Open Source im BACARDI Projekt

Martin Stoffers, Michael Meinel

Simulations- und Softwaretechnik (SC)
Deutsches Zentrum für Luft- und Raumfahrt e. V.

FrOSCon 14, 10.-11.08.2019 Sankt Augustin

DLR

Überblick

1. Vorstellung
2. Raumfahrtrückstände
3. BACARDI
4. Open Source Software im BACARDI Projekt
5. F2x

Vorstellung

DLR

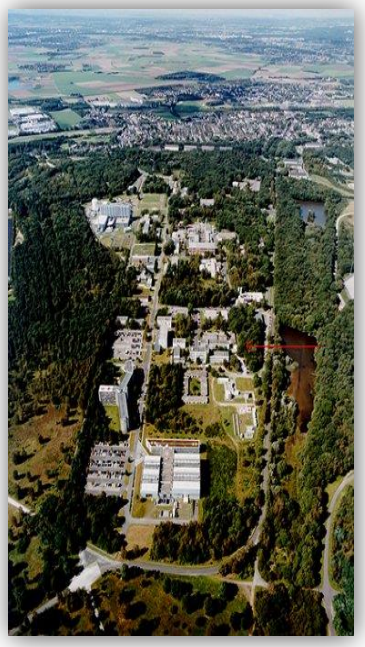
Einrichtung Simulations- und Softwaretechnik

- Einrichtung für **Softwareforschung**, **Research Software Engineering** und **wissenschaftliches Rechnen**
- Ca. 70 Mitarbeiterinnen und Mitarbeiter
- Standorte
  - Köln-Porz
  - Berlin-Adlershof
  - Braunschweig
  - Oberpfaffenhofen
  - Bremen

`http://www.DLR.de/sc`

Standorte


STADE
BREMERHAVEN
HAMBURG
NEUSTRELITZ
OLDENBURG
BREMEN
TRAUEN
BERLIN
BRAUNSCHWEIG
GÖTTINGEN
DRESDEN
KÖLN
JÜLICH
BONN
JENA
LAMPOLDSHAUSEN
STUTTGART
AUGSBURG
OBERPFAFFENHOFEN
WEILHEIM



Die Vortragenden

Michael Meinel

- Softwareentwickler & Python-Enthusiast
- Professionelle Rubber-Duck
- Seit 2004 am DLR

