- Software-Reviews
- Mitarbeit an den Software-Anforderungen
- Erstellen der Interface-Dokumente
- Erstellung und Modellierung der Softwarearchitektur
- Implementierung und Integration der Lageregelungssoftware in die Flug-Software
- Software-Konfigurationsmanagement
- Beratung in der Softwarequalitätssicherung
- Unterstützung bei Systemtests

OOV-TET-1 ist ein Kompaktsatellit (Abbildung 51), der im nationalen Raumfahrtprogramm entstand und 2012 gestartet wurde. Das OOV-Programm (On-Orbit Verification) soll Herstellern raumfahrttauglicher Produkte eine kostengünstige Möglichkeit bieten, diese im Weltraum zu qualifizieren. Der TET-1 wurde durch eine öffentlich-private Partnerschaft gefördert. Der Industrieanteil hatte eine Missionsdauer von einem Jahr. TET-1 wird nun durch das DLR in der FireBird-Mission betrieben.

Der Satellitenbus des TET-1 beruht auf dem Bus des BIRD-Satelliten, der 2001 gestartet wurde. Daher konnte die generische Lageregelungssoftware des BIRD in den Satellitenbus übernommen und erweitert werden.

2015 soll der Kompaktsatellit **BIROS** (Berlin InfraRed Optical System) starten, dessen Bus erweiterte Funktionalitäten im Vergleich zum TET-1-Satelliten bietet (Abbildung 52). Beide Satelliten zusammen bilden die **FireBird-Mission**. Sie werden sich im gleichen Orbit befinden und ein Beobachtungsgebiet kurz nacheinander überfliegen. Beide Satelliten besitzen als primäre Nutzlast Infrarot-Sensoren.

BIROS hat im Gegensatz zum TET-1 ein Antriebssystem, welches für einen Formationsflug mit TET-1 und zur Erprobung von In-Orbit-Servicing-Flugmanövern genutzt wird. Die Lageregelungs-Software wird daher um Orbit-Manöver erweitert.

Die DLR-Mission **Eu:CROPIS** (Euglena and Combined Regenerative Organic-Food Production in Space) soll die Langzeitstabilität eines biologischen Lebenserhaltungssystems demonstrieren. Der Start ist für 2017 geplant. Eu:CROPIS wird als spinstabilisierter Satellit ausgelegt. Der Satellit (Abbildung 53) soll nacheinander Gravitationsverhältnisse von Mond und Mars simulieren, indem er seine Rotationsgeschwindigkeit variiert.

Die Spin-Stabilisierung führt zu einigen tiefergreifenden Änderungen in der Lageregelungssoftware bei Eu:CROPIS. Das betrifft vor allem die Ansteuerung der Aktuatoren und Sensoren. Der modulare Aufbau der Software erlaubt eine effiziente Umsetzung dieser neuen Anforderungen.

Die Beispiele zeigen, dass sich eine objektorientierte Software-Architektur für Systeme mit hohen Echtzeitanforderungen bewährt hat. Große Teile der Lageregelungskomponenten konnten und können für die verschiedenen Satellitenmissionen wiederverwendet werden.

Command & Data Handling
Neben der Lageregelungs-Software unterstützt SC die Entwicklung der Software für das Command & Data Handling (C&DH) bei der **Eu:CROPIS-**Mission.

Ein C&DH-System empfängt Befehle (Telecommands) von der Bodenstation und leitet diese sofort oder zeitgesteuert an die Subsysteme weiter. Die Daten der Nutzlasten (z. B. die einzelnen Experimente) werden vom C&DH-System gesammelt, zwischengespeichert und bei Funkkontakt über die Telemetrie-Strecke zum Boden gesendet.

In enger Zusammenarbeit mit dem Institut für Raumfahrtsysteme in Bremen wirkt SC an der Erzeugung und Verteilung von Telemetrie- und Telecommand (TM/TC) an Bord des Satelliten mit. Dabei werden die etablierten TM/TC-Standards der ESA umgesetzt (Consultative Committee for Space Data Systems – CCSDS, hier insbesondere der Packet Utilisation Standard – PUS). Des Weiteren implementieren wir eine Massenspeicher-Verwaltung, um Daten der einzelnen Nutzlasten bis zur erfolgreichen Übertragung zum Boden zu speichern.

Navigation
Wir beteiligen uns im Bereich der Navigation hauptsächlich an Projekten, die mit optischen Verfahren arbeiten. Ein Navigationsfilter fusioniert die Eingaben verschiedener Sensoren, um die Position und Lage des Raumfahrzeugs zu schätzen. Schon seit Jahren werden erfolgreich Beschleunigungs- und Kreiselsysteme (Inertial Measurement Units – IMUs) eingesetzt. Daneben kommen auch optische Sensoren zum Einsatz, etwa Kameras, Sternenkameras und laser-basierte Systeme wie LIDARs und Laser-Altimeter.

Im Flugexperiment **SHEFEX II** (SHarp Edge Flight EXperiment) kam erstmals ein vom DLR entwickeltes Navigationsexperiment zum Einsatz (Abbildung 54). Das Experiment wurde 2012 geflogen. SHEFEX untersucht neuartige Bauweisen und Steuerungskonzep-

Abbildung 53: Künstlerische Darstellung von Eu:CROPIS.

Abbildung 54: SHEFEX-II-Navigationsexperiment.

Abbildung 55: Kraterdetektion in ATON zur Absolut-Positionsschätzung.

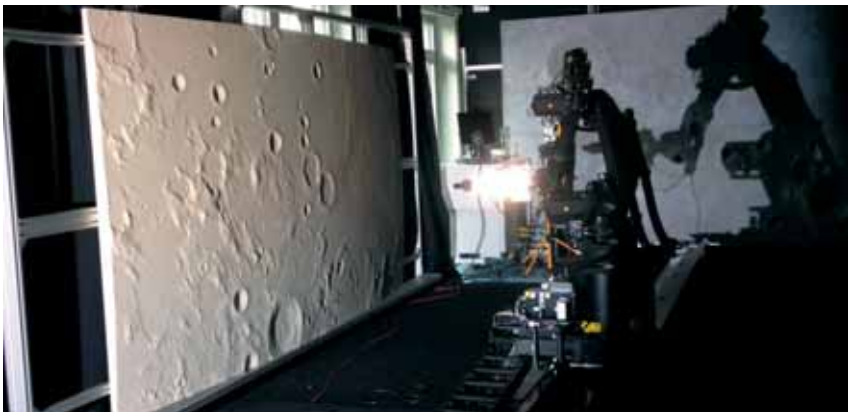


Abbildung 56: Testbed for Robotic Optical Navigation (TRON) am Institut für Raumfahrtsystem in Bremen. Zu sehen ist ein Roboter mit einer Kamera über einer maßstabsgerechten Mondlandschaft.

te für einen scharfkantigen Wiedereintrittskörper. Die Flüge werden mit Höhenforschungsraketen durchgeführt. Für eine präzise Steuerung während des Wiedereintritts ist eine genaue Navigation notwendig. Dies wurde mit dem Navigationsexperiment auf SHEFEX II erfolgreich demonstriert. Die Navigationslösung errechnet sich durch die Verbindung einer IMU, einer Sternenkamera und eines GPS-Empfängers. Für den Nachfolger SHEFEX III wird das Navigationssystem als Sensorsystem in die Regelung des Flugkörpers eingebunden.

Bei SHEFEX II beteiligten wir uns zusammen mit dem Institut für Raumfahrtsysteme an der Entwicklung des Navigationssystems. Wir leisteten Beiträge zur Software-Entwicklung und -Qualitätssicherung.

Mit dem Projekt **ATON** (Autonomous Terrain-based Optical Navigation) soll ein wesentlicher Schritt in Richtung autonomer Landesysteme unternommen werden. Im Mittelpunkt steht die Entwicklung einer Demonstrationshardware, die die Leistungsfähigkeit optischer Navigationssysteme evaluieren soll.

Das Beispielszenario ist die präzise Landung auf dem Mond. Dabei soll das optische Navigationssystem für den gesamten Landeanflug verwendbar sein. Dies umfasst neben der Anflugnavigation auch die Landeplatzbewertung. Neben den rein optischen Bildinformationen sollen Daten weiterer Sensoren genutzt werden (IMU, Sternsensor, LIDAR, Laser Altimeter). Das Ziel ist eine integrierte und robuste Navigationslösung sowie eine zuverlässige Hazard Avoidance (Risikovermeidung).

Von zentraler Bedeutung für das Projekt ATON sind effiziente Bildverarbeitungs- und Regelungsalgorithmen. Ein Beispiel dafür ist die Kraternavigation, die in ATON zur absoluten Positionsschätzung genutzt wird. Sie wurde am Institut für Raumfahrtsysteme entwickelt (Abbildung 55).

ATON hat inzwischen Closed-Loop-Versuche im Labor bestanden (Abbildung 56). Für 2015 sind Open-Loop-Versuche auf einem unbemannten Helikopter geplant, 2016 auch Closed-Loop-Versuche.

Unsere Aufgaben im Projekt ATON:

- Analyse und Systemdefinition,
- Entwicklung einer Simulationsumgebung,
- Software-Integration und -Bereitstellung,
- Beteiligung bei Systemtests in allen Entwicklungsstufen,
- Unterstützung und Beratung der Projektpartner in Fragen des Software-Engineering.

Hardware-Treiber
In vielen Projekten entwickeln wir Software-Treiber für Hardware-Komponenten, beispielsweise Kameras, Sternensensoren, Magnetometer und IMUs. Ein wesentliches Merkmal der Raumfahrt ist die Heterogenität der Onboard-Betriebssysteme. Dadurch haben wir inzwischen weitreichende Kompetenzen für unterschiedliche Betriebssysteme aufgebaut. Herauszuheben sind dabei Linux und die Echtzeitbetriebssysteme QNX, RTEMS und RODOS.

Kommunikationsprotokolle
In den vergangenen Jahren haben wir nicht nur die Treiberentwicklung deutlich ausgebaut, sondern auch die Spezifikation und Implementierung von Kommunikationsschnittstellen und -protokollen. Immer häufiger wird unsere Expertise auf diesem Feld angefragt. So konnten wir für das Projekt **MAIUS** erfolgreich die benötigten Protokolle für Telemetrie und Telecommands umsetzen. Für das Projekt **OBC-NG** entwickeln wir Protokolle für die Datenübertragung in einem SpaceWire-Netzwerk (ein europäischer Onboard-Netzwerk-Standard). Auch im Projekt **ATON** kommen die grundlegenden Protokolle für die Kopplung von Sensorsimulatoren und Labor von uns.

Beratung
Mitarbeiter der Arbeitsgruppe sind nicht nur als Entwickler in DLR-internen und -externen Projekten aktiv. Unsere Software-Expertise ist auch in Raumfahrt-Missionen gefragt, die wir als Berater begleiten. So hat uns z. B. die Firma dSPACE GmbH beauftragt, sie in der Anwendung der ECSS-Standards zu beraten (European Cooperation on Space Standardization). Des Weiteren sind wir immer wieder für die Astro- und Feinwerktechnik Adlershof GmbH tätig. Wir haben sie u.a. dabei unterstützt, den Software-Entwicklungsplan und die Software-Architektur für die DEOS- und TET-2-Missionen zu erstellen. Auch für die Missionen und Experimente TechnoSat (TU Berlin), GOSSAMER (DLR) und Vidana (Universität Würzburg) waren wir als Berater und Gutachter tätig.

Ausblick
Wir möchten unsere Erfahrungen mit modularer und wiederverwendbarer Onboard-Software ausbauen und in weitere Projekte einbringen. Besonders geeignet ist hierfür das Projekt **CLAVIS-S2TEP**, das 2015 unter Federführung des Instituts für Raumfahrtsysteme startete. Ziel ist die Erarbeitung eines Kleinsatellitenkonzepts, in dem möglichst viele vorkonfektionierte Komponenten zum Einsatz kommen. Zusammen mit dem Systemkonzept soll ein integrativer Entwicklungsprozess spezifiziert werden. Kürzere Entwicklungszyklen sollen dazu führen, dass z. B. neue Technologien schneller im Orbit getestet werden können. Wir werden in CLAVIS-S2TEP vor allem effizientere Entwicklungsprozesse spezifizieren. Unsere Expertise in der Entwicklung modularer Onboard-Software wird dabei von Vorteil sein.

Die Standardisierungs-Aktivitäten der ESA für Onboard-Software sind für uns richtungsweisend. Beispiele sind die Space Avionics Open Interface Architecture (SAVOIR) und die On-board Software Reference Architecture (OSRA). Wir verfolgen diese aufmerksam und werden die Anwendbarkeit für kommende Software-Entwicklungen prüfen.

Laufzeit-Plattformen für Raumfahrtsysteme

Modulare Laufzeitsysteme werden benötigt, damit Onboard-Applikationen zuverlässig arbeiten. Laufzeitsysteme sind in diesem Kontext ein Betriebssystem und eine optionale Middleware. Die Middleware bietet zusätzliche Dienste wie Kommunikation an.

Üblicherweise werden für Raumfahrt-Anwendungen eingebettete Echtzeit-Betriebssysteme eingesetzt. Zunehmend kommen aber auch „Standard"-Betriebssysteme wie Linux zum Einsatz. Gerade bei komplexen Applikationen wie der Bildverarbeitung bietet Linux den Vorteil, dass viele freie Software-Bibliotheken zur Verfügung stehen.

Echtzeit-Betriebssystem
Wir waren am Projekt **Core Avionics** des DLR beteiligt. Dabei ging es um die Entwicklung eines generischen Betriebssystems für Kompaktsatelliten. Es entstand ein Betriebssystem mit dem Namen RODOS. Seine Hauptbestandteile sind ein Echtzeitkern, eine Middleware und eine Netzwerkarchitektur zur Anbindung peripherer Komponenten und Nutzlasten (Abbildung 57). RODOS besitzt eine objektorientierte Architektur und erfordert die Anwendungsentwicklung in C++. Dank des modularen Aufbaus werden nur diejenigen Betriebssystem-Komponenten verwendet, die für ein konkretes Missionsziel benötigt werden. Dies führt zu einer sehr schlanken Onboard-Software.

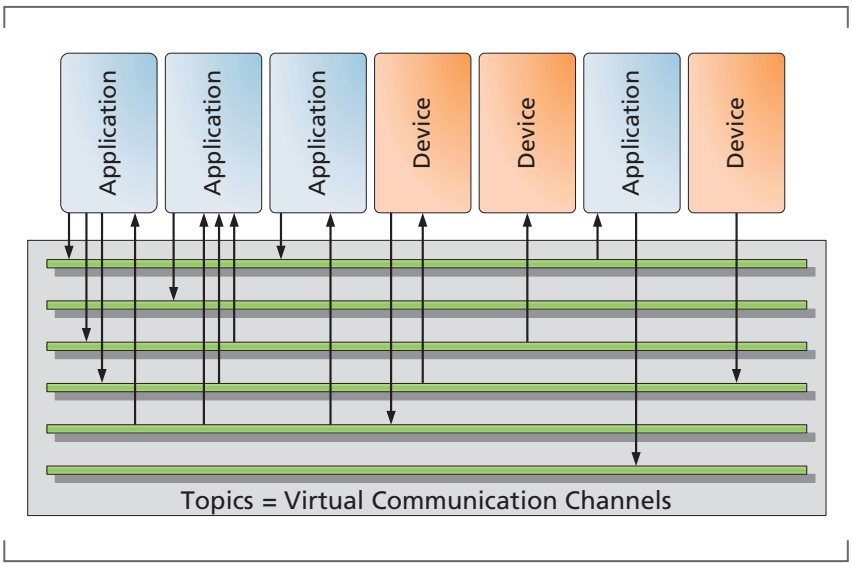


Abbildung 57: Konzept der RODOS Middleware für Avionik-Systeme.

Bei Core Avionics haben wir hauptsächlich die Nutzerdokumentation erstellt, sowie RODOS verifiziert und validiert. Im aktuellen Projekt **OBC-NG** soll RODOS ebenfalls eingesetzt werden. Dafür wird ein bisher nicht vorhandener Multicore-Support für den Prozessor „ARM Cortex A9" benötigt. Derzeit erweitern wir RODOS um diese Funktionalität.

Tasking Framework
Für Echtzeit-Kontrollsysteme ist die zeitgerechte Ausführung der Algorithmen essentiell. Verantwortlich hierfür ist das Scheduling. Es muss gewährleisten, dass alle Tasks rechtzeitig und in vorgegebener Reihenfolge ihre Ergebnisse liefern. Die Ergebnisse dienen üblicherweise wiederum als Eingangsdaten weiterer Tasks. Werden diese nachfolgenden Tasks ausgeführt, noch bevor aktualisierte Eingangsdaten vorliegen, kann es zu Fehlern kommen.

Im folgenden Beispiel ist dies tatsächlich aufgetreten: Die Tasks der Lageregelung des BIRD und TET-1-Satelliten hatten eine zeitlich feste und zyklische Ausführungsreihenfolge. Dieses vordefinierte Scheduling wurde beim TET-1 durch eine andere Anwendung zeitlich gestört und führte zu einem unerwarteten Zustand der Lageregelung.

Glücklicherweise ging der Satellit nicht verloren. Auf Basis unserer Erfahrungen entwickelten wir ein deutlich flexibleres Scheduling-Rahmenwerk mit dem Namen Tasking Framework. Es führt erst dann einen Task aus, sobald alle benötigten Eingangsdaten für die Verarbeitung zur Verfügung stehen – und nicht mehr in einer festgelegten Reihenfolge.

Das in C++ entwickelte Tasking Framework wird bereits heute in den Projekten **ATON, Eu:CROPIS** und **MAIUS** eingesetzt. Dabei ist es nicht an ein bestimmtes Betriebssystem gebunden. Das Tasking Framework ist auch die die Basis für die Laufzeit-Umgebung im Projekt **OBC-NG.**

Rekonfigurierbarer Onboard-Computer
Im Projekt **OBC-NG** (Onboard Computer – Next Generation) arbeiten wir an einer neuen Bord-Computer-Architektur für zukünftige Raumfahrtsysteme. Das Projekt ist aus dem Vorhaben **SHARC** (Software/Hardware Architecture for Reconfigurable Computing) hervorgegangen.

Es ist absehbar, dass der Bedarf an Rechenleistung für zukünftige Satelliten und Raumsonden zunehmen wird. Komponenten und Architekturen aktueller Bord-Computer stoßen aber an ihre Leistungsgrenzen. Das Problem lässt sich nicht ohne weiteres durch Hochleistungskomponenten beheben, da für Raumfahrtmissionen nur raumfahrtqualifizierte Prozessoren und FPGAs (Field-Programmable Gate Arrays) in Frage kommen. Und hier ist der Markt sehr überschaubar.

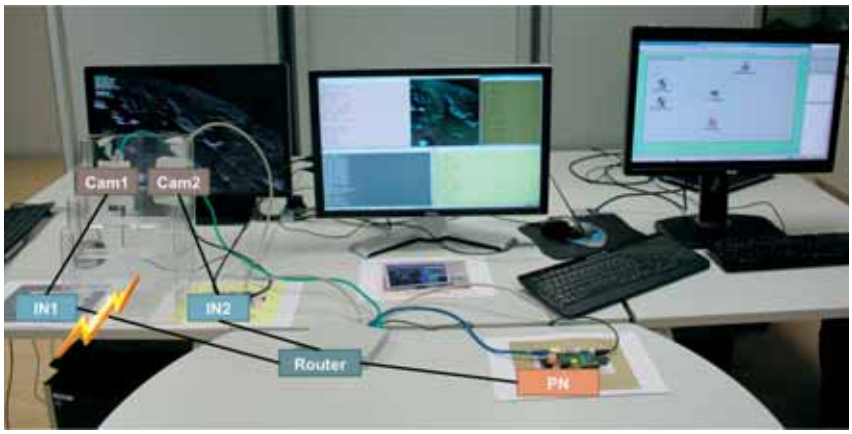


Abbildung 58: Demonstration von OBC-NG. Das Bild von Kamera 1 (Cam1) wird über einen Interface-Knoten (IN1) und über einen Router zu einem Knoten im Netz übertragen (PN, mittlerer Bildschirm). Wird die Verbindung von IN1 und Router unterbrochen, rekonfiguriert sich das System selbstständig. Ab jetzt überträgt Kamera 2 (Cam2) die Bilder über einen weiteren Interface-Knoten zum Empfänger.

Außerdem bauen aktuelle Architekturen auf unflexible Redundanzkonzepte: Im Fehlerfall wird eine Komponente durch eine baugleiche Komponente ersetzt. Diese Ersatzkomponente kann üblicherweise keine anderen Aufgaben übernehmen.

Genau hier möchte das Projekt OBC-NG ansetzen. Ein neues Konzept soll das unflexible Redundanzkonzept ersetzen. Es baut auf einem verteilten, rekonfigurierbaren System auf. Ein weiteres Merkmal von OBC-NG ist der verstärkte Einsatz von Standard-Bauteilen (Commercial Off-The-Shelve – COTS). Unter Weltraumbedingungen steigt zwar die Ausfallwahrscheinlichkeit einzelner Bauteile. Die geringere Zuverlässigkeit lässt sich aber voraussichtlich sehr gut durch das neue Redundanzkonzept ausgleichen. Der große Vorteil der COTS-Komponenten ist deren Performance. So lassen sich deutlich höhere Rechenleistungen für zukünftige Onboard-Systeme erreichen. Und diese würden nicht nur dem jeweiligen Subsystem des Satelliten zur Verfügung stehen, sondern je nach Konfiguration auch für weitere rechenintensive Aufgaben.

