Einleitung Fallstudien Ergebnisse

01 **Leonie Pick** Page no. **1**

19/6/2013 13:51

Im Rahmen dieses Seminars habe ich mich mit wissenschaftlichen Papern zur Interaktion von Informatik und Naturwissenschaften bei der Softwareentwicklung beschäftigt. Basierend auf der Auswertung von Fallstudien wird der Fokus meines Vortrags auf der Darstellung der speziellen Entwicklungsumgebung von interdisziplinären HPC (High Performance Computing) Projekten liegen. Anhand ausgesuchter Projekt-Parameter werde ich typische Missverhältnisse zwischen beiden Domänen aufzeigen.

02 **Leonie Pick** Page no. **2**

19/6/2013 14:17

Ich habe meinen Vortrag folgendermaßen gegliedert: Einleitend werde ich die Auseinandersetzung mit der Interaktion zwischen Informatikern und Naturwissenschaftlern in HPC Projekten kurz motivieren, bevor ich in den Hintergründen auf die zugrundeliegenden Arbeiten und deren Ziele eingehe. Es folgt ein Abschnitt über die bereits angesprochenen Fallstudien, in dem ich zunächst das methodische Vorgehen erläutern werde. In Vorbereitung auf die Vorstellung der in den Fallstudien ausgewählten Projekte werde ich anschließend den heterogenen Charakter der HPC Entwicklungsumgebung hervorheben. In den Ergebnissen stelle ich 7 Erkenntnisse, welche die Autoren aus den Fallstudien gewonnen haben, vor. Anschließend fasse ich die Essenz des Vortrags in den "Take Home Messages" zusammen.

03 **Leonie Pick** Page no. **3**

19/6/2013 14:36

Zur Motivation habe ich diese Illustration (fast eine Karikatur) aus dem Artikel von Segal und Morris von 2008 ausgesucht, der eigentlich in Renkos Themengebiet fällt. Trotzdem passt sie als Einstieg hier gut, weil die Unterhaltung des Softwareentwicklers mit der Wissenschafterin über den Bedarf eines wahrscheinlich bevorstehenden Projekts bereits einen wesentlichen Unterschied in der Herangehensweise beider Berufsgruppen deutlich macht: Während der Softwareentwickler versucht, zwecks Planung vorab eine Bedarfsliste zusammenzustellen, hat die Wissenschaftlerin zu diesem Zeitpunkt noch keine Ahnung, was auf sie zukommt. Hinsichtlich der Planung eines Projekts bestehen offenbar große Unterschiede, die einer symbiotischen Zusammenarbeit im Wege stehen (beide Seufzen verständnislos). Es lohnt sich deshalb, die Interaktion beider Professionsgruppen zu untersuchen, um die Zusammenarbeit zukünftig effektiver zu gestalten.

04 **Leonie Pick** Page no. **4**

19/6/2013 15:00

Die Entwicklung von Software für wissenschaftliche Anwendungen unterscheidet sich von kommerzieller IT in drei wesentlichen Punkten: 1) Oftmals starten Projekte als Forschungsprojekte und wachsen auf Grundlage des wissenschaftlichen Erfolgs. Deswegen ist eine Bedarfsermittlung oder die Festlegung eines Ablaufplans a priori nicht möglich, was auch die Illustration gerade deutlich gemacht hat. 2) Das Produzieren von wissenschaftlich korrekten Ergebnissen hat eine größere Priorität als die Sicherstellung der Softwarequalität im Sinne eines Informatikers. Die Teams bestehen daher überwiegend aus Naturwissenschaftlern, weswegen es an Expertise in formaler Softwareentwicklung mangelt. 3) Letztlich scheint eine psychologisch verankerte Abneigung gegenüber traditionellen prozess-orientierten Entwicklungsmethoden vorzuherrschen. Individuelle Flexibilität wird klar bevorzugt.

05 **Leonie Pick** Page no. **5**

19/6/2013 16:23

Die Anzahl von naturwissenschaftlichen Applikationen, die auf parallelen Hochleistungsrechnern laufen, hat zugenommen. Derartige Anwendungen sind typischerweise die Simulation physikalischer Phänomene wie z.B. globaler Klimawandel, Erdbeben, Atomreaktionen, Materialverhalten usw., die 10.000de Prozesse simultan beantspruchen, wobei die Kommunikation meist über MPI geregelt wird. Aufgrund der wachsenden Relevanz macht es Sinn, Projekte zu evaluieren, in denen diese Klasse von (HPC-) Software entwickelt wird.