Martin Stoffers

- Wissenschaftlicher Mitarbeiter
- Ambitionierter KSP-Pilot
- Seit 2016 am DLR
- 

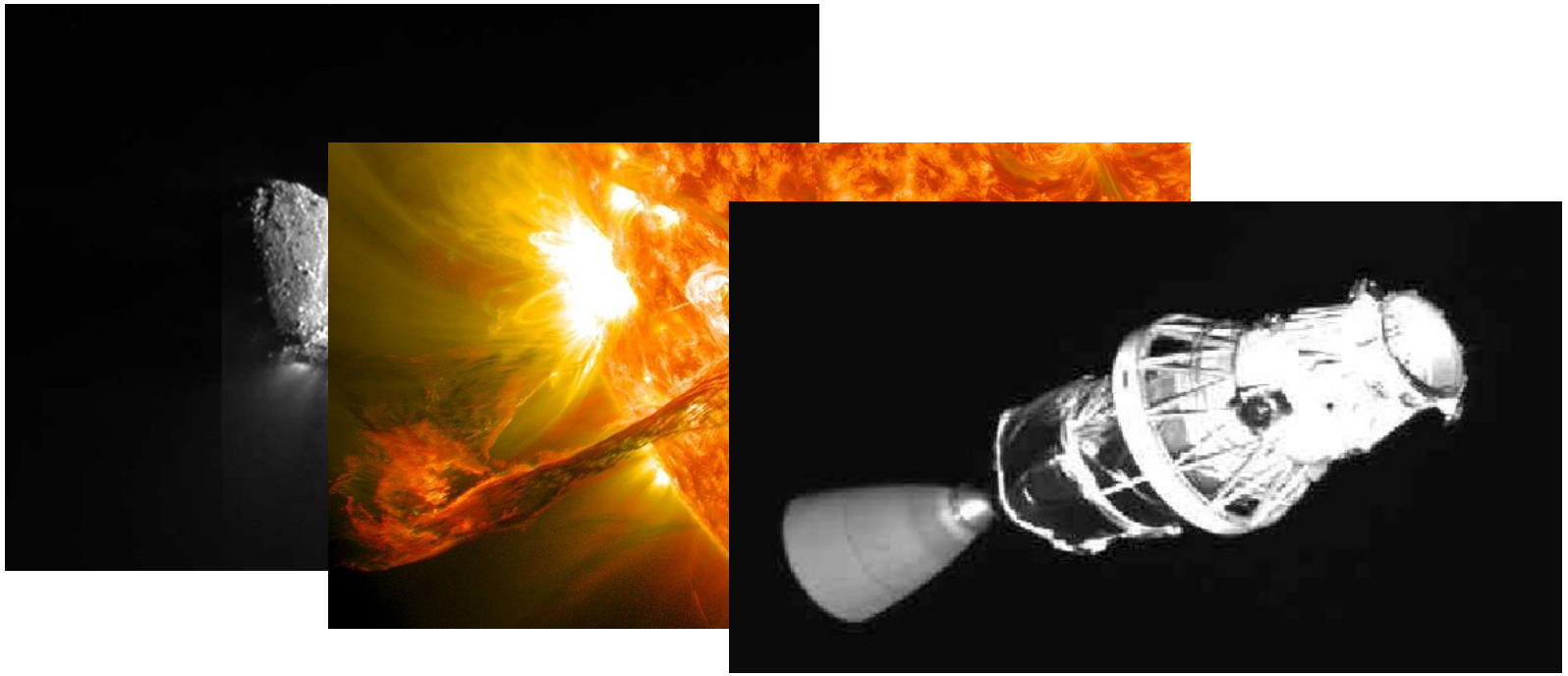
Raumfahrtrückstände

DLR

Space Situational Awareness (SSA)

Unterteilt in:

- Near-Earth objects (NEO)
- Space Weather (SWE)
- Space Surveillance and Tracking of satellites and space debris (SST)



Ein Menschen gemachtes Problem

Satelliten

- Ca. 8950 Starts seit 1975
- Ca. 3950 wieder eingetreten
- Ca. 1950 aktiv
- Ca. **3050** defekt oder ausrangiert