Bei OBC-NG übernimmt Software die Aufgabe, Fehler zu erkennen und zu beheben. Hierbei werden klassische Subsystem-Grenzen aufgebrochen: Im Prinzip können alle Prozessoren des Satelliten an dem OBC-NG-Netzwerk teilnehmen. Das System wird im Fehlerfall oder für verschiedene Missionsphasen rekonfiguriert. Dabei können Applikationen von einem Prozessor/FPGA auf einen anderen verschoben werden. Auch der Wechsel der Plattform, beispielsweise von einem FPGA zu einem Mikrocontroller, soll möglich sein (sogenanntes Task Morphing). Ein typischer Testaufbau von OBC-NG zur Simulation des Ausfalls einer Kamera ist in Abbildung 58 dargestellt.

Wir sind in diesem Projekt für die System-Architektur mitverantwortlich und erstellen die System-Software. Auf Basis des Tasking Framework entwickeln wir eine Programmierschnittstelle und eine Middleware, die Kommunikation und Rekonfiguration in einem verteilten System unterstützt. Ende 2015 soll ein erster Prototyp zur Verfügung stehen, bestehend aus vier Rechenknoten, einem SpaceWire-Netzwerk und System- und Anwendungs-Software.

Ausblick
Für verteilte Systeme ist der Trend erkennbar, dass Betriebssysteme immer mehr an Bedeutung verlieren und Funktionalität in Middlewares verlagert werden. Wir werden uns deshalb in Zukunft auf die Middleware konzentrieren. In **OBC-NG** entwickeln wir beispielsweise die Middleware auf Basis des Tasking Framework für Linux und RODOS. So entstehen Applikationen, die über Betriebssystem-Grenzen hinweg kommunizieren können.

Nach einer erfolgreichen Demonstration von OBC-NG in den nächsten Jahren lässt sich das Konzept auf Schwärme von Satelliten oder Explorationsrobotern erweitern (Arbeitstitel „SpaceCloud"). So könnte ein lunarer Lander seine Rechenleistung für einen Schwarm von Krabblern zur Verfügung stellen. Oder ein Satellitenschwarm verteilt die Rechenlast auf mehrere Satelliten.

Modellbasierte Software-Entwicklung

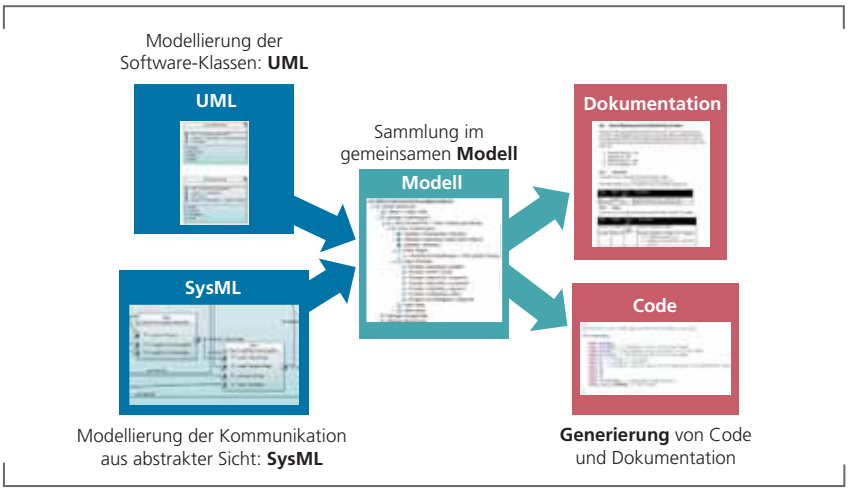
Das Forschungsgebiet „modellbasiertes Software-Engineering“ haben wir in den vergangenen Jahren neu aufgenommen. Dabei werden formale Modelle für Teile der Software erstellt. Aus diesen wird der Quellcode und ggf. Tests generiert.

Ein Beispiel ist die Flug-Software (Entry, Descend, and Landing) des Mars Science Laboratory („Curiosity“). Sie besteht aus ca. 3,5 Millionen Zeilen C-Code. Davon wurden ca. 2,5 Millionen Zeilen automatisch generiert und etwa 1 Million Zeilen von Hand geschrieben. Dieses prominente Beispiel zeigt, wie wichtig Code-Generierung und damit modellbasierte Ansätze im Raumfahrtbereich inzwischen sind.

Wir setzen modellbasierte Ansätze zur Code-Generierung derzeit in zwei Projekten ein. In **ATON** arbeiten wir aktuell an einem UML/SysML-Modell (Unified Modeling Language / Systems Modeling Language). Aus diesem Modell wird unter anderem die Kommunikation zwischen einzelnen Komponenten modelliert, z. B. zwischen Kratererkennung und Navigationsfilter. Daraus wird dann der C++-Code für das Tasking Framework generiert (Abbildung 59).

Dieser Ansatz hilft Fehler in der Flug-Software zu vermeiden. Daneben werden Teile der Dokumentation automatisch aus dem Modell generiert. Dieses Vorgehen reduziert den Dokumentationsaufwand für die Entwickler.

Abbildung 59: Schematischer Ablauf der Code-Generierung im Projekt ATON.



Eine andere Modellierungsstrategie verfolgen wir bei der Software-Entwicklung für das MAIUS-Experiment (Materiewellen Interferometrie unter Schwerelosigkeit). Die MAIUS-Mission führt quantenoptische Experimente mit ultrakalten Atomen in Schwerelosigkeit auf einer Höhenforschungsrakete durch (Start 2015). Wir entwickeln die Experiment-Kontroll-Software inklusive Command & Data Handling. Die im Experiment verwendete Hardware wurde für diesen Zweck maßgefertigt und benötigt spezielle Treiber. Für die Umsetzung dieser Treiber haben wir ein Modell entwickelt, das die Schnittstellen zu den Hardwarekomponenten beschreibt.

Zur Modell-Erstellung setzen wir domänenspezifische Sprachen (Domain Specific Language – DSL) ein. Für MAIUS haben wir vier unterschiedliche DSLs entwickelt, die aufeinander aufbauen. Die erste DSL ist speziell für Experiment-Kontroll-Boards auf der untersten Ebene vorgesehen (Abbildung 60). Diese Boards bestehen aus Frequenzgeneratoren, Stromtreibern, Sensoren etc. Sie sind zu Stacks zusammengesteckt, die über eine zweite DSL beschrieben werden. Die Experiment-Sequenzen – unterteilt in wiederverwendbare Subsequenzen – werden ebenfalls mit zwei verschiedenen DSLs programmiert.

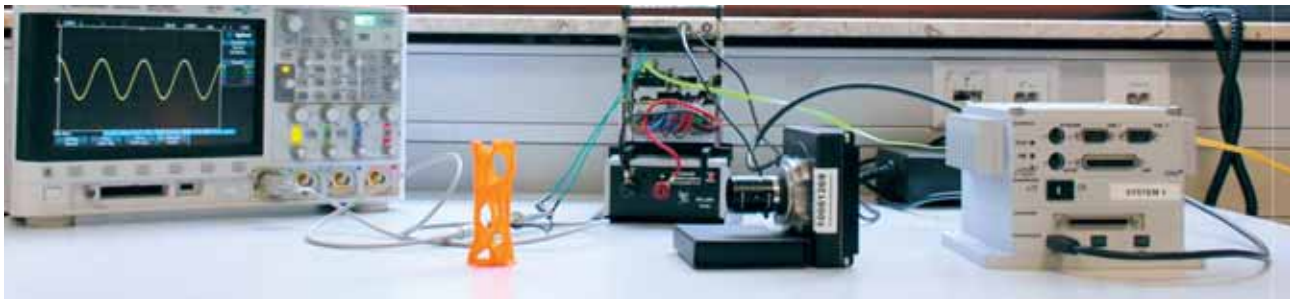
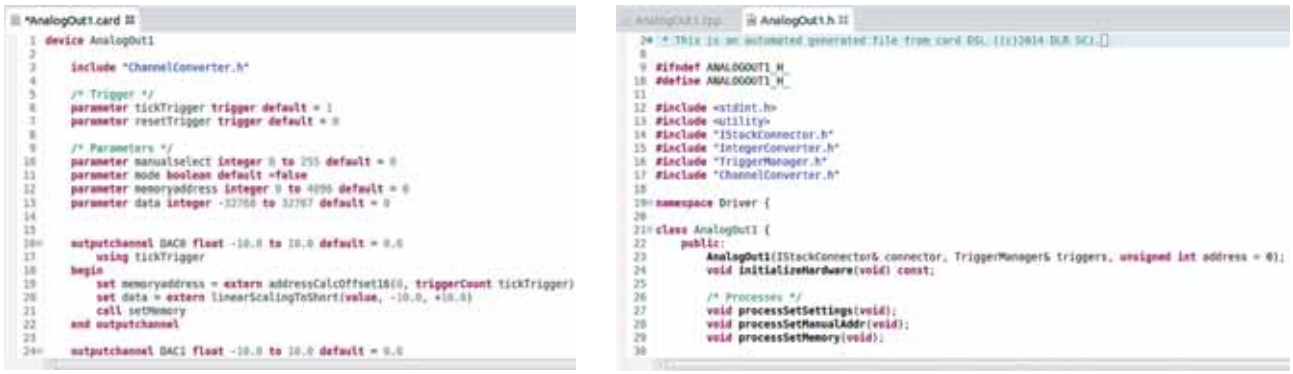


Abbildung 60: Demonstrationsaufbau des MAIUS-Experiments. Test von Kamera und Experimente-Steuerungs-Hardware.

Die textuellen, formalen Beschreibungen in den einzelnen DSLs werden zunächst in ein baumartiges Modell konvertiert und anschließend mit einem Code-Generator in ausführbaren C++-Code überführt (Abbildung 61). Durch die hohe Zahl verschiedener Experiment-Kontroll-Boards hat dieses Vorgehen zu einer deutlichen Zeitersparnis und erhöhten Zuverlässigkeit der Software-Entwicklung geführt. Derzeit liegt der Anteil der generierten Quellcode-Zeilen bei über 70 Prozent.



```
*AnalogOut1.card
device AnalogOut1

    include "ChannelConverter.h"

    /* Trigger */
    parameter tickTrigger trigger default = 1
    parameter resetTrigger trigger default = 0

    /* Parameters */
    parameter manualselect integer 0 to 255 default = 0
    parameter mode boolean default =false
    parameter memoryaddress integer 0 to 4096 default = 0
    parameter data integer -32768 to 32767 default = 0


    outputchannel DAC0 float -10.0 to 10.0 default = 0.0
        using tickTrigger
    begin
        set memoryaddress = extern addressCalcOffset16(0, triggerCount tickTrigger)
        set data = extern linearScalingToShort(value, -10.0, +10.0)
        call setMemory
    end outputchannel

    outputchannel DAC1 float -10.0 to 10.0 default = 0.0
```

```
AnalogOut1.cpp
AnalogOut1.h
/* This is an automated generated file from card DSL (c)2014 DLR SCI.

#ifndef ANALOGOUT1_H
#define ANALOGOUT1_H

#include <stdint.h>
#include <utility>
#include "IStackConnector.h"
#include "IntegerConverter.h"
#include "TriggerManager.h"
#include "ChannelConverter.h"

namespace Driver {

class AnalogOut1 {
    public:
        AnalogOut1(IStackConnector& connector, TriggerManager& triggers, unsigned int address = 0);
        void initializeHardware(void) const;

        /* Processes */
        void processSetSettings(void);
        void processSetManualAddr(void);
        void processSetMemory(void);
```

Abbildung 61: DSL-Beschreibung eines Analog-Output-Boards (links) und der daraus generierte C++-Quellcode (rechts).

Ausblick

In Zukunft möchten wir den Anteil des generierten Codes in unseren Projekten erhöhen. Dabei werden wir uns nicht auf eine Modellierungssprache wie UML oder DSL festlegen, sondern die passende Sprache für den jeweiligen Anwendungszweck wählen. Aktuell entwickeln wir Ansätze zur automatischen Code-Generierung für Teile einer Lageregelungs-Software. Dabei haben wir uns wiederum für eine DSL entschieden.

Aber nicht nur der Treiber-Code lässt sich automatisch erzeugen. Für die Nachfolgemissionen von **MAIUS** (MAIUS-2 und MAIUS-3) sollen aus den Hardware-Beschreibungen gleichzeitig die passenden Software-Simulationen der Hardware generiert werden. Damit können wir die automatisch erzeugten Hardware-Treiber gleich kontinuierlich mit automatischen Tests verifizieren.

Zuverlässigkeit von Flug-Software

Flug-Software sollte jederzeit zuverlässig arbeiten. Eine grobe Schätzung besagt, dass 30 Prozent der Entwicklungszeit einer Onboard-Software für Tests aufgewendet wird.

Um zuverlässige Software zu entwickeln, verwenden wir Vorgehensweisen aus dem agilen Software-Engineering. Wir entwickeln Software testgetrieben: Im Idealfall werden Tests für ein Software-Modul bereits vor dem eigentlichen Modul geschrieben. So erhält man nebenbei auch noch eine sehr gute Spezifikation des Verhaltens dieses Moduls.

Darüber hinaus nutzen wir Werkzeuge zur ständigen Qualitätskontrolle der entwickelten Software. Es werden täglich automatische Builds samt Tests ausgeführt, statische und dynamische Code-Analysen durchgeführt sowie Statistiken über die Testabdeckungen erzeugt.

Treiber-Tests
Im Bereich der eingebetteten Systeme wird sehr viel Software entwickelt, die direkt mit Hardware interagiert. Es ist schwierig, diese Software automatisch zu testen. Die spezielle Hardware steht oft nicht zur Verfügung oder ist schwer in ein automatisches Test-Setup einzubinden.

Für automatische Tests einer Kamera haben wir ein Verfahren entwickelt, um in einem abgeschlossenen Raum reproduzierbare Bilder zu erzeugen. Anschließend vergleicht eine Software das Bild mit einem Referenzbild. Damit lässt sich kontinuierlich testen, ob die Treiber für die Kamera zuverlässig funktionieren.

Steht die Hardware gar nicht zur Verfügung, behilft man sich mit Simulationsmodellen der Hardware. Im Projekt **ATON** wurde in der ersten Phase ausschließlich mit simulierten Sensoren gearbeitet.

Ausblick
Die Validierung und die Verifikation unserer Flug-Software haben einen hohen Stellenwert. Der Fokus liegt auf der

- Automatisierung von Tests,
- automatischen Generierung von Testfällen,
- Simulation von Hardware,
- Einbindung von zu testender Hardware in einen automatisch ablaufenden Prozess.

Aber auch formale Methoden werden in den kommenden Jahren einen Schwerpunkt bilden. Beispiele sind das Model Checking sowie statische und dynamische Code-Analysen.

Fazit

Die Entwicklung von Onboard-Software zeigt in den unterschiedlichsten Bereichen gute Ergebnisse. Die Arbeitsgruppe verfügt über ein breitgefächertes Portfolio, von der Lageregelung und Navigation über das Command & Data Handling bis hin zu Experiment-Steuerung. Der Einsatz wiederverwendbarer, robuster Software-Komponenten macht sich bezahlt.

Großes Potential hat das Projekt **OBC-NG.** Dank seiner verteilten, rekonfigurierbaren Rechnerarchitektur könnte es international Beachtung finden. Es bestehen gute Chancen, dass sich das Konzept nach 2015 in einer Flugkampagne bewähren kann.

In unseren aktuellen Projekten hat es sich als sehr nützlich erwiesen, Code-Generatoren bei der modellbasierten Software-Entwicklung einzusetzen. Daher wollen wir unser Engagement auf diesem Gebiet weiter ausbauen.

Insbesondere in diesem Bereich kooperieren wir eng mit der Arbeitsgruppe Modellierung und Simulation. Das langfristige Ziel ist es, den gesamten Raumfahrt-Entwicklungszyklus mit modellbasierten Ansätzen abzudecken.

Wissenschaftliche Visualisierung

Visualisierung spielt in fast allen Bereichen der Technik und Wissenschaft eine große Rolle. In den meisten Publikationen oder Vorträgen findet man Graphiken oder andere Abbildungen, um Ergebnisse und Daten zu veranschaulichen. Daten, oft große Datenmengen, entstehen in der Forschung und Entwicklung in allen DLR-Schwerpunkten. Dies sind Daten aus Simulationen (z. B. zur Optimierung von Triebwerksauslegungen), aus Experimenten (z. B. Windkanalversuche) oder aus Beobachtungen (z. B. Wetterdaten aus Satellitenbildern). Visualisierung bietet oft den einzigen Zugang zu den Daten. So unterschiedlich wie die Anwendungen sind allerdings auch die Anforderungen an die Visualisierung. Die Arbeitsgruppe „Wissenschaftliche Visualisierung" entwickelt Analyse- und Darstellungsmethoden, die einen intuitiven Zugang zu den Daten ermöglichen. Eine enge Zusammenarbeit mit den Wissenschaftlern garantiert die Entwicklung effektiver Visualisierungsmethoden. Die Gruppe widmet sich insbesondere zwei zentralen Themen: Visualisierung und Analyse großer Datenmengen, und Merkmalsextraktion aus wissenschaftlichen Daten.

Analyse großer Datenmengen

Die Analyse und die Visualisierung großer Datenmengen sind ein Schwerpunkt der Gruppe „Wissenschaftliche Visualisierung". Die zunehmende Rechenkapazität der HPC-Hardware-Systeme ermöglicht es Wissenschaftlern, physikalische Phänomene und Prozesse mit enormer räumlicher und zeitlicher Genauigkeit zu simulieren. Dadurch entstehen zunehmend große Datenmengen (siehe Tabelle 1). Die Größe der Datensätze ist in vielerlei Hinsicht eine Herausforderung. Eine effektive Auswertung und Visualisierung ist in der Regel nur mit hohem Zeitaufwand möglich.

Das Ziel ist, Methoden der wissenschaftlichen Visualisierung als ein nützliches Werkzeug in den täglichen Arbeitsablauf für Anwender im DLR zu integrieren. Wir erforschen und entwickeln neue Konzepte. Anschließend implementieren wir diese im Rahmen einer umfangreichen Software-Infrastruktur, welche den DLR-Anwendern zur Verfügung gestellt wird. Der System-Prototyp wurde bereits bei einigen Kooperationspartnern im DLR erfolgreich installiert.

Tabelle 1: Entwicklung der Größe von numerischen Simulationen.

	2010	2018	Faktor
Systemleistung	2 PFlop/s	1 EFlop/s	500
Energieverbrauch	6 MW	20 MW	3
Speicher	0,2 PB	10 PB	33
Prozessoren pro Knoten	12 CPUs	1000 CPUs	83
Systemgröße	20K Knoten	1M Knoten	50
I/O-Bandbreite	0,2 TB/s	20 TB/s	100

Durch die Zusammenarbeit mit dem Institut für Antriebstechnik und dem Institut für Planetenforschung stehen zudem große, hochwertige Datensätze zur Erprobung des Gesamtsystems zur Verfügung. Der in Abbildung 62 verwendete Strömungsdatensatz ist mehrere Terabytes groß. Mit herkömmlichen Methoden benötigt die Berechnung einer solchen Darstellung Stunden. Sie kann nun in wenigen Sekunden durchgeführt werden.

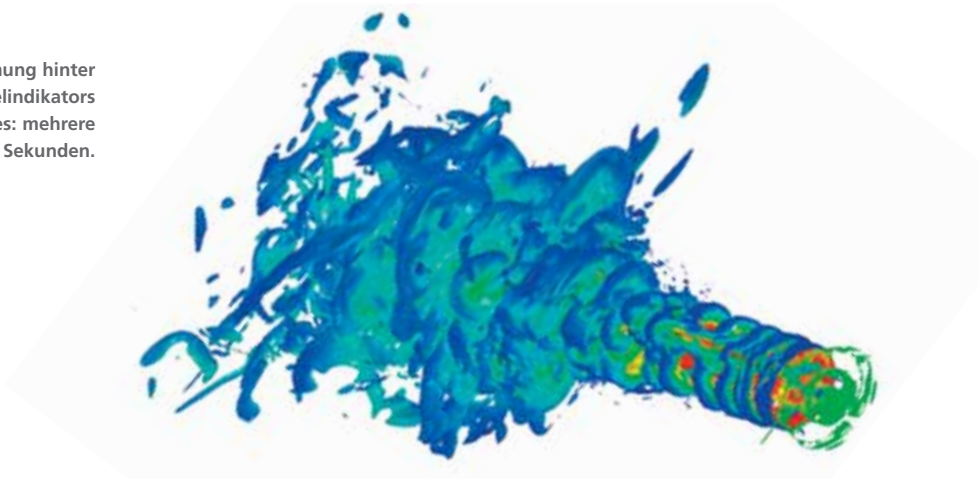


Abbildung 62: Turbulente Strömung hinter einer Turbine – Isofläche des Wirbelindikators Lambda-2. Größe des Datensatzes: mehrere Terabyte. Bildberechnung: wenige Sekunden.

