Seminar „Softwareentwicklung in der Wissenschaft“

Studien zur Interaktion von Informatik und Naturwissenschaften

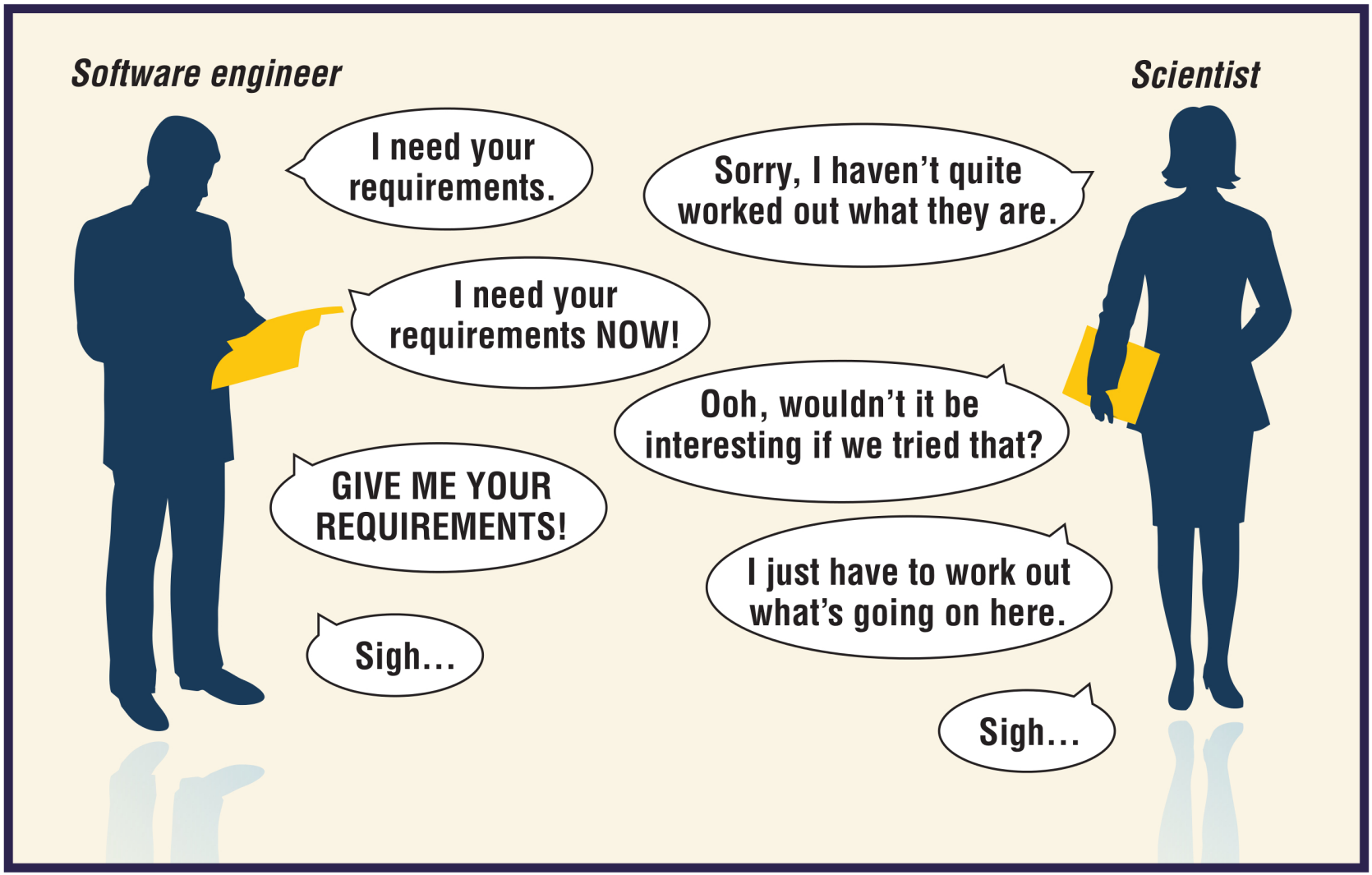
Leonie Pick

Arbeitsbereich Wissenschaftliches Rechnen

24. Juni 2013

Gliederung

Motivation



Bildquelle: Segal and Morris, 2008: Develloping Scientific Software, IEEE Software pp. 19

