Programmieren und Mehr: Erfahrungen mit der Entstehung von research software engineering

Dr. Benjamin Fuchs

Über mich:

Python seit 2008

Data Science seit 2009

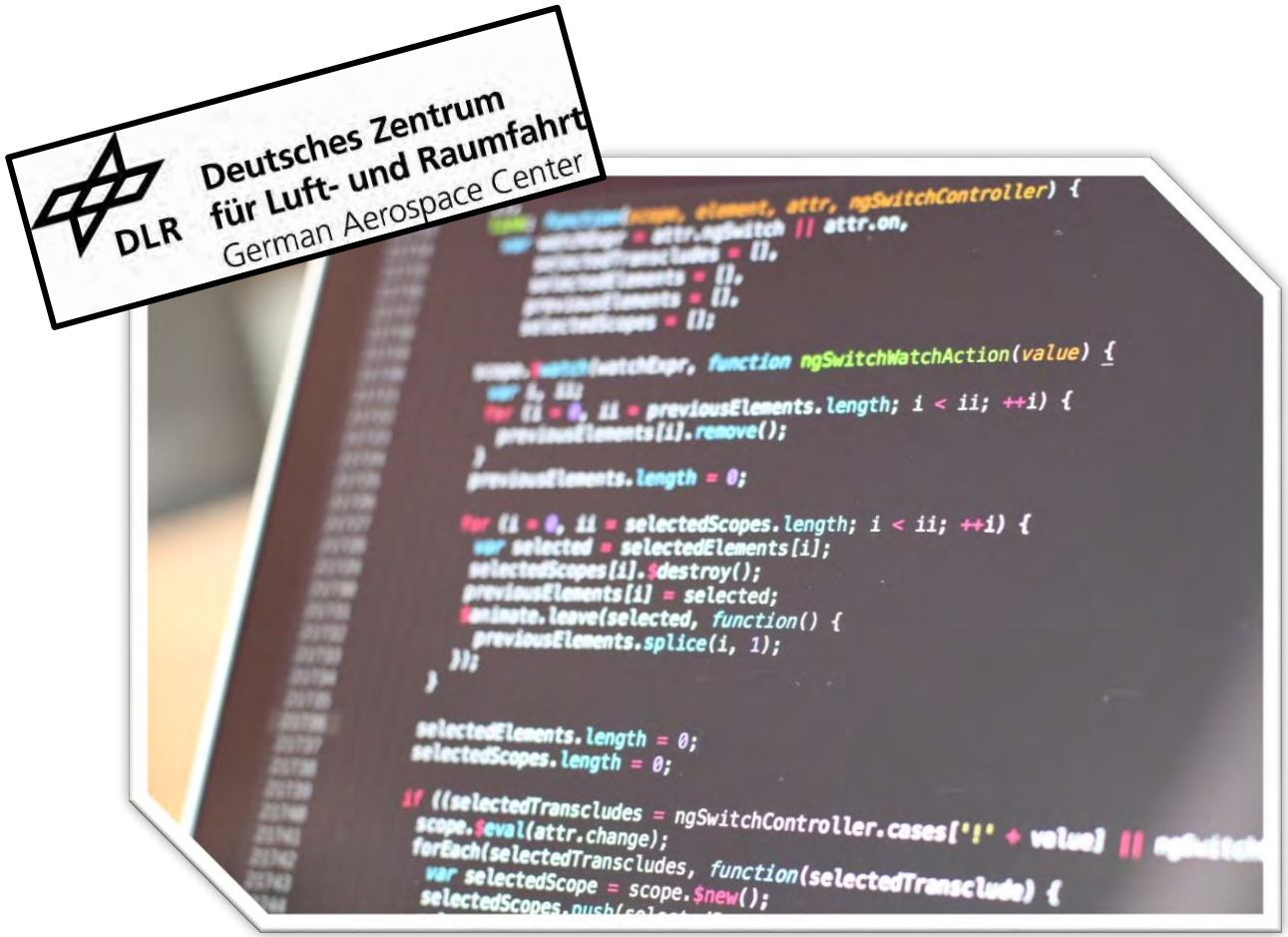
Research Software Engineering seit 2012

RSE = Forschungs-Software Entwickl(ung/er:in)

3 Jahre Doktorarbeit in der Astroteilchenphysik

Data Science, Data Science everywhere!