Die Softwaregrundlage für die Implementierung bildet das ViSTA-Framework. Entworfen wurde es von der RWTH Aachen. Seit einigen Jahren entwickeln wir es gemeinsam mit der RWTH weiter. Dabei zielt ViSTA vor allem auf den Einsatz in 3D-Arbeitsumgebungen unter Verwendung von Techniken der virtuellen Realität (VR). Zusätzlich wurde das Visualisierungssystem ParaView angebunden, das vielen Anwendern gut vertraut ist. So ist auch eine Integration des Systems an Desktop-Arbeitsplätzen möglich. Folgende Komponenten haben die enorme Beschleunigung des Visualisierungsprozesses ermöglicht:

- **Verteiltes Post-Processing:** Entwicklung einer verteilten Software-Infrastruktur für die parallele Verarbeitung großer Datenmengen.
- **Hybrides Rendering:** Optimierung der Interaktivität durch die Verteilung der Rendering-Lasten.
- **In-Situ-Visualisierung:** Online-Visualisierung laufender Simulationen.

Diese Themen werden im Folgenden genauer ausgeführt.

Verteiltes Post-Processing
In den meisten Anwendungen ist eine Darstellung der gesamten Daten nicht mehr möglich und oft auch nicht wünschenswert. Um den Wissenschaftler nicht mit Information zu überladen, durchlaufen die Daten eine Reihe von Nachbearbeitungsschritten (Post-Processing). Semantische Filter extrahieren dabei die wesentliche Information aus den Rohdaten. Herkömmliche Methoden erwarten allerdings häufig, dass die Daten vollständig in lokalen Speichern gehalten und lokal prozessiert werden können. Für Simulationen, die schon heute bis in den Terabyte-Bereich gehen, ist diese Vorrausetzung nicht mehr gegeben. Daher erforscht und entwickelt die Arbeitsgruppe eine beliebig skalierbare Software-Infrastruktur, die die anfallenden Arbeitsschritte auf die verfügbaren Ressourcen verteilt (Abbildung 63). Da hierbei die rechenintensive Datenfilterung auf große HPC-Systeme ausgelagert wird, müssen auf dem Frontend-Rechner nur noch die erheblich reduzierten Extraktionsdaten für die interaktive Exploration verarbeitet werden. Der Prozess besteht aus drei Schritten: Merkmalsextraktion, Rendering, Visualisierung. Diese werden wir im Folgenden erläutern.

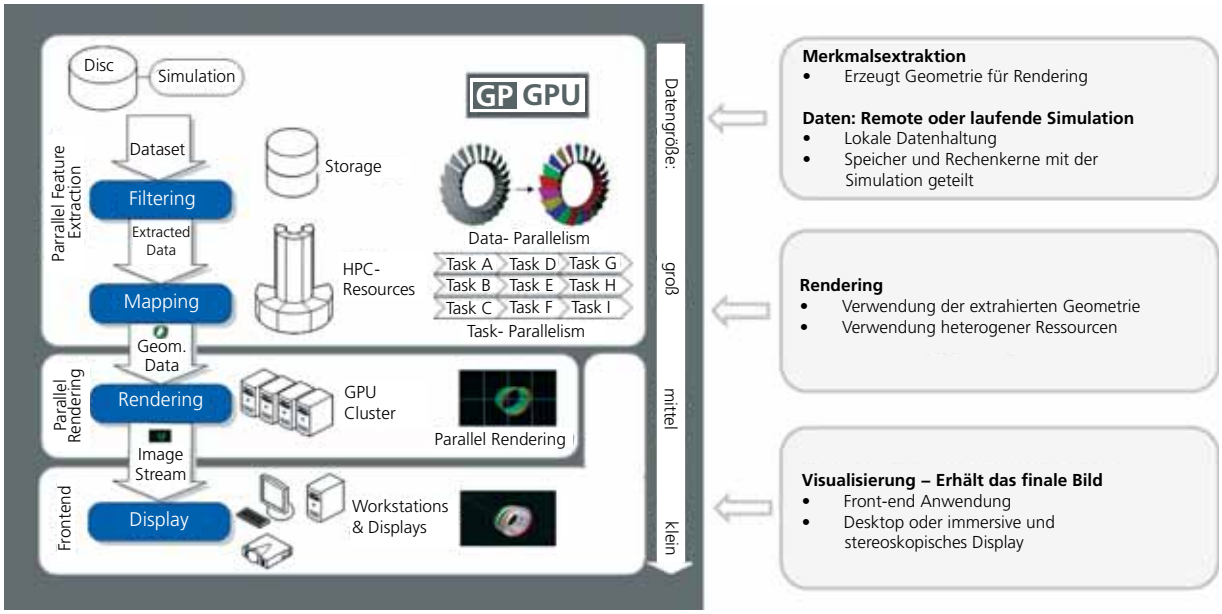


Abbildung 63: Verteilung der Visualisierungs-Pipeline für die Analyse und Darstellung großer Daten.

1. Merkmalsextraktion – Datenfilterung: Der erste Schritt in der Pipeline ist das Filtern der Rohdaten, wie z. B. die Extraktion einer Isofläche oder eines Teilvolumens. Dies ist der rechen- und datenintensivste Schritt der Pipeline. Er wird direkt auf dem HPC-System ausgeführt. Solche Systeme ermöglichen einen sehr schnellen Zugriff auf die Originaldaten, die nach der Simulation auf File-Servern abgespeichert wurden. Die eingelesenen Daten werden gleichmäßig auf viele Prozessoren verteilt, was eine parallele Bearbeitung erlaubt. Scheduling-Methoden weisen die Datenfragmente dynamisch den Prozessoren zu. Dabei kommen entsprechend der Filtermethode und des Datenformats unterschiedliche Heuristiken zum Einsatz.

2. Rendering der gefilterten Daten: Das Ergebnis der Datenfilterung sind im Allgemeinen Geometriedaten, z. B. die Dreiecke einer Isofläche. Im Vergleich zu den Rohdaten ist die Datenmenge deutlich reduziert. Sie liegt jedoch meist noch immer über den Kapazitäten eines einzelnen Desktop-Systems. Durch den Einsatz paralleler Rendering-Techniken kann dennoch eine flüssige Darstellung erreicht werden. Dabei werden die Daten als nächstes von einem mit mehreren Graphikkarten bestückten Render-Cluster verarbeitet. Jeder Grafikprozessor (GPU) erhält dabei einen Teil der Gesamtgeometrie. Für die Aufteilung der Geometrie gibt es verschiedene Strategien. Bewährt hat sich das so genannte „Sort Last Rendering", das bezüglich der Datengröße gut skaliert. Das Ergebnis sind mehrere Teilbilder, die zusammengefügt das endgültige Bild ergeben.

3. Visualisierung: Im letzten Schritt wird das finale Bild auf dem Display ausgegeben.

Hybrides Rendering
Die direkte Interaktion mit den Daten ermöglicht eine intuitive Exploration der Simulationsergebnisse. Dies kann eine Änderung der Perspektive oder die Wahl einer anderen Visualisierungsmethode sein. Solche Änderungen bewirken kontinuierlich neue Anfragen an das Post-Processing-System. Interaktive Systeme verlangen die Visualisierung von Änderungen in Echtzeit (Bildwiederholungsraten von mindestens 30 Hz). Da wird die Render-Leistung der Grafikkarte schnell zum Flaschenhals. Sollen dann auch noch hochauflösende Displays angesteuert werden, ist dies selbst bei der Verwendung von Render-Clustern eine Herausforderung. Das Problem ist dann nicht mehr die Render-Leistung, sondern die Latenz und Bandbreite von Netzwerken zur Übertragung der fertigen Bilder an das Front-end. Auch eine sehr hohe Bildwiederholrate auf dem GPU-Cluster kann nicht verhindern, dass Änderungen am Visualisierungs-Setup verzögert angezeigt werden.

Die Idee von Hybrid-Rendering-Methoden ist es daher, die Darstellung in zwei Komponenten zu zerlegen: Die Simulationsdaten und die Kontext-Geometrie, z. B. die Schaufeln eines Verdichters oder der Rumpf eines Flugzeuges. Die Kontext-Geometrien sind im Allgemeinen nicht sehr groß und können auf dem lokalen System interaktiv gerendert werden. Diese Kontextvisualisierung ist die Basis für die Nutzerinteraktion und vermittelt den Eindruck einer flüssigen Visualisierung. Die großen, oft zeitabhängigen Simulationsdaten werden dagegen auf entfernten Ressourcen gerendert und dann in die lokale Darstellung integriert (Abbildung 64 a-c). Aufgrund der Verzögerungen bei der Bildübertragung kann es allerdings passieren, dass die Bilder kurzzeitig nicht zusammenpassen (Abbildung 64 d). Die Arbeitsgruppe hat daher neue Bildkompositions- und Blending-Algorithmen entwickelt, die diese negativen Nebeneffekte verringern (siehe Abbildung 64 e und f).

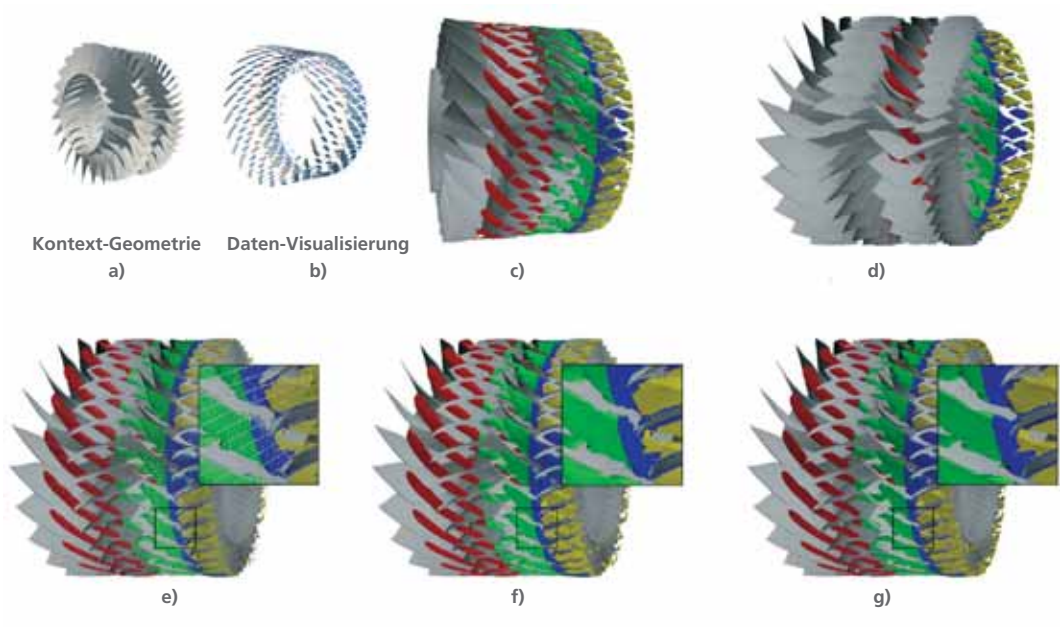


Abbildung 64: Rendering einer Strömung in einem Verdichter. (a) Die Kontext-Geometrie der Turbine ändert sich im Laufe der Simulation nicht und kann daher lokal gerendert werden. (b) Die zeitabhängigen Strömungsdaten werden dagegen auf einem entfernten Cluster visualisiert. (c) Zusammengefügte Geometrie und Visualisierung passen zusammen. (d) Der Benutzer rotiert die lokale Geometrie, was zu einem Offset der beiden Bilder führt. (e) Eine lokale Transformation fügt die Bilder wieder zusammen. (f) Zusätzliche Anpassung der Visualisierung nach der Transformation führt zu einer guten Approximation des richtigen Bildes. (g) Korrekt gerendertes Bild im Vergleich.

In-Situ-Visualisierung
Eine wissenschaftliche Simulation ist ein aufwendiger Prozess. Die Berechnung kann Tage bis Wochen dauern. Dabei hängt die Spezifikation einer Simulation von vielen Parametern ab. Ist das Simulationsergebnis nicht zufriedenstellend, müssen Parameterwerte nachträglich modifiziert werden. Dann beginnt die gesamte Simulationsrechnung von vorn. Glücklicherweise beruhen die meisten wissenschaftlichen Simulationsberechnungen auf iterativen Algorithmen, die sich in viele Schritte unterteilen lassen. Frühzeitige Einblicke in Teilergebnisse können daher Fehler in der Simulationskonfiguration aufdecken, ohne dass das Endergebnis abgewartet werden muss. Ein Abbruch und Neustart des Simulationslaufs mit modifizierten Parametern ist dann möglich. Die Analyse und Visualisierung schon während der laufenden Simulation wird als In-Situ-Visualisierung bezeichnet.

Im EU-Projekt CRESTA haben wir Methoden entwickelt, um Analyse- und Visualisierungsprozesse in die Simulation zu integrieren. Die hier verwendete Visualisierungs-Architektur ähnelt in ihrer Grundidee der Post-Processing-Architektur.

Der wesentliche Unterschied ist, dass die Daten nun direkt aus der Simulation abgegriffen werden und die Visualisierung sich die HPC-Ressourcen mit der Simulation teilen muss. Die Integration führt somit zu neuen Herausforderungen: Gelöst werden müssen die Synchronisation des Datenzugriffs und das Ressourcen-Management. Ein Vorteil ist der schnelle Datenzugriff über den Hauptspeicher anstatt der langsamen Festplatten-Zugriffe.

Wir evaluieren die neu entwickelten In-Situ-Methoden in Kooperation mit dem University College London (UCL) in einem Prototyp. Der dabei betrachtete Anwendungsfall behandelt die Blutfluss-Simulation in Aneurysmen unter Verwendung des Simulationscodes HemeLB (Abbildung 65, Abbildung 66).

Abbildung 65: Partitionierung des Netzes für eine Blutflusssimulation mit HemeLB.

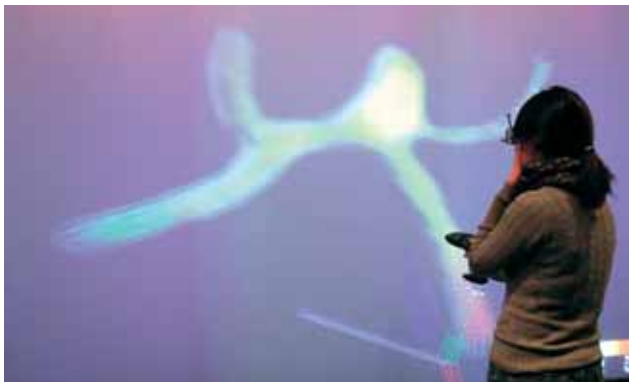


Abbildung 66: Eine Wissenschaftlerin untersucht den Blutfluss in einem Aneurysma während der laufenden HemeLB-Simulation.

Merkmalsbasierte Datenanalyse und Visualisierung

Das zweite Schwerpunkt-Thema der Arbeitsgruppe „Wissenschaftliche Visualisierung“ ist die Extraktion von Merkmalen und Strukturen aus Daten. Diese werden im Zusammenhang mit Visualisierung häufig auch „Features“ genannt. Dabei steht die Komplexität und weniger die Größe der Daten im Vordergrund. Das Ziel ist die Reduktion der Visualisierung auf bedeutungstragende Features, um den Zugang zu den Daten zu erleichtern. Diese Arbeiten stehen in engem Zusammenhang mit der Arbeitsgruppe „3D-Interaktion“, die die Ergebnisse in interaktiven Umgebungen integriert.

Die Spezifikation relevanter Features ist bereits die erste Herausforderung. Der nächste Schritt ist dann die Entwicklung effizienter und robuster Extraktionsalgorithmen. Anwendungen, mit denen sich die Gruppe beschäftigt hat, umfassen Simulationsdaten für Strömungen, Materialbeanspruchungen, Energieflüsse in technischen Systemen, sowie Klima- und Beobachtungsdaten auf Erde und Mars. Im Folgenden stellen wir die wichtigsten Projekte kurz vor: Klima-Visualisierung, Visualisierung von Energieflüssen in Zugsystemen, Urbane Sicherheit, Tensor- und Strömungsvisualisierung.

Klimadaten-Visualisierung
Vulkanausbrüche haben große Auswirkungen auf die Atmosphäre, das Klima sowie den Luftverkehr. Um diese besser zu verstehen, nutzt man Beobachtungsdaten sowie Simulationsergebnisse (siehe Abbildung 67):

- Bilddaten aus Satellitenbeobachtungen,
- punktuelle Messdaten von Satelliten,
- Partikelsimulationen.

Abbildung 67: Visualisierung der Aschewolken nach einem Vulkanausbruch: Diese Abbildung zeigt eine direkte Visualisierung der Daten. (a) Tropopause der Erde, (b) Partikelbahnen aus einer Simulation, (c) Sattelitenbilddaten der Aschewolke, (d) Detektionen von Asche- und Sulfat-Partikeln.

Auflösung und Abdeckung der einzelnen Datenquellen sind sehr unterschiedlich. Um dennoch die Daten im Kontext der anderen Daten untersuchen zu können, haben wir ein exploratives Framework entwickelt. Alle Methoden sind für die direkte Interaktion mit den Daten ausgelegt. Ziele waren dabei: die Erstellung eines Gesamtbildes und ein detailliertes Verständnis der verschiedenen Datenquellen.

1. Integriertes Gesamtbild: Verschiedene Perspektiven auf die Situation unterstützen den Fachexperten dabei, die einzelnen Aspekte der verschiedenen Datenquellen zu einem vollständigen Bild zusammenzufügen. Um die 4D-Daten (Raum + Zeit) erfassen zu können, stehen verschiedene Kontext- und Bezugssysteme zur Verfügung. Der Nutzer kann zwischen den verschiedenen Darstellungen wechseln.

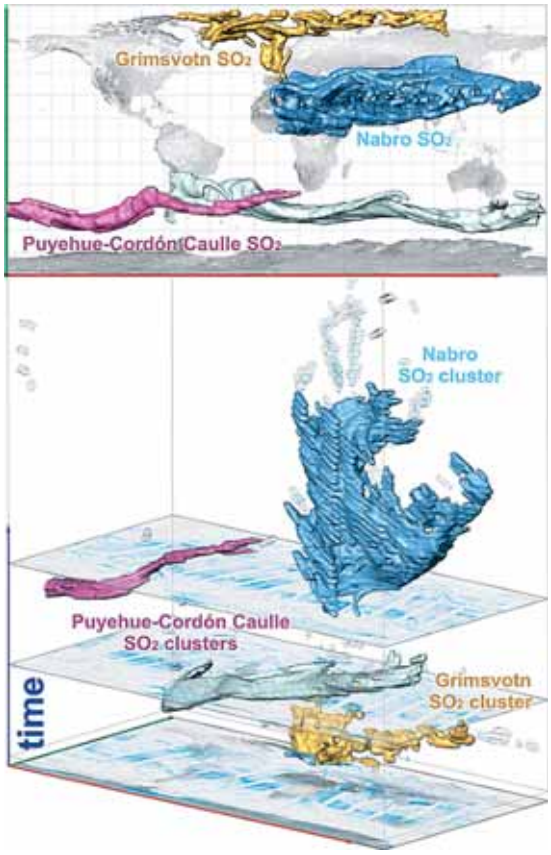


Abbildung 68: Interpolation und Clustering ermöglicht die Trennung der einzelnen Wolken und deren Verfolgung über die Zeit.

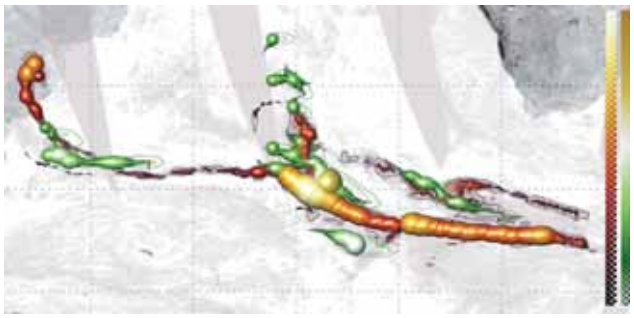


Abbildung 69: Vergleich der beiden satelliten-basierten Daten mit Hilfe des topologischen Skeletts der jeweiligen Aschewolken.

Abbildung 68 zeigt ein Beispiel, in dem die Zeit als dritte Dimension über einer Kartenansicht dargestellt ist. Für die Visualisierung wurde hier eine Clustering-Methode umgesetzt.

2. Detailliertes Verständnis der einzelnen Datenquellen: Jede einzelne Datenquelle hat ihre eigenen Qualitäten, aber auch Grenzen. Ein gezielter Vergleich der Daten illustriert die Zuverlässigkeit der einzelnen Quellen, so z. B. der Simulationsergebnisse. Die Wolken können auf Basis eines „Wolkenskeletts" (Extremalstrukturen) zeitlich verfolgt und verglichen werden (Abbildung 69).

