Universität Hamburg
Fakultät für Mathematik, Informatik und Naturwissenschaften

64 – 169

Seminararbeit

Softwareentwicklung in der Wissenschaft

Dr. Hermann Lenhart

Masihola Schojai Aziz
Thema: „Softwareentwicklung in der Praxis“
Matrikel Nummer: 6583184
Studiengang: B.Sc Wirtschaftsinformatik / 4. Fachsemester
E – Mail: masiaziz@live.de
Abgabedatum: 25.08.2014

Gliederung

Einführung

Software hat in den letzten Jahren eine enorme Verbreitung entdeckt bzw. gefunden. Heutzutage gibt es kaum noch irgendwelche Geräte oder Maschinen, in denen die Bedienung und Steuerung nicht über die Software realisiert werden. Software trägt ganz entscheidend zum Funktionieren der Geräte bei. Darüber hinaus, ist Software auch im Unternehmen ein wichtiger Bestandteil. Softwaresysteme müssen gut und zuverlässig sein, damit die Geschäftsprozesse reibungs- sowie Problemlos ablaufen und somit bestimmte Aufgaben im Unternehmen gelöst werden können.

Die gesamte Lebenszeit eines Softwareproduktes - von der Entstehung des Produktes, über die Inbetriebnahme und Wartung, bis zur Ablösung des Produktes durch ein anderes - bezeichnet man als Software – Lebenszyklus. Es gibt bestimmte Aufgabenbereiche in der Softwareentwicklung. Die folgenden Phasen wie zB. Analyse , Entwurf, Implementierung und Einführung ist weit verbreitet. In der Praxis laufen die Phasen allerdings nicht immer zeitlich nacheinander ab. Die frühen Phasen, Analyse und Entwurf, sind ziemlich kritisch, denn mehr als die Hälfte der Fehler der fertiggestellten Softwareprodukte haben ihre Ursache in den fehlerhaften Anforderungen. Solche Fehler sind meistens ziemlich teuer und aufwendig zu beheben. Vorgehensmodelle haben die Aufgabe, den komplexen Prozess der Entwicklung und die anschließenden Wartung in überschaubare Teilaktivitäten zu zerlegen und deren Ergebnisse, logischen- und zeitlichen Zusammenhang festzulegen und zu definieren.

Ich habe mich für dieses Thema entschieden, da ich mich schon immer für Softwareentwicklung interessiert habe. Darüber hinaus haben wir (Wirtschaftsinformatiker) im 1. Fachsemester das Modul „Grundlagen der Wirtschaftsinformatik“ und dort haben wir das Thema Softwareentwicklung nochmal intensiv behandelt.

In dieser Arbeit möchte ich den Ablauf einer Softwareentwicklung darstellen. Zunächst einmal gehe ich auf die Theorie ein, welche ich auch in meiner Powerpoint – Präsentation intensiv behandelt habe. Zum Ende der Arbeit möchte ich die Theorie mit der Praxis anhand des Papers *„ Understanding the High – Performance – Computing Community“* verknüpfen.

Softwareentwicklung

Softwaretechnik bzw. Softwareentwicklung ist eine deutschsprachige Übersetzung des engl. Begriffs *software engineering* und beschäftigt sich mit der Herstellung von Software.

„Zielorientierte Bereitstellung und systematische Verwendung von Prinzipien, Methoden und Werkzeugen für die arbeitsteilige, ingenieurmäßige Entwicklung und Anwendung von umfangreichen Softwaresystemen“ [Allgemeine Definition von Helmut Balzert]

Bestandteile guter Softwareentwicklung

Um Software zu entwicklen bzw. herzustellen, benötigt man Prinzipien. Diese bestimmten Prinzipien sollen das Programmieren optimieren und besser machen. Im Vordergrund stehen zwei Punkte. Zunächst einmal muss das Programm schnell und gut entwickelt werden. Man muss jedoch beachten, dass das Programm auch nach dem Erstellen Veränderbar sein muss. Denn es kommt häufiger vor, dass ein Programm Fehler enthält und diese beheben werden müssen. Durch die Veränderung entsteht dann ein neuer Code. Dann muss auch die Qualität stimmen. Qualität ist in meinen Augen sogar sehr wichtig, denn man sollte ein möglichst gutes Programm schreiben. Eine gute Qualität bedeutet, dass mit wenig Zeitaufwand irgendwelche Änderungen vorgennommen werden können. Eine Softwareentwicklung muss vorher gut geplant werden. Mehrere Schritte müssen vorher beachtet werden. Da die Entwicklung einer Software ziemlich komplex und aufwendig ist, muss vorher ein Plan erstellt werden. Dieser Plan ist wiederrum unterteilt in zeitlich und inhaltlich begrenzte Phasen. Diese Phasen durchlaufen den allgemeinen Prozess. Somit wird die Software Schritt für Schritt hergestellt. Man kann nicht von 0 auf 100 direkt mit der Programmierung bzw. