Informatik für die Welt von Morgen

Keynote

PyCon DE 2011 (07.10.2011, Leipzig)

Andreas Schreiber *Andreas.Schreiber@dlr.de*
Deutsches Zentrum für Luft- und Raumfahrt e.V. (DLR)
Berlin-Adlershof / Braunschweig / Köln-Porz
http://www.dlr.de/sc

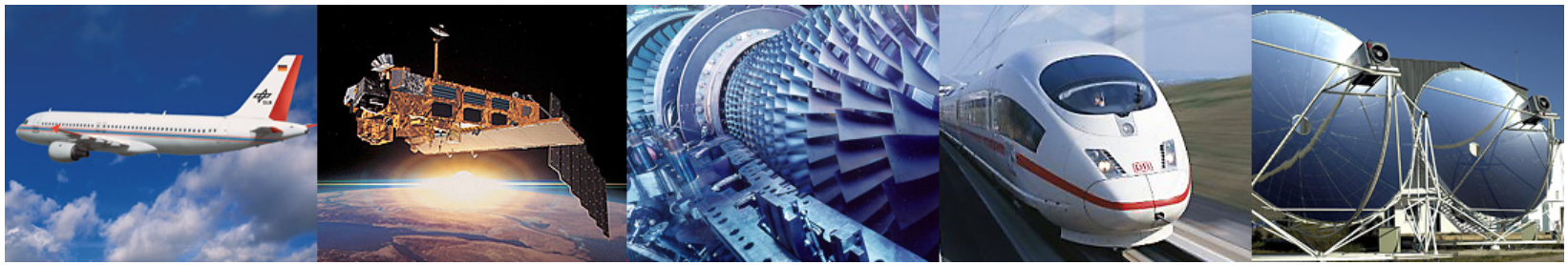
Informatik für die Welt von Morgen

Überblick

- Das DLR
- Software im DLR
- Informatik und Softwaretechnologie
- Python auf alle Systeme!
- Hinweise


DLR

Das DLR
Deutsches Zentrum für Luft- und Raumfahrt



- Forschungseinrichtung
- Raumfahrt-Agentur
- Projektträger

Standorte und Personal

6.900 Mitarbeiterinnen und Mitarbeiter arbeiten in 32 Instituten und Einrichtungen in

- 15 Standorten.

Büros in Brüssel, Paris und Washington.

Mission

- **Erforschung der Erde und des Sonnensystems,**
- **Forschung zum Erhalt der Umwelt, zur Mobilität, zur Gewährleistung der Sicherheit**
- **Forschung zur Bearbeitung gesellschaftlicher Fragen im öffentlichen Auftrag**

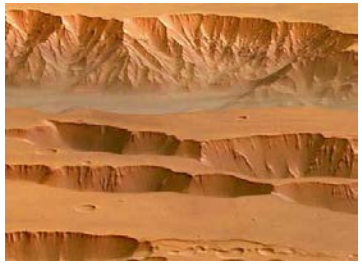
- Brückenfunktion von Grundlagenforschung und innovativen Anwendungen sowie Transfer von Wissen und Forschungsergebnissen zu Industrie und Politik durch Vermittlung, Beratung sowie Dienstleistungen
- Gestaltung des deutschen Raumfahrtengagements und internationale Interessenvertretung als hoheitliche Aufgabe
- Leistung eines signifikanten Beitrags zum Wissenschafts- und Wirtschaftsstandort Deutschland und zum europäischen Wachstumsraum
- Ausbildung des wissenschaftlichen Nachwuchses zur Steigerung der Innovationsfähigkeit Deutschlands

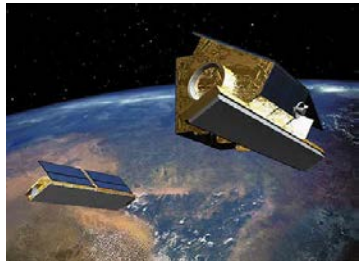
DLR Forschungsbereich Luftfahrt

- Optimierung der Leistung und der Umweltverträglichkeit des Gesamtsystems „Flugzeug“
- Erweiterung des Flugbereichs von Hubschraubern auf alle Wetterbedingungen
- Effiziente und umweltfreundliche Flugtriebwerke
- Sicherer, umweltfreundlicher und effizienter Luftverkehr (Flugsicherung, Flugbetrieb)

DLR Forschungsprogramm Raumfahrtforschung und -technologie

- Erforschung des Weltraums
- Forschung unter Schwerelosigkeit
- Erdbeobachtung
- Kommunikation & Navigation
- Raumtransport
- Technik für Raumfahrtsysteme



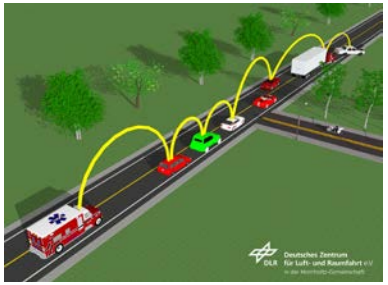



DLR Forschungsbereich Verkehr

- Nachhaltige Mobilität erreichen in einer Balance von
  - Ökonomie
  - Gesellschaft
  - Ökologie

durch

- Sicherung der Mobilität für Menschen und Güter
- Schutz von Umwelt und Ressourcen
- Verbesserung der Sicherheit



DLR Forschungsbereich Energie

Der DLR Forschungsbereich Energie konzentriert sich auf

- CO2-Vermeidung durch Effizienz und Erneuerbare Energien
- Synergien im DLR
- energiewirtschaftlich relevante und großforschungsspezifische Themen.

Die Welt von Morgen

Die Entwicklungen beeinflussen unsere Welt von Morgen...

Neue Flugzeuge, neue Verkehrs-konzepte, Raumfahrt, ...

Informatik und Software hat entscheidenden Anteil

Software im DLR

Deutsches Zentrum
für Luft- und Raumfahrt e.V.
in der Helmholtz-Gemeinschaft

Software im DLR

Größenordnung der Software-Entwicklung

Über 1000 Mitarbeiter des DLR entwickeln Software

Das sind >100 Millionen EUR Vollkosten pro Jahr

DLR ist eines der größten Software-Häuser Deutschlands

Software im DLR
Individualsoftware

- Ein Großteil der entwickelten Software im DLR ist Individualsoftware
- Entwicklung gemäß der speziellen Anforderungen des DLR

- Gründe
  - Es gibt keine verfügbare geeignete Standardsoftware am Markt
  - Es gibt Standardsoftware, die Individualsoftware wird jedoch monetär günstiger bewertet
  - Man möchte vollständige Kontrolle über die weitere Entwicklung
  - Die Software soll einen Wettbewerbsvorteil verschaffen
  - ***„Das Rad neu erfinden“:*** Die angestrebte Lösung soll noch besser werden, als die verfügbare Standardsoftware

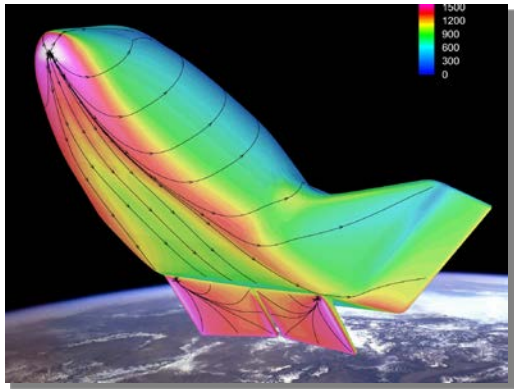
Software in der Luft- und Raumfahrt

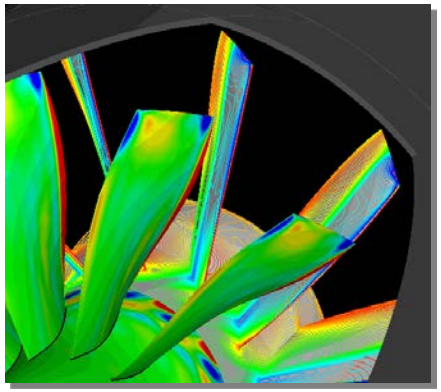
Software mit hoher Kritikalität

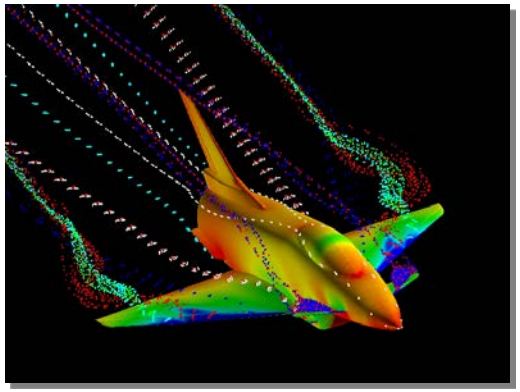
- Echtzeitfähige Software und Software für eingebettete Systeme
  - Bsp.: Lageregelungssysteme für Flugzeuge und Raumfahrzeuge
- Hohe Anforderungen an Ausfallsicherheit und Fehlerfreiheit
- Steuert oft technische Systeme
- Häufig ist das Leben von Menschen von ihr abhängig

Software in der Luft- und Raumfahrt
Simulationssoftware

- Simulation physikalischer Vorgänge oder komplexer Systeme
  - Bsp.: Numerische Strömungssimulation
- Oft hohe Anforderungen an Genauigkeit und Performanz
- Ausführung auf High-Performance-Computing-Systemen („Supercomputing“)
- Erzeugt oft große Datenmengen



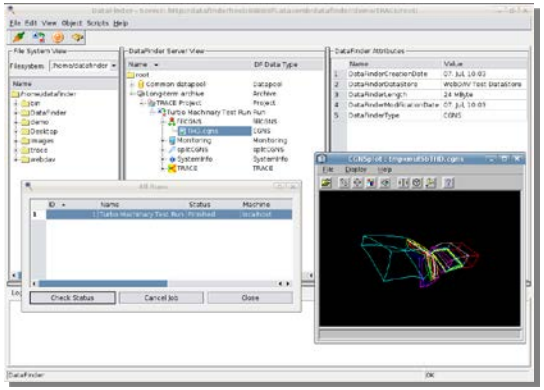


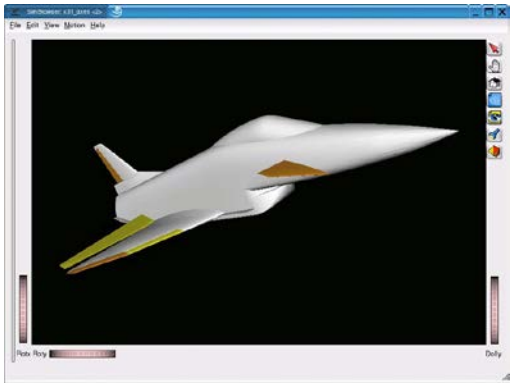


Software in der Luft- und Raumfahrt

Unterstützende Software

- Unterstützt die Arbeit der Wissenschaftler
- Erhöht die Produktivität
- Beispiele:
  - Verwaltung von wissenschaftlichen Daten
  - Wissensmanagement und Expertensysteme
  - Grafische Auswertung und Visualisierung

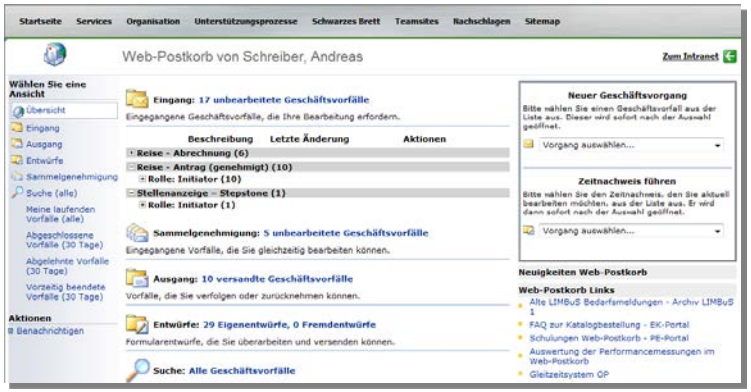




Software in der Luft- und Raumfahrt

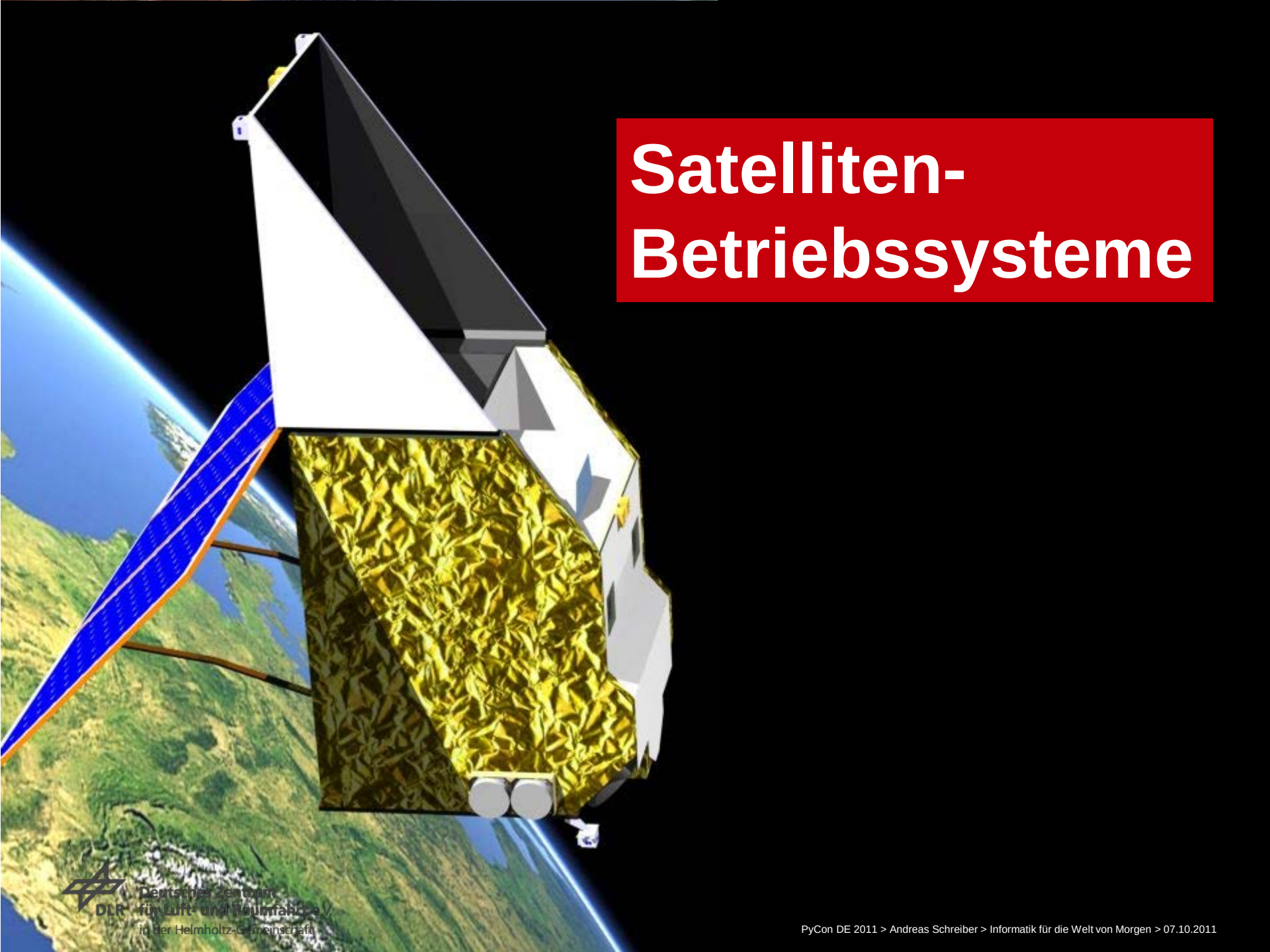
Administrative Software (SW für große Nutzerzahlen)

- Häufig Web-basierte Software für Internet oder Intranet
- Oft mit Anbindung an Unternehmenssoftware (SAP)
- Beispiele:
  - Beantragen von Reisen oder Urlaub
  - Verwaltung von IT-Ressourcen
  - Information der Öffentlichkeit



Beispiele für Software-Anwendungen

Management wiss. Daten von
- Simulationen
- Experimenten
- Erdbeobachtungen



Satelliten-Betriebssysteme

Deutsches Zentrum
DLR für Luft- und Raumfahrt e.V.
in der Helmholtz-Gemeinschaft

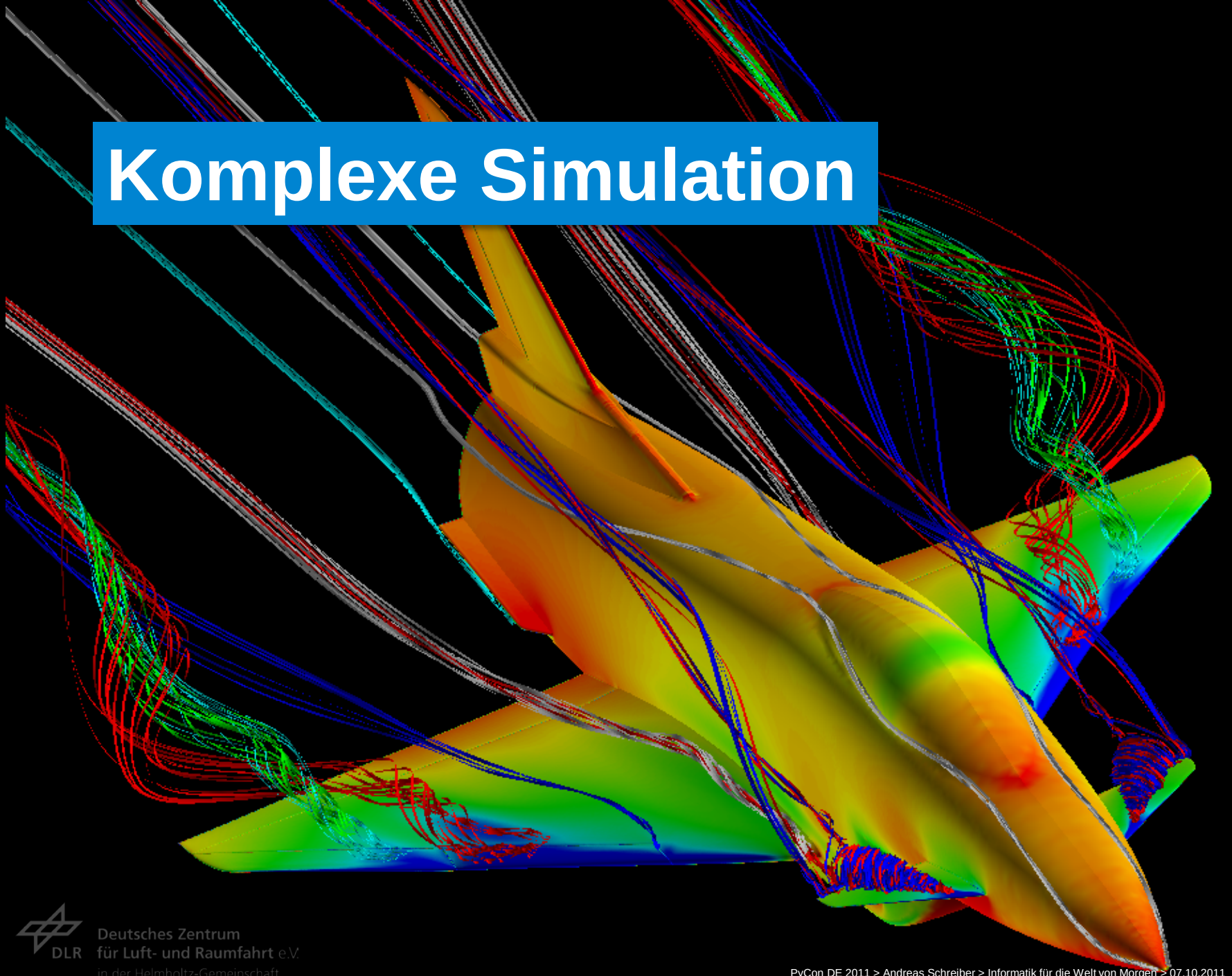
Klima-Forschung

Deutsches Zentrum
für Luft- und Raumfahrt e.V.
in der Helmholtz-Gemeinschaft

Simulation und Management … von Luft-Verkehr

Simulation und Management ... von Straßenverkehr

Komplexe Simulation



Deutsches Zentrum
für Luft- und Raumfahrt e.V.
in der Helmholtz-Gemeinschaft

Bild: P. Winandy

DLR

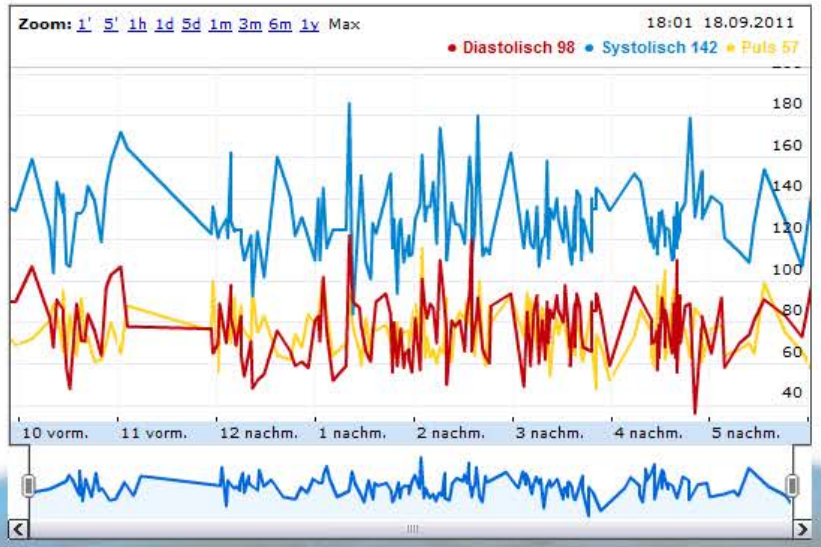
BLUTDRUCK KÖRPERGEWICHT STATISTIKEN

Deutsch

Home
» Blutdruck

Blutdruck & Puls
Körpergewicht
Statistiken

Blutdruck & Puls



Filter Tageszeiten

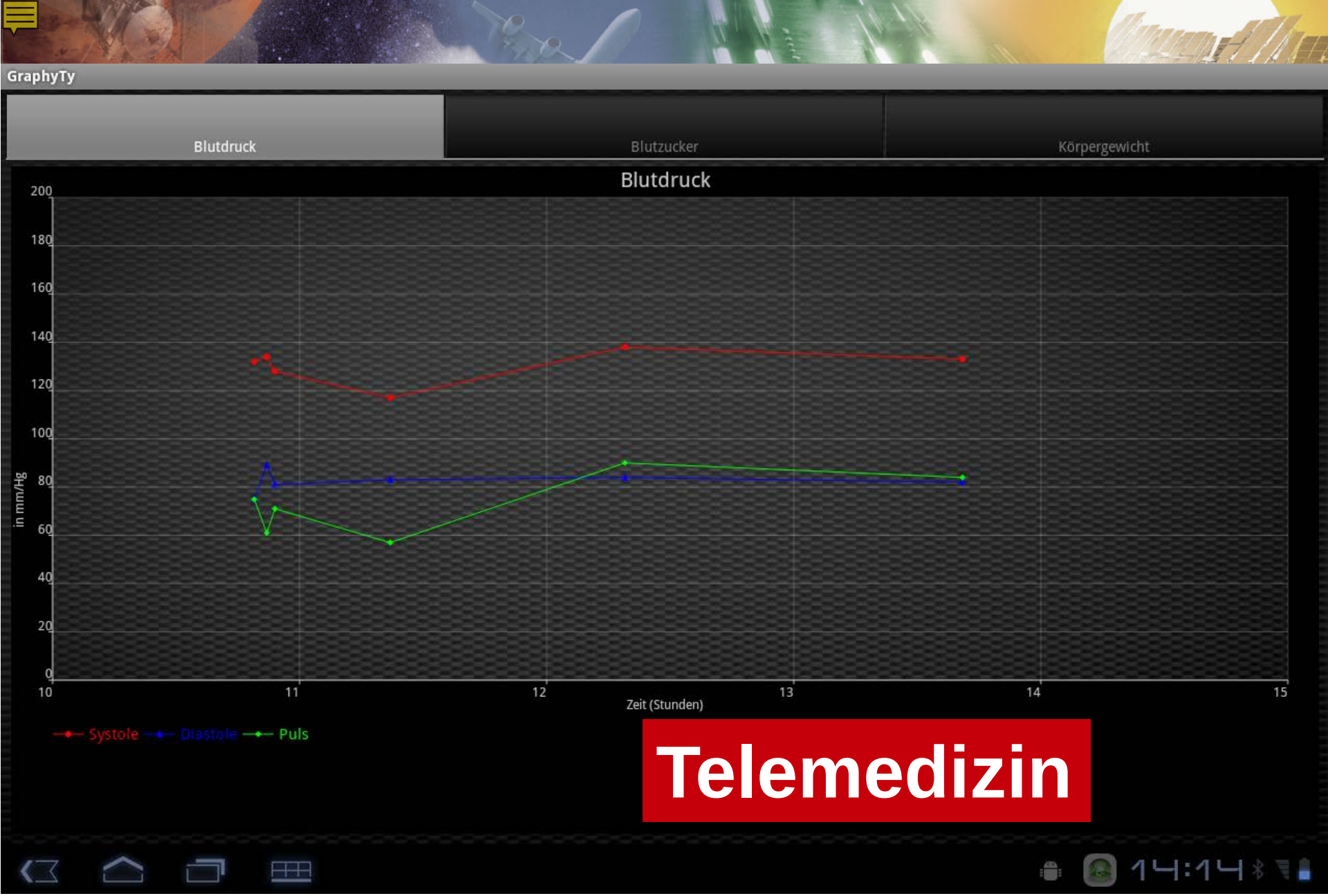
Jede Tageszeit

Telemedizin

Deutsches Zentrum
DLR für Luft- und Raumfahrt

DLR
Tag der Luft- und Raumfahrt
DLR Simulations- und Softwaretechnik
DLR Software Portal

Home Twitter Admin Kontakt Impressum


GraphyTy
Blutdruck
Blutzucker
Körpergewicht
Blutdruck
200
180
160
140
120
100
80
60
40
20
0
in mm/Hg
10
11
12
13
14
15
Zeit (Stunden)
Systole
Diastole
Puls
Telemedizin
14:14


Informatik und
Softwaretechnologie

Informatik für Wissenschaftler und Ingenieure

Software zum Lösen von Problemen

Wissenschaftler und Ingenieure wollen eigentlich keine Software entwickeln sondern ihre Probleme lösen

Möglichst schnelles Umsetzen ihrer Ideen in laufenden Code

Wenn sie Code schreiben, sollte es so einfach wie möglich sein

Informatik für Wissenschaftler und Ingenieure
Randbedingungen und Anforderungen

Informatik und Softwaretechnologie müssen die Anwender effektiv unterstützen

Notwendig sind benutzbare, einfach zugängliche Softwaretechnologien und nahtlos Integration in bestehende Arbeitsumgebungen

Die wissenschaftliche Freiheit und Kreativität darf nicht behindert werden

Informatik für Wissenschaftler und Ingenieure

Beispiele für notwendige Software(-technologien)

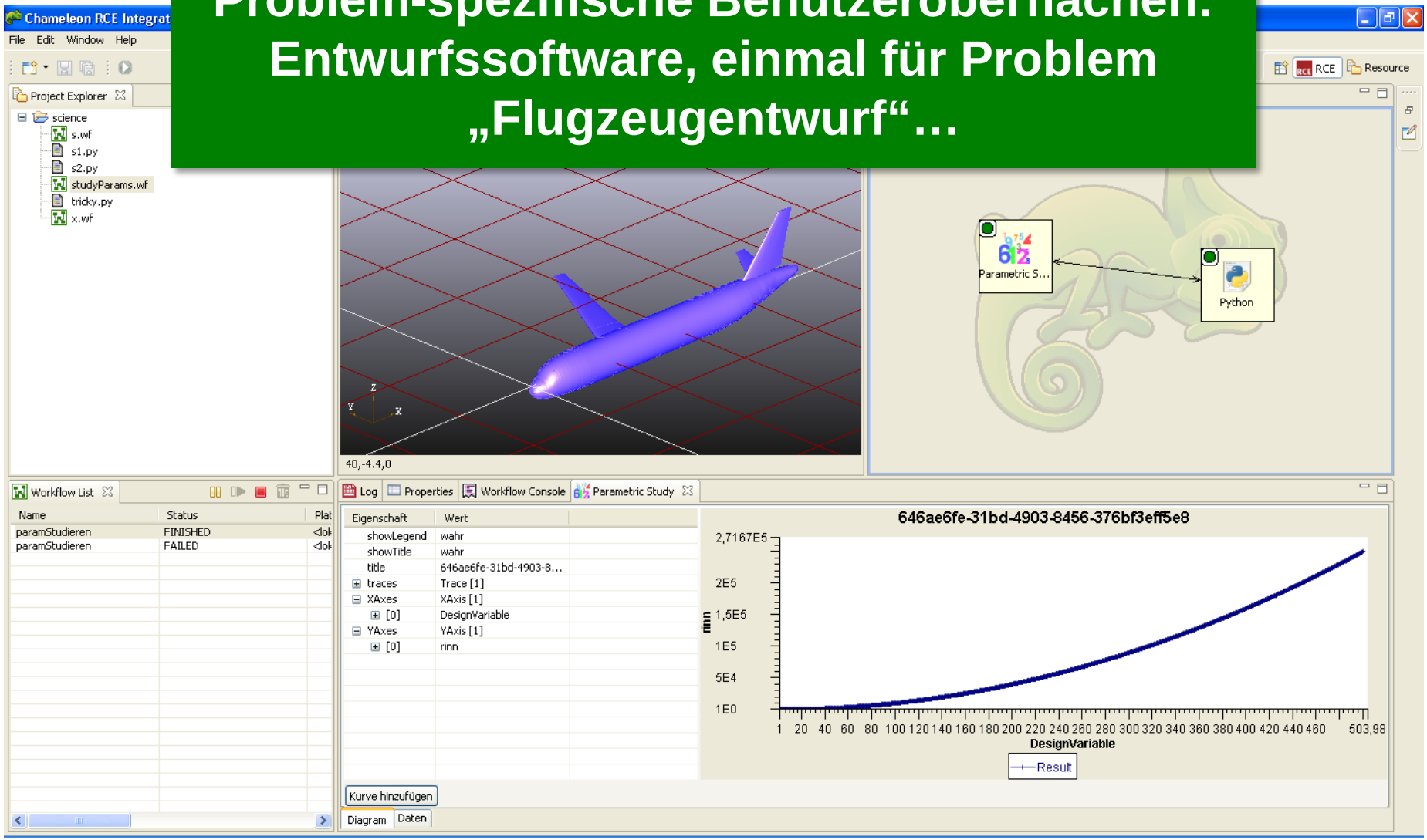
Usability

Problem-spezifische Benutzeroberflächen

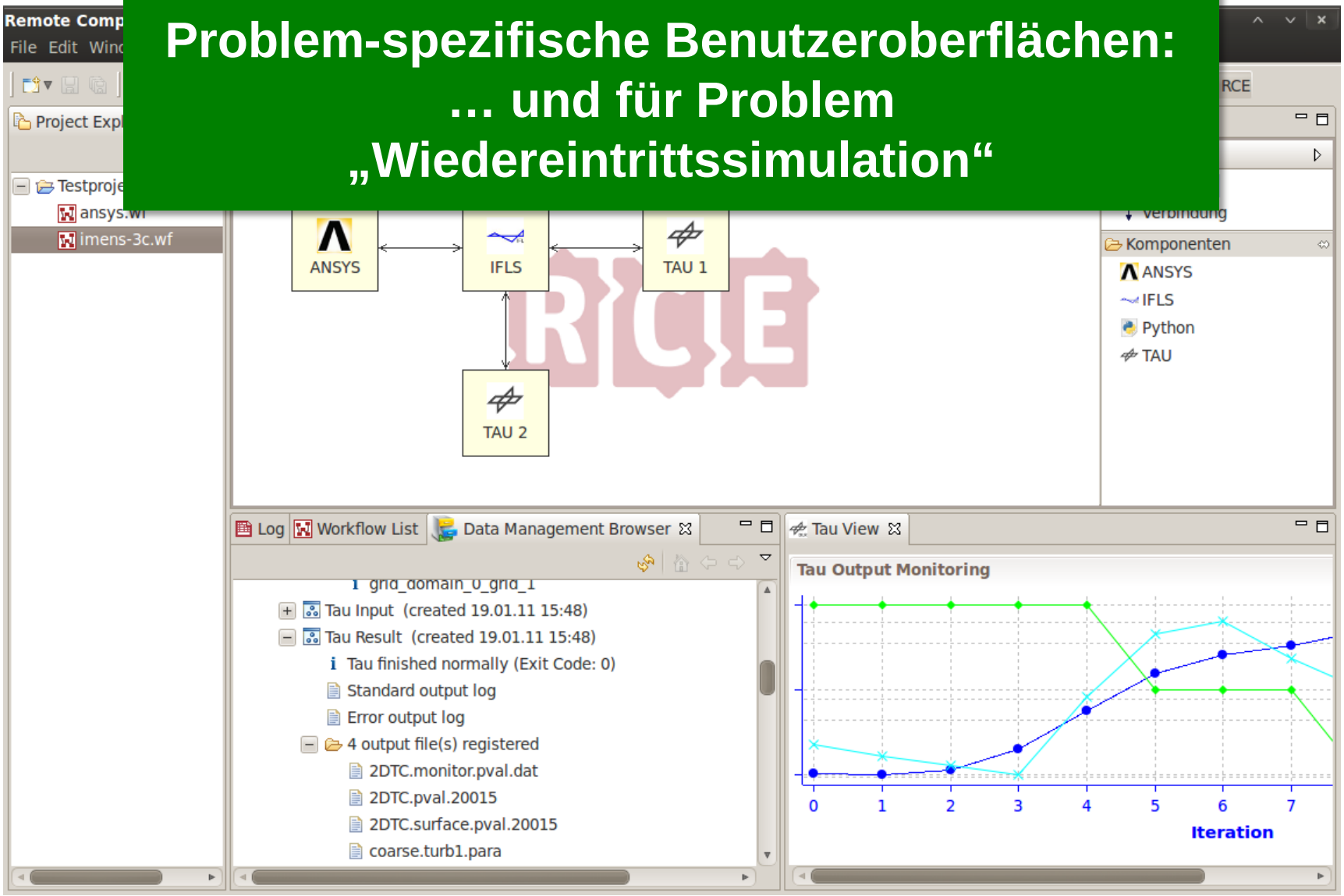
(Schmerzfreies) Software Engineering

Leicht zu erlernende Programmiersprachen

Problem-spezifische Benutzeroberflächen: Entwurfssoftware, einmal für Problem „Flugzeugentwurf“…



Problem-spezifische Benutzeroberflächen: … und für Problem „Wiedereintrittssimulation“



Software Engineering

- **Software Engineering wichtig für Software von hoher Qualität**
  - Softwareentwicklungsprozesse
  - Geeignete Entwicklungswerkzeuge
  - Software-Tests

- **Softwareentwicklung durch Wissenschaftler und Ingenieure**
  - Oft nur Mittel zum Zweck
  - Jedoch erheblicher Anteil an täglicher Arbeitszeit
  - Software-Engineering-Technologien sollen Produktivität und Qualität verbessern, aber wissenschaftliche Arbeit nicht behindern

Software Engineering
Herausforderung der Zukunft

Wie macht man Wissenschaftlern Software Engineering schmackhaft?

Wird immer wichtiger, weil…

Software Engineering

Software entwickeln ist Teamarbeit

- Software wird im DLR meist in **interdisziplinären Teams** entwickelt
- Informatiker entwickeln gemeinsam mit Ingenieuren, Mathematikern, Physikern, Chemikern, Medizinern, Ökonomen, …
- Informatiker bekommen viele Einblicke in verschiedenste Fachdisziplinen

Software Engineering
Notwendige Kompetenzen für Informatiker

- Gute Kommunikation in Entwicklungsprojekten notwendig für erfolgreiche Resultate
- Interesse, Lust und Spaß am kommunizieren mit „fremden“ Fachdisziplinen ist daher sehr wichtig
- Natürlich bleibt: Programmieren selber muß Spaß machen!!
  - Z.B. durch leicht zu erlernende Programmiersprachen…

Leicht zu erlernende Programmiersprachen
Die Vielfalt der Sprachen…

- In Praxis viele Sprachen im Einsatz
- Im DLR allein ca. 30 Sprachen
- Oft eng begrenzte Anwendungsgebiete

- Viele Faktoren beeinflussen die Wahl der Sprache:
  - Anforderungen
  - Ziel-Plattformen und Plattformunabhängigkeit
  - Vorhandene Software
  - Vorhandenes Know-How (Personal!)
  - Performanz
  - Erlernbarkeit
  - …

Programmiersprachen im DLR

Leicht zu erlernende Programmiersprachen
Die Sprache der Wahl für Wissenschaftler: **Python**

- Allgemein verwendbare Skriptsprache
- Sehr leicht zu erlernen und einfach zu benutzen (= *steile Lernkurve*)
- Rapid Application Development (= *kurze Entwicklungszeit*)
- ***„Inherent great maintainability“*** (= *Investitionsschutz*)
- Sehr geeignete Lehr- und Einsteigersprache

“There seems to be two sorts of people who love Python: those who hate brackets, and scientists.

“If it’s good enough for Google and NASA, it’s good enough for me, baby.

“Python has the cleanest, most-scientist- or engineer friendly syntax and semantics.

Paul F. Dubois

Leicht zu erlernende Programmiersprachen
Python-Beispiel

```
def fakultaet(x):
    if x > 1:
        return x * fakultaet(x - 1)
    else:
        return 1
```

Python für Wissenschaftler
NumPy

ENTHOUGHT
SCIENTIFIC COMPUTING SOLUTIONS

- Website: http://numpy.scipy.org/
- Bietet Funktionalität wie MATLAB… aber in Python!
  - Lineare Algebra, FFT, Data I/O, Grafik, …
- Zum Beispiel N-dimensionale Arrays (ndarray)

```
>>> a = array([[ 0, 1, 2, 3],
               [10,11,12,13]])
>>> a
array([[ 0, 1, 2, 3],
       [10,11,12,13]])
```

```
>>> a[1,3]
13
>>> a[1,3] = -1
>>> a
array([[ 0, 1, 2, 3],
       [10,11,12,-1]])
```

SciPy
Scientific Tools for Python

- Website: http://www.scipy.org
- Große Bibliothek wissenschaftlicher Algorithmen
- Erweitert NumPy um viele Tools für Forschung und Entwicklung

MATLAB?

- Quasi-Standard für Auswertung und Visualisierung von Daten, mathematische Modellierung und Entwicklung von Algorithmen

Prognose:
Python w/SciPy wird MATLAB in vielen Bereichen ablösen

Python auf alle Systeme!

Deutsches Zentrum
DLR für Luft- und Raumfahrt e.V.
in der Helmholtz-Gemeinschaft

Python auf alle Systeme!
Jenseits von Desktop- und Webanwendungen…

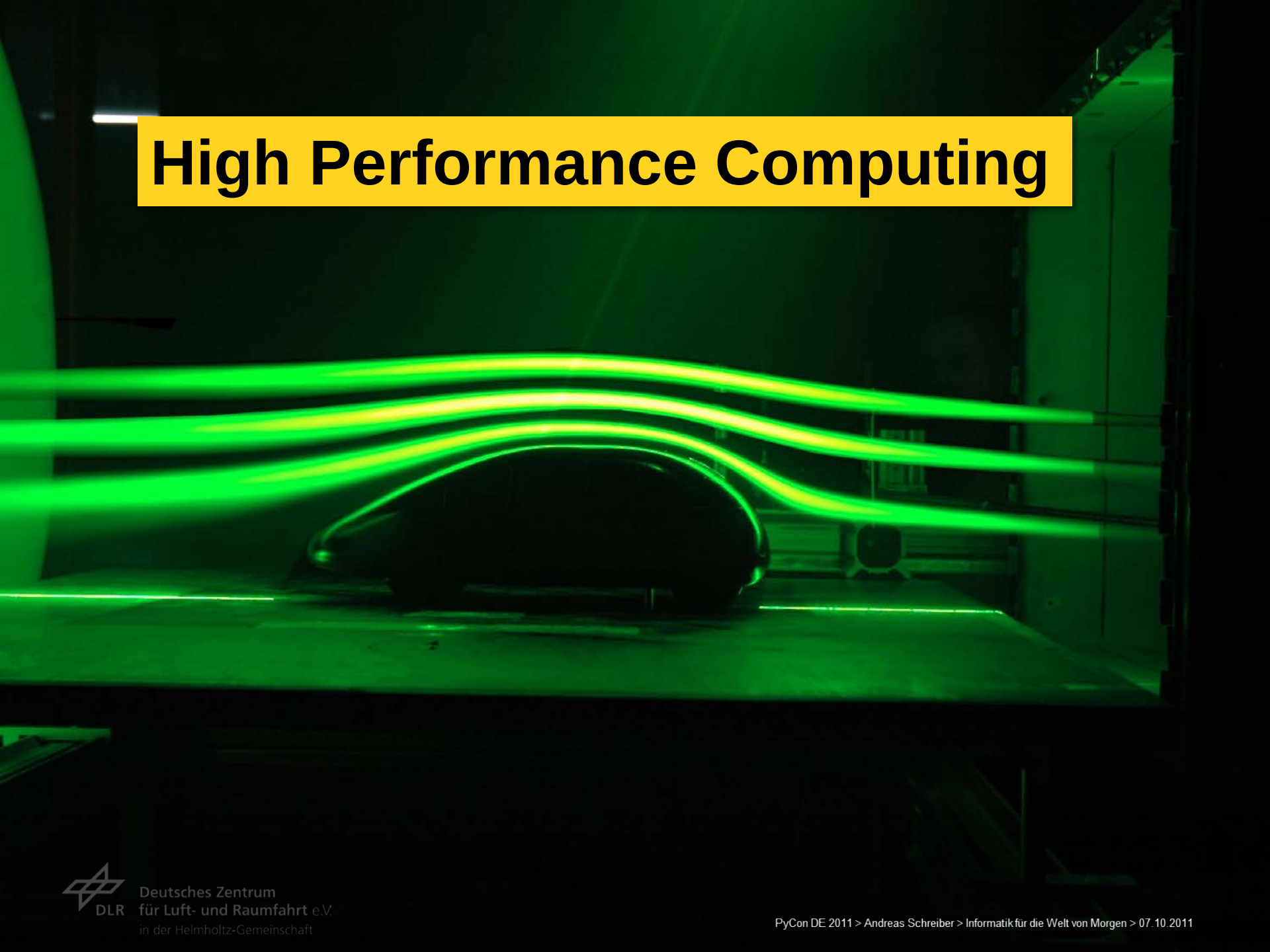
Embedded Systems

Eingebettet in andere Sprachen

Mobile Systeme

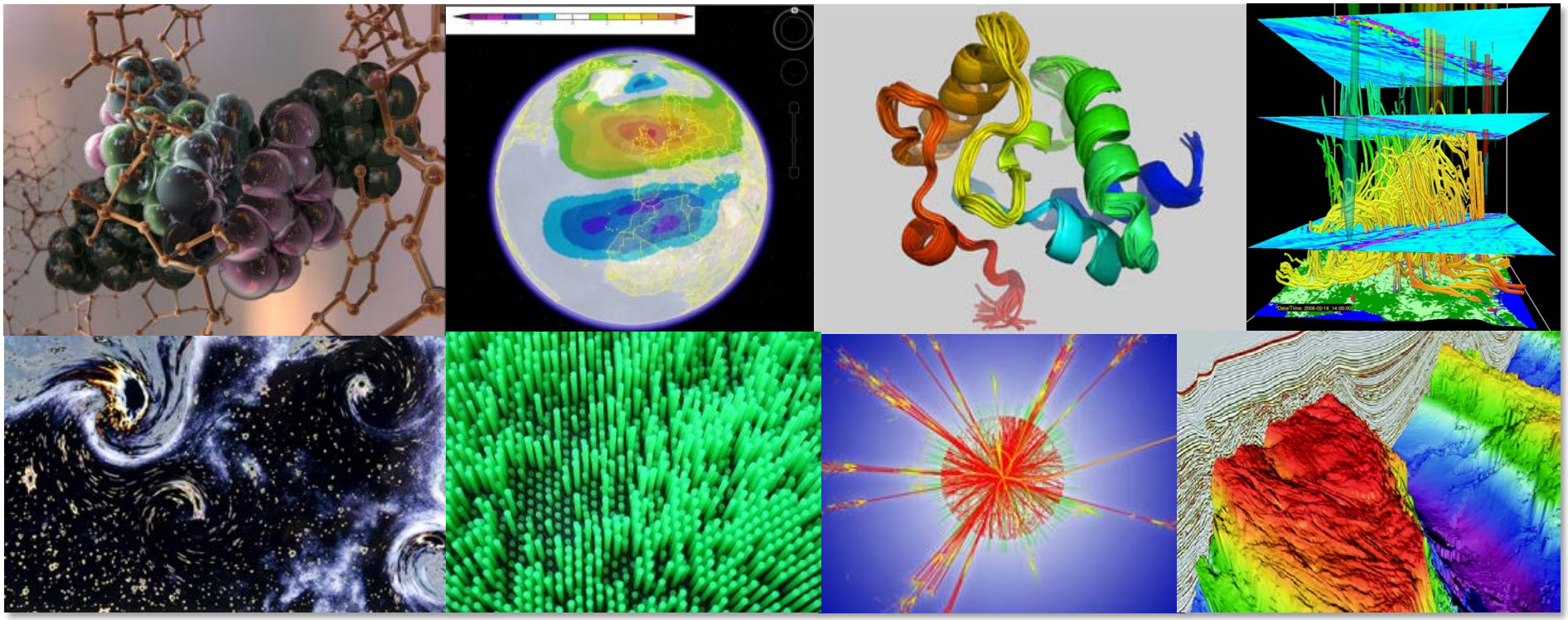
HPC-Systeme

High Performance Computing



Überblick über Anwendungen

Meteorologie, Astrophysik, Teilchenphysik, Biologie, Genetik, Quantenchemie, Strömungsmechanik, Finanzen, Erdöl-Exploration, …



Images from: http://www.isgtw.org

High-Performance-Computing-Systeme

- Aktuelle Top-Systeme haben > 500,000 cores und > 8,000 TFlops
- Siehe TOP500-Liste der Supercomputer (http://www.top500.org)
- Top 3 im Juni 2011 sind:
  1. K computer, Japan
  2. Tianhe-1A, China
  3. Jaguar, USA

#1: *K computer, SPARC64 VIIIfx 2.0GHz, Tofu interc. (RIKEN Advanced Institute for Computational Science (AICS), Japan)*

#2: *Tianhe-1A - NUDT TH MPP, X5670 2.93Ghz 6C, NVIDIA GPU, FT-1000 8C (National Supercomputing Center in Tianjin, China)*

#3: *Jaguar - Cray XT5-HE Opteron 6-core 2.6 GHz (DOE/SC/Oak Ridge National Laboratory, United States)*

Tools und Bibliotheken für HPC mit Python

Deutsches Zentrum
für Luft- und Raumfahrt e.V.
in der Helmholtz-Gemeinschaft

Wissenschaftliche Tools und Bibliotheken
Allgemeine Tools

Sehr allgemein

- NumPy
- SciPy

Visualisierung

- Matplotlib
- VisIt
- MayaVi
- Chaco
- VTK

High Performance Computing

Parallel Computing

- PETSc
- PyMPI
- Pypar
- mpi4py

GPGPU Computing

- PyCUDA
- PyOpenCL

Wissenschaftliche Tools und Bibliotheken
Anwendungsspezifische Tools

KI
- pyem
- ffnet
- pymorph
- Monte
- hcluster

Biologie
- Brian
- SloppyCell
- NIPY
- PySAT

Chemie
- PyMOL
- Biskit
- GPAW

Geowissenschaften
- GIS Python
- PyClimate
- ClimPy
- CDAT

Elektromagnetismus
- PyFemax

Astronomie
- AstroLib
- PySolar

Dynamische Systeme
- Simpy
- PyDSTool

Finite Elemente
- SfePy

Wissenschaftliche Tools und Bibliotheken
Spezielle Tools

Wrappen von Code in anderen Sprachen

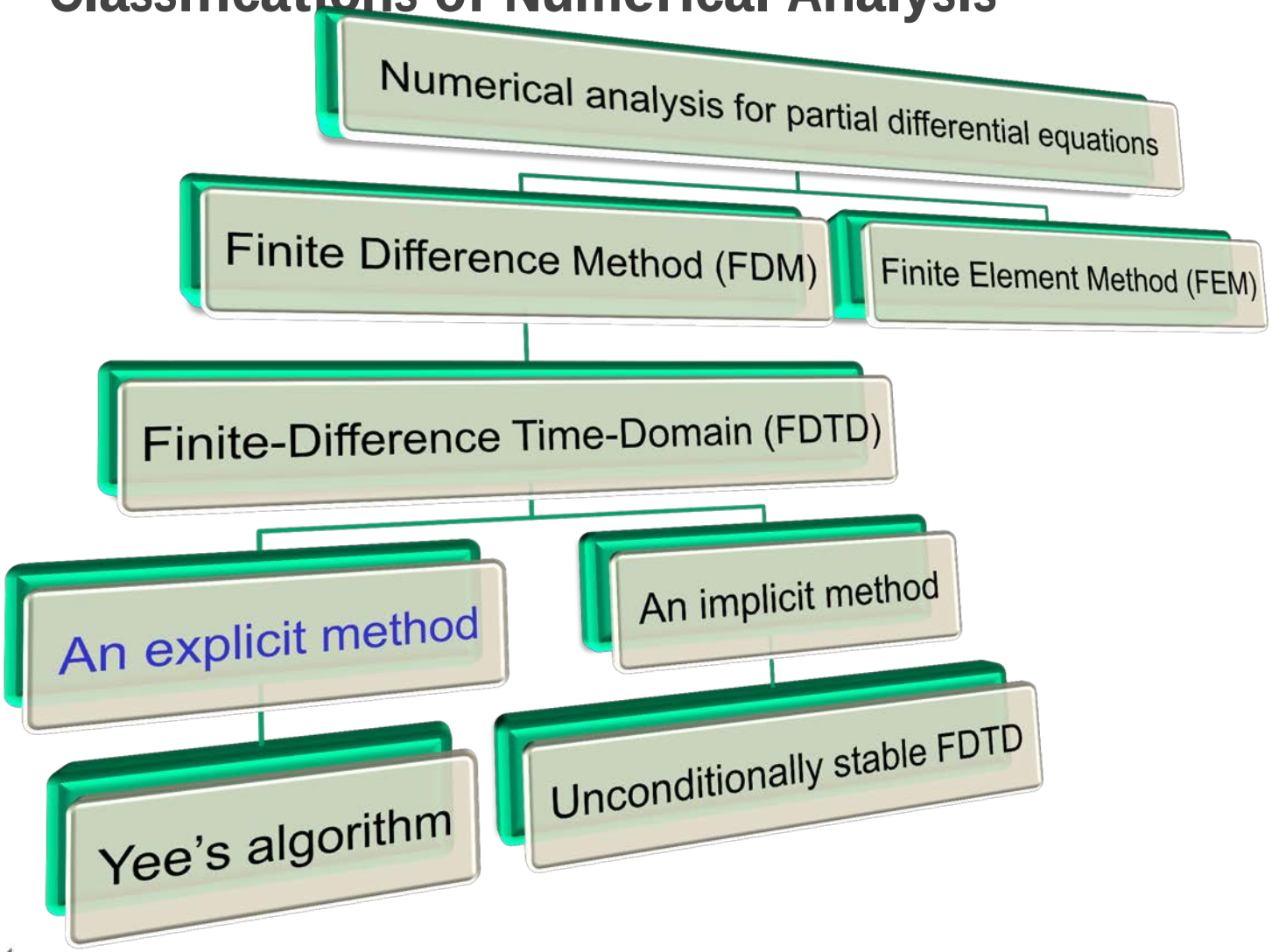
- weave (C/C++)
- f2py (Fortran)
- Cython
- Ctypes (C)
- SWIG (C/C++)
- RPy / RSPython (R)
- MatPy (Matlab)
- Jython (Java)
- IronPython (.NET)

Drei Beispiele...

Beispiel:
GIST Maxwell's Equation Solver (GMES)

- Object-Oriented Implementation of the Finite-Difference Time-Domain Method in Parallel Computing Environment
- *Slides by Kyungwon Chun et al. (GIST - Gwagnju Institute of Science and Technology, Südkorea)…*

Classifications of Numerical Analysis



Application Areas

- Near-DC (ultralow-frequency geophysics involving the entire Earth-ionosphere waveguide)
- Microwaves (radar signature technology, antennas, wireless communications devices, digital interconnects, biomedical imaging/treatment)
- visible light (photonic crystals, nanoplasmonics, solitons, and biophotonics)

From http://en.wikipedia.org/wiki/FDTD

FDTD Method Is …

The FDTD is a grid-based differential time-domain numerical modeling methods to solve Maxwell's equations.

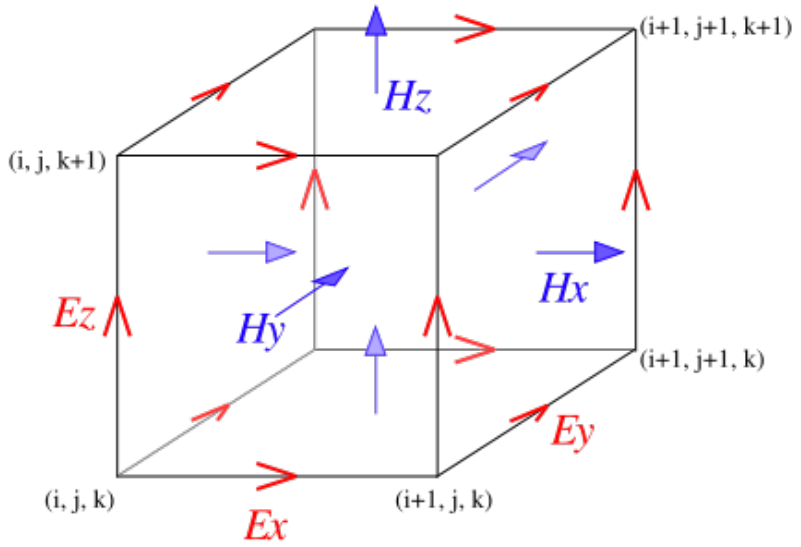
GMES implements the explicit FDTD method using Python/C++.

$$\nabla \cdot \mathbf{D} = \rho$$

$$\nabla \times \mathbf{H} - \frac{\partial \mathbf{D}}{\partial t} = \mathbf{J}$$

$$\nabla \times \mathbf{E} + \frac{\partial \mathbf{B}}{\partial t} = 0$$

$$\nabla \cdot \mathbf{B} = 0$$

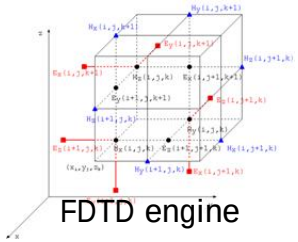


$$E_x\big|_{i+1/2,j,k}^{n+1/2} = E_x\big|_{i+1/2,j,k}^{n-1/2} + \frac{\Delta t}{\varepsilon_{i+1/2,j,k}}\left(\frac{H_z\big|_{i+1/2,j+1/2,k}^{n} - H_z\big|_{i+1/2,j-1/2,k}^{n}}{\Delta y} - \frac{H_y\big|_{i+1/2,j,k+1/2}^{n} - H_y\big|_{i+1/2,j,k-1/2}^{n}}{\Delta z}\right)$$

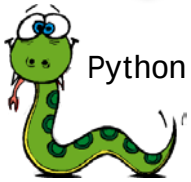
…

Design Goal

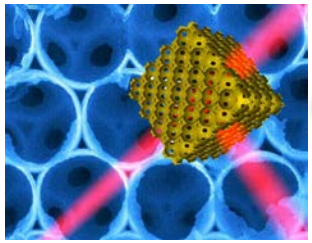
From the general simulation engine



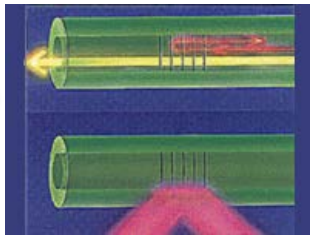
FDTD engine



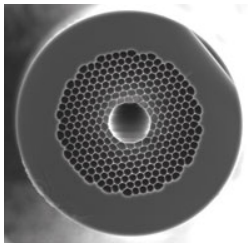
Python interface



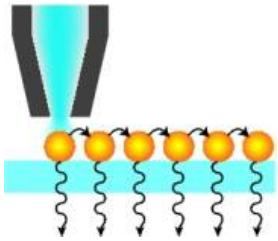
Photonic crystals



Optical fiber grating



Photonic crystal fiber



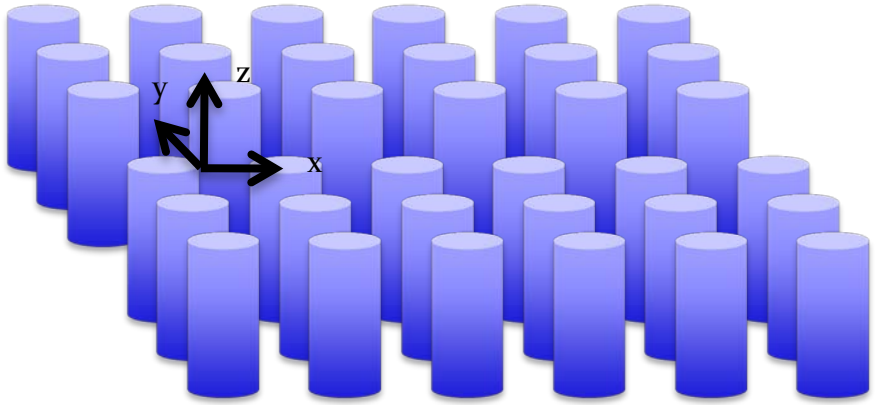
Plasmonics

- Reuse simulation program.
- Keep the Simplicity.
- Provide the flexibility.

Provide various specific contents

Results

```
NANO = 10**-6
PETA = 10**15
SIZE = (5,5,0)
def makeRod(x, y):
    axis = (0,0,1)
    radius = 0.29
    height = 10
    material = Dielectric(epsilonR=8.9)
    center = (x,y,0)
    c = Cylinder(axis, radius, height, material, center)
    return c
def makeCrystals(xLow, xHigh, yLow, yHigh):
    rodList = []
    for i in xrange(xLow, xHigh+1):
        for j in xrange(yLow, yHigh+1):
            rodList.append(makeRod(i,j))
      return rodList
geomList = [DefaultMaterial(material=Dielectric())] + \
           makeCrystals(-3,3,1,3) + makeCrystals(-3,3,-3,-1) + \
           [Boundary(material=Cpml(thickness=0.5, size=SIZE), \
            thickness=0.5, size=SIZE)]
startTime = time.time()
TMzFDTD(coordinate=Cartesian(SIZE), geometry=geomList, source=())
print time.time() - startTime
```



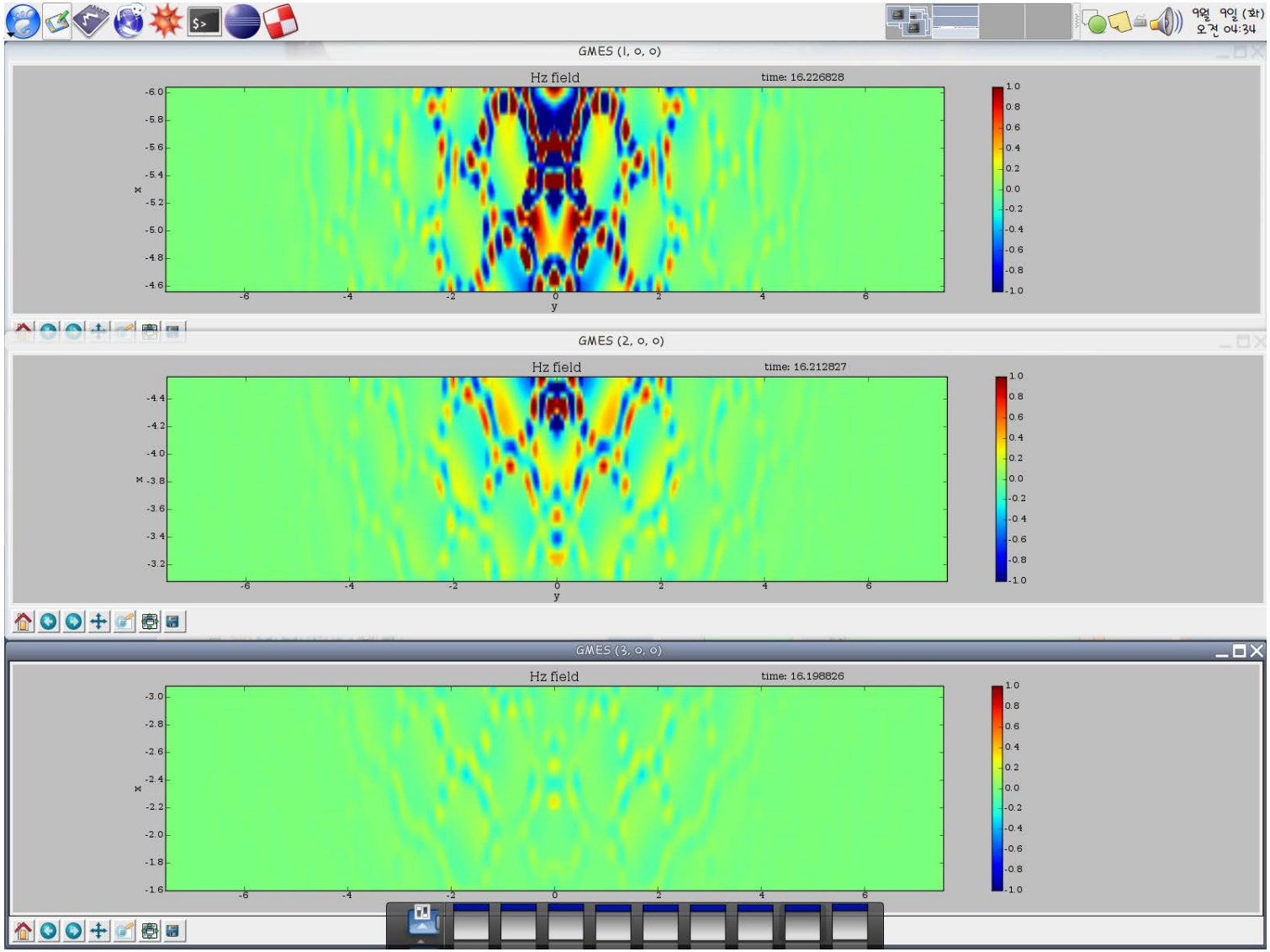
Sequential search + memory alloc.: 552s

Binary tree search + memory alloc.: 216s

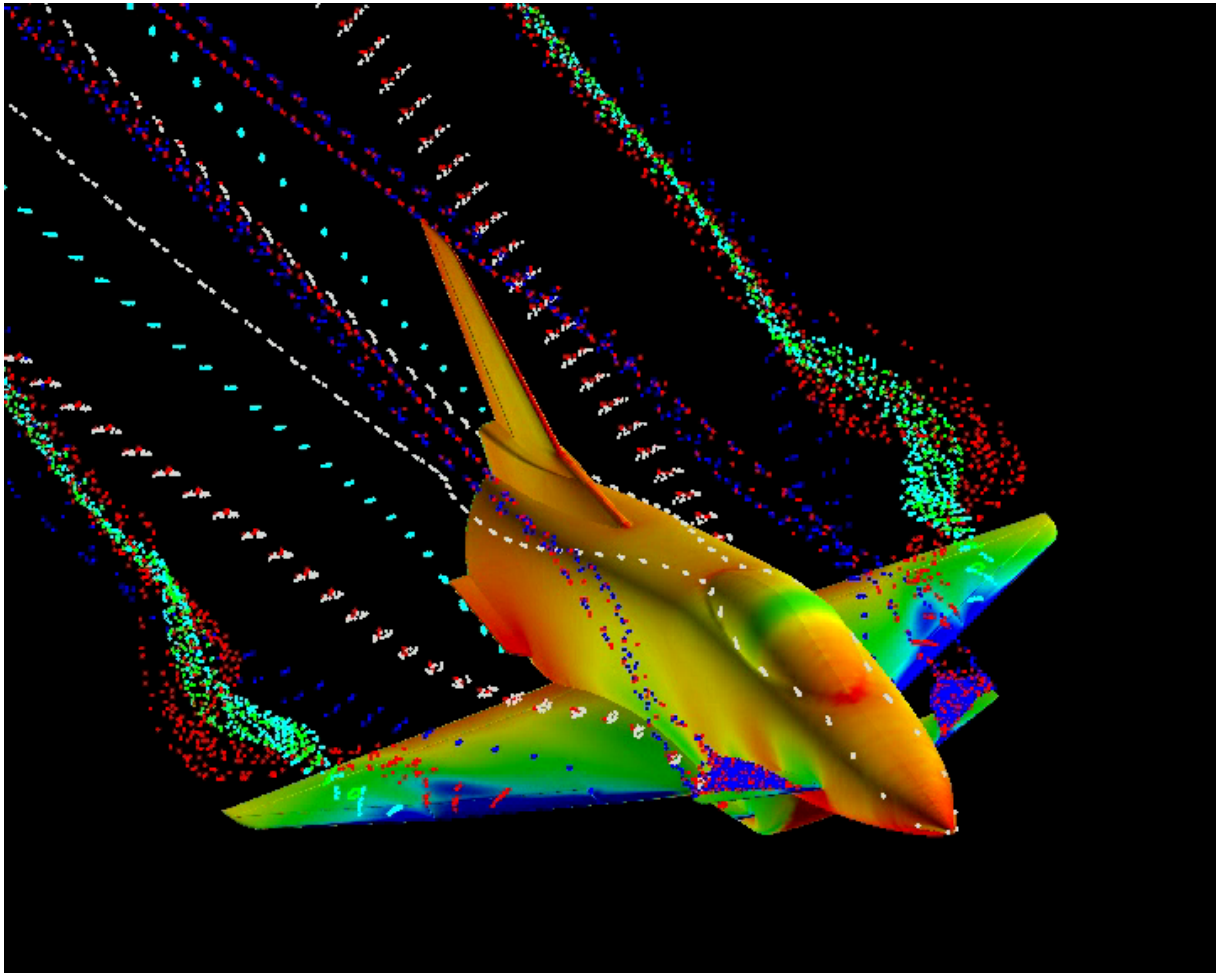
Including memory allocation

Parallel Execution

- Two ways of parallelism are used.
- Thread
  - Python does not support full-thread.
  - Thread provides a way to describe a algorithmic parallelism.
- MPI
  - pyMPI
  - Fully functional.

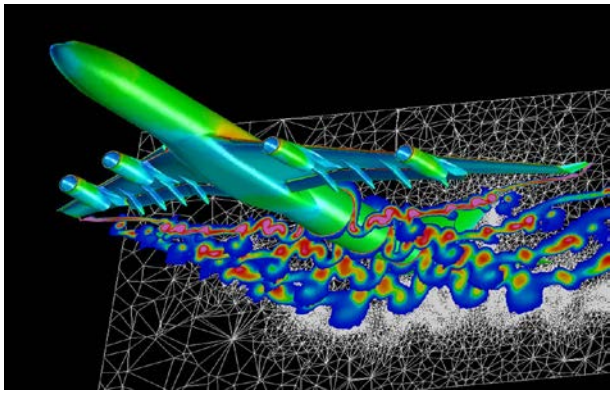

GMES (1, 0, 0)
Hz field
time: 16.226828
GMES (2, 0, 0)
Hz field
time: 16.212827
GMES (3, 0, 0)
Hz field
time: 16.198826

Beispiel: **FlowSimulator –** A Python-controlled framework to unify massive parallel CFD workflows

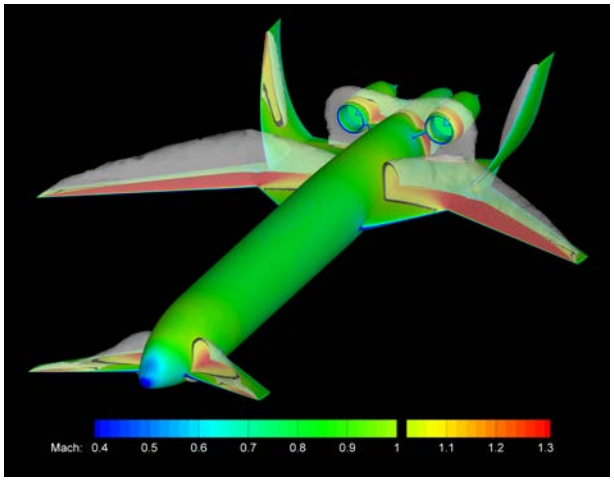


What does CFD mean?