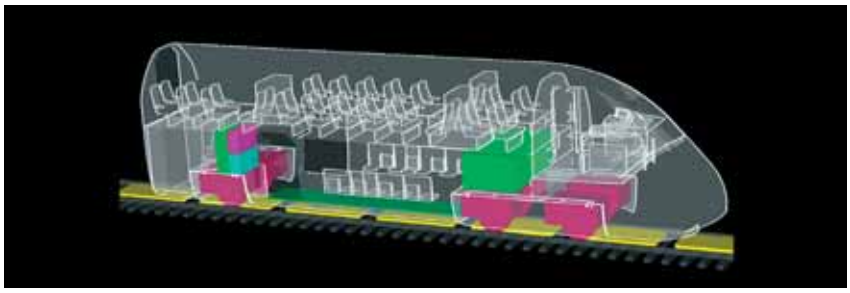
Dieses Projekt ist im Rahmen des „IEEE Vis-Contest 2014" entstanden. Die Daten sind öffentlich und wurden vom Forschungszentrum Jülich zur Verfügung gestellt. Die Entwicklung der Visualisierungsmethoden basiert auf Diskussionen mit Experten aus dem DLR-Institut für Physik der Atmosphäre. Basierend auf dieser ersten Kooperation sind weitere gemeinsame Arbeiten geplant.

„Next Generation Train" – Visualisierung von Energieflüssen

Neun Forschungsinstitute sind am DLR-Projekt "Next Generation Train (NGT)" beteiligt. In diesem Projekt sollen die Bedingungen für einen Hochgeschwindigkeitszug der Zukunft untersucht werden. Ein Teilprojekt beschäftigt sich mit dem Energiemanagement der Züge (Hauptverantwortlich: Institut für Fahrzeugkonzepte). Das Ziel ist, den Energieverbrauch zu senken und ein teilelektrifiziertes Zug-System zu erproben. Die Basis dazu bildet die Simulation der Energieflüsse des gesamten Systems. Die Simulationen basieren auf einer Vielzahl von Parametern und erzeugen hunderte von Ausgabevariablen.

Unsere Aufgabe in diesem Teilprojekt ist es, ein Visualisierungsmodul zur Steuerung und Analyse in die Simulation zu integrieren. Das Projekt ist auf einen längeren Zeitraum angelegt; erste Schritte zur Visualisierung der Energieflüsse wurden aber bereits umgesetzt. Ein Schnappschuss aus der Visualisierung ist in Abbildung 70 dargestellt. Die einzelnen dargestellten Komponenten mit aktuellem Energieverbrauch, Energiereserven sowie Energieflüssen sind frei konfigurierbar. Die Vielzahl der unterschiedlichen Softwarekomponenten, die in das Simulationssystem integriert werden müssen, ist die größte Herausforderung in diesem Projekt.

Abbildung 70: Energieflussvisualisierung eines Zuges. Die Kontextgeometrie ist als transparente Hülle dargestellt. Die eingefärbten Teile sind Komponenten, die am Energiefluss beteiligt sind.



MoSec – Zivile Sicherheit

Für den Katastrophenschutz ist es wichtig, schnelle und effiziente Destaster-Management-Systeme zur Hand zu haben – zum Beispiel im Falle einer Katastrophe durch Feuer, Überschwemmung oder Erdbeben. Im Notfall ist es zwingend notwendig, Zugang zu Stadtplänen und Karten zu haben, die sowohl eine Übersicht der Region als auch detaillierte Informationen über einzelne Gebäude beinhalten. Zugleich müssen aktuelle Informationen über die Lage vor Ort integriert werden können. Aktuelle Krisenmanagement-Tools erlauben bereits die Integration dieser Information, basieren allerdings häufig auf digitalen 2D-Karten. Detaillierte 3D-Darstellungen stehen im Allgemeinen nicht zur Verfügung.

"Mobile Worlds – Civil Security" (MoSec) ist ein gemeinsames Forschungsprojekt unserer Einrichtung und des DLR-Instituts für Optische Sensorsysteme. In einem ersten Schritt sollen hochaufgelöste 3D-Beobachtungsdaten aus Überflug-Kampagnen interaktiv visualisiert und in immersive virtuelle Arbeitsumgebungen integriert werden. Basierend auf diesem Prototyp soll anschließend ein verteiltes Krisen- und Desaster-Management-System entwickelt werden. Dies ermöglicht einen genauen Überblick über die aktuelle Situation und unterstützt eine effiziente Einsatzplanung. Zusätzliche semantische Informationen ergänzen die interaktiven Darstellung der 3D-Karten (siehe Abbildung 71 für ein erstes Ergebnis). Weiter sollen Interaktionen möglich sein, z. B. die Messung der Entfernung zum nächsten Hydranten.

Abbildung 71: Erster Schnappschuss einer Visualisierung eines Stadtmodells (Berlin-Kreuzberg), mit Darstellung der expliziten Hülle eines Gebäudes aus einer City-GML-Datei.

Die Herausforderungen sind vielfältig:

- Verschiedene Datenquellen mit unterschiedlicher Auflösung und Abdeckung müssen fusioniert werden.
- Der Wechsel zwischen Überflugansicht (2,5D) und Straßenansicht (3D) erfordert unterschiedliche Rendering-Strategien.
- Visualisierung und Interaktion sollen nicht nur in VR-Umgebungen sondern auch auf mobilen Geräten am Einsatzort flüssig funktionieren. Dies muss in der Gesamtarchitektur berücksichtigt werden.

Ein Überblick über das System-Konzept ist in Abbildung 72 gegeben.

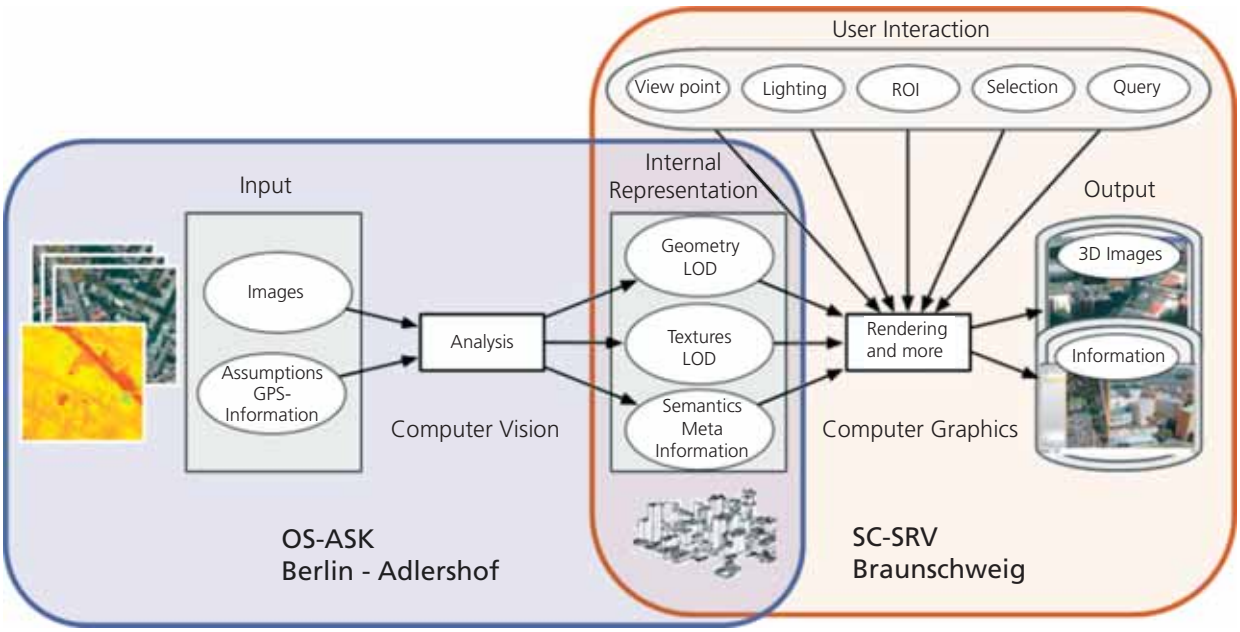


Abbildung 72: Mobile Sicherheit – Konzept zur Entwicklung einer interaktiven Umgebung zur Unterstützung z. B. von Feuerwehrleuten im Einsatz.

Tensorvisualisierung im Produktentwicklungsprozess
Dieses Projekt basiert auf einer engen Zusammenarbeit unserer Visualisierungsforscher mit Maschinenbauern von der Universität des Saarlandes. Es zeigt, wie interdisziplinäre Arbeit zu neuen Erkenntnissen und Fortschritten in beiden Bereichen führen kann. In dieser Zusammenarbeit geht es um einen Schritt in dem Produktentwicklungsprozess: „Entwurf und Auslegung mechanischer Bauteile“. Die Bauteile sollen funktionsfähig sein, die geforderten Qualitätsansprüche erfüllen und mit Standard-Fertigungsverfahren hergestellt werden können.

Die Zusammenarbeit begann mit unspezifischen Zielen und ersten Experimenten mit den vorhandenen Daten und Visualisierungsmethoden. Im Zuge der Zusammenarbeit entstanden viele konkrete Fragen. Es wurde eine Hypothese entwickelt, die die Kraftübertragung in einem Bauteil mit sogenannten Tensorlinien in Zusammenhang bringt. Tensorlinien sind Integrallinien, die den Hauptspannungsrichtungen folgen. Basierend auf dieser Beobachtung haben wir eine Visualisierungsmethode zur Unterstützung der Auslegung der Bauteile entwickelt.

Wir haben die Hypothese in einer Fallstudie überprüft. Abbildung 73 illustriert die einzelnen Schritte. Das Beispiel untersucht die Konstruktion einer Verstärkungsstruktur eines Bremshebels durch eine Kunststoff-Verrippung. Drei neue Hebel-Geometrien wurden auf Grundlage unserer Hypothese entwickelt und sowohl miteinander als auch mit einem Referenzmodell verglichen. Die Validierung umfasst standard-numerische und experimentelle Untersuchungen. In unserem Fall waren alle neuen Strukturen der Referenzgeometrie überlegen. Die Ergebnisse sind sehr vielversprechend und verdeutlichen das Potenzial von Visualisierung in der Produktentwicklung auch für komplexere Szenarien.

Rand-Geometrie: Design-Space

Handskizze basierend auf Tensorlinien-Visualisierung

3D CAD Modell

Virtuelle Evaluierung: Simulation + Tensorlinien-Visualisierung

„Rapid Prototyping" und experimentelle Validierung

Abbildung 73: Tensor-Visualisierung zur Unterstützung der Auslegung von mechanischen Komponenten.

Strömungsanalyse mit Methoden aus der Mustererkennung

Die Analyse von Strömungsdaten ist oft geprägt von der Suche nach charakteristischen Strukturen mit semantischer Bedeutung. Eine Möglichkeit, solche Muster zu definieren, ist, einen menschlichen Beobachter interessante Muster manuell selektieren zu lassen. Die Herausforderung besteht dann darin, ähnliche Strukturen im selben oder in anderen Datensätzen zu finden. Dabei sollen die Muster auf verschiedenen Skalen und mit verschiedenen Orientierungen erkannt werden.

In dieser Arbeit verallgemeinern wir das Konzept invarianter Momente als effektive Deskriptoren für die Muster in Strömungsfeldern. Invariante Momente wurden ursprünglich für die Beschreibung von Objekten und Mustern in der Bilderkennung entwickelt. Die von uns entwickelten Momente sind auch dazu geeignet, Strömungsmerkmale mit unterschiedlichen Geschwindigkeitsprofilen zu erkennen. Das Konzept von Strömungsmustern wird in Abbildung 74 anhand eines 2D-Beispiels verdeutlicht.

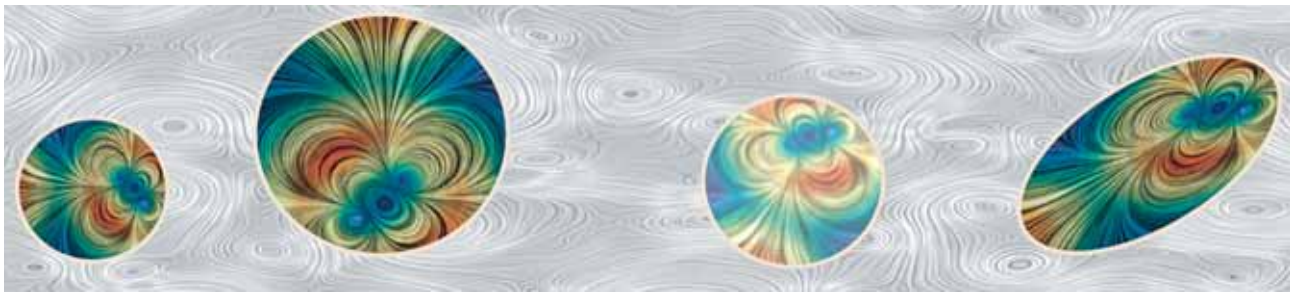


Abbildung 74: Mustererkennung in der Strömungsanalyse.

3D-Interaktion

Virtuelle Realität (VR) ist eine Technologie, die es erlaubt, computer-generierte Welten so zu erleben, als wären sie real. VR vereint Visualisierung, Echtzeit-Simulation und Mensch-System-Interaktion im drei-dimensionalen Raum. Die Anwendungsgebiete sind vielfältig, sie reichen vom Astronauten-Training bis zum Design-Review eines Flugzeugs. Ziel dieser Arbeitsgruppe ist es, effektives und intuitives Arbeiten in VR zu ermöglichen. Wir entwickeln und optimieren Algorithmen zur echtzeitfähigen Simulation und Bilderzeugung. Außerdem erforschen wir innovative Methoden, um intuitiv und effektiv mit der simulierten Welt interagieren zu können. Die Anforderungen dazu leiten sich ab aus Gemeinschaftsprojekten mit DLR-Instituten oder internationalen Partnern.

Virtuelle Montage-Simulation

Der Weltraumschrott nimmt stetig zu. Damit werden Entsorgung, Reparatur und Wartung von Raumfahrtsystemen immer wichtiger. VR-OOS (Virtual Reality On-Orbit Servicing) ist ein Simulator, mit dem man in einer sicheren und kostengünstigen Umgebung die telerobotische Wartung eines Satelliten im Orbit planen und trainieren kann (Abbildung 75). Der Simulator macht es extrem einfach, neuartige Satelliten zu entwickeln, Astronauten für einen Außeneinsatz zu trainieren und die entsprechenden Serviceroboter zu entwickeln und zu steuern.

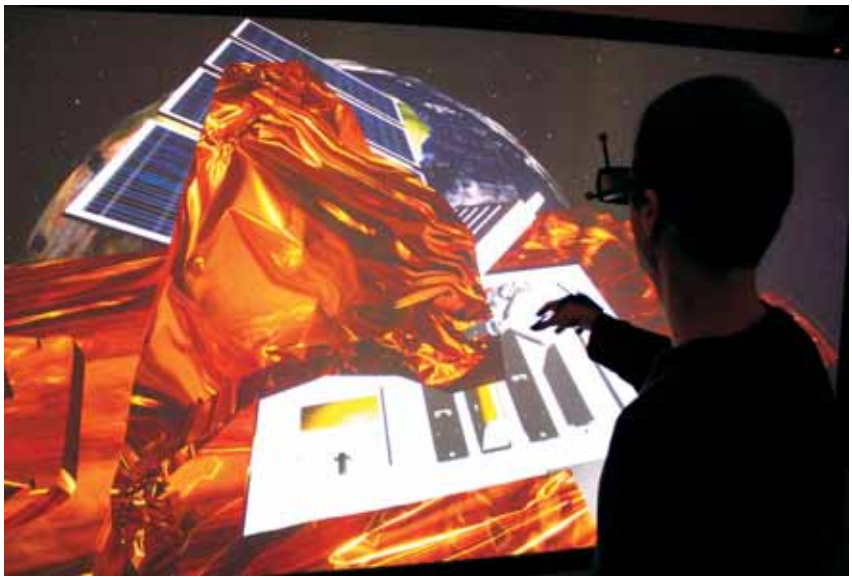


Abbildung 75: Der VR-OOS Simulator auf einer immersiven Projektions-Anlage für die Simulation und das Training von telerobotischen On-Orbit- Servicing-Aufgaben.

Wir entwickeln VR-OOS gemeinsam mit dem DLR-Institut für Robotik und Mechatronik. Unser Partner-Institut übernimmt dabei die Ansteuerung des Service-Roboters sowie die Anbindung der haptischen Geräte. Wir sind verantwortlich für die Physiksimulation, die Visualisierung, die Interaktions-Methoden sowie für die Integration des Gesamtsystems. Die Systemarchitektur mit den Kernkomponenten ist in Abbildung 76 dargestellt.

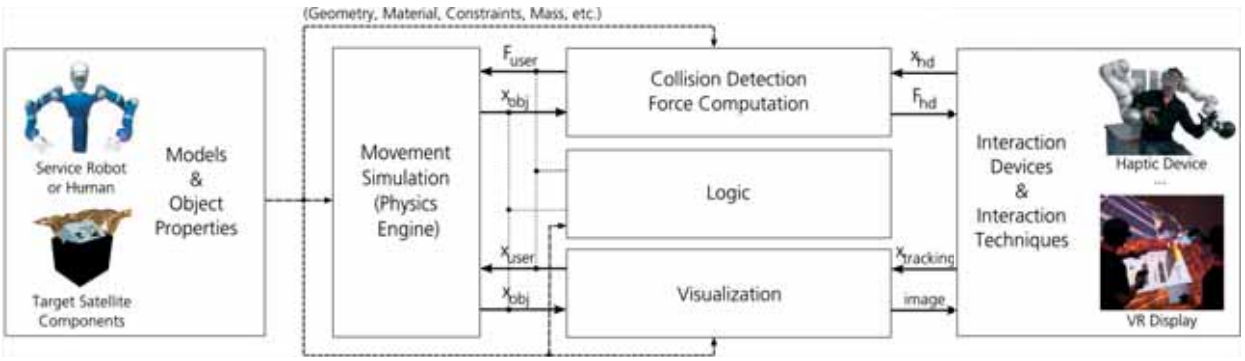


Abbildung 76: Überblick der System-Architektur mit Kern-Komponenten und Daten-Fluss in VR-OOS.

Die Wartung eines Satelliten lässt sich in mehrere Einzeloperationen unterteilen. Die wesentlichen Arbeitsschritte sind bereits in VR-OOS abgebildet und lassen sich daher in virtuellen Umgebungen simulieren. Dazu zählt das Abziehen oder Aufschneiden der metallischen Schutzfolie, die den Satelliten vor großen Temperaturschwankungen und kosmischer Strahlung schützt. Des Weiteren können Schalter betätigt, Schrauben gelöst, elektronische Baugruppen aus ihrem Schacht entnommen und eingeschoben sowie an einer Sicherungsleine befestigt werden. Besonders herausfordernd sind dabei die präzise Simulation der Bewegungen und die Berechnung von Kollisionen aller virtuellen Objekte. Für die Planung und das Training ist es ebenso wichtig, die Montage-Umgebung möglichst fotorealistisch darzustellen. Dazu gehören visuelle Effekte, zum Beispiel die Visualisierung der charakteristischen Materialoberflächen, Licht-Reflektion zwischen den Objekten oder der akkurate Schattenwurf. Damit virtuelle Montage-Simulationen tatsächlich von den Wissenschaftlern und Ingenieuren akzeptiert werden, müssen sie nicht nur so plausibel aussehen, wie man es z. B. von modernen Computerspielen gewohnt ist. Sie müssen auch in allen Belangen physikalisch korrekt sein.

Verteiltes System
Interaktion in VR erfordert Simulation in Echtzeit. Das heißt, das System muss schnell auf die Eingaben des Benutzers reagieren und Ergebnisse anzeigen. Berechnungsbedingte Verzögerungen müssen so kurz sein, dass sie nicht bemerkt werden können. Ansonsten wirkt die Simulation unrealistisch, vermittelt nicht das gewünschte Trainings-Szenario oder wirkt schnell ermüdend. In schlimmsten Fall kann es zu Kopfschmerzen und Übelkeit des Benutzers kommen. Typische Grenzwerte für die Visualisierung der Änderungen vom Zeitpunkt der Eingabe bis zur Ausgabe liegen bei etwa 20 Millisekunden.

Sogar noch höher sind die Anforderungen, wenn man zusätzlich virtuelle Objekte bei Berührung spüren möchte. Solch eine Kraftrückkopplung (engl.: Force Feedback) ermöglichen üblicherweise Force-Feedback-Geräte. Wird z. B. die Oberfläche eines festen Körpers berührt, muss die Kraft innerhalb einer Millisekunde berechnet und an das Gerät übermittelt werden.

Um diesen Anforderungen gerecht zu werden, nutzt VR-OOS eine verteilte System-Architektur. Der Vorteil liegt darin, dass rechenintensive Prozesse auf dedizierte Ressourcen verteilt werden können. Dazu werden einzelne Prozesse als eigenständige Simulationen abstrahiert, etwa Bewegungssimulation, Force-Feedback-Berechnung und Visualisierung. Diese können dann voneinander unabhängig in ihrer höchstmöglichen Geschwindigkeit laufen.