1 Jahr Entwicklung eines Open Data Portals
Lizenzen, Webframeworks, Application Development



Wechsel ans DLR, Institut VE, Abteilung Energiesystemanalyse

Entwicklung von Forschungssoftware (RSE)

Wartung und Entwicklung einer bestehenden Forschungssoftware

Problem:
“gewachsene Forschungs-Software”

Folgen:

strukturelle Mängel

inhomogene Konzeptanwendung

wenig Abstraktion

aufwändige Wartung

Was sind die fundamentalen Ursachen?

Großes Hindernis:

Fehlendes SE Konzept-Wissen

z.B. zu Architektur & Design

Konsequenz:

Entwicklungs-Entscheidungen werden

unbewusst und in technischer Unkenntnis

getroffen

Kommunikation und Umsetzungen

werden erschwert

Was tun?

Umgekehrte Perspektive:

Was ist das Besondere an RSE Wissen?

RSE Wissen umfasst:

Komplexitätsreduktion

Lösungsansätze (Architektur & Design)

Methodische Limitierungen

Tradeoffs

Was/Wieviel davon muss die Wissenschaftler:in wissen?

Meine Antwort:

Soviel, dass Wissenschaftler:innen
eigenständige Entscheidungen
während der Entwicklung treffen können

Wann sollten RSE Spezialisten konsultiert werden?

Was sind relevante Zusammenhänge (Tradeoffs)?

Wann sind bewusste Entscheidungen erforderlich?

Viele Wissenschatler:innen würden es gerne besser machen

Kein Mangel an Bereitschaft

aber

SE ist und bleibt Mittel zum Zweck

Wie bekommt man notwendiges SE Wissen an die Wissenschaftler:in?

SE Seminar

2 Wochen / 1,5h

Konzept oder Mini-Workshop

Vorteil: Skaliert gut mit Personenanzahl

Nachteil: beschränkt auf theoretische Konzepte

Fortgeschrittene Programmier-Kurse

6-8 Einheiten à 1-2h

1 Thema/Konzept mit Anwendungskontext + Hausaufgaben + Nachbesprechung

Vorteil: Verknüpft Praxis mit Theorie

Nachteil: Zeitintensiv

Begleitetes Programmieren

Selbstgewähltes Projekt der Wissenschaftler:in

Regelmäßiger Austausch, Code Reviews
Fokus: gezielt Erfahrungen schaffen

Vorteil: Effizient und Synergetisch

Nachteil: Flaschenhals RSE

Beispiel: ioProc

Entwicklung und Schulung unter einem Hut

Gitlab: https://gitlab.com/dlr-ve/esy/ioproc
PyPI: https://pypi.org/project/ioproc/
Paper: https://elib.dlr.de/136084/

Entwickler: Felix Nitsch, Jan Buschmann und Benjamin Fuchs

Was macht ioProc?

Wissenschaftlicher Workflow Manager

Gitlab: https://gitlab.com/dlr-ve/esy/ioproc
PyPI: https://pypi.org/project/ioproc/
Paper: https://elib.dlr.de/136084/

Entwickler: Felix Nitsch, Jan Buschmann und Benjamin Fuchs

Konzeption:

Bietet modulare, nachnutzbare, und reproduzierbare Workflows

Setzt Minimales Konzeptwissen voraus = Niedrige Anwendungshürde

Fördert modulare, kurze source code Einheiten + Doku

Einfache Weiterentwicklung vom Workflow zur Applikation

In Zukunft: Metadaten & Provenance

Gitlab: https://gitlab.com/dlr-ve/esy/ioproc
PyPI: https://pypi.org/project/ioproc/
Paper: https://elib.dlr.de/136084/

Entwickler: Felix Nitsch, Jan Buschmann und Benjamin Fuchs

Ausbildungsaspekte in ioProc

Gitlab: https://gitlab.com/dlr-ve/esy/ioproc
PyPI: https://pypi.org/project/ioproc/
Paper: https://elib.dlr.de/136084/

Entwickler: Felix Nitsch, Jan Buschmann und Benjamin Fuchs

ioProc wendet fortgeschrittene Methoden an:

Meta Programmierung,
Abstract Base Classes,
Code Introspection,
...