06 **Leonie Pick** Page no. **6**

19/6/2013 16:36

Die zugrundeliegenden Artikel präsentieren Teilergebnisse des "High Productivity Computing Systems" Programm, das 2004 von der "Defense Advanced Research Projects Agency", einer Behörde des US Verteidigungsministeriums, ins Lebens gerufen wurde. Übergeordnetes Ziel des Programms ist die Weiterentwicklung des Hochleistungsrechnens, sowohl im Bereich der Hardware als auch der Software. Hinsichtlich der Software wurde der Einfluss existierender Parallelisierungsmodelle, wie z.B. MPI, openMP, Unified Parallel C usw., auf die Produktivität anhand von Studenten in Programmierungskursen acht verschiedener Universitäten getestet. Außerdem wurde sogenannte "Folklore" in der HPC Gemeinschaft zusammengetragen, womit eine stillschweigende informale Übereinkunft hinsichtlich der Parallelisierungsmethoden gemeint ist. Den Terminus "HPC Gemeinschaft" werde ich im Anschluss noch diskutieren. Die Ergebnisse dieser Untersuchungen werden in dem Artikel "Understanding the High-Performance-Computing Community: A Software Engineer's Perspective", geschrieben von Basili et al. und erschienen 2008 in IEEE Software, zusammengestellt. Aufgrund des Titels und weil die Autoren ausnahmslos Informatiker sind, habe ich diesen Block unter "Sicht eines Softwareentwicklers" zusammengefasst.

07 **Leonie Pick** Page no. **7**

19/6/2013 17:05

Mein Fokus liegt allerdings mehr auf dem Teil des HPCS Programms, in dem Fallstudien 6 repräsentativer Softwareprojekte durchgeführt und 2007 bzw. 2008 von Carver et al. bzw. Kendall et al. wiederum in IEEE Software veröffentlicht wurden. Das Autorenteam besteht sowohl aus Softwareentwicklern als auch aus Naturwissenschaftern und einer Anthropologin. Aufgrund der breiten Übereinstimmung der Ergebisse in beiden Blöcken, habe ich mich um eine integrative Darstellung der Ergebnisse bemüht. Aussagen, die explizit auf den oberen Artikel (Basili et al.) zurückgehen, habe ich in lila farbkodiert.

08 **Leonie Pick** Page no. **8**

19/6/2013 17:20

Die Ziele der genannten Studien sind sowohl Wege zu finden, ausgebildete Softwareentwickler produktiv in wissenschaftliche HPC Projekte zu integrieren,

09 **Leonie Pick** Page no. **9**

19/6/2013 17:25

als auch kritische Erfolgsfaktoren für das Projekt als solches zu identifizieren. Die Menge der Fallstudien kann in einem Referenzkörper zusammengefasst werden und die abgeleiteten "Lessons learned", die ich in den Ergebnissen besprechen werde, der Community zugänglich gemacht werden.

10 **Leonie Pick** Page no. **10**

19/6/2013 17:44

Das Vorgehen bei den Fallstudien lässt sich in drei Einheiten gliedern, nämlich die Vorbereitungsphase (grüne Ellipse), die Kernphase, in der die eigentliche Befragung stattfindet (rote Ellipse) und die Nachbereitungsphase (blaue Ellipse). Zur Vorbeitung gehört zunächst die Identifikation eines geeigneten Projekts mit dessen Sponsoren und die Verhandlung mit beiden Parteien über die Teilnahme an der Studie. Oftmals fühlen sich die beteiligten Programmierer sicherer, wenn der Geldgeber mit der Evaluation einverstanden ist. In diese Phase gehören außerdem Absprachen zur Diskretion, alle Projekte und die beteiligten Personen werden anonymisiert, und Logistik. Vorbereitend auf die vis-à-vis Interviews vorort wird dem Projektteam ein Fragenkatalog zugeschickt, was sowohl der Reduktion der Interviewzeit dient, als auch den Befragten Recherchezeit einräumt. Die aufgenommenen Interviews werden dann vom Befragungsteam analysiert, in der Nachbereitungsphase vom Projektteam und den Sponsoren rückgeprüft und die Ergebnisse letztlich veröffentlicht.