Gemeinsam verwendete Daten werden über das Netzwerk synchronisiert. Das darf aber nicht dazu führen, dass die unabhängigen Simulationsprozesse durch die ständige Datensynchronisation ausgebremst werden. Ebenso dürfen die Teilsimulationen nicht durch die Verwendung veralteter Datenbestände auseinanderlaufen. Die richtige Balance muss also gefunden werden zwischen der Konsistenz der Daten und geringen Übertragungszeiten. Um dies zu gewährleisten, wurden folgende Techniken integriert:

- Intelligente Message-Queuing-Mechanismen,
- Trennung zwischen zuverlässigen und unzuverlässigen Nachrichten,
- Kommunikationsprotokolle mit geringem Overhead.

VR-OOS ist ein offenes Rahmenwerk. Das hat einige Vorteile: Weitere Funktionalitäten lassen sich leicht hinzufügen, bestehende Module erweitern und externe Software-Pakete integrieren, z. B. Simulink als externe Simulation. Neue Erkenntnisse im Satellitenbau oder der Satellitenwartung sollen direkt während der Montagesimulation gesammelt und durch Wissensmanagementsoftware ausgewertet werden. Daher ist z. B. die Anbindung an den „Virtuellen Satelliten“ geplant. Darüber hinaus dient VR-OOS als Forschungsplattform für die Evaluierung neuer Methoden und Ideen. Die Benutzerstudien, die wir im Abschnitt „Mensch-Maschine-Schnittstellen“ beschreiben, konnten wir hiermit schnell durchführen.

Bewegungs-Simulation
Die korrekte Abbildung der physikalischen Prozesse ist ein wichtiger Bestandteil einer Montage-Simulation. VR-OOS beschränkt sich auf die Dynamik und Kinematik von Körpern. Dazu gehört die Bewegung und Kollision in Schwerelosigkeit. Darüber hinaus lassen sich Federverhalten oder Drehungen von Scharnieren abbilden. Um das Setup zu vervollständigen, werden zudem biegbare Objekte angeboten. Diese werden z. B. für die Simulation der beweglichen Schutzfolie, für die Modellierung der Sicherheitsleinen sowie für die Verlegung von Kabeln benötigt. Wir haben die physikalischen Eigenschaften der Objekte um Materialeigenschaften erweitert. Damit können wir Reibung simulieren, z.B. beim Greifen eines Objekts mit den virtuellen Fingern des Benutzers.

Das Ziel von VR-OOS war es nicht, eine vollständig neue Physiksimulation zu entwickeln. Vielmehr sollte auf bestehende Implementierungen zurückgegriffen werden. Eine interne Performance-Studie gängiger echtzeitfähiger Physik-Engines zeigte, dass Bullet im Vergleich zu den anderen Produkten (NVIDIA PhysX, Intel Havok, ODE, Newton, etc.) die besten Ergebnisse in punkto Genauigkeit und Geschwindigkeit liefert. So haben wir Bullet in VR-OOS integriert. Allerdings lassen sich damit nicht alle Anforderungen zufriedenstellend umsetzen. Daher planen wir, weiter nach Alternativen zu suchen. Vielversprechend erscheint das Simulationsprogramm Modelica, das vom DLR mitentwickelt wurde.

Echtzeit-Rendering
Die flexible System-Architektur von VR-OOS erlaubt uns, mehrere Ansätze für die Visualisierung parallel zu untersuchen. Zurzeit arbeiten wir an drei Implementierungen.

Eine basiert auf InstantReality vom Fraunhofer- Institut IGD. Sie wird seit Jahren vom Institut für Robotik und Mechatronik genutzt und ist daher bereits gut in die dort existierende Soft- und Hardware-Umgebung integriert.

Eine andere Lösung basiert auf dem VR-Toolkit ViSTA, das wir in Kooperation mit der RWTH Aachen entwickeln. ViSTA skaliert von Desktop-Arbeitsplätzen zu immersiven Mehrkanal-VR-Systemen und vereinfacht den Zugriff auf spezialisierte VR-Eingabegeräte.

Als C++-Framework erlaubt es, eigene Visualisierungs-Algorithmen zu entwickeln und tief im VR-System zu verankern. Die höchste Effizienz für Echtzeit-Rendering bieten dabei OpenGL-Shader-Programme. Fast alle visuellen Licht- und Materialieneffekte, die wir entwickelt haben, basieren auf der Shader-Technologie. Beispiele sind metallische Oberflächen, eine realistische Atmosphäre der Erde und ca. eine Million Sterne (Abbildung 77). Die Shader-Technologie erlaubt es, diese Effekte in Echtzeit zu visualisieren.

Die genannten Methoden versuchen, sich visuellen Effekten der Realität möglichst effizient anzunähern. Häufig werden dabei gute Bildwiederholraten zulasten der Genauigkeit erreicht. Um die Licht- und Schattenverhältnisse möglichst physikalisch korrekt abbilden zu können, arbeiten wir an einer dritten Implementierung, die auf Ray-Tracing als globalem Beleuchtungsverfahren beruht. Hierfür verwenden wir die Bibliothek OptiX von NVIDIA. Um dieses aufwendige Verfahren echtzeitfähig zu bekommen, teilen wir ein zu renderndes Bild in kleinere Abschnitte auf. Diese werden dann parallel auf unserem GPU-Cluster berechnet, der aus 12 Grafikkarten besteht. Unser System kann so bis zu 15 Bilder pro Sekunde in HD-Auflösung erzeugen.

Abbildung 77: VR-OOS: Fotorealistische Visualisierung der Weltraum-Umgebung beim Anflug an einen Satelliten.

Interaktive Exploration von Planeten

Bei der Erforschung von Planeten fallen sehr große Datenmengen an. Sie entstehen z. B. durch hochauflösende Satellitenbilder, Radardaten oder Messungen in der Atmosphäre. Um solche Daten effektiv auswerten zu können, sind geeignete Datenstrukturen und Software-Werkzeuge erforderlich. Seit 2009 erforschen wir Methoden, die die interaktive Exploration von Planeten und anderen Himmelskörpern in VR-Umgebungen ermöglichen.

Terrain-Visualisierung
Der Terrain-Renderer ist ein zentrales Software-Projekt der Arbeitsgruppe. Er erlaubt die interaktive Visualisierung der gesamten Oberfläche eines Planeten. So müssen sich Planeten-Forscher nicht wie bisher auf einen kleinen Ausschnitt begrenzen, sondern können sich frei an jede beliebige Stelle des Planeten bewegen, um dort wissenschaftliche Daten zu analysieren und zu vergleichen (Abbildung 78). Der Terrain-Renderer kann die in der Planetenforschung am häufigsten verwendeten Datenformate einlesen. Üblicherweise liegen die Teildaten als Streifen vor. Damit sie übereinander gelegt werden können, sind sie eindeutig georeferenziert. Unterschiedliche Auflösungen der Instrumente werden dabei berücksichtigt. Verwendet wird immer nur die höchste verfügbare Auflösung. Das Ergebnis ist ein einheitliches digitales Terrain-Modell (DTM).

Zugrunde liegt dem DTM eine HEALPix-Datenstruktur als Ergebnisraster. Es unterteilt die Oberfläche einer Kugel in 12 gleichgroße, viereckige Flächen (Patches). Die Rasterauflösung jedes einzelnen Patch wird durch die Auflösung des feinsten Datenstreifens innerhalb dieses Patches bestimmt. Dabei deckt jeder Rasterpunkt (Pixel) innerhalb eines Patches jeweils gleichgroße Bereiche der Planetenoberfläche ab. Der entscheidende Vorteil dieser Struktur ist, dass es zu keinen Verzerrungen der Pixel an den Polen kommt, wie es z. B. bei der Rastereinteilung nach Längen- und Breitengraden der Fall wäre.

Das hochaufgelöste DTM ließe sich aber ohne so genannte Level-of-Detail-Techniken (LoD) nicht annähernd in Echtzeit rendern. Daher wird jeder einzelne HEALPix-Patch in einer Quadtree-Datenstruktur organisiert. Mithilfe dieser LoD-Struktur werden immer nur Daten derjenigen Auflösungsstufe geladen, die benötigt werden, um in Abhängigkeit vom Fokus des Benutzers die maximal wahrnehmbare Detaillierung darzustellen. Dieser Ansatz ermöglicht nicht nur interaktive Bildwiederholungsraten, sondern schont auch Speicherressourcen. Erst so lassen sich giga- bis terabyte große Planetendaten effizient verarbeiten.

Abbildung 78: Ein Benutzer betrachtet Valles Marineris auf dem Mars mit dem Terrain-Renderer. Die Daten stammen vom HRSC-Instrument des MarsExpress-Orbiters der ESA. Die Auflösung beträgt ca. 50-100 Meter pro Pixel für das DTM und ca. 12 Meter pro Pixel für die Schwarz-Weiß-Bilddaten. Der gesamte Datensatz ist 600 GB groß.

Ein wesentliches Merkmal unserer Implementierung ist die Genauigkeit. Im Unterschied zu Computerspielen geht es nicht darum, so schnell wie möglich und so plausibel wie nötig zu rendern. Vielmehr möchten wir den Planetenforschern ein wissenschaftlich exaktes Arbeitswerkzeug anbieten. Daher dürfen die LoD-Ansätze keine Fehler induzieren, die zu falschen Annahmen führen könnten. Wir haben Verifikationsmethoden entwickelt, um die Exaktheit der LoD-Darstellung in Abhängigkeit von der Bildschirmauflösung und im Vergleich zur maximalen Datenauflösung zu überprüfen. Diese dienen zum einen als Beleg gegenüber den Wissenschaftlern. Zum anderen sollen sie künftig fester Bestandteil bei automatischen Integrationstests werden.

Interaktive Analyse-Werkzeuge
Die Genauigkeit ist nicht nur für die Visualisierung nötig. Viel wichtiger sind präzise Ausgabeergebnisse bei der interaktiven Analyse der Daten. Das einfachste Analyseelement ist das Einblenden von Zusatzinformationen. Neben Textinformationen in Form von 3D-Annotationen erlaubt es der Terrain-Renderer, Hilfsobjekte direkt auf der Terrain-Oberfläche zu überblenden. Ein Beispiel sind Höhenlinien, die Höhenunterschiede z. B. in 1000-Meter-Schritten verdeutlichen. Höhenunterschiede lassen sich aber auch schnell durch die Einfärbung des Terrains entsprechend der Höhenwerte vermitteln. Als Referenz für den Mars hat sich die Farbtabelle durchgesetzt, die üblicherweise für Daten des MOLA-Sensors verwendet wird.

Neben Höhenlinien können dem Terrain auch die Koordinaten-Linien der Breiten- und Längengrade überlagert werden. Das ist hilfreich bei der Lokalisierung bestimmter Regionen. Für das Zeichnen z. B. der Hilfslinien greifen wir wiederum auf die Shader-Technik zurück. Linien, die im gerenderten Bild unterbrochen erscheinen, sind dann auch tatsächlich an diesen Stellen durch das Gelände verdeckt. Im Gegensatz dazu werden in den meisten GIS-Programmen (GIS, Geographical Information System) herkömmliche Rendertechniken eingesetzt. Unterbrochene Linien sind dann häufig lediglich render-bedingte Artefakte (Bildfehler) und führen zu Fehlinterpretationen.

Ein weiteres wichtiges Werkzeug für die Arbeit von Geologen ist die Profillinie. Sie definiert eine Linie, entlang derer man genau Höhen und Höhenunterschiede im Gelände ablesen kann. Die Spezifizierung erfolgt durch das interaktive Platzieren von Start- und Endpunkt im Gelände. Das dazwischen liegende Höhenprofil wird dann automatisch aus der am höchsten aufgelösten LoD-Stufe des DTM-Datensatzes ermittelt. Es lassen sich mehrere solcher Linien an beliebigen Stellen auf der Planetenoberfläche zugleich platzieren. Die erzeugten Profile können für externe Programme exportiert werden, um sie z. B. mithilfe von 2D-Diagrammen genauer zu analysieren.

Auf ähnliche Weise kann das Volumen von Bergen oder Kratern gemessen werden. Dazu werden interaktiv Punkte gesetzt, um grob deren Umriss zu markieren. Die Punkte auf dem dadurch bestimmten geschlossenen Polylinienzug bilden gleichzeitig die Schnittebene. Nun werden alle Geländepunkte, die durch diese Ebene überdeckt werden, ermittelt und die Höhendifferenzen zur Ebene aufintegriert. Um Merkmale im Terrain präziser zu identifizieren, gehen wir noch einen Schritt weiter und forschen an semiautomatischen Algorithmen, die z. B. nach dem Setzen eines Ankerpunkts im Krater automatisch den Kraterrand finden können.

Um die Interaktion mit dem Terrain weiter zu verbessern, haben wir haptische Geräte integriert. Dadurch kann man die Planetenoberfläche abtasten und gleichzeitig Analyse-Werkzeuge nutzen, um z. B. den tiefsten Punkt eines Tals zu finden oder sich an der Kante eines Kraterabhangs entlang zu tasten. Weitere Methoden können den Benutzer zu speziellen Geländemerkmalen führen. Virtueller Magnetismus z. B., der das Eingabegerät auf einem Bergkamm hält oder in das Kraterinneren zieht, kann Untersuchungsmethoden beschleunigen.

Es besteht aber auch die Möglichkeit, das Terrain zu verändern. Dies ist hilfreich, wenn Hypothesen über die Entstehung von Verwerfungen, Gräben oder geologischer Diskontinuitäten verifiziert werden sollen. Exemplarisch haben wir eine Methode integriert, mit der man die Ränder von Gräben interaktiv markieren kann. Diese dienen als Schnittkanten. Der ausgeschnittene Teil lässt sich dann in alle Richtungen verschieben, verdrehen oder ganz löschen. Das umliegende Gelände wird dabei automatisch angepasst. Diese Analysearbeit lässt sich ohne 3D-Immersion und entsprechender 3D-Interaktionstechnik kaum akkurat durchführen.

Viele weitere Werkzeuge sind denkbar, die für die interaktive Exploration in immersiven 3D-Arbeitsumgebungen optimiert werden. Daher wird sich die Arbeitsgruppe „3D-Interaktion“ weiterhin intensiv in diesem Forschungsbereich engagieren.

Abbildung 79: Mit dem Terrain-Renderer gerendertes Bild des Kraters „Valles Marineris" auf dem Mars. Links: Ohne Atmosphäre. Mitte: Mit Atmosphäre. Rechts: Mit Atmosphäre bei Sonnenuntergang.

Atmosphären-Visualisierung
Weist ein Planet eine Atmosphäre auf, können globale Atmosphäreneffekte eine hilfreiche Ergänzung in der Visualisierung sein. Ein realistischer Eindruck der Umgebung verstärkt nicht nur den Immersionseffekt, sondern hilft auch beim Abschätzen von Distanzen im Gelände (Abbildung 79). Der im Terrain-Renderer implementierte Algorithmus berücksichtigt physikalische Eigenschaften der Atmosphäre:

- **Dunst:** Dieser Effekt entsteht durch Streuung des Sonnenlichtes an Molekülen und Aerosolen auf dem Weg durch die Atmosphäre.
- **Lichtfarbe:** Die Farbe wird durch die Länge des Weges und die Zusammensetzung der Gase beeinflusst.

Unsere Implementierung basiert auf der Rayleigh- und Mie-Streuung. Um eine Anzeige in Echtzeit zu ermöglichen, werden alle benötigten Integrale vorberechnet. Nach jedem Zeichnen der Planetenoberfläche wird in einem separaten Render-Pass ein Shader-Programm aufgerufen, um die Atmosphärenhülle zu generieren (Deferred Shading). Dabei wird durch jeden Pixel des Bildes ein Strahl in die Atmosphäre geschossen. Die vorberechneten Streuungswerte werden über eine vorgegebene Anzahl von Abtastpunkten (ca. 50-200 Punkte) aufsummiert. Durch die Modifikation der Koeffizienten der Streuungsintegrale lässt sich der Algorithmus für beliebige Planeten anpassen.

Kooperative Virtuelle Umgebung
Missionen zur Exploration des Weltraums und ihre Planung erfordern die intensive Zusammenarbeit eines interdisziplinären Experten-Teams. In der Praxis ist der enge Informationsaustausch jedoch oft eingeschränkt. Das liegt zum einen an der Verteilung der Daten und Personen über die ganze Welt. Zum anderen haben Wissenschaftler und Ingenieure unterschiedliche Methoden, ihre Daten und Intensionen zu kommunizieren. Es fehlt ein Werkzeug, das die Daten zusammenbringen und einer multidisziplinären Gruppe veranschaulichen kann.

Wir koordinieren das EU-Projekt „CROSS DRIVE" (FP7, N° 607177). „CROSS DRIVE" steht für „Collaborative Rover Operation and Space Science in a Distributed and Interactive Virtual Environment". Ziel ist es, die Infrastruktur für eine kooperative Arbeitsumgebung für Raumfahrt-Missionen zu entwickeln. Das Projekt-Konsortium umfasst führende Atmosphären-Forscher, Geologen, VR-Experten sowie Vertreter der Raumfahrt-Industrie. Wir übernehmen in diesem Projekt die Gesamtleitung und bringen den Terrain-Renderer ein. Dieser soll durch weitere interaktive Visualisierungsmethoden um geologische und atmosphärische Wissenschaftsdaten ergänzt werden. Ein weiteres wichtiges Thema wird die Entwicklung entsprechender Analyse-Werkzeuge sein, die sich in verteilten, kooperativen Umgebungen besonders intuitiv einsetzen lassen. Abbildung 80 zeigt die erste Demonstration einer kooperativen Sitzung.

Abbildung 80: Erste Demonstration einer kooperativen Sitzung im CROSS-DRIVE-Projekt. Ein lokaler Nutzer (im Vordergrund) am DLR-Standort in Braunschweig untersucht die Oberfläche des Mars gemeinsam mit zwei weiteren Teilnehmern (im Hintergrund) an anderen Standorten (zwei verschiedene VR-Labs der University of Salford, UK).

Weitere Anwendung(en)
Auch wenn aktuell der Mars im Mittelpunkt des Interesses steht, eignet sich der Terrain-Renderer für die Prozessierung und Analyse beliebiger Himmelskörper. Sobald die Rohdaten von laufenden Raumfahrtmissionen wie Rosetta oder Dawn vorliegen, können sie prozessiert und interaktiv exploriert werden.

Da der Terrain-Renderer als Modul von ViSTA realisiert wurde, lassen sich auch Anwendungen erzeugen, die auf spezielle Anforderungen ausgerichtet sind. So arbeitet der Terrain-Renderer z. B. auch als Anflugsimulator im DLR-Projekt ATON. Wir möchten untersuchen, welche Navigationsalgorithmen für ein autonomes und nur optisch gesteuertes Mond-Landesystem zum Einsatz kommen können. Der Terrain-Renderer kann jederzeit um Algorithmen erweitert werden, die von den Anwendungen verlangt werden. Sie stehen anschließend auch anderen Anwendungen zu Verfügung. Ein gutes Beispiel ist die Berechnung harter Schatten in Kratern (Abbildung 81). Sie wird von den Kratererkennungsalgorithmen von ATON benötigt.

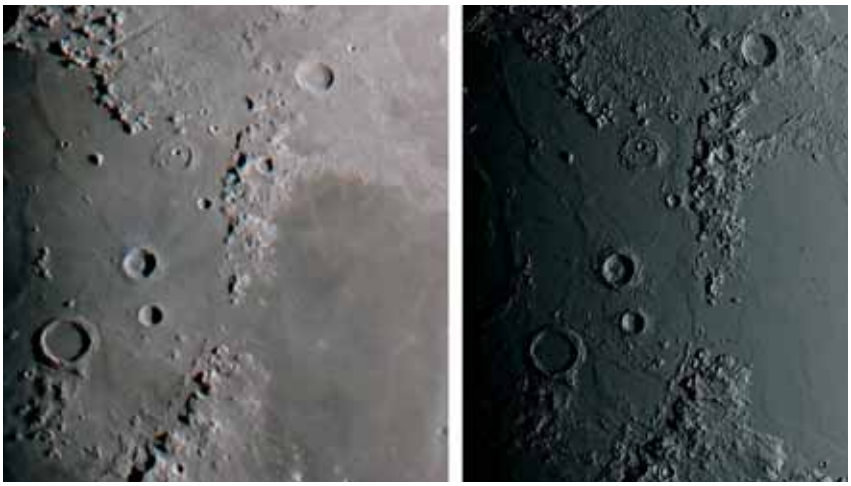


Abbildung 81: Vergleich: Fotoaufnahme (links) und ein mit dem Terrain Renderer erzeugtes Bild (rechts). Die Bilder zeigen den östlichen Rand des Mare Imbrium auf dem Mond.

Mensch-Maschine-Schnittstellen

Eine der wichtigsten Eigenschaften von virtuellen Umgebungen ist die echtzeitfähige, multimodale Interaktion zwischen dem Benutzer und der virtuellen Welt. Ziel ist es zum einen, eine möglichst realitätsnahe Simulation zu erreichen. Dies ist besonders für Trainings-Anwendungen wichtig, z. B. im VR-OOS-Vorhaben. Wenn die virtuelle Interaktion allerdings weniger oder mehr Freiheitsgrade als in der Realität aufweist, wird der gewünschte Lerneffekt ggf. nicht erreicht oder unter Umständen sogar die falsche Aktion trainiert.