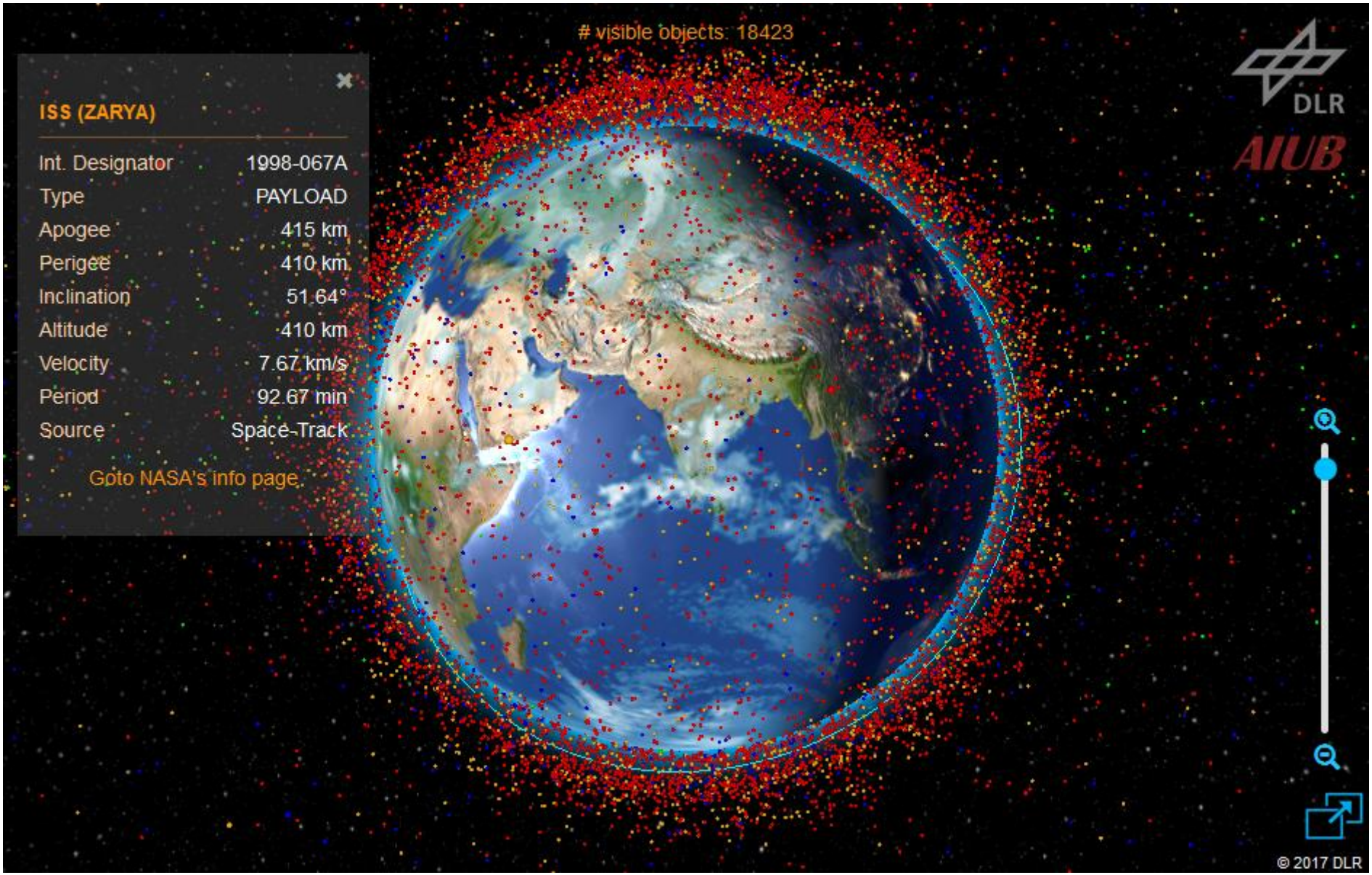
500 „Vorfälle“ seit 1975

Objekte im Erdorbit

>10cm:	Ca. 34000
>1cm & <10cm:	Ca. 900000
>1mm & <1cm:	Ca.128 Mio.

Nur Ungefähr 22300 sind katalogisiert

Quelle: https://www.esa.int/Our_Activities/Space_Safety/Space_Debris/Space_debris_by_the_numbers


visible objects: 18423
ISS (ZARYA)
Int. Designator 1998-067A
Type PAYLOAD
Apogee 415 km
Perigee 410 km
Inclination 51.64°
Altitude 410 km
Velocity 7.67 km/s
Period 92.67 min
Source Space-Track
Goto NASA's info page
DLR
AIUB
© 2017 DLR

DLR

BACARDI

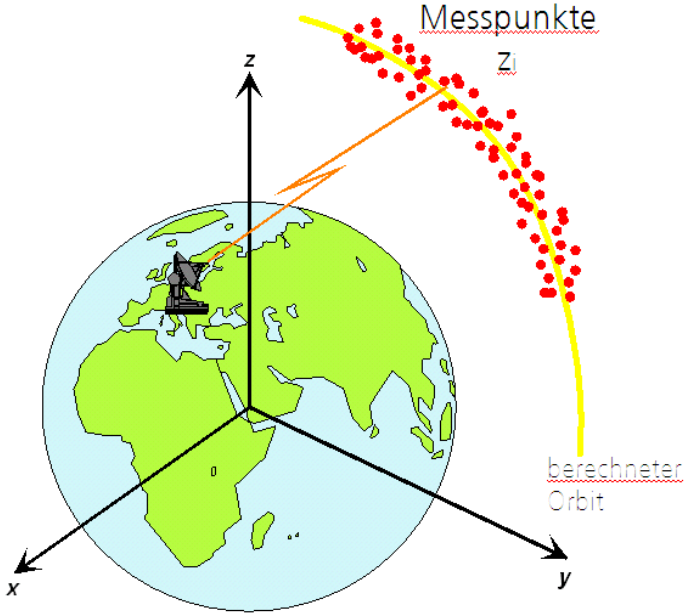
DLR

Was ist BACARDI

Zielsetzung

- Entwicklung einer Bahndatenbank mit möglichst hoher Vollständigkeit und Bahngenauigkeit
- Unterstützung des Missionsbetriebs am GSOC bei Kollisionsvermeidung
- Unabhängige Bahnbestimmung aus Sensordaten

- Entwicklung und Optimierung von Algorithmen:
  - Beobachtungskorrelation
  - Bahnpropagation & -bestimmung
  - Detektion & Vorhersage von Manövern, Fragmentationen, Wiedereintritten



Bahnschätzung durch Mittelung:

$$\sum_i \left[z_i - f_i(\mathbf{r}, v)\right]^2 = Min$$

Systemüberblick

Sensornetzwerk

Tracking Radar | Überwachungsradar | Laser Tracking | Raumgestützt | Optisches Teleskop

BACARDI

Datenbank

- Sensordaten
- Korrelationsergebnisse
- Katalogobjekte und Objektkandidaten
- Bahnephemeriden inkl. Kovarianz
- Bahnmanöver geplant / ausgeführt
- Meta- und Logdaten
- Datenrechte je nach Datenpolitik

Prozessoren

- Beobachtungskorrelation
- Bahnbestimmung
- Bahnpropagation
- Manöverdetektion
- Fragmentationsdetektion
- Kollisionsvorhersage
- Wiedereintrittsvorhersage

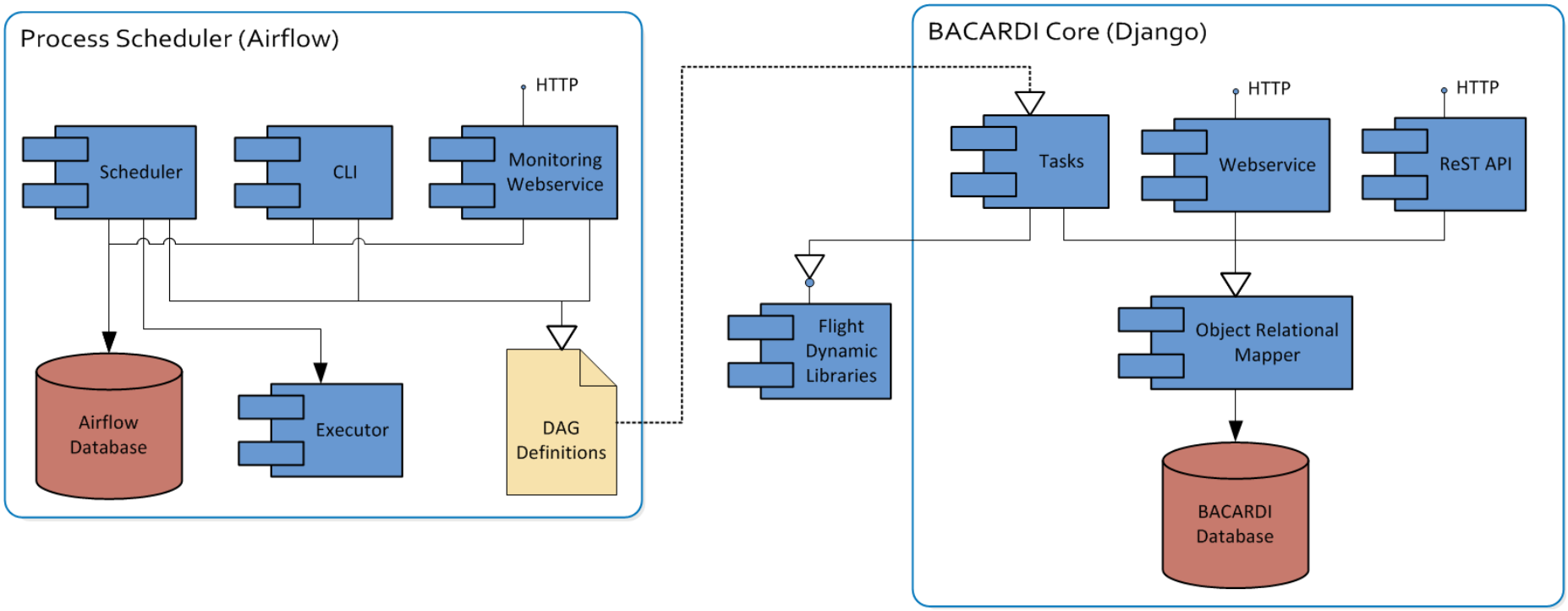
Externe Daten

Bahninformationen | Objekteigenschaften | Sonnenaktivität

Schnittstellen

Nutzer | Sensorplanung

Systemarchitektur



Open Source Software

im BACARDI Projekt

DLR

Gründe für die Nutzung von Open Source Software

- Verfügbarkeit von Dokumentation und Code
  - Hinzufügen von benötigten Features in bestehenden Code
- Spezielle Anforderungen im Projekt
  - Nicht durch verfügbare kommerzielle Produkte abgedeckt
  - OSS als Baustein für eine selbst entwickelte Softwarelösung
- Ressourcenknappheit in Projekten
  - Ersatz für kommerzielle verfügbare Produkte
  - Häufig Betriebssysteme oder Serverlösungen

Kriterien für die Auswahl von OSS im BACARDI Projekt

1. Unterstützter Funktionsumfang
2. Stabilität der Schnittstellen
3. Benutzbarkeit der Software
4. Qualität und Umfang der Dokumentation
5. Projektaktivität
6. Codequalität
7. Umfang der Softwareabhängigkeiten zu Drittsoftware
8. Verwendete Lizenz
9. Lizenzeinhaltung zu Abhängigkeiten
10. Veröffentlichungszyklen der Software

Django

- Python Framework zur Erstellung von Webseiten
- Initiiert im Juli 2005
- Modifizierte BSD Lizenz
- Große und aktive Community und regelmäßige Konferenzen

- Veröffentlichungsstrategie
  - Feature-Releases alle 8 Monate
  - Patch-Releases
  - Long-Term Support Releases (LTS)

- Ausführliche Dokumentation, Tutorials
- HOWTO-Artikel zu speziellen Themen

Verwendung von Django in BACARDI

- Primär zur Definition des Datenbankmodells

- Django Migrations
  - Stabiles Releasedeployment
  - Tracking der erfolgten Änderungen

- Dedizierte Applikationen (Apps) innerhalb des Projektes
  - projektinterne ReST API zum Abruf von Trackletdaten
  - Webservices für Administration und Monitoring der Datenbank

Verwendung von Airflow in BACARDI

- Python Framework zur Prozessautomatisierung
- Initiiert im Oktober 2014 von Maxime Beauchemin bei Airbnb
- Seit Juni 2015 unter der Apache 2 Lizenz veröffentlicht
- Mitglied im Inkubator Programm der Apache Software Foundation

- Veröffentlichungsstrategie
  - Halbjährliche Feature-Releases
  - Unregelmäßige Patch-Releases

- Software unterliegt zwischen Releases starken Änderungen,
  - Darunter leidet die Dokumentation

Verwendung von Airflow in BACARDI

- Löst den Cron basierenden Prototyp ab
- Verwaltung und Monitoring der Prozessketten
- Modellierung der Prozessketten mittels DAGs
  - Tasks nur aus ,,Tasks" Komponente im Core eingebunden
  - Lose Kopplung zwischen den beiden Hauptkomponenten
- Gewährleistung der Skalierbarkeit des BACARDI Systems
  - Austauschbarkeit der Ausführungseinheit (Executer).
  - Verteilung von Tasks über nachrichtenorientierte Middleware

F2x

DLR

Wieso kein f2py?

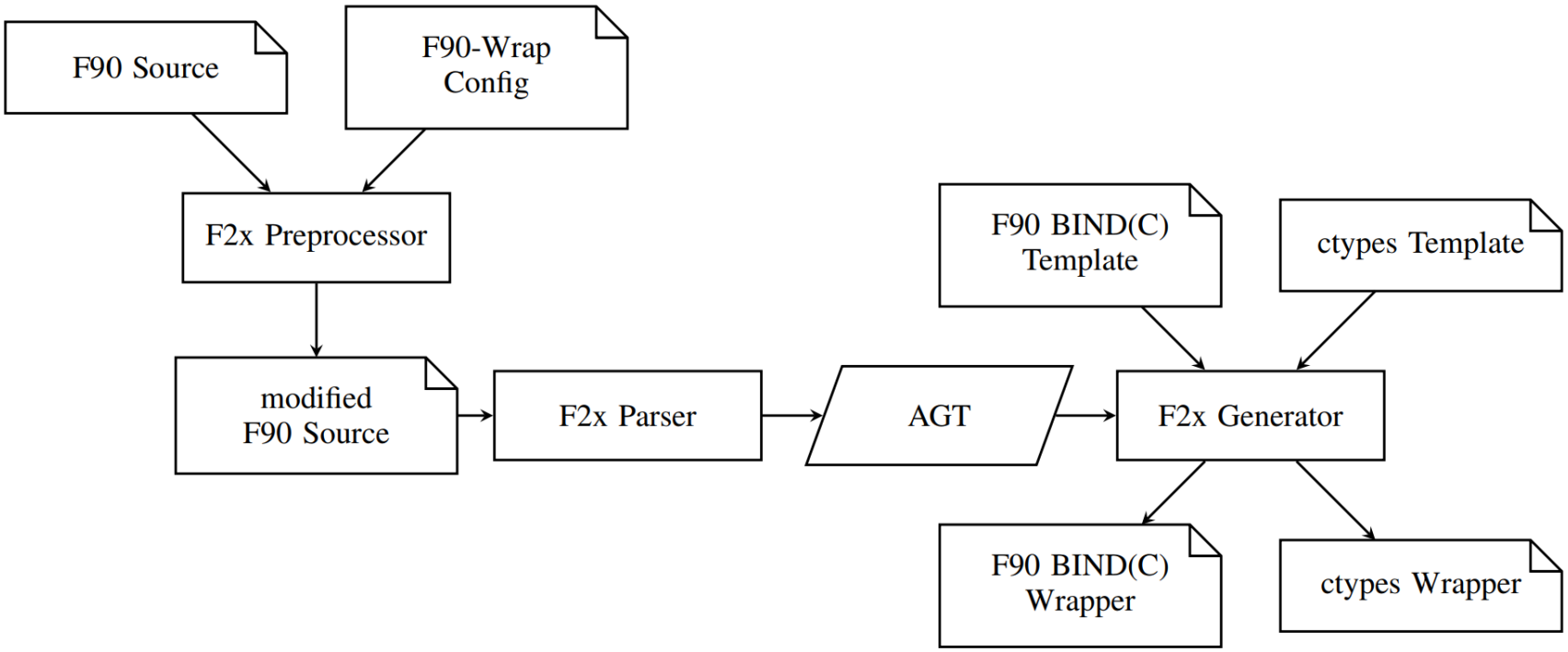
Pro

- Quasi Standard Lösung
- Stabil
- Hoch automatisiert
- Einfache Benutzung
  (Für einfach Anwendungsfälle)

Contra

- Keine abgeleiteten Datentypen verfügbar
- Starke Abhängigkeit des Wrappercodes von Compiler und Version
- Schwer erweiterbar
- Bietet nur Unterstützung für Python

F2x Implementierung
Übersicht



F2x Implementierung
Parser

- Vollständiger LALR(1) Parser unter Nutzung von plyplus
  - Fortran hat keine vollständige LALR(1) Grammatik…
  - plyplus wurde durch den Autor abgekündigt ☹
  - Nachfolger „lark“ nicht ausreichend ☹
- Grammatik vom OpenFortran Projekt geborgt
- AST umgewandelt in „AGT“
  - Entkopplung
  - Verbesserung der Templates
- Weitere Ansätze in Planung (ANTLR)

F2x Implementierung
Template-basierte Codeerzeugung

- Ansatz aus der *Modell getriebenen Entwicklung*

- „Transformation der Modelinstanz in Text (Quellcode) mit Hilfe von Templates.“

- Eingaben
  - Modell
  - Modellinstanz
  - Vorlage

- Ausgabe
  - Quellcode

- Unterstützung für weitere Zielsprachen (C#, Java, …):
  Schreib ein neues Template!
- Es steckt viel Wissen in den Templates → nicht-trivial ☹