Hintergründe

- Entwicklung wissenschaftlicher Software:
  - Unklarer Bedarf

  $\Rightarrow$ Codes wachsen mit wissenschaftlichem Erfolg

  - Wissenschaft hat Priorität

  $\Rightarrow$ Fehlende Expertise in formaler Softwareentwicklung

  - Ablehnung „traditioneller“ Entwicklungsmethoden

- Steigende Anzahl von Applikationen auf Hochleistungsrechnern (HPC) im wissenschaftlichen Sektor

Hintergründe

- Entwicklung wissenschaftlicher Software:
  - Unklarer Bedarf

  $\Rightarrow$ Codes wachsen mit wissenschaftlichem Erfolg

  - Wissenschaft hat Priorität

  $\Rightarrow$ Fehlende Expertise in formaler Softwareentwicklung

  - Ablehnung „traditioneller“ Entwicklungsmethoden

- Steigende Anzahl von Applikationen auf Hochleistungsrechnern (HPC) im wissenschaftlichen Sektor

DARPA HPCS Programm 2004:

1. Sicht eines Softwareentwicklers:
   - Einfluss paralleler Modelle auf Produktivität testen
   - „Folklore“ in der „HPC Gemeinschaft“ zusammentragen

⇒ Basili et al. (2008)

2. Sechs repräsentative Fallstudien

⇒ Carver et al. (2007), Kendall et al. (2008)

DARPA HPCS Programm 2004:

1. Sicht eines Softwareentwicklers:
   - Einfluss paralleler Modelle auf Produktivität testen
   - „Folklore“ in der „HPC Gemeinschaft“ zusammentragen

⇒ Basili et al. (2008)

2. Sechs repräsentative Fallstudien

⇒ Carver et al. (2007), Kendall et al. (2008)