11 **Leonie Pick** Page no. **11**

19/6/2013 17:56

Ein Zoom auf den Punkt "Interviews vor Ort" zeigt die einzelnen Interviewetappen und deren wichtigste inhaltliche Themen, auf die auch in den Ergebnissen Bezug genommen wird: Nachdem das Projektteam ihr Projekt selbst vorgestellt und dabei Prioritäten artikuliert hat, wird das Team erst als Ganzes und ggf. anschließend in Kleingruppen befragt. Inhaltlich betreffen die Fragen vor allem die Gebiete Personal, Projekt- und Code Charakteristika sowie Erfolgsmessung. Hinsichtlich des Personals interessieren vor allem die Zusammensetzung, wie lange die Mitglieder bereits zusammenarbeiten und welche Qualifikationen sie haben. Projekt Charakteristika umfassen natürlich auch Punkte wie Ziele, Bedarf und Leistungen, hier in besonderem Maße aber auch die Nutzergruppe, Sponsoren und identifizierte Risiken. Unter Code Charakteristika ist neben der Programmiersprache besonders das, was man unter "Entwicklungsphilosophie" zusammenfassen kann, von Bedeutung. Selbstverständlich muss ein erfolgreicher Code im Rahmen der Möglichkeiten verifiziert und validiert sein. Kritisch im Milieu der wissenschaftlichen Applikationen ist dagegen die Beziehung zwischen Erfolgmessung und Leistungsfähigkeit.

12 **Leonie Pick** Page no. **12**

19/6/2013 18:12

Bevor ich die sechs ausgewählten Projekte überblicksartig vorstelle, möchte ich anhand von drei Parametern der Entwicklungsumgebung deutlich machen, dass keine einheitliche "HPC Community" existiert: Die Teamgrößen können erheblich variieren. Der "lone researcher" ("einsame Wissenschaftler") steht "community codes" ("Gemeinschaftscodes") gegenüber.

13 **Leonie Pick** Page no. **13**

19/6/2013 18:17

Neben der Teamgröße ist der Lebenszyklus eines Codes ein weiterer charakteristischer Projekt Parameter: Ein Code, der wenige Male ausgeführt wird, mag zwar weniger Entwicklungszeit brauchen, dafür aber umso mehr Ausführungszeit. Ein häufig ausgeführter Code braucht länger in der Entwicklung, um Portabilität und Wartbarkeit zu garantieren, dafür weniger Ausführungsdauer.

14 **Leonie Pick** Page no. **14**

19/6/2013 18:23

Letztlich variiert die Nutzergruppe: Der ausschließlichen Eigennutzung von Entwicklern

15 **Leonie Pick** Page no. **15**

19/6/2013 18:24

steht die Nutzung von anderen Mitgliedern der Organisation (z.B. andere US Regierungslabore) und die kommerzielle Nutzung gegenüber.

16 **Leonie Pick** Page no. **16**

19/6/2013 18:26

Diese Tabelle stellt die Charakteristika der sechs ausgewählten HPC Projekte zusammen. OSPREY (alle Projekte erhalten Pseudonyme) ist hervorgehoben, weil es separat im Artikel von Kendall et al. behandelt wurde (alle anderen in Basili et al.). Hinichtlich des Anwendungsgebietes passen FALCON, CONDOR und HAWK zusammen, weil sie alle ein Produktverhalten (z.B. von Verbundmaterial beim Herstellungsprozess (HAWK)) unter spezifischen Bedingungen (z.B. großer Spannung (CONDOR)) simulieren. Dabei treten stark gekoppelte physikalische Effekte über mehrere Größenordnungen auf. Der Sponsor kann so aufwendige, teure, potentiell gefährliche Tests (wenn überhaupt möglich) oder Prototypen minimieren. HAWK ist das einzige Projekt, an dem die Arbeit eingestellt wurde, worauf ich später noch eingehen werde. NENE und OSPREY sind aufwendige Modellierungsapplikationen. NENE berechnet Moleküleigenschaften auf Grundlage quantenphysikalischer Modelle als Teil einer Software-Suite. OSPREY gehört einer Wettervorhersage-Suite an und kombiniert großskalige atmosphärische Modelle mit ozeanographischen Modellen. Ein Vergleich zischen NENE und OSPREY zeigt die angesprochene Heterogenität in der HPC-Domäne auf: Während NENE im universitären Umfeld mit 100ten stark fluktuierenden Mitwirkenden und externen Softwarezulieferungen entwickelt wird, besteht das Kernteam vob OSPREY aus 10 Physikern, die seit 10 Jahren eng zusammenarbeiten. Einer Codeverwaltung mit "Subversion" und der Finanzierung von Code-orientierten Zielen bei OSPREY steht die manuelle Integration jeder Codezeile und der wissenschafts-orientierte Erwerb von Geldern bei NENE gegenüber. Weil EAGLE ursrünglich nicht als Applikation geplant war und entsprechend keine Nutzer hat, fällt dieses Projekt etwas heraus. Im Rahmen eines Forschungsprojekts soll die Anwendbarkeit von Echtzeit-Processing von Sensor-Daten auf spezialisierter HPC Hardware im Feldeinsatz demonstriert werden, weswegen der Status "Demonstation" lautet.