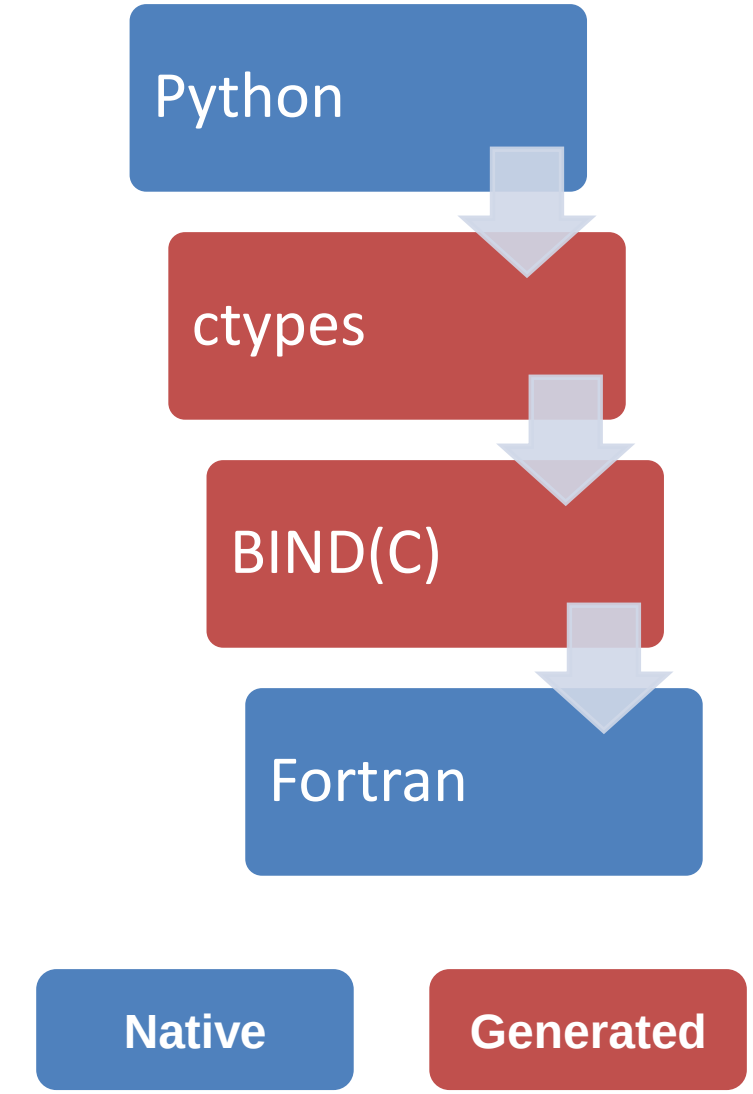
F2x Implementation
Ergebnisse

- Zwei Schichten
  - Compiler neutrale Schnittstelle
  - Nutzung der Schnittstelle aus ...
    Python, C#, …

- Unterschiedliche Templates
  - Fortran – BIND(C)*°
  - Python – ctypes*°, Cython
  - C – Longjmp error handling*°
  - C# – Pinvoke

* Aktuell im produktiven Einsatz

° Teil des F2x Package



F2x Implementation
Beispiel

example.f90

```
MODULE EXAMPLE
    TYPE, PUBLIC :: POLYNOM
        INTEGER :: DEGREE
        REAL(8), DIMENSION(:), &
            ALLOCATABLE :: COEFF
    END TYPE
    PUBLIC EVAL_AT
CONTAINS
    FUNCTION EVAL_AT(P, X)
        REAL(8) :: EVAL_AT
        TYPE(POLYNOM), INTENT(IN) :: P
        REAL(8), INTENT(IN) :: X
        [...]
    END FUNCTION
END
```

example.py

```
from example_glue import POLYNOM, EVAL_AT

# p: f(x) = 3.4 * x^3 + 1.2 * x.
p = POLYNOM(DEGREE=3,
            COEFF=[0, 1.2, 0, 3.4])
c = EVAL_AT(p, 5.6)

# Update offset of p.
p.COEFF[0] = -c
assert abs(EVAL_AT(p, 5.6)) < 0.0001
```

Nutzung im DLR

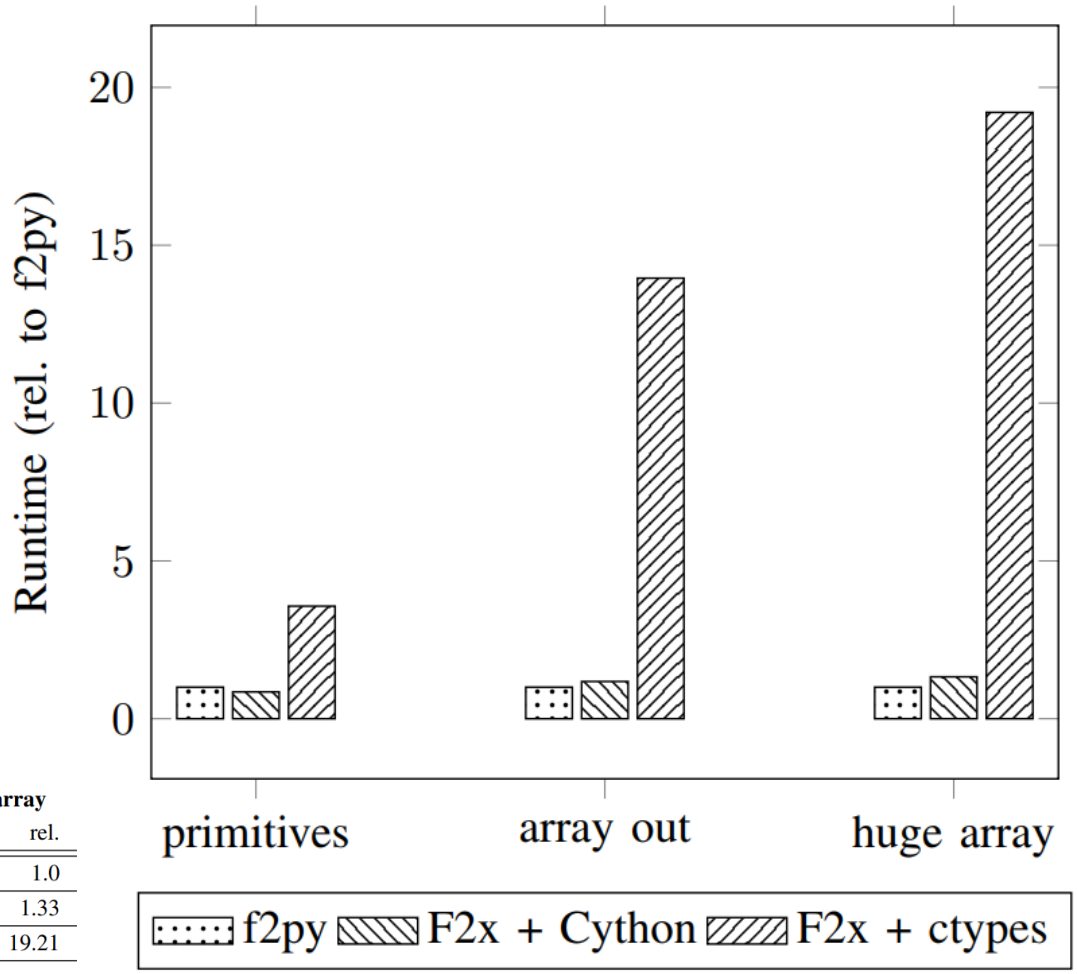
- Initial für BACARDI entwickelt
  - Ziel: Nutzung des Fortran Flugdynamikcodes aus Python