Ziele

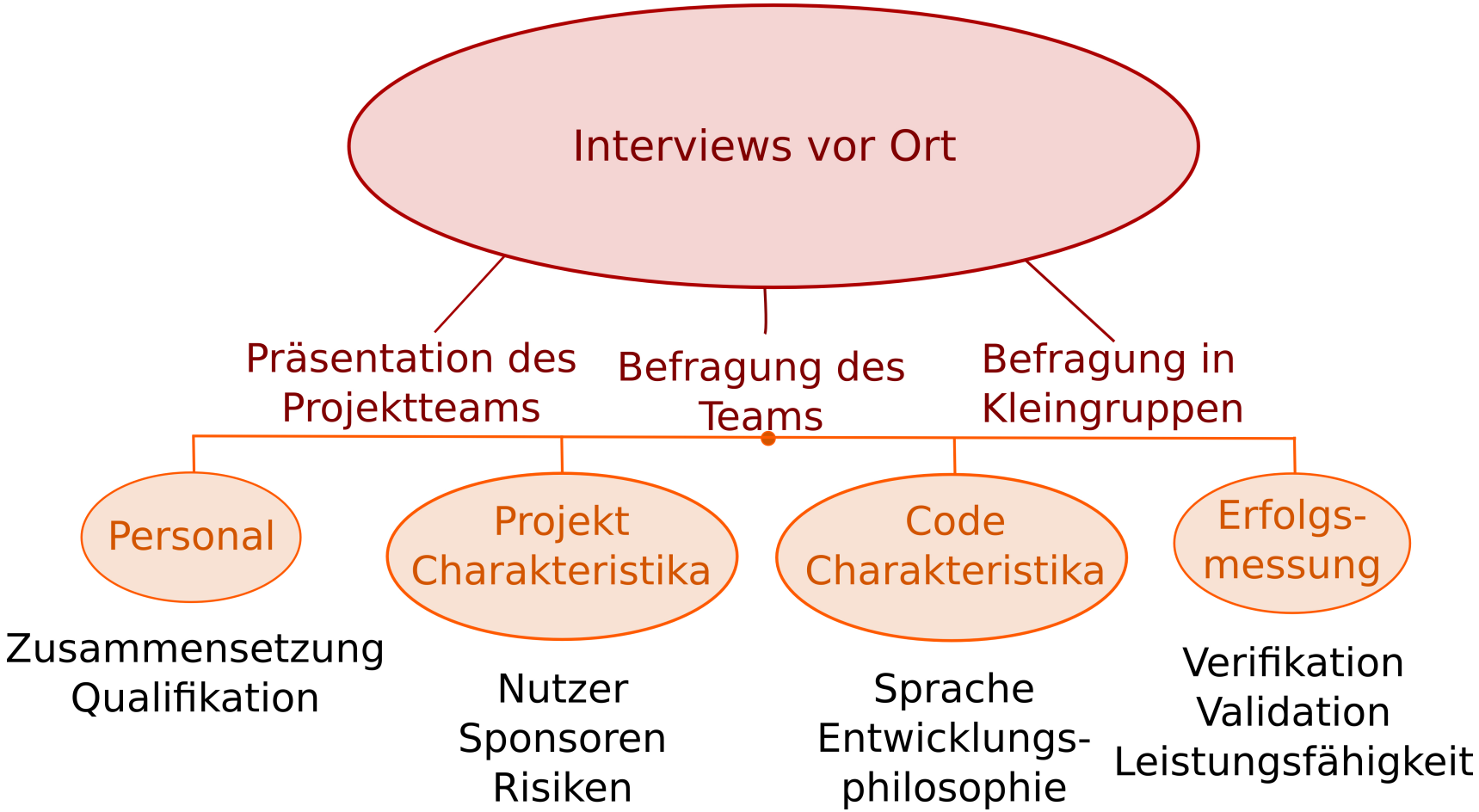
- Softwareentwickler produktiv in HPC Projekte integrieren
- Kritische Erfolgsfaktoren identifizieren
- Referenzkörper von Fallstudien erstellen
- Satz von „lessons learned“ ableiten

Ziele

- Softwareentwickler produktiv in HPC Projekte integrieren
- Kritische Erfolgsfaktoren identifizieren
- Referenzkörper von Fallstudien erstellen
- Satz von „lessons learned“ ableiten

Methodik


Projekt identifizieren
Teilnahme verhandeln
Fragenkatalog
Interviews vor Ort
Analyse
Nachbesprechung
Veröffentlichung


Interviews vor Ort
Präsentation des Projektteams
Befragung des Teams
Befragung in Kleingruppen
Personal
Projekt Charakteristika
Code Charakteristika
Erfolgs-messung
Zusammensetzung
Qualifikation
Nutzer
Sponsoren
Risiken
Sprache
Entwicklungs-philosophie
Verifikation
Validation
Leistungsfähigkeit

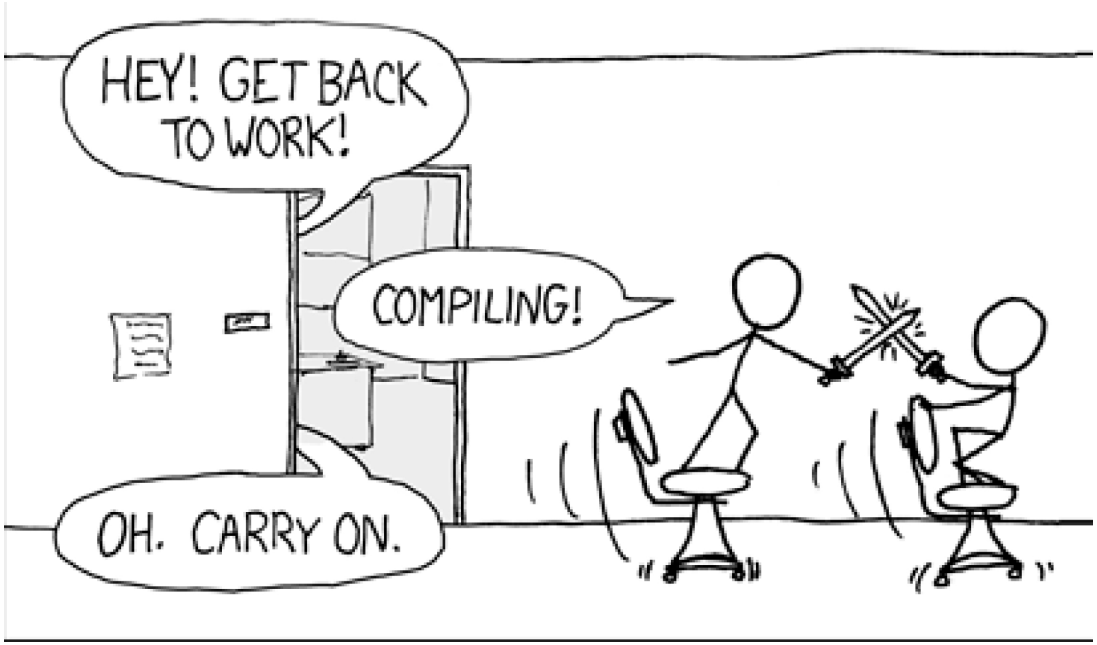
Entwicklungsumgebung - Teamgröße

„Lone researcher“ vs. „Community codes“

Erstellt mit Material von `http://xkcd.com/`

Lebenszyklus

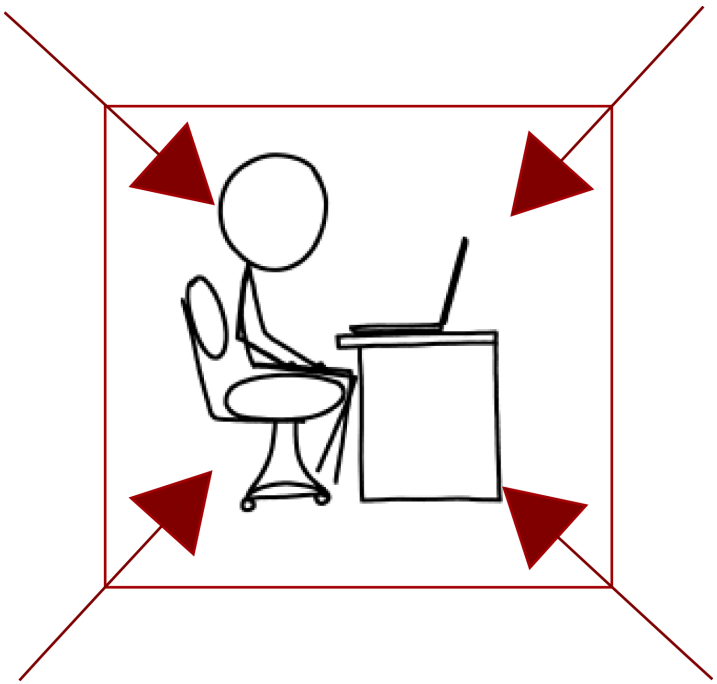
Entwicklungsdauer vs. Ausführungsdauer



Bildquelle: `http://xkcd.com/303/`

Nutzung

Interne Nutzung



Erstellt mit Material von `http://xkcd.com/`

Nutzung

vs. externe Nutzung

Erstellt mit Material von `http://xkcd.com/`

Projekte

	FALCON	CONDOR	NENE	OSPREY	HAWK	EAGLE
Anwendung	Produkt-verhalten	Produkt-verhalten	Modellierung	Modellierung	Fabrikation	Signal-bearbeitung
Lebenszyklus (Jahre)	$\sim$ 10	$\sim$ 20	$\sim$ 25	$\sim$ 10	$\sim$ 6	$\sim$ 3
Releases	9	7	?	$>$ 10	1	1
Personal (FTEs)	15	3-5	$\sim$ 10, 100te Helfer	10	3	3
Kunden	$<$ 50	100te	$\sim$ 100.000	$\sim$ 10.000	10er	keine
Codelänge (LOC)	$\sim$ 405.000	$\sim$ 200.000	750.000	$\sim$ 100.000	$\sim$ 134.000	$<$ 100.000
Sprache	F77 (24%), C (12%)	F77 (85%)	F77 (95%)	Fortran	C++ (67%), C (18%)	C++, Matlab
Status	✓	✓	✓	✓	✗	Demonstration

Nachzeichnung – basiert auf Carver et al. (2007), Tabelle 1

Projekte

„Fortran“ Gruppe „OO“ Gruppe

	FALCON	CONDOR	NENE	OSPREY	HAWK	EAGLE
Anwendung	Produkt-verhalten	Produkt-verhalten	Modellierung	Modellierung	Fabrikation	Signal-bearbeitung
Lebenszyklus (Jahre)	$\sim$ 10	$\sim$ 20	$\sim$ 25	$\sim$ 10	$\sim$ 6	$\sim$ 3
Releases	9	7	?	$>$ 10	1	1
Personal (FTEs)	15	3-5	$\sim$ 10, 100te Helfer	10	3	3
Kunden	$<$ 50	100te	$\sim$ 100.000	$\sim$ 10.000	10er	keine
Codelänge (LOC)	$\sim$ 405.000	$\sim$ 200.000	750.000	$\sim$ 100.000	$\sim$ 134.000	$<$ 100.000
Sprache	F77 (24%), C (12%)	F77 (85%)	F77 (95%)	Fortran	C++ (67%), C (18%)	C++, Matlab
Status	✓	✓	✓	✓	✗	Demonstration