- CFD = Computational Fluid Dynamics
  - to solve the compressible flow equations to
  - predict an aircrafts behavior and to
  - optimize its features.



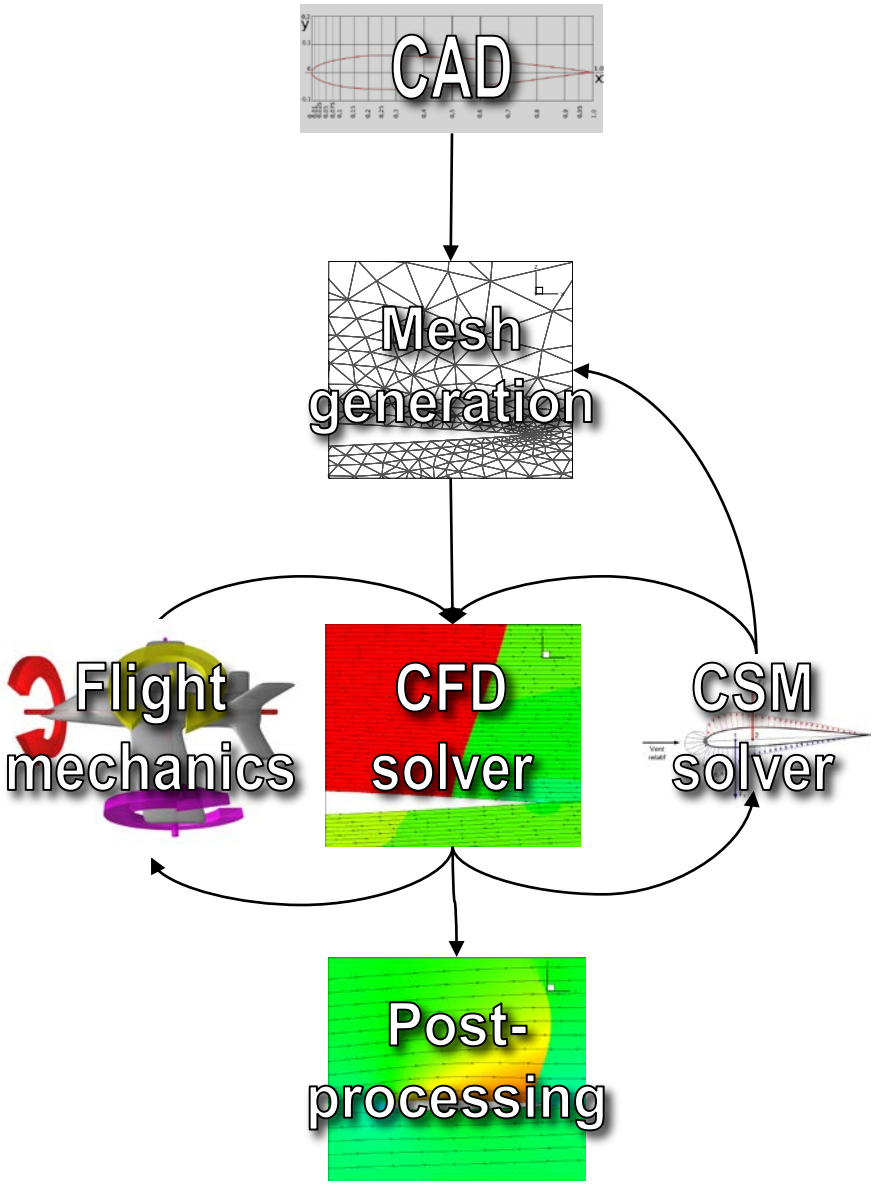
- Numerical approach:
  - Spatial discretization
  - Simplified models for turbulent effects
  - Temporal discretization (to support unsteady flows)



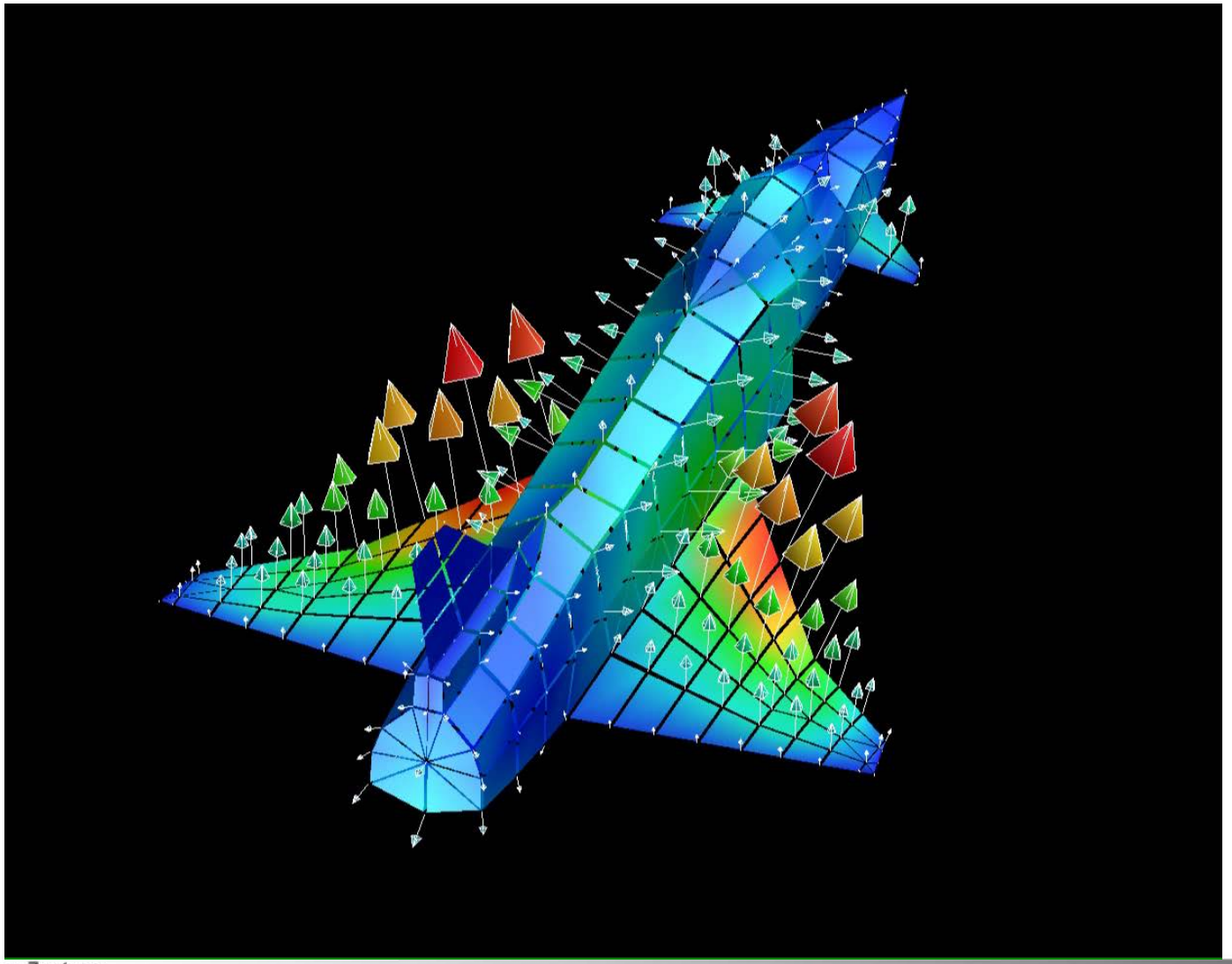
CFD Process Chain

Multi-disciplinary simulations

- Multi-disciplinary chain
  - output of solver is input for
    - structural mechanics (deformation)
    - flight mechanics (control surface movement)
  - output of structural/flight mechanics modifies mesh
- Tools:
  - solver as Python module
  - stand-alone CSM
  - Python-based flight mechanics