17 **Leonie Pick** Page no. **17**

19/6/2013 19:39

Der Übersichtlichkeit wegen habe ich die sechs Projekte gemäß des Parameters Sprache in die "FORTRAN GRUPPE" (blau) und die "OO Gruppe" (grau, OO steht für objektorientiert) untergliedert. Hintergrund dieser Einteilung ist die Tatsache, dass die Wahl der vorrangig genutzen Programmiersprache von anderen Projekt-charakterisierenden Parametern wie Portabilität und Wartbarkeit abhängt, was "Sprache" sozusagen zu einem übergeordneten Charakteristikum macht.

18 **Leonie Pick** Page no. **18**

19/6/2013 19:52

In den Ergebnissen werde ich jetzt die von den Autoren sogenannten "lessons learned" (erlernte Lektionen) aus den Fallstudien vorstellen: Offensichtlich ist die Tatsache, dass Verifikation und Validation in diesem Bereich der Applikationen problematisch sind. Verifikation ist der Nachweis, dass die Applikation die Gleichungen innerhalb des Algorithmus korrekt löst. Validation ist der Nachweis, dass die Applikation alle wichtigen Effekte modelliert. Simulationen produzieren eine Approximation für einen Gleichungssatz, der analytisch nicht lösbar ist. Insofern ist Verifikation lediglich qualitativ hinsichtlich gewünschter Eigenschaften (z.B. Stabilität, Gültigkeit von Erhaltungsgleichungen) oder einer problemabhängigen Genauigkeit möglich. Validation ist prinzipiell mittels eines Vergleichs zwischen Vorhersage und Messungen möglich. Praktisch ist es aber häufig so, dass die experimentelle Überprüfung aufgrund von Sicherheit, Kosten- , Zeitaufwand oder rechtlichen Einschränkungen nicht möglich ist.

19 **Leonie Pick** Page no. **19**

19/6/2013 20:11

Dazu ein passendes Zitat des CONDOR Teamleiters: "[...] 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."

20 **Leonie Pick** Page no. **20**

19/6/2013 20:14

Ein Kommentar aus einem Programmierkurs soll zum Thema Programmiersprache überleiten: " Java is for drinking"

21 **Leonie Pick** Page no. **21**

19/6/2013 20:20

Das Zitat macht deutlich, dass objektorientierte Programmiersprachen in der HPC-Domäne wenig Anwendung finden, was sich in der Größe der "FORTRAN Gruppe", verglichen mit der "OO Gruppe" widerspiegelt. Im Gegensatz zu den "OO-Projekten" haben die "FORTRAN-Projekte" längere Lebenszyklen von bis zu 25 Jahren (NENE) mit einer großen Nutzer-Community (10.000 OSPREY, 100.000 NENE). Das Beibehalten von FORTRAN über diese Jahre garantiert eine Stetigkeit, die es den Nutzern möglich macht, sich an den Code zu gewöhnen und auf Code-Level zu interagieren. FALCON wird beispielsweise signifikant von den Nutzern weiterentwickelt, verifiziert und validiert. FORTRAN ist außerdem verglichen mit C++ einfach zu erlernen, was im universitären Umfeld mit großer Fluktuationsrate wie im Fall von NENE essentiell ist. FORTRAN erleichtert außerdem Portabilität und Wartung, da komplizierte "make-" und "link-" Befehle obsolet sind. Ein zusätzliches Argument für die Benutzung von Fortran liefert ein CONDOR Entwickler in seinem Kommentar: "I'd rather be closer to machine language than more abstract." Ein größeres Abstraktionsniveau mindert die Verständlichkeit des Codes. Die Entwickler haben bei der Nutzung von FORTRAN größeres Vertrauen in die resultierenden maschinellen Anweisungen.