Nachzeichnung – basiert auf Carver et al. (2007), Tabelle 1

Verifikation und Validation

Definition (Verifikation)

Nachweis, dass die Applikation die Gleichungen innerhalb des Algorithmus korrekt löst.

Definition (Validation)

Nachweis, dass die Applikation alle wichtigen Effekte modelliert.

CONDOR Teamleiter: „[...] you have to understand that the answer you are going to get is going to have a certain level of uncertainty in it. The neat thing about it is that it is easy to get an answer in the general sense.“

Verifikation und Validation

Definition (Verifikation)

Nachweis, dass die Applikation die Gleichungen innerhalb des Algorithmus korrekt löst.

Definition (Validation)

Nachweis, dass die Applikation alle wichtigen Effekte modelliert.

CONDOR Teamleiter: „[...] you have to understand that the answer you are going to get is going to have a certain level of uncertainty in it. The neat thing about it is that it is easy to get an answer in the general sense."

Programmiersprache

Kurs für paralleles Programmieren: „Java is for drinking.“

- Fortran Gruppe: FALCON, CONDOR, NENE, OSPREY
- OO Gruppe: HAWK und EAGLE
- FORTRAN bietet
  - Stetigkeit $\Rightarrow$ Gewöhnung der Nutzer
  - Leichte Erlernbarkeit
  - Portabilität, Wartung

CONDOR Entwickler: „I'd rather be closer to machine language than more abstract.“

Programmiersprache

Kurs für paralleles Programmieren: „Java is for drinking.“

- Fortran Gruppe: FALCON, CONDOR, NENE, OSPREY
- OO Gruppe: HAWK und EAGLE
- FORTRAN bietet
  - Stetigkeit ⇒ Gewöhnung der Nutzer
  - Leichte Erlernbarkeit
  - Portabilität, Wartung

CONDOR Entwickler: „I'd rather be closer to machine language than more abstract.“

Entwicklungsphilosophie

- Rigide Software Management Ansätze werden vermieden
- Keine Wiederbenutzung existierender HPC frameworks
- Keine Nutzung von IDEs

EAGLE Entwickler: „They all try to impose a particular style of development on me and I am forced into a particular mode.“

Entwicklungsphilosophie

- Rigide Software Management Ansätze werden vermieden
- Keine Wiederbenutzung existierender HPC frameworks
- Keine Nutzung von IDEs

EAGLE Entwickler: „They all try to impose a particular style of development on me and I am forced into a particular mode.“

Software von Drittanbietern

- Extern entwickelte Software ist ein Risiko
- Häufig keine Tools zur parallelen Entwicklung (Debugging)
- Tools aus Eigenherstellung oder Open-Source
- Neue Technologien, die mit älteren koexistieren können, haben eine größere Erfolgschance

Leistungsfähigkeit

- Allgemeine Entwicklungsziele:
  - Korrektheit
  - Leistungsfähigkeit
  - **Portabilität**
  - Wartbarkeit

IBM scientific users group conference: „[Floating-point operations per second] rates are not a useful measure of science achieved.“

- Wissenschaftliche Ziele bestimmen die benötigte Leistungsfähigkeit
- Leistungsfähigkeit konkurriert mit Portabilität, Wartbarkeit und Verständlichkeit

Leistungsfähigkeit

- Allgemeine Entwicklungsziele:
  - Korrektheit
  - Leistungsfähigkeit
  - **Portabilität**
  - Wartbarkeit

  IBM scientific users group conference: „[Floating-point operations per second] rates are not a useful measure of science achieved.“

- Wissenschaftliche Ziele bestimmen die benötigte Leistungsfähigkeit
- Leistungsfähigkeit konkurriert mit Portabilität, Wartbarkeit und Verständlichkeit

Teamzusammensetzung

- Komplexität der Domäne und Software Komplexität
- Weniger als 20% Informatiker

HAWK Entwickler: „[...] You need an equal mixture of subject theory, the actual physics, and technology expertise.“

Nutzer und Sponsoren

- Von kommerzieller IT abweichendes Geschäftsmodell
- Finanzierung oft von staatlichen Behörden
- Nutzer sind häufig unabhängig
- HAWK: Technischer Erfolg, trotzdem keine ausreichende Nutzer-Basis

Take home

- Verifikation und Validation sind problematisch
- FORTRAN statt objektorientierter Sprachen
- Flexible Entwicklungsphilosophie
- Externe Softwarelieferungen gelten als Risiko
- Trade-Off zwischen Leistungsfähigkeit, Portabilität und Wartbarkeit
- Expertise in Teams ist ungleich verteilt
- Zufriedenheit von Nutzer und Sponsoren ist kritisch

Take home

Was sollte getan werden?

- Existierende SE Praktiken für HPC Domäne zuschneiden
- Wiederbenutzung von Frameworks auf großem Maßstab
- HPC Kurse an Universitäten speziell für Wissenschaftler (`http://www.hpcbugbase.org/`)

Literatur

Victor R. Basili, Jeffrey C. Carver, Daniela Cruzes, Lorin M. Hochstein, Jeffrey K. Hollingsworth, Forrest Shull, and Marvin V. Zelkowitz.

Understanding the high-performance-computing community: A software engineer's perspective.

IEEE Softw., 25(4):29–36, July 2008.

Jeffrey C. Carver, Richard P. Kendall, Susan E. Squires, and Douglass E. Post.

Software development environments for scientific and engineering software: A series of case studies.

In *Proceedings of the 29th international conference on Software Engineering*, ICSE '07, pages 550–559, Washington, DC, USA, 2007. IEEE Computer Society.

R. Kendall, J.C. Carver, D. Fisher, D. Henderson, A. Mark, D. Post, C.E. Rhoades, and Susan Squires.

Development of a weather forecasting code: A case study.

Software, IEEE, 25(4):59–65, 2008.