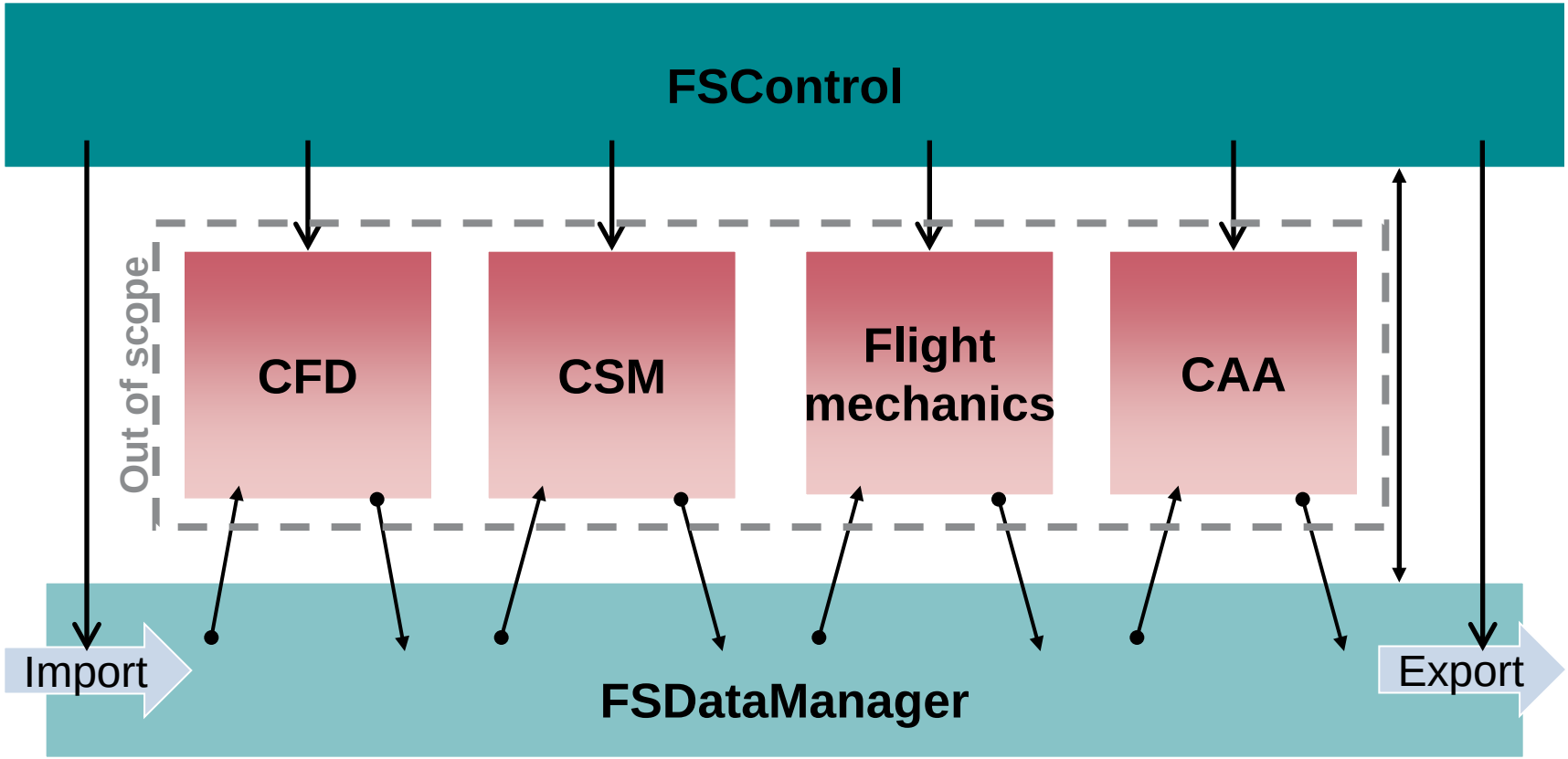


Structure and Flight Mechanics



FlowSimulator Environment

Single Python script controls the whole workflow



FSControl

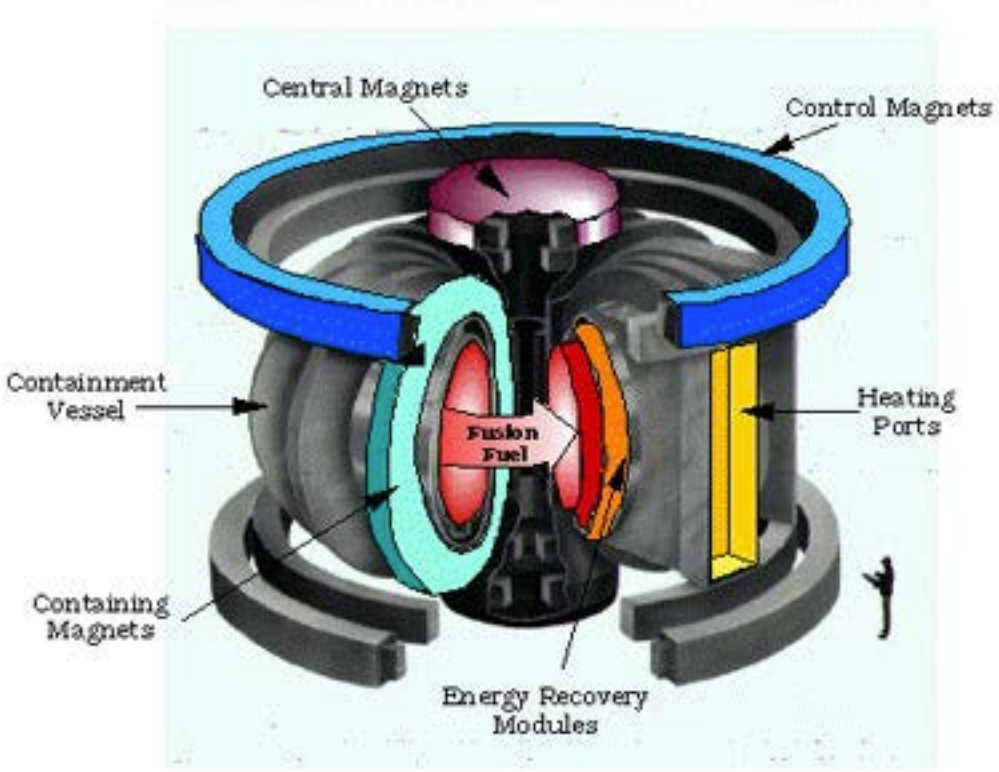
- Pure Python library
- Set of simple Python classes for different modules/tools:
  - simple interface: Import, Run, Export
  - data stored in FSDataManager

FSDataManager (FSDM)

- C++ library, wrapped to Python
  - FSDataManager is a real Python module!
- Parallelization context
  - all data structures “distributed”
  - facilitated transfers (Gather, Distribute, Send, …)

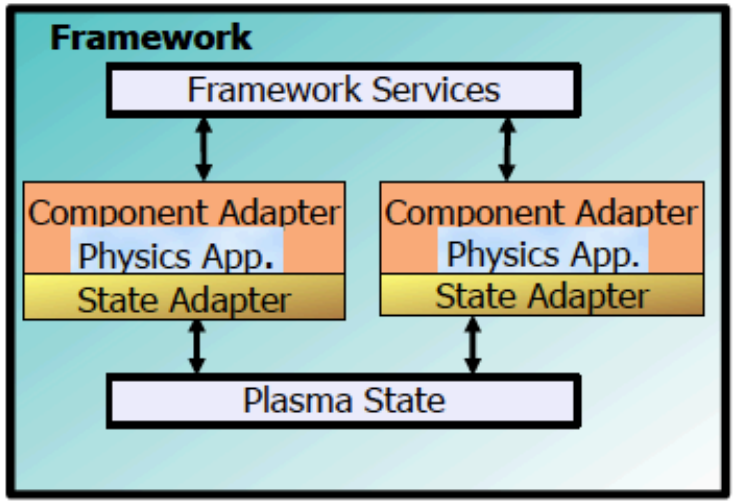
Beispiel:
Python Framework for Coupled Fusion Simulations

- *Slides by Samantha Foley et al. (ORNL – Oak Ridge National Laboratory, United States) …*

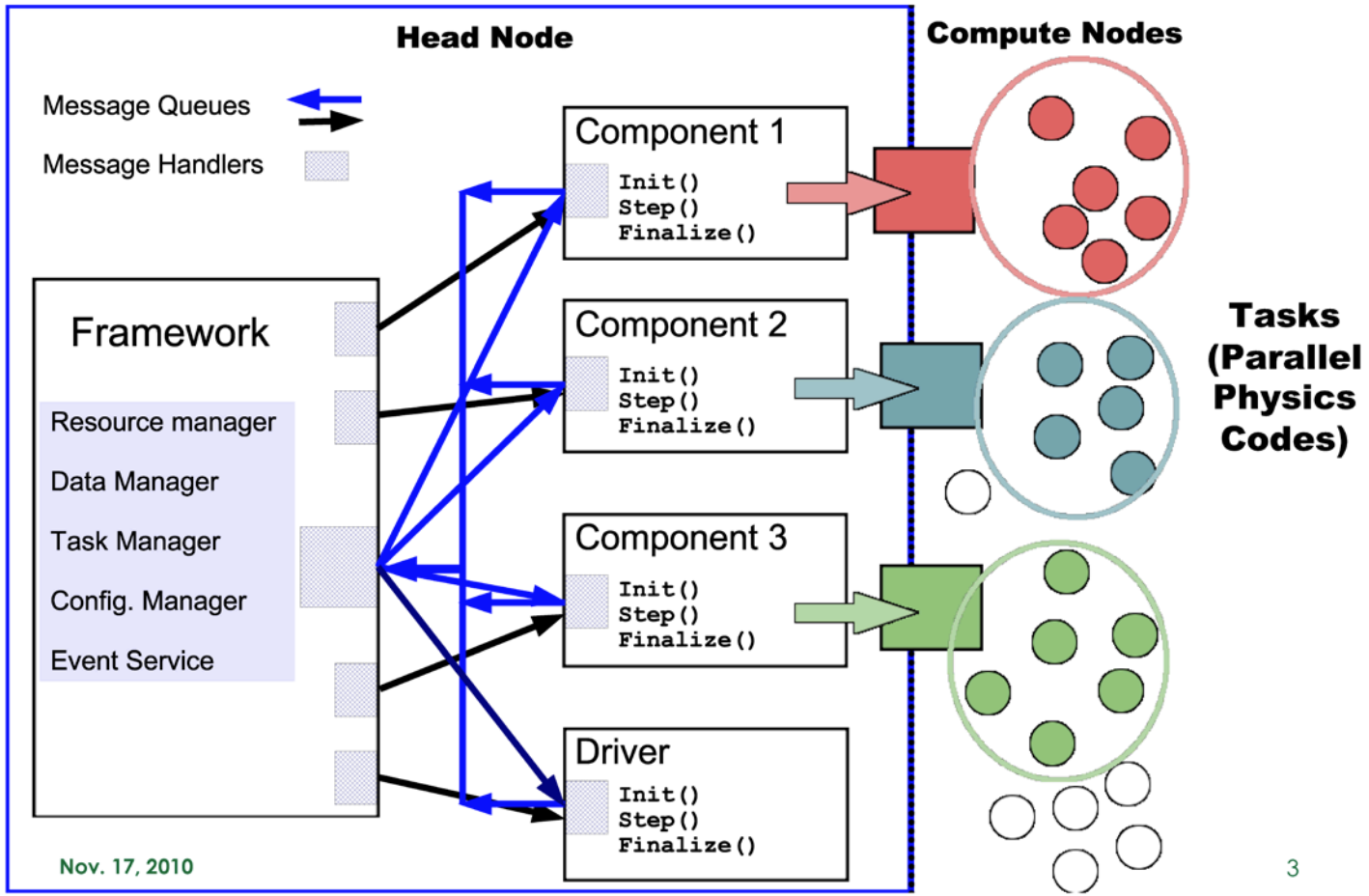


Integrated Plasma Simulator (IPS)

- The **Integrated Plasma Simulator (IPS)** is a component framework for fusion energy simulation for the **Center for Simulation of RF Wave Interactions with Magnetohydrodynamics (SWIM)**
- Design landscape
  - Code re-factoring and/or rewriting ruled out
  - Modest data exchange between components
  - Loose-coupling via files
  - Flexible execution environment
  - Platforms range from laptops/desktops to clusters to supercomputers



IPS: Architecture



OAK RIDGE National Laboratory

Why we chose Python

- Powerful and expressive
- Easy to learn and manage for new users
- Portable and widely available
- Wide variety of modules that we can leverage
  - **Matplotlib** – plotting to examine execution characteristics of coupled simulations
  - **ConfigObj** – configuration file parser
  - **multiprocessing** – multiple process capability helps us provide multiple levels of parallelism
  - **socket**, **subprocess**, **urllib**, **shutil**, **os**, **sys** – a variety of shell and file interactions
  - **logging** – package to manage various levels and destinations of output
  - **Sphinx** – plan to start using for documentation soon

Wo sich Python- und HPC-Communities treffen…

- **SciPy**
  - Python for Scientific Computing Conference (seit 2008)

- **EuroSciPy**
  - Annual European Conference for Scientists using Python (seit 2008)

- **SCxx („Supercomputing“)**
  - The International Conference for High Performance Computing, Networking, Storage, and Analysis (jährliche Konferenz seit 1988)

Supercomputing-Konferenzen (SCxx)

Birds-of-a-Feather Sessions, seit 2009

Workshop Python for High Performance and Scientific Computing (PyHPC 2011)

- **Wissenschaftlicher Workshop auf der SC11 (18. November, Seattle)**
- **Themen**
  - Python-based scientific applications and libraries
  - High performance computing
  - Parallel Python-based programming languages
  - Scientific visualization
  - Scientific computing education
  - Python performance and language issues
  - Problem solving environments with Python
  - Performance analysis tools for Python application

Programmkomitee

Achim Basermann, David Beazley, William E. Hart, Konrad Hinsen, Andreas Klöckner, Guy K. Kloß, Maurice Ling, Stuart Mitchell, Mike Müller, Travis Oliphant, Fernando Pérez, Massimo Di Pierro, Marc Poinot, William Scullin, Argonne Andy R. Terrel, Gaël Varoquaux

Hinweise

Credits

Achim Basermann (DLR)
Kyungwon Chun (GIST)
Samantha Foley (ORNL)
Michael Meinel (DLR)
Travis Oliphant (Enthought)
Andreas Schütte (DLR)
William R. Scullin (ANL)

@

@DLR_de

@onyame

Andreas Schreiber

Andreas.Schreiber@dlr.de

http://www.dlr.de/sc

Getting Python ready for the Enterprise…

ncc1701

getting python ready for the enterprise

Search projects

Project Information

Starred by 0 users

Activity None

Project feeds

Code license

MIT License

Members

haraldarminmassa, onyame

2 committers

Your role

Owner

provides classes and methods to:

initiate, target and fire phasers

reroute power

eject the warp core

code.google.com/p/ncc1701