22 **Leonie Pick** Page no. **22**

19/6/2013 20:41

Unter Entwicklungsphilosophie haben ich drei Punkte aus den Artikeln zusammengefasst: Zunächst wird festgestellt, dass rigide, prozess-orientierte Software Management Ansätze vermieden und agile Entwicklungsmethoden (unwissentlich) klar bevorzugt werden. Grund hierfür könnte der unbekannte a-priori-Bedarf sein. Hinsichtlich der Ablehnung von Rigidität passt auch die Feststellung, dass vorhandene Frameworks (z.B. Common Component Architecture - CCA) nicht genutzt werden. Weil man Frameworks nicht inkrementell anwenden kann, fehlt benötigte Flexibilität, um die erwünschten physikalischen Gleichungen zu implementieren. Das resultiert in der Überzeugung, es koste mehr Aufwand das Problem in das Framework zu integrieren als ein eigenes zu erstellen.

23 **Leonie Pick** Page no. **23**

19/6/2013 20:57

Fehlende Flexibilität ist weiterhin ein Grund, warum Integrated Development Environments (IDEs, z.B. Eclipse) wenig Anwendung finden, was auch das Zitat des EAGLE Entwicklers verdeutlicht: "They all try to impose a particular style of development on me and I am forced into a particular mode." Zusätzlich wird IDEs nicht zugetraut, eine Aufgabe genauso auszuführen, wie man es in der Unix Command Line tun würde, wo man "näher am Metall" ist. Ein dritter Grund ist der Fakt, dass IDEs keine Einrichtungen haben, um Jobs an entfernte batch queues zu schicken und Debugging zu betreiben. Solange diese Funktionen nicht eingerichtet werden, ist es unwahrscheinlich, dass IDEs in Zukunft mehr genutzt werden.

24 **Leonie Pick** Page no. **24**

20/6/2013 12:03

Die Tool-Situation in der HPC-Entwicklung ist heikel: Extern entwickelte Software gilt als Risiko, weil aufgrund des instabilen Markts nicht garantiert werden kann, dass das Tool wärend des gesamten Lebenszyklus der Applikation lieferbar ist. Bei NENE wird dieses Risiko abgemildert, indem externe Software lediglich in unkritischen Codeteilen genutzt wird. Außerdem existieren bislang schlicht keine guten Tools zur parallelen Entwicklung (speziell für das Debugging). Diejenigen, die existieren, können häufig nicht effektiv genutzt werden, weil sie die Latenzen von Fernverbindungen auf Clustern (batch queues) nicht berücksichtigen. Deshalb werden bislang häufig lediglich Tools aus Eigenherstellung oder Open-Source Tools genutzt. Allgemein kann man sagen, dass neue Technologien eine größere Chanche haben, in HPC Projekten Verwendung zu finden, wenn sie mit alten koexistieren können und nicht eine komplette Umstellung erfordern.

25 **Leonie Pick** Page no. **25**

20/6/2013 12:35

Das Thema Leistungsfähigkeit (Performance) sorgt oft für Missverständnisse zwischen Wissenschaftlern und Informatikern. Weil auf parallelen Hochleistungsrechnern gearbeitet wird, liegt die Vermutung nahe, Leistungsfähigkeit müsse eine übergeordnete Priorität haben. Bei einem Ranking der wichtigsten Projektziele schafft es die Leistungsfähigkeit allerdings lediglich auf Platz zwei hinter "wissenschaftliche Korrektheit". Übertragbarkeit (Portabilität) ist das einzige Ziel, das von allen Befragten als "wichtig" eingestuft wurde.

26 **Leonie Pick** Page no. **26**

20/6/2013 12:46

Leistungsfähigkeit hat also keine Monopolstellung, was das folgende Zitat von einer IBM Konferenz unterstreicht: "[Floating-point operations per second] rates are not a useful measure of science achieved." Ein wahrscheinlich angebrachteres Leistungsmaß in den betrachteten Projekten wäre wohl "wissenschaftlich korrekte und nutzbare Ergebnisse pro Zeiteinheit". Die benötigte Leistungsfähigkeit wird von den wissenschaftlichen Zielen bestimmt. Können Deadlines (z.B. für eine Publikation, Präsentation beim Sponsor etc.) eingehalten werden, gibt es keinen Grund, das Programm hinsichtlich der Performance zu optimieren. Zusätzlich konkurriert die Leistungsfähigkeit mit Portabilität, Wartbarkeit und Verständlichkeit. Weil die Applikation auf möglichst vielen Plattformen (SGI, Linux Networkx, IBM...) lauffähig sein soll, kann der Code nicht System-spezifisch optimiert werden. Änderungen des Source-Codes zwecks Performance-Tuning machen das Programm unverständlicher und es ist schwieriger zu warten. Wenn die Leistungsfähigkeit verbessert werden soll, passiert das meist auf größerem Maßstab als Code-tuning und involviert ein komplettes Überdenken der zugrundeliegenden Physik.