Gitlab: https://gitlab.com/dlr-ve/esy/ioproc
PyPI: https://pypi.org/project/ioproc/
Paper: https://elib.dlr.de/136084/

Entwickler: Felix Nitsch, Jan Buschmann und Benjamin Fuchs

Aber: Anwendungskomplexität ist gering (lineares Workflow-Management)

Gitlab: https://gitlab.com/dlr-ve/esy/ioproc
PyPI: https://pypi.org/project/ioproc/
Paper: https://elib.dlr.de/136084/

Entwickler: Felix Nitsch, Jan Buschmann und Benjamin Fuchs

komplexe Konzepte im Anwendungskontext erfahrbar und vermittelbar

Gleichzeitig hoher Bezug zur Forschungsrealität

Gitlab: https://gitlab.com/dlr-ve/esy/ioproc
PyPI: https://pypi.org/project/ioproc/
Paper: https://elib.dlr.de/136084/

Entwickler: Felix Nitsch, Jan Buschmann und Benjamin Fuchs

Ausbildung ist nur ein Teilansatz …

Arbeitsmethoden

Herausforderungen:

Wissensfluktuation durch dynamisch entstehende Teams

Meist zeitlich unterbrochene Entwicklung von Software

Umfang ändert sich durch Erkenntnisgewinn aus Entwicklung

Zielsetzung oft unklar oder unscharf

Agil oder lieber nicht?

Agile Methoden lassen sich adaptieren - Frameworks eher nicht

Erfolgreich adaptierte Methoden:

KANBAN Boards :: hilft der Koordination bei zerstückelten Arbeitszeiten

Kurze Sprints (1T / 1W) :: hilft Arbeiten zu Fokussieren

Standups :: hilft Kommunikation zu fokussieren

SCRUM Poker :: hilft bei der Aufwandsschätzung

Retrospektiven :: ermöglichen Verbesserungen am Prozess

Code Reviews :: verbessert Qualität + guter Wissenstransfer

Und zahlt sich das aus?

Ja

SE Wissenstand ist stetig gestiegen

Neuzugänge lernen mittlerweile von den „Alten“

Komplexe Entwicklungen werden erst möglich

Entwicklungen sind besser steuerbar

Wartungsarbeiten verteilen sich auf mehr Schultern

Synergien aus wiederverwendbaren Softwarebausteine

Aber: Das sind nur 2 der vielen Aspekte die eine Rolle spielen …

… Kommunikation

… Koordination

… psychologische/soziologische Faktoren

…

Meine Perspektive nach 10 Jahren RSE:

- RSE ist mehr als nur Programmieren oder Data Science
- Als RSE bin ich Teil des Teams und zugleich techn. Leitung
- nicht Programmieren für sondern mit der Wissenschaftler:in
- WAS und WIE das WARUM ist Sache der Wissenschaftler:in
- Vermitteln von SE Wissen, Strukturieren, befähigen informierte Entscheidungen zu treffen und bei der Umsetzung mithelfen

RSEs ähneln der Meister:in eines Werkstattlabors –
sie sind hoch spezialisiert und mit komplementärer Kompetenz zur Wissenschafler:in. RSE helfen Wissenschaftler:innen durch Spezialwissen und Methodenkompetenz. Sie leitet bei SE Arbeiten an und Unterstützen in der Umsetzung von Forschungsfragen.

Vielen, herzlichen Dank für den Campus Source Award und die Würdigung wissenschaflticher Software Entwicklung!

Und mein Dank geht auch an alle Anwesenden für ihre Zeit und die Gelegenheit meine Erfahrungen teilen zu dürfen.