Zum anderen sollen möglichst effiziente und intuitive Interaktions-Methoden zur Verfügung stehen. Dafür kann es mitunter sinnvoll sein, von einer realitätsnahen Abbildung bewusst abzuweichen. Ein Beispiel: Ein selektiertes Objekt wird über eine größere Distanz lediglich durch eine einfache Handgeste verschoben. Oder: Objekte werden ohne aufwendige Montage-Aktionen automatisch an eine bestimmte Position bewegt und anschließend ausgerichtet bzw. eingerastet. Solche Hilfsmittel können die Arbeit erheblich vereinfachen und verkürzen.

Wichtig ist jedoch, dass die VR-Geräte und Interaktionsmethoden ihren Aufgaben entsprechend ausgewählt werden und aufeinander abgestimmt sind. Daher ist das Thema Mensch-Maschine-Schnittstelle eines der wichtigsten Schwerpunkte der Arbeitsgruppe.

Elektrotaktiles Feedback
Um eine hohe Immersion zu erreichen, unterstützen fortschrittliche virtuelle Umgebungen nicht nur visuelle Sinnesmodalitäten, sondern auch andere, wie die haptische Modalität. Haptische Rückkopplung macht virtuelle Objekte oder Daten fühlbar. Man unterscheidet zwischen Force-Feedback (Rückkopplung von Kräften) und taktilem Feedback (Ertasten von Oberflächen). Haptische Geräte sind sehr komplex. Sie sind meist teuer, unterstützen nur einen geringen Arbeitsbereich oder sind schwer und unbequem zu tragen. Besonders in immersiven VR-Umgebungen ist es schwierig, haptische Geräte zu integrieren. Wir suchen daher nach Alternativen, die mobil, leicht und kostengünstig sind und dennoch ein geeignetes haptisches Feedback bieten.

Abbildung 82 zeigt den von uns entwickelten Prototyp eines Geräts, das taktiles Feedback an den Fingerspitzen des Benutzers erzeugt. Man kann damit typische Empfindungen generieren, die beim Greifen und bei der Manipulation von Gegenständen auftreten: Berührung, Druck, Bewegung und Rutschen. Die Stimulation wird durch einen Strom erzeugt, der durch die Haut fließt und dabei diejenigen Nerven anregt, die für eine taktile Empfindung verantwortlich sind. Der Ausgabesensor an den Fingerspitzen besteht dabei aus einem Array kleiner Elektroden. Durch eine Folge kurzer elektrischer Signale und durch ein zeitlich variierendes Ausgabemuster lassen sich die unterschiedlichen Empfindungstypen abbilden. Die begleitende Nutzerstudie hat gezeigt, dass hiermit bereits sehr gut die Gefühle von Festhalten (Berührung) und Weggleiten (Rutschen) simuliert werden. Es besteht aber insgesamt noch viel Potential für Verbesserungen.

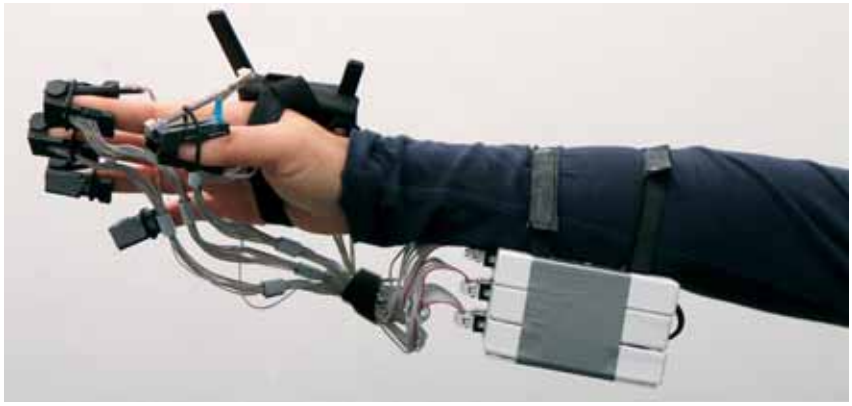


Abbildung 82: Erster Prototyp des elektrotaktilen Feedback-Gerätes.

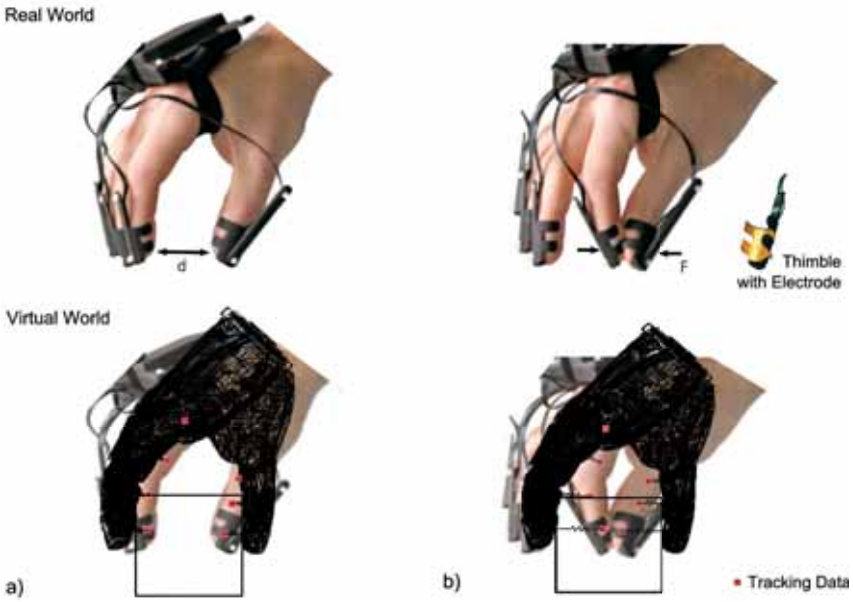


Abbildung 83:
Nutzung des Pseudo-Haptik-Effekts als Alternative zu haptischen Geräten.
a) Die Eindring-Tiefe steuert die Kraft des Greifens.
b) Die Druck-Kraft der Finger steuert die Greifkraft der virtuellen Finger.

Pseudo-Haptik
Falls haptisches Feedback in einer VR-Umgebung nicht verfügbar oder integrierbar ist, bietet Pseudo-Haptik eine Alternative. Darunter versteht man die Wahrnehmung bzw. Identifizierung einer Kraftgröße, ohne dass eine Kraft aktiv über ein Gerät auf den Benutzer ausgeübt wird.

Wir haben dazu drei Interaktionsmethoden entwickelt und in mehreren Studien untersucht. Die erste Methode basiert auf indirekter Interaktion und nutzt das Standard-Eingabegerät unseres VR-Systems, den A.R.T. Flystick2. Es handelt sich dabei um ein 3D-Eingabegerät, das man in die Hand nehmen kann. Es besitzt u. a. einen kleinen analogen Joystick, der leicht mit dem Daumen bedienbar ist. Drückt man ihn nach vorn, öffnet sich eine virtuelle Hand. Je weiter man den Joystick nach hinten drückt, desto mehr schließt sich die Hand. Will man ein Objekt fest greifen, muss man den Joystick entsprechend weit nach hinten drücken.

Die zweite Methode verwendet ein Finger-Tracking-Gerät. Dadurch lässt sich eine direkte Interaktion realisieren. Die reale Hand greift nach dem virtuellen Objekt. Es wird jedoch zusätzlich eine virtuelle Hand abgebildet. Wird nun ein Objekt ergriffen, bleiben die virtuellen Finger an der Oberfläche „kleben". Die reale Hand kann aber in das virtuelle Objekt „eintauchen" (Abbildung 83 a). Diese Methode basiert auf der Eindringtiefe der Finger beim Greifen eines virtuellen Objektes. Zwischen der tatsächlichen Finger-Positionen und den virtuellen Fingern sind virtuelle Federn gekoppelt. Die Ausdehnung der Federn steuert die Kraft, die auf das virtuelle Objekt wirkt. Aber auch auf den Benutzer wirkt eine Kraft. Diese beruht auf dem so genannten kinästhetischen und propriozeptiven Feedback, das sich durch den Abstand zwischen Finger und Daumen ergibt.

Die dritte Methode ist ebenfalls eine direkte Interaktion und basiert auf dem haptischen Feedback beim Berühren der eigenen Finger (Abbildung 83 b). Die Kraft beim Zusammendrücken der Finger steuert dabei die benötigte Kraft zum Greifen des virtuellen Objekts. Wir nutzen dazu eine Spezialanfertigung eines optischen Finger-Tracking-Systems. Es hat Elektroden unter den Finger-Hauben, um den Hautwiderstand zu messen. Dieser ändert sich, wenn man die Finger mehr oder weniger fest zusammendrückt. In einer der Benutzerstudien haben wir die beiden direkten Interaktionsmethoden miteinander verglichen. Es hat sich herausgestellt, dass die meisten Nutzer in der Lage waren, das relative Gewicht zwischen virtuellen Objekten richtig abzuschätzen. Das tatsächliche Gewicht kann dagegen keiner der dargestellten Pseudo-Haptik-Ansätze vermitteln.

Myoelektrische Eingabe
In einer weiteren aktuellen Studie zur Pseudo-Haptik untersuchen wir myoelektrische Signale am Unterarm, die durch das Zudrücken und Bewegen einer Hand erzeugt werden. Unter Myoelektrik versteht man die elektrische Spannung, die sich in den Muskelzellen während der Muskelkontraktion ergibt. Sie liegt im Mikrovoltbereich und kann durch empfindliche Elektroden auf der Haut gemessen werden. Der Vorteil für den Benutzer ist, dass Interaktionen in virtuellen Umgebungen ohne unbequeme Sensoren an den Fingern möglich sind.

Es existieren zwar kommerzielle Systeme. Die sind jedoch entweder sehr teuer, zu invasiv oder bieten keine geeigneten Programmier-Schnittstellen. Daher entwickeln wir ein eigenes Gerät (Abbildung 84), in das unsere Forschungsergebnisse einfließen sollen. Es besteht aus einem Armband mit acht Elektroden, das über den Unterarm geschoben wird. Ein kleines Mikrocomputer-Board übernimmt die Verstärkung und Auswertung der Signale. Das Gerät ist leicht, drahtlos und batteriebetrieben.

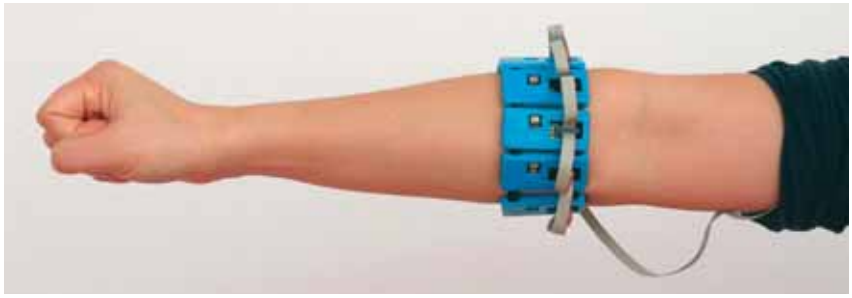


Abbildung 84: Prototyp des myoelektrischen Geräts.

Wissens-Austausch-Workshop: Visualisierung großer Datenmengen in der Wissenschaft

In vielen DLR-Instituten ist die wissenschaftliche Visualisierung bereits ein wichtiges Hilfsmittel in der täglichen Arbeit. Allerdings werden die Datenmengen größer und komplexer, und die Anforderungen gehen deutlich über die Funktionalitäten heute üblicher Desktopanwendungen hinaus (z. B. ParaView und TechPlot). Im DLR gibt es einen hohen Bedarf, Visualisierungsmethoden weiterzuentwickeln. Dabei ist auch erkennbar, dass die Anwender nicht nur die Daten anschauen wollen, sondern mit ihnen interaktiv arbeiten wollen. Echtzeitfähige 3D-Methoden werden mehr und mehr eingefordert.

2013 fand daher in Köln ein DLR-interner Workshop statt. Ziel war der Wissensaustausch zum Thema „Visualisierung großer Datenmengen in der Wissenschaft". Die Organisation lag zu einem großen Teil bei unseren Arbeitsgruppen „Wissenschaftliche Visualisierung" und „3D-Interaktion". Ein Nachfolgeworkshop fand 2014 am Institut für Lufttransportsysteme in Hamburg statt. Ein weiterer Workshop ist für 2015 geplant am deutschen Fernerkundungsdatenzentrum in Oberpfaffenhofen.

Ausbildung und Lehre

Diplom-, Bachelor- und Masterarbeiten

Angegeben sind jeweils: Name (Jahr), Thema. Hochschule (DLR-seitiger Betreuer)

1. Pielicke, Stefan (2011): Requirements Engineering im Deutschen Zentrum für Luft- und Raumfahrt. Eine Untersuchung zum Stand der Praxis. Masterarbeit, Universität Duisburg-Essen (Tobias Schlauch)
2. Ney, Miriam (2011): Enabling a Data Management System to Support the „Good Laboratory Practice". Masterarbeit, Freie Universität Berlin (Andreas Schreiber)
3. Kautz, Frank (2011): Implementierung exemplarischer, paralleler, numerischer Verfahren aus dem CFD-Bereich auf Grafikkarten unter Verwendung von OpenCL. Masterarbeit, Hochschule Darmstadt (Achim Basermann)Leinemann, Gero (2011): Steuerung von Simulationen des Virtuellen Satelliten mit Hilfe Interaktiver Visualisierung. Bachelorarbeit, Duale Hochschule Mannheim (Robin Wolff)
4. Zöllner, Melven (2011): Parallel iterative solution of sparse linear systems in computational fluid dynamics. Bachelorarbeit, RWTH Aachen (Achim Basermann)
5. Stünz, Hermann (2012): Automatisierte Tests verteilter Systeme unter Nutzung von Cloud-Ressourcen. Masterarbeit, Fachhochschule Köln (Robert Mischke)
6. Kroll, Phillip (2012): Dynamic node discovery for the open source framework RCE for usage at the German Aerospace Center (DLR). Masterarbeit, Bonn-Rhein-Sieg, University of Applied Sciences, St. Augustin, Germany (Robert Mischke)
7. Schlegel, Sophia Veronika (2012): Konzeption und Implementierung einer Visualisierung von Simulationsdaten. Bachelorarbeit, Deutsches Zentrum für Luft- und Raumfahrt (Markus Kunde)
8. Krüger, Franziska (2012): Entwicklung von parallelisierbaren Gradienten-basierten Verfahren zur automatisierten, Ersatzmodell-gestützten Optimierung unter Nebenbedingungen für CFD-FEM-Verdichterdesign. Masterarbeit, Technische Universität Berlin (Achim Basermann)
9. Schneider, Matthias (2012): Entwicklung und Evaluierung von intuitiven GUI Elementen und Strukturen für immersive Virtuelle Realität zur Anwendung an einer Powerwall. Bachelorarbeit, Otto-von-Guericke Universität Magdeburg (Johannes Hummel)
10. Stein, Tobias (2012): Prozess zum Benchmarking von Physik-Engines für orbitale Simulationen. Bachelorarbeit, Otto-von-Guericke Universität Magdeburg (Johannes Hummel)
11. Schumann, Robert (2012): Techniken für installationsfreie Client-Software am Beispiel einer Versionsverwaltung von Simulations-modellen. Diplomarbeit, Humboldt-Universität zu Berlin (Daniel Lüdtke)
12. Franke, Marian (2012): Echtzeitfähige und realistische Visualisierung des Sternenhimmels für Weltraumanwendungen. Bachelor-arbeit, Duale Hochschule Mannheim (Markus Flatken)
13. Teichmann, Clemens (2013): Analysis of Software-Engineering-Processes. Masterarbeit, Technische Universität Ilmenau (Andreas Schreiber)
14. Tischler, Timo (2013): Entwicklung eines systematischen Testkonzepts für ein hoch paralleles Softwarepaket aus dem Bereich der Materialwissenschaften. Bachelorarbeit, Duale Hochschule Baden-Württemberg, Mannheim (Achim Basermann, Christin Keutel)
15. Seebach, Oliver (2013): Conception and Implementation of a Usability Problem Analysis Tool and its Comparison with another Usability Technique on the Example of a Workflow-driven Integration Environment. Masterarbeit, Rheinische Friedrich-Wilhelms-Universität Bonn (Doreen Seider, Robert Mischke)
16. Schäfer, Thorsten (2013): Semantische Suche in Wissensportalen: Konzeption und Evaluation der Erweiterung eines Suchframeworks um semantische Technologien. Masterarbeit, Hochschule Bonn-Rhein-Sieg (Doreen Seider, Sinan Mece)
17. Biskoping, Lukas-Maria (2013): Aufbau einer Kommunikations-Infrastruktur auf Basis von MQTT. Bachelorarbeit, Fachhochschule Südwestfalen (Doreen Seider)
18. Klein, Florian (2013): A Model-driven Approach to Design Multi-device Forms for Capturing Data in a Medical Study Environment. Masterarbeit, Rheinische Friedrich-Wilhelms Universität (Thomas Sauerwald)
19. Tiede, Michael (2013): Evaluierung von Suchalgorithmen zur formalen Verifikation von Raumfahrtmissionen. Bachelorarbeit, Ostfalia Hochschule für angewandte Wissenschaften (Philipp Fischer)
20. Collienne, Peter (2013): Interactive Visualization of Parameterized Atmospheres in Virtual Reality. Bachelorarbeit, RWTH Aachen (Robin Wolff)
21. Weps, Benjamin (2013): Entwicklung einer Schnittstelle für verteiltes I/O in einer neuen on-board Computerarchitektur, Masterarbeit, Fachhochschule Bingen (Daniel Lüdtke)
22. Müller, Juliane (2013): Bidirektionale Anbindung eines grafischen Editors an ein Systemmodell. Bachelorarbeit, DHBW Mannheim (Volker Schaus)

23. Berthold, Frieder (2013): Entwicklung eines Systems für verteiltes, paralleles Echtzeit-Raytacing. Bachelorarbeit, DHBW Mannheim (Robin Wolff)
24. Illmer, Joachim (2014): Parallele Python-Programmierung auf Multi-Core-Architekturen und Grafikkarten für numerische Algorithmen aus der Strömungstechnik und den Materialwissenschaften. Bachelorarbeit, Duale Hochschule Baden-Württemberg, Mannheim (Achim Basermann, Melven Röhrig-Zöllner)
25. Roder, Simon (2014): Konzeption der Generalisierbarkeit und Implementierung einer automatisierten Erkennung von Usability-Schwächen am Beispiel einer Java SWT Anwendung. Masterarbeit, Fachhochschule Aachen (Oliver Seebach)
26. Röhrig-Zöllner, Melven (2014): Parallel solution of large sparse eigenproblems using a Block-Jacobi-Davidson method. Masterarbeit, RWTH Aachen (Jonas Thies)
27. Isidro, Pedro (2014): Automatic Code Generation for Attitude and Orbit Control Systems Using Domain-Specific Languages. Masterarbeit, Instituto Superior Técnico, Lisbon (Meenakshi Deshmukh)
28. Wegener, Ronny (2014): Adding Haptic Feedback to Geodesy Analysis Tools used in Planetary Surface Exploration. Masterarbeit, Otto-von-Guericke Universität Magdeburg (Robin Wolff)
29. Li, Weichen (2014), Myoelectric Interaction Device to Enhance Grasping in Virtual Environments. Masterarbeit, OvGU Magdeburg (Johannes Hummel)
30. Schartel, Andreas (2014): Filter-Konzept für diagnostische Reports. Bachelorarbeit, Universität Würzburg (Olaf Maibaum)
31. Merkel, Jonas (2014): Interaktive Visualisierung großer Simulationsdaten. Bachelorarbeit, TU Kaiserslautern (Markus Flatken)
32. Altgelt, Max (2014): Semi-Automatic Crater Detection. Bachelorarbeit, Uni Münster (Wito Engelke)
33. Zhang, Wie (2014): Design and Implementation of Multi-core Support for an Embedded Real-time Operating System for Space Applications. Masterarbeits, KTH Royal Institute of Technology (Ting Peng)
34. Jastrzembksi, Tim (2015): Wissenschaftliche Visualisierung von Wirbelschleppen landender und startender Flugzeuge. Bachelorarbeit, Fachhochschule Bielefeld (Robin Wolff)

Betreute Dissertationen

Nachdem die Simulations- und Softwaretechnik als wissenschaftlich-technische Einrichtung etabliert war, haben wir begonnen, neben Bachelor- und Masterstudierenden auch Doktoranden zu betreuen. Sie leisten wichtige Beiträge zu unserer Grundlagenforschung. Da das DLR über kein Promotionsrecht verfügt, erfolgt die Promotion jeweils in Kooperation mit einer Hochschule. Im Berichtszeitraum haben wir die folgenden Dissertationen betreut:

1. Christian Wagner, Thema: „Online Monitoring and Computational Steering of Massive Parallel CFD Simulations“, TU Kaiserslautern, Fachbereich Informatik (Prof. Dr. Hans Hagen), Betreuung bei SC: 2009 – 2013, Promotion: 2013.
2. Rolf Westerteiger, Thema: „Enabling Geoscientific Research in Virtual Reality“, TU Kaiserslautern, Fachbereich Informatik (Prof. Dr. Hans Hagen), Betreuung bei SC: 2009 – 2013, Promotion: 2014.
3. Hao Zhang, Thema: „Application of Model Driven Architecture (MDA)“, The University of Chinese Academy of Sciences, China, (Prof. Bo Liu), Betreuung bei SC: 2011 - 2013, Promotion: 2014.
4. Roxana Bujack, Thema: „Pattern recognition in flow fields using moment Invariants“, Universität Leipzig (Prof. Gerik Scheuermann), Betreuung bei SC: 2013 – 2014, Promotion: 2014.