27 **Leonie Pick** Page no. **27**

20/6/2013 12:59

HPC Projekte zeichnen sich durch die Komplexität der Domäne (nämlich Naturwissenschaften-meist Physik) einerseits und der Softwarekomplexität (parallel auf Hochleistungsrechnern) andererseits aus. Diese Bipolarität spiegelt sich nicht in der Zusammensetzung und Expertise der Projektmitglieder wider. Die Teams bestehen im Schnitt lediglich aus 20 % ausgebildeten Softwareentwicklern. Ursache ist der Glaube, dass es einfacher ist, einem Naturwissenschaftler programmieren beizubringen, als einem Informatiker die breiten naturwissenschaftlichen Grundlagen für das Problem zu erklären. Ein HAWK Entwickler sieht das allerdings (wie auch ich persönlich) anders: "[...] You need an equal mixture of subject theory, the actual physics, and technology expertise."

28 **Leonie Pick** Page no. **28**

20/6/2013 13:06

Der letzte Punkt, auf den ich eingehen möchte, ist das wichtige Verhältnis zu Nutzern und Sponsoren. Den untersuchten HPC-Projekten liegt ein von kommerzieller Software abweichendes Geschäftsmodell zugrunde. Sie werden meist von staatlichen Behörden finanziert, die nicht mit der Gruppe der Nutzer zusammenfallen muss. OSPREY wird beispielsweise kostenlos mit einem Lizenz-Manager über das WEB unabhängigen Privatpersonen zugänglich gemacht. So kann es passieren, dass ein Projekt in den Augen des Sponsors erfolgreich ist und trotzdem keine Nutzer-Basis hat, die groß genug ist, um Weiterarbeit an diesem Projekt zu rechtfertigen. Genau das ist der Grund, weswegen die Arbeit an HAWK eingestellt wurde.

29 **Leonie Pick** Page no. **29**

20/6/2013 13:18

Ich fasse die Kernaussagen der sieben genannten Punkte zusammen: Verifikation und Validation sind bei wissensachftlicher Simulations-Software im herkömmlichen Sinne problematisch, weil keine exakte Lösung bekannt und eine experimentelle Überprüfung praktisch nicht möglich ist. Hier muss ein Umdenken seitens der Informatiker stattfinden. FORTRAN wird objektorientierten Sprachen vorgezogen, hauptsächlich aufgrund von Gewöhnung, leichter Erlernbarkeit und einfacher Übertragbarkeit. Keiner dieser Gründe schließt eine stärkere Nutzung von oo-Sprachen aus. Es überwiegt eine Entwicklungsphilosophie, in der individuelle Flexibilität rigiden Prozessabläufen eindeutig vorgezogen wird. Existierende HPC Frameworks werden deshalb genauso ungern genutzt wie IDEs. Extern entwickelte Tools werden kaum eingesetzt und gelten als potenzielles Risiko, weswegen auf die Entwicklung eigener Tools und Open-Source Produkte gesetzt wird. Leistungsfähigkeit ist nicht das primäre Ziel, sondern konkurriert mit Portabilität und Wartbarkeit. Allgemein bestimmen wissenschaftliche Ziele die benötigte Leistungsfähigkeit. In den Teams überwiegt der Anteil an reinen Naturwissenschaftlern deutlich. Das ist in Anbetracht der Software-Komplexität nicht zielführend und sollte geändert werden. Letztlich ist nicht nur die Zufriedenheit der Kunden kritisch für den Erfolg des Projekts, sondern ebenso die der (teilweise unabhängigen) Nutzer.

30 **Leonie Pick** Page no. **30**

20/6/2013 13:36

Was sollte also getan werden, um Softwareingenieure produktiv in HPC-Projekte zu integrieren? Existierende SE Praktiken müssen für die HPC Domäne zugeschnitten werden. Die Nutzung von IDEs und Debugging Tools für batch queues sind Beispiele. Die Wiederbenutzung von Frameworks sollte auf größerem Maße stattfinden. Informatiker können helfen, diese Frameworks anzupassen. Letztlich sollte bereits an den Universitäten begonnen werden, Wissenschaftler in HPC Entwicklung zu schulen, was hier an der Uni Hamburg bereits praktiziert wird. Als Support wurde die "HPC-Bugbase" eingerichtet, auf der sich auch für uns ein Besuch lohnt.