- Nutzung durch andere Projekten am GSOC
  - Verwendung der Fortran Flugdynamikbibliotheken
  - Initialer Plan zur Implementierung von Templates für C#, Java, …
  - Verfügbarmachung unterschiedlicher RPC Schnittstellen aus Python (REST, MQTT, …)

- Nutzung in der Produktion durch unterschiedliche Frameworks
  - Super Tester
  - Sehr viel Feedback

Benchmark Results
Vergleich mit f2py

Implementation	primitives		array out		huge array	
	abs.	rel.	abs.	rel.	abs.	rel.
f2py	3.6	1.0	1.16	1.0	0.82	1.0
F2x + Cython	3.06	0.85	1.37	1.18	1.1	1.33
F2x + ctypes	12.84	3.57	16.22	13.96	15.84	19.21



https://doi.org/10.5281/zenodo.1405459

Neue (mögliche) Anwender

- Mark Paris (LANL)
  - Erste Tests mit F2x
  - Nicht seine Hauptaufgabe
  - Dankbar für die gute Dokumentation, gibt wertvolles Feedback

- Pyccel (Christopher Albert, Ahmed Ratnani)
  - „Python nach Fortran Compiler“
  - Sucht nach einem besseren Backend
  - Benutzt SymPy-AST → Gemeinsames Modell

- Mike Müller (Python Academie)
  - Sehr interessiert
  - Benötigt Callbacks um f2py zu Ersetzen

Stand und Ausblick

- Verfügbar unter der Apache Lizenz 2.0
- https://github.com/DLR-SC/F2x

- Unterstützt viele Fortran Konstrute
- Compilerunabhängig (getestet mit GNU und Intel)

- Templates:
  - BIND(C)
  - ctypes (Langsam!)

- Viele Verbesserungen in Planung (leider nur wenig Entwickler)
  - Callbacks
  - Verbesserter Parser
  - Übernahme des pyccel Modells

F2x „Roadmap“

- FD templates
  - Support for **`OPTIONAL`** (required for BACARDI)
  - Automatic proposal for interface configuration („*.f90-wrap“ file)
  - Support for **`INTERFACE`** construct

- F2x
  - Refactoring of CLI
  - Finish new templates
    - Cython
    - Support for Callbacks
  - Replace F2x AGT with Pyccel AST
  - Better (faster) Parser – different approaches possible

- Students wanted!

Zusammenfassung

- Raumrückstände sind ein Problem
- Tracking erfordert spezielle Software → BACARDI
- Open Source erleichtert Softwareentwicklung
  - Django
  - Airflow
  - Auf Nachhaltigkeit achten
- Open Source ist nicht immer der Universalhammer
  - F2x … jetzt schon

DLR