Wissenschaftleraustausch

Angegeben sind jeweils: Name, Gastinstitution, Thema der Kooperation, Jahr, Dauer, Status

1. Robin Wolff, ThinkLab – University Salford – UK, „Collaborative Exploration of Planet Mars“, 2014, 3 Wochen, abgeschlossen.
2. Zhiqi Guo (BUAA University, China, Prof. Zili WANG), DLR Simulations- und Softwaretechnik – Braunschweig, Thema: „Virtual Maintenance Simulation Planning and Credibility Evaluation Technologies of On-Orbit Servicing (OOS)“, 2013, 3 Monate, abgeschlossen mit Promotion: 2014.

Lehrtätigkeit

Angegeben sind jeweils: Name, Fach, Art der Veranstaltung, Hochschule, Jahr

1. Lüdtke, Daniel: Digitale Systeme, Seminaristischer Unterricht + Praktikum, Hochschule für Technik und Wirtschaft Berlin, 2012
2. Hotz, Ingrid: Scientific Computing / Grundlagen der wissenschaftlichen Visualisierung, Vorlesung + Übungen, TU Braunschweig, 2014 / 2015

Veröffentlichungen

Veröffentlichungen und Vorträge werden im DLR in der elektronischen Bibliothek elib registriert. Für den Zeitraum 2011 – 2014 enthält die Bibliothek 197 Einträge für die Simulations- und Softwaretechnik. Die folgenden Listen geben einen Überblick über die schriftlichen Veröffentlichungen aus dem Berichtszeitraum.

ISI-gelistete Veröffentlichungen

1. Murgia, Alessio und **Wolff, Robin** und Sharkey, Paul M. und Clark, Ben (2011): Low-cost optical tracking for collaborative applications in immersive virtual environments. International Journal on Disability and Human Development, Volume 10, Issue 4, De Gruyter, Seiten 359-364, Nov. 2011
2. **Westerteiger, Rolf** und Compton, Tracy und Bernardin, Tony und Cowgill, Eric und Gwinner, Klaus und Hamann, Bernd und **Gerndt, Andreas** und Hagen, Hans (2012): Interactive Retro-Deformation of Terrain for Reconstructing 3D Fault Deplacements. Transaction on Visualization and Computer Graphics (TVCG), Vol. 18, Issue 12, (Proceedings, IEEE Visualization, Seattle, WA, Oktober 14 - 19), Seiten 2208 - 2215
3. Kratz, Andrea und Baum, Daniel und **Hotz, Ingrid** (2013): Anisotropic Sampling of Planar and Two-Manifold Domains for Texture Generation and Glyph Distribution. IEEE Transaction on Visualization and Computer Graphics (TVCG), 2013, 19, 1782-1794
4. **Seider, Doreen und Basermann, Achim und Mischke, Robert und Siggel, Martin und Tröltzsch, Anke und Zur, Sascha** (2013): Ad hoc Collaborative Design with Focus on Iterative Multidisciplinary Process Chain Development applied to Thermal Management of Spacecraft. In: CEAS Aerospace Aerodynamics Research Conference, Linkoping University Electronic Press. 4th CEAS Air & Space Conference, 16-19 September 2013, Linköping, Sweden. ISBN 78-91-7519-519-3, ISSN 0001-9240.
5. **Wolff, Robin** und Preusche, Carsten und **Gerndt, Andreas** (2014): A modular architecture for an interactive real-time simulation and training environment for satellite on-orbit servicing. Journal of Simulation (JOS), Palgrave Macmillan, Vol. 8, Issue 1, Seiten 50-63, Februar 2014
6. **Tröltzsch, Anke** (2014): A Sequential Quadratic Programming Algorithm for Equality-Constrained Optimization without Derivatives. Optimization Letters. Springer. DOI: 10.1007/s11590-014-0830-y. ISSN 1862-4472

Weitere begutachtete Veröffentlichungen

1. **Litz, Markus** und **Seider, Doreen** und Otten, Tom und **Kunde, Markus** (2011): Integration Framework for Preliminary Design Toolchains. In: CEAS Aeronautical Journal. Deutscher Luft- und Raumfahrtkongress 2011, 27.-29. September 2011, Bremen, Deutschland.
2. **Bachmann, Arne** und **Bergmeyer, Henning** und **Schreiber, Andreas** (2011): Evaluation of aspect-oriented frameworks in Python for extending a project with provenance documentation features. The Python Papers, 6 (3), Seiten 1-18. The Python Papers Anthology. ISSN 1834-3147.

3. **Basermann, Achim** und **Kersken, Hans-Peter** (2011): Scalable Distributed Schur Complement Methods for CFD Simulation on Many-Core Architectures. International Conference On Preconditioning Techniques For Scientific And Industrial Applications, May 16-18, 2011, Bordeaux, France.
4. **Basermann, Achim** und **Kersken, Hans-Peter** (2011): Scalable Distributed Schur Complement Solvers for Internal and External Flow Computations on Many-Core Architectures. ICIAM 2011, July 18 – 22, 2011, Vancouver, BC, Canada.
5. **Bachmann, Arne** (2011): A Python Wrapper Code Generator for Dynamic Libraries. The Python Papers, 6 (2). ISSN 1834-3147.
6. **Maibaum, Olaf** und Terzibaschian, Thomas und Raschke, Christian und **Gerndt, Andreas** (2011): Software Reuse of the BIRD ACS for the TET Satellite Bus. Proceedings, 8th IAA Symposium on Small Satellites for Earth Observation, Berlin, Seiten 409-412, April 4 - 8, 2011
7. **Wagner, Christian** (2011): Open Problems in Computational Steering of Massive Parallel Unstructured Grid Based CFD Simulations. Visualization of Large and Unstructured Data Sets - Applications in Geospatial Planning, Modeling and Engineering (IRTG 1131 Workshop), VLUDS 2010, March 19-21, 2010, Bodega Bay, CA, USA. OASICS 19 Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Deutschland
8. **Westerteiger, Rolf** (2011): *Cartography of Mars in a Virtual Reality Environment.* Visualization of Large and Unstructured Data Sets - Applications in Geospatial Planning, Modeling and Engineering (IRTG 1131 Workshop), VLUDS 2010, March 19-21, 2010, Bodega Bay, CA, USA. OASICS 19 Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Deutschland
9. **Fischer, Philipp M.** und **Schaus, Volker** und **Gerndt, Andreas** (2011): Design Model Data Exchange Between Concurrent Engineering Facilities by Means of Model Transformation. 13th NASA-ESA Workshop on Product Data Exchange 2011, Cypress, California, USA
10. **Schaus, Volker** und Großekatthöfer, Karsten und **Lüdtke, Daniel** und **Gerndt, Andreas** (2011): Collaborative Development of a Space System Simulation Model. 20th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE 2011), 27-29th June 2011, Paris, France
11. **Wolff, Robin** und Preusche, Carsten und **Gerndt, Andreas** (2011): A Modular Architecture for an Interactive Real-Time Simulation and Training Environment for Satellite On-Orbit Servicing. Proceedings, 15th IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications (DS-RT), Salford, MediaCity, UK, Seiten 72-80, Sept. 4-7, 2011
12. **Chen, Fang** und **Flatken, Markus** und **Basermann, Achim** und **Gerndt, Andreas** und Hetherington, James und Krüger, Timm und Matura, Gregor und Nash, Rupert (2012): Enabling In-situ Pre- and Post-Processing for Exascale Hemodynamic Simulations – A Co-Design Study with the Sparse Geometry Lattice Boltzmann Code HemeLB. In: IEEE Xplore, ACM Digital Library (Online-Proceedings). SC 2012, 10.-16. Nov. 2012, Salt Lake City, USA.
13. **Basermann, Achim** und **Zöllner, Melven** (2012): Scalable Two-Level Preconditioners for CFD Computations on Many-Core Systems. PMAA 2012, 28.-30. Juni 2012, London, England.
14. **Basermann, Achim** und **Zöllner, Melven** (2012): New Block Distributed Schur Complement Preconditioners for CFD Simulation on Many-Core Architectures. 2012 SIAM Conference on Applied Linear Algebra, 18.-22.06.2012, Valencia, Spanien.
15. **Basermann, Achim** und **Kersken, Hans-Peter** und **Schreiber, Andreas** und Gerhold, Thomas und Jägersküpper, Jens und Kroll, Norbert und Backhaus, Jan und Kügeler, Edmund und Alrutz, Thomas und Simmendinger, Christian und Feldhoff, Kim und Krzikalla, Olaf und Müller-Pfefferkorn, Ralph und Puetz, Mathias und Aumann, Petra und Knobloch, Olaf und Hunger, Jörg Hunger (2012) HICFD – Highly Efficient Implementation of CFD Codes for HPC Many-Core Architectures. Springer Berlin Heidelberg. ISBN 978-3-642-24024-9.
16. **Basermann, Achim** und **Zöllner, Melven** (2012): Block Distributed Schur Complement Preconditioners for CFD Computations on Many-Core Systems. SIAM Conference on Parallel Processing for Scientific Computing, 15.-17.02.2012, Savannah, Georgia, US.
17. **Schreiber, Andreas** und **Ney, Miriam** und **Wendel, Heinrich** (2012): The Provenance Store prOOst for the Open Provenance Model. In: Provenance and Annotation of Data and Processes, 7525, Seiten 240-242. Springer Berlin Heidelberg. 4th International Provenance and Annotation Workshop, 19.-21. Jun. 2012, Santa Barbara, USA. DOI: 10.1007/978-3-642-34222-6_26. ISBN 978-3-642-34221-9. ISSN 0302-9743
18. Böhnke, Daniel und Moerland, Erwin und **Seider, Doreen** und **Kunde, Markus** und **Litz, Markus** und Ziemer, Sven und Stenz, Gernot (2012): Challenges for Collaborative Data Management in an MDAO Process. Deutscher Luft- und Raumfahrt Kongress, 10.-12. Sep. 2012, Berlin, Deutschland.
19. Kreutzer, Moritz und Hager, Georg und Wellein, Gerhard und Fehske, Holger und **Basermann, Achim** und Bishop, Alan R. (2012): Sparse matrix-vector multiplication on GPGPU clusters: A new storage format and a scalable implementation. Proceedings of the 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops (IPDPS 2012), 1696 -1702. IEEE Conference Publications. Workshop on Large-Scale Parallel Processing to be held at the IEEE International Parallel and Distributed Processing Symposium 2012, 21.-25. Mai 2012, Shanghai, China. ISBN 978-1-4673-0974-5.

20. **Seider, Doreen** und **Fischer, Philipp M.** und **Litz, Markus** und **Schreiber, Andreas** und **Gerndt, Andreas** (2012): Open Source Software Framework for Applications in Aeronautics and Space. 2012 IEEE Aerospace Conference, Big Sky, Montana, USA, März 3-10, 2012
21. **Fischer, Philipp M.** und **Wolff, Robin** und **Gerndt, Andreas** (2012): Collaborative Satellite Configuration Supported by Interactive Visualization. 2012 IEEE Aerospace Conference, Big Sky, Montana, USA, März 3-10, 2012
22. **Westerteiger, Rolf** und **Gerndt, Andreas** und Hamann, Bernd und Hagen, Hans (2012): Spatial Analysis of Terrain in Virtual Reality. Short Paper, IEEE Virtual Reality Workshop, Immersive Visualization Revisited: Challenges and Opportunities, Orange County, CA, USA, März 4, 2012
23. **Wagner, Christian** und **Gerndt, Andreas** und Hansen, Charles und Hagen, Hans (2012): Interactive In-Situ Online Monitoring of Large Scale CFD Simulations with Cut-Planes. Short Paper, IEEE Virtual Reality Workshop, Immersive Visualization Revisited: Challenges and Opportunities, Orange County, CA, USA, März 4, 2012
24. **Hummel, Johannes** und **Wolff, Robin** und **Gerndt, Andreas** und Kuhlen, Torsten (2012): Comparing Three Interaction Methods for Manipulating Thin Deformable Virtual Objects. Poster + Video, IEEE VR, Orange County, CA, USA, März 4-8, 2012
25. **Lüdtke, Daniel** und Ardaens, Jean-Sébastien und **Deshmukh, Meenakshi** und Paris Lopez, Rosa und Braukhane, Andy und Pelivan, Ivanka und Theil, Stephan und **Gerndt, Andreas** (2012): Collaborative Development and Cataloging of Simulation and Calculation Models for Space Systems. Proceedings, 21th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), Toulouse, France, Seiten 244-249
26. **Lüdtke, Daniel** und **Mece, Sinan** und **Deshmukh, Meenakshi** und **Bock, Michael** und **Schreiber, Andreas** und **Gerndt, Andreas** (2012): A Framework to Model Metadata for Knowledge Management Tools. 4th International Conference on Knowledge Management for Space Missions, Toulouse Space Show 2012, Toulouse, France, Juni 25-28, 2012
27. **Hummel, Johannes** und **Wolff, Robin** und **Stein, Tobias** und **Gerndt, Andreas** und Kuhlen, Torsten (2012): An Evaluation of Open Source Physics Engines for Use in Virtual Reality Assembly Simulations. Proceedings, 8th International Symposium on Visual Computing (ISVC), Rethymnon, Crete, Greece, Juli 16 - 18, 2012
28. **Deshmukh, Meenakshi** und **Schaus, Volker** und **Fischer, Philipp** und Quantius, Dominik und Maiwald, Volker und **Gerndt, Andreas** (2012): Decision Support Tool for Concurrent Engineering in Space Mission Design. Proceedings, 19th ISPE International Conference on Concurrent Engineering (CE2012), Trier, September 3 - 7, 2012
29. Shaw, D., und Patera, M., und Paparidou, E. und **Wolff, R.** (2012): Evaluation of the prototype mobile phone app Pugh: a 3D cartoon character designed to help deaf children to speech read. In Sharkey, P (ed.), Proc. 9th Intl Conf. Disability, Virtual Reality & Associated Technologies (ICDVRAT), Laval, France, September 10-12, 2012
30. **Zhang, Hao** und **Gerndt, Andreas** und Liu, Bo (2012): 3D State Decomposition Modeling Method for MBSE in Space Engineering. Proceedings, AIAA SPACE 2012 Conference & Exposition, Pasadena, CA, September 11 - 13, 2012
31. **Wagner, Christian** und **Flatken, Markus** und **Chen, Fang** und **Gerndt, Andreas** und Hansen, Charles und Hagen, Hans (2012): Interactive Hybrid Remote Rendering for Multi-pipe Powerwall Systems. Proceedings, 9. GI-Workshop Virtuelle und Erweiterte Realität, Düsseldorf, September 19 - 20, 2012
32. **Westerteiger, Rolf** und **Chen, Fang** und **Gerndt, Andreas** und Hamann, Bernd und Hagen, Hans (2012): Remote GPU-Accelerated Online Pre-processing of Raster Maps for Terrain Rendering. Proceedings, 9. GI-Workshop Virtuelle und Erweiterte Realität, Düsseldorf, September 19 - 20, 2012
33. **Fischer, Philipp** und **Schaus, Volker** und **Lüdtke, Daniel** und **Maibaum, Olaf** und **Gerndt, Andreas** (2012): Formal Verification in Early Mission Planning. Proceedings, Workshop on Simulation for European Space Programmes (SESP), Noordwijk, Niederlande, September 25 - 27, 2012
34. **Zhang, Hao** und **Gerndt, Andreas** und Liu, Bo (2012): Static Simulation Scheduling for the Validation of Space System Requirement Decomposition. Proceedings, 63rd International Astronautical Congress (IAC), Neapel, Italien, Oktober 1 - 5, 2012
35. **Westerteiger, Rolf** und **Gerndt, Andreas** und Hamann, Bernd (2012): Spherical Terrain Rendering using the hierarchical HEALPix grid. Proceedings, Visualization of Large and Unstructured Data Sets (VLUDS), IRTG 1131 Workshop 2011, Kaiserslautern, Juni 10 - 11, 2011, OASIcs, Vol. 27, Dagstuhl Publishing
36. **Hummel, Johannes** und **Wolff, Robin** und **Dodiya, Janki** und **Gerndt, Andreas** und Kuhlen, Torsten (2012): Towards Interacting with Force-Sensitive Thin Deformable Virtual Objects. Short Paper, Joint Virtual Reality Conference (JVRC), Madrid, Spanien, Oktober 16 - 19, 2012
37. **Schaus, Volker** und **Fischer, Philipp** und Quantius, Dominik und **Gerndt, Andreas** (2012): Automated Sensitivity Analysis in Early Space Mission Design. Proceedings, 5. International Workshop on Systems & Concurrent Engineering for Space Applications (SECESA), Lissabon, Portugal, Oktober 17 - 19, 2012

38. **Chen, Fang** und **Flatken, Markus** und **Basermann, Achim** und **Gerndt, Andreas** und Hetherington, James und Krüger, Timm und **Matura, Gregor** und Nash, Rupert (2012): Enabling In-situ Pre- and Post-Processing for Exascale Hemodynamic Simulations – A Co-Design Study with the Sparse Geometry Lattice Boltzmann Code HemeLB. Proceedings, Preparing Applications for Exascale Through Co-design, SC 2012 Workshop, 16. Nov. 2012, Salt Lake City, USA
39. Schumann, Ulrich und **Hempel, Rolf** und Flentje, Harald und Garhammer, Markus und Graf, Kaspar und Kox, Stephan und Lösslein, Heinz und Mayer, Bernhard (2013): Contrail study with ground-based cameras. Atmospheric Measurement Techniques Discussions (AMTD), 2013, 3597-3612, Copernicus Publication
40. **Fischer, Philipp** und **Lüdtke, Daniel** und **Schaus, Volker** und **Gerndt, Andreas** (2013): A Formal Method for Early Spacecraft Design Verification. Proceedings, IEEE Aerospace, Big Sky, Montana, März 2-9, 2013
41. Paris Lopez, Rosa und **Lüdtke, Daniel** und **Deshmukh, Meenakshi** und Soragavi, Geeta (2013): Knowledge Management Tools Integration within DLR's Concurrent Engineering Facility. IEEE Aerospace Conference 2013, Big Sky, Montana, USA, März 2-9, 2013
42. **Westerteiger, Rolf** und Streletz, Gregory und Kreylos, Oliver und Gebbie, Geoffrey A. und Spero, Howard J. und Kellogg, Louise H. und **Gerndt, Andreas** und Hamann, Bernd und Hagen, Hans (2013): Exploration of Time-dependent Paleoceanographic Flow Data in Virtual Reality. Extended Abstract, Online Proceedings, GeoViz Workshop, Hamburg, März 6-8, 2013
43. **Hummel, Johannes** und **Dodiya, Janki** und **Wolff, Robin** und **Gerndt, Andreas** und Kuhlen, Torsten (2013): An Evaluation of Two Simple Methods for Representing Heaviness in Immersive Virtual Environments. Proceedings, IEEE Symposium on 3D User Interfaces (3DUI), Orlando, FL, März 16-17, 2013
44. **Maibaum, Olaf** und **Gerndt, Andreas** (2013): Safe Integration of Payload Experiments in an AOCS. 9th IAA Symposium on Small Satellites for Earth Observation, Berlin, April 8 - 12, 2013
45. **Matura, Gregor** und **Basermann, Achim** und **Chen, Fang** und **Flatken, Markus** und **Gerndt, Andreas** und Hetherington, James und Krueger, Timm und Nash, Rupert (2013): A Pre-Processing Interface for Steering Exascale Simulations by Intermediate Result Analysis through In-Situ Post-Processing. Abstract, Exascale Application and Software Conference (EASC), Solving Software Challenges for Exascale, Edingburgh, Scotland, UK, April 9-11, 2013
46. **Schaus, Volker** und **Fischer, Philipp M.** und **Gerndt, Andreas** (2013): Taking Advantage of the Model: Application of the Quantity, Units, Dimension, and Values Standard in Concurrent Spacecraft Engineering. Proceedings, 23rd Annual INCOSE International Symposium, Philadelphia, PA, USA, Juni 24-27, 2013
47. **Maibaum, Olaf** und **Lüdtke, Daniel** und **Gerndt, Andreas** (2013): Tasking Framework: Parallelization of Computations in Onboard Control Systems. Ext. Abstract, ITG/GI Fachgruppentreffen Betriebssysteme, Berlin, Nov. 7-8, 2013
48. **Deshmukh, Meenakshi** und Adolf, Florian-Michael und Heisel, Maritta (2013): Pattern-based Requirements Model using SysML for a Helicopter's Pilot Assistance System. Proceedings, Deutscher Luft- und Raumfahrtkongress, Stuttgart, Sept. 10-12, 2013
49. **Schaus, Volker** und **Tiede, Michael** und **Fischer, Philipp M.** und **Lüdtke, Daniel** und **Gerndt, Andreas** (2013): A Continuous Verification Process in Concurrent Engineering. AIAA Space Conference, San Diego, CA, USA, Sept. 10-12, 2013
50. **Hummel, Johannes** und **Dodiya, Janki** und **Wolff, Robin** und **Gerndt, Andreas** und Kuhlen, Torsten (2013): Usability Evaluation of Two Simple Methods for Representating Heaviness of Virtual Objects. In M. E. Latoschik, O. Staadt, F. Steinicke (Eds.), Virtuelle und Erweiterte Realität, 10. Workshop der GI-Fachgruppe VR/AR, Würzburg, Shaker Verlag, Sept. 19-20, 2013
51. **Collienne, Peter** und **Wolff, Robin** und **Gerndt, Andreas** und Kuhlen, Torsten (2013): Physical Based Rendering of the Martian Atmosphere. In M. E. Latoschik, O. Staadt, F. Steinicke (Eds.), Virtuelle und Erweiterte Realität, 10. Workshop der GI-Fachgruppe VR/AR, Würzburg, Shaker Verlag, Sept. 19-20, 2013
52. **Thies, Jonas** und Galgon, Martin und Krämer, Lukas und Pieper, Andreas (2014): A 3D-Parallel Interior Eigenvalue Solver. Numerical Methods on High Performance Computers, 1.-3. Dez. 2014, Heidelberg, Deutschland.
53. Ilic, Caslav und Führer, Tanja und Banavara, Nagaraj und Abu-Zurayk, Mohammad und Einarsson, Gunnar und Kruse, Martin und Himisch, Jan und **Seider, Doreen** und Richard-Gregor, Becker (2014): Towards cooperative high-fidelity aircraft MDO: comparison of Breguet and ODE evaluation of the cruise mission segment. STAB Symposium 2014, 4-5 Nov 2014, München.
54. **Schreiber, Andreas** (2014): Open Source SW Development within DLR. 8th ESA Workshop on Avionics, Data, Control and Software Systems - ADCSS 2014, 27.-29. Okt. 2014, Noordwijk, Niederlande.
55. Wubs, Fred W**.** und **Thies, Jonas** und Song, Weiyan (2014): HYMLS: A Multilevel ILU approach for coupled uid and transport equations. European Multigrid Conference, EMG 2014, 8.-12. Sept. 2014, Leuven.

56. Führer, Tanja und Görtz, Stefan und Abu-Zurayk, Mohammad und Ilic, Caslav und Keye, Stefan und Banavara, Nagaraj und Kruse, Martin und Brodersen, Olaf und Liepelt, René und Becker, Richard-Gregor und Bach, Tobias und Jepsen, Jonas und Ciampa, Pier Davide und Kohlgrüber, D. und Scherer, Julian und Kier, Thiemo und Leitner, Martin und **Siggel, Martin** (2014): Entwicklung einer Softwareplattform für die Multidisziplinäre Optimierung eines Gesamtflugzeugs. 63. Deutscher Luft- und Raumfahrtkongress 2014, 16.-18. Sept. 2014, Augsburg, Deutschland.
57. **Zur, Sascha** und **Tröltzsch, Anke** (2014): Optimization of the DLR SpaceLiner inside the integration environment RCE. In: Engineering Optimization 2014, 1, Seiten 757-761. CRC Press 2014. EngOpt 2014, Lissabon. ISBN 978-1-138-02725-1.
58. **Schreiber, Andreas** und Galoppini, Roberto und **Meinel, Michael** und **Schlauch, Tobias** (2014): An Open Source Software Directory for Aeronautics and Space. In: Proceedings of The International Symposium on Open Collaboration, 46:1-46:7. ACM, New York, NY, USA. OpenSym'14, 27.-29. Aug. 2014, Berlin. DOI: 10.1145/2641580.2641630. ISBN 978-1-4503-3016-9.
59. **Schreiber, Andreas** (2014): Quantified Self: Analyzing the Big Data of our Daily Life. PyData Berlin 2014, 25.-27. Jul. 2014, Berlin.
60. Song, Weiyan und Wubs, Fred W. und **Thies, Jonas** (2014): A highly parallel code for strongly coupled fluid-transport equations. In: Proceedings of the 11th world congress on computational mechanics (WCCM XI), Seiten 199-210. WCCM XI, 2014, 20.-25. Juli, Barcelona, Spanien.
61. **Tröltzsch, Anke** (2014): Different Second Order Approximations in a model-based SQP Trust-Region DFO Method. In: Optimization 2014 Conference. University of Minho. Optimization 2014, 28.-30. Juli 2014, Guimaraes, Portugal.
62. **Schlauch, Tobias** und Beckmann, Dirk und Passchier, Igor und Zahariev, Nikola und Mikkelsen, Lars (2014): MOBiNET – an innovative approach for a European-wide ITS service platform. 10th ITS European Congress, 16–19 June 2014, Helsinki, Finland.
63. **Tröltzsch, Anke** (2014): An SQP trust-region algorithm for optimization without derivatives. In: SIAM Conference on Optimization. SIAM OP14, 19.-22. Mai 2014, San Diego, USA.
64. Alvermann, Andreas und **Basermann, Achim** und Fehske, Holger und Galgon, Martin und Hager, Georg und Kreutzer, Moritz und Krämer, Lukas und Lang, Bruno und Pieper, Andreas und **Röhrig-Zöllner, Melven** und Shahzad, Faisal und **Thies, Jonas** und Wellein, Gerhard (2014): ESSEX: Equipping Sparse Solvers for Exascale. In: Euro-Par 2014 Workshop Part II, 8806, Seiten 578-589. Springer. Euro-Par 2014, 25.-29. Aug. 2014, Porto, Portugal.
65. **Lüdtke, Daniel** und Westerdorff, Karsten und Stohlmann, Kai und Boerner, Anko und **Maibaum, Olaf** und **Peng, Ting** und **Weps, Benjamin** und Fey, Görschwin und **Gerndt, Andreas** (2014): OBC-NG: *Towards a Reconfigurable On-board Computing Architecture for Spacecraft.* Proceedings, IEEE Aerospace Conference 2014, Big Sky, Montana, USA, März 1-8, 2014
66. **Deshmukh, Meenakshi** und Schwarz, René und Braukhane, Andy und Paris Lopez, Rosa und **Gerndt, Andreas** (2014): *Model Linking to Improve Visibility and Reusability of Models during Space System Development.* Proceedings, IEEE Aerospace Conference 2014, Big Sky, Montana, USA, März 1-8, 2014
67. **Hotz, Ingrid** und Kratz, Andrea und Schoeneich, Marc und Zobel, Valentin und Burgeth, Bernhard und Scheuermann, Gerik und Stommel, Markus (2014): *Tensor Visualization Driven Mechanical Component Design.* Full Paper, IEEE Pacific Visualization Symposium (PacificVis), Yokohama, Japan, März 4-7, 2014
68. Bujack, Roxana und **Hotz, Ingrid** und Scheuermann, Gerik und Hitzer, Eckhard (2014): *Moment invariants for 2D flow fields using normalization.* Full Paper, Best Paper Award, IEEE Pacific Visualization Symposium (PacificVis), Yokohama, Japan, März 4-7, 2014
69. **Flatken, Markus** und **Gerndt, Andreas** (2014): *Hybrid Rendering: Enabling Interactivity in a Distributed Post-Processing Environment.* Extended Abstract, Exascale Application and Software Conference (EASC), Stockholm, Schweden, Apr. 2-4, 2014
70. Stohlmann, Kai und Fey, Görschwin und **Lüdtke, Daniel** (2014): *Automatic Performance Tracking of a SpaceWire Network.* Proceedings, International SpaceWire Conference 2014, Athen, Griechenland, Sept. 22-26, 2014
71. **Aslandere, Turgay** und Dreyer, Daniel und Pantkratz, Frieder und Schubotz, Rene (2014): *A Generic Virtual Reality Flight Simulator.* In Proceedings, GI-Workshop AR/VR, Bremen, Sept. 25-26, 2014
72. Schöneich, Marc und Stommel, Markus und Kratz, Andrea und Zobel, Valentin und Scheuermann, Gerik und **Hotz, Ingrid** und Burgeth, Bernhard (2014): *Optimization strategy for the design of ribbed plastic components.* Zeitschrift Kunststofftechnik / Journal of Plastics Technology, Vol. 10, Issue 4, Seiten 160 - 175, Carl-Hanser-Verlag, 2014
73. Heidecker, Ansgar und Kato, Takahiro und **Maibaum, Olaf** und Hölzel, Metthew (2014): *Attitude Control System of the Eu:CROPIS Mission.* In Proceedings, 65th International Astronautical Congress (IAC), Toronto, Kanada, 29.09.-03.10, 2014
74. **Schaus, Volker** und **Müller, Juliane** und **Deshmukh, Meenakshi** und Braukhane, Andy und **Gerndt, Andreas** (2014): *Bidirectional Graphical Modelling Supporting Concurrent Spacecraft Design.* In Proceedings, 6th International Conference on Systems & Concurrent Engineering for Space Applications (SECESA), Okt. 8-10, Stuttgart, Germany, 2014

75. Gianni, Daniele und **Schaus, Volker** und D'Ambrogio, Andrea und **Gerndt, Andreas** und Lisi, Marco und De Simone, Pierluigi (2014): Model-based Interface Engineering in Concurrent Engineering Facilities: *Motivations and Possible Applications to Systems and Service Systems Engineering.* In Proceedings, 6th International Conference on Systems & Concurrent Engineering for Space Applications (SECESA), Okt. 8-10, Stuttgart, 2014
76. **Schaus, Volker** und **Lüdtke, Daniel** und **Gerndt, Andreas** (2014): *Advanced Spacecraft Systems Design using Model-based Techniques.* Extended Abstract, 2nd Federated Satellite Systems Workshop, Moskau, Russland, Okt. 13-15, 2014
77. **Lüdtke, Daniel** und **Schaus, Volker** und **Gerndt, Andreas** (2014): *Space Cloud: From a Distributed On-board Computer to a Federated System-of–Systems in Space.* Extended Abstract, 2nd Federated Satellite Systems Workshop, Moskau, Russland, Okt. 13-15, 2014
78. **Chen, Fang** und **Flatken, Markus** und **Hotz, Ingrid** und **Gerndt, Andreas** (2014): *In-situ Processing and Interactive Visualization for Large-Scaled Numerical Simulations.* Poster / Extended Abstract, IEEE Large Data Analysis and Visualization (LDAV), Paris, Frankreich, Nov. 9-10, 2015
79. Bujack, Roxana und **Hotz, Ingrid** und Kasten, Jens und Scheuermann, Gerik und Hitzer, Eckhard (2014): *Moment Invariants for 3D Flow Fields.* Poster, SCIVIS HONORABLE MENTION AWARD, IEEE Visualization, Paris, Frankreich, Nov. 9-14, 2014
80. Gianni, Daniele und D'Ambrogio, Andrea und **Schaus, Volker** und **Gerndt, Andreas** und Lisi, Marco und De Simone, Pierluigi (2014): *Interface Management in Concurrent Engineering Facilities for Systems and Service Systems Engineering: A Model-based Approach.* In Proceedings, INCOSE Italia Conference on Systems Engineering (CIISE 2014), Rom, Italien, 24. – 25. November 2014

Nicht begutachtete Veröffentlichungen

1. **Schaus, Volker** und **Fischer, Philipp M.** (2011): *Ein Werkzeug für alle - Software für den Satellitenentwurf.* DLR Magazin, No. 129, Seiten 20-23, April 2011.
2. **Basermann, Achim** und **Kersken, Hans-Peter** und **Zöllner, Melven** (2011): Scalable Preconditioned Solvers for Internal and External Flow Computations on Many-Core Systems. 17th Conference of the International Linear Algebra Society, 22.-26.08.2011, Braunschweig, Deutschland.
3. **Hempel, Rolf** (2013): Panoramafotografie des Mondes. Sterne und Weltraum, 2013 (10), 78-85, Spektrum der Wissenschaft Verlagsgesellschaft mbH
4. **Schreiber, Andreas** (2013): Software mit offenen Quellen. DLR Magazin 138, Seiten 32-35. DLR. ISSN 2190-0094.
5. **Schreiber, Andreas** (2013): Quantified Self: Self Tracking for Health. 4th International TEMOS CONFERENCE, 1.-3. Dez. 2013, Bonn.
6. **Tröltzsch, Anke** (2013): A new Algorithm for Equality- and Bound-Constrained DFO. In: Recent Advances on Optimization. Recent Advances on Optimization, 24.-26. Juli 2013, Toulouse, Frankreich.
7. **Hempel, Rolf** (2014): Endlich scharf sehen am Horizont – Korrektur der atmosphärischen Dispersion. Sterne und Weltraum, 2014 (6), 66-74, Spektrum der Wissenschaft Verlagsgesellschaft mbH
8. **Schreiber, Andreas** und Lindlar, Markus (2014): Telemedizin für jedermann. DLR Magazin, 143, Seiten 44-47. DLR. ISSN 2190-0094.
9. **Gerndt, Andreas** und **Dodiya, Janki** und Hertkorn, Katharina und Hulin, Thomas und **Hummel, Johannes** und Sagardia, Mikel und **Wolff, Robin** (2014): *Fallbeispiele für VR/AR - Virtuelle Satellitenreparatur im Orbit.* In: R. Dörner, W. Broll, P. Grimm, B. Jung (Eds.), Virtual und Augmented Reality (VR / AR) - Grundlagen und Methoden der Virtuellen und Augmentierten Realität, eXamen.press. Springer Vieweg, Seiten 303-305, Febr. 2014
10. Grimm, Paul und Herold, Rigo und **Hummel, Johannes** und Broll, Wolfgang (2014): *VR-Eingabegeräte.* In: R. Dörner, W. Broll, P. Grimm, B. Jung (Eds.), Virtual und Augmented Reality (VR / AR) - Grundlagen und Methoden der Virtuellen und Augmentierten Realität, eXamen.press. Springer Vieweg, Seiten 97-125, Febr. 2014
11. Bremer, Peer-Timo und **Hotz, Ingrid** und Pascucci, Valerio und Peikert, Ronald (Eds.) (2014): *Topological Methods in Data Analysis and Visualization III.* (TopoInVis 2013 Proceedings), Theory, Algorithms, and Applications, Mathematics and Visualization, Springer, Mai 2014

12. Zobel, Valentin und Reininghaus, Jan und **Hotz, Ingrid** (2014): *Visualization of Two-Dimensional Symmetric Tensor Fields Using the Heat Kernel Signature.* In: P.-T. Bremer, I. Hotz, V. Pascucci; R. Peikert (Eds.), „Topological Methods in Data Analysis and Visualization III", (TopoInVis'13), Springer, Seiten 249-262, Mai 2014
13. Kratz, Andreas und Auer, Cornelia und **Hotz, Ingrid** (2014): *Tensor Invariants and Glyph Design.* In: Carl-Fredrik Westin, Anna Vilanova, Bernhard Burgeth (Eds.), „Visualization and Processing of Tensors and Higher Order Descriptors for Multi-Valued Data", Mathematics and Visualization, Springer, Seiten 17-34, Aug. 2014
14. Hlawitschka, M. und **Hotz, I.** und Kratz, A. und Marai, L. und Scheuermann, G. und Stommel, M. und Wiebel, A. und Zhang, E. (2014): *Top Challenges in the Visualization of Engineering Tensor Fields.* In: Carl-Fredrik Westin, Anna Vilanova, Bernhard Burgeth (Eds.), „Visualization and Processing of Tensors and Higher Order Descriptors for Multi-Valued Data", Mathematics and Visualization, Springer, Seiten 3-15, Aug. 2014
15. **Hotz, Ingrid** und Peikert, Ronald (2014): *Definition of a Multifield.* In: M. Chen, H. Hagen, C. Hansen, C. Johnson, and A. Kaufman (Eds), „Scientific Visualization: Uncertainty, Multifield, Biomedical, and Scalable Visualization", Mathematics and Visualization, Springer, London, Seiten 105-109, Sept. 2014
16. **Flatken, Markus** und **Wagner, Christian** und **Gerndt, Andreas** (2014): *Distributetd Post-Processing and Rendering for Large-Scale Scientific Simulations.* In: M. Chen, H. Hagen, C. Hansen, C. Johnson, and A. Kaufman (Eds), „Scientific Visualization: Uncertainty, Multifield, Biomedical, and Scalable Visualization", Mathematics and Visualization, Springer, London, Seiten 381-398, Sept. 2014
17. **Engelke, Wito** und Kuhn, Alexander und **Flatken, Markus** und **Chen, Fang** und Hege, Hans-Christian und **Gerndt, Andreas** und **Hotz, Ingrid** (2014): *Atmospheric Impact of Volcano Eruptions.* Contest Contribution, Proceedings, IEEE Visualization, IEEE Scientific Visualization Contest, Paris, France, Nov. 9-14, 2014
18. **Schaus, Volker** und **Lüdtke, Daniel** und **Fischer, Philipp** und **Gerndt, Andreas** (2014): *Collaborative Modeling and Simulation in Spacecraft Design.* In: Daniele Gianni, Andrea D'Ambrogio, Andreas Tolk (Eds.), „Modeling and Simulation-based Systems Engineering Handbook", Chapter 7, CRC Press, Seiten 149-182, Dez. 2014
19. **Wubs, Fred W.** und Song, Weiyan und **Thies, Jonas** (2014): HYMLS: a parallel solver for steady coupled fluid-transport problems. PMAA 2014, 2.-4. Juli 2014, Lugano, Schweiz.
20. **Tröltzsch, Anke** und **Siggel, Martin** und Kopp, Alexander und Schwanekamp, Tobias (2014): Multidisciplinary Analysis of the DLR SpaceLiner Concept by different Optimization Techniques. In: 5th European Conference on Computational Mechanics. International Center for Numerical Methods in Engineering. 11th World Congress on Computational Mechanics (WCCM XI), 20.-25. Juli 2014, Barcelona, Spanien. ISBN 978-84-942844-7-2.
21. Schwanekamp, Tobias und Meyer, Frank und Reimer, Thomas und Petkov, Ivaylo und **Tröltzsch, Anke** und **Siggel, Martin** (2014): System Studies on Active Thermal Protection of a Hypersonic Suborbital Passenger Transport Vehicle. In: 19th AIAA International Space Planes and Hypersonic Systems and Technologies Conference. AIAA. 19th AIAA International Space Planes and Hypersonic Systems and Technologies Conference, 16.-20. Juni 2014, Atlanta, GA, USA.
22. Tiefers, Rüdiger und Aguilar, Julio und Dresbach, Christian und Buske, Clemens und Schmidt, Thomas und **Zur, Sascha** (2014): Titanium Aluminide Turbine Toolbox – A new tool for optimizing the design process for gamma-TiAl low pressure turbine blades. 28th International Congress ‚Excellence in the art of Investment Casting', 15.-18. Jun. 2014, Lugano, Switzerland.

Das DLR im Überblick

Das DLR ist das nationale Forschungszentrum der Bundesrepublik Deutschland für Luft- und Raumfahrt. Seine umfangreichen Forschungs- und Entwicklungsarbeiten in Luftfahrt, Raumfahrt, Energie, Verkehr und Sicherheit sind in nationale und internationale Kooperationen eingebunden. Über die eigene Forschung hinaus ist das DLR als Raumfahrt-Agentur im Auftrag der Bundesregierung für die Planung und Umsetzung der deutschen Raumfahrtaktivitäten zuständig. Zudem fungiert das DLR als Dachorganisation für den national größten Projektträger.

In den 16 Standorten Köln (Sitz des Vorstands), Augsburg, Berlin, Bonn, Braunschweig, Bremen, Göttingen, Hamburg, Jülich, Lampoldshausen, Neustrelitz, Oberpfaffenhofen, Stade, Stuttgart, Trauen und Weilheim beschäftigt das DLR circa 8.000 Mitarbeiterinnen und Mitarbeiter. Das DLR unterhält Büros in Brüssel, Paris, Tokio und Washington D.C.

Die Mission des DLR umfasst die Erforschung von Erde und Sonnensystem und die Forschung für den Erhalt der Umwelt. Dazu zählt die Entwicklung umweltverträglicher Technologien für die Energieversorgung und die Mobilität von morgen sowie für Kommunikation und Sicherheit. Das Forschungsportfolio des DLR reicht von der Grundlagenforschung bis zur Entwicklung von Produkten für morgen. So trägt das im DLR gewonnene wissenschaftliche und technische Know-how zur Stärkung des Industrie- und Technologiestandorts Deutschland bei. Das DLR betreibt Großforschungsanlagen für eigene Projekte sowie als Dienstleistung für Kunden und Partner. Darüber hinaus fördert das DLR den wissenschaftlichen Nachwuchs, betreibt kompetente Politikberatung und ist eine treibende Kraft in den Regionen seiner Standorte.

Statusbericht_D_03/15

DLR **Deutsches Zentrum für Luft- und Raumfahrt**

Simulations- und Softwaretechnik
Linder Höhe
51147 Köln

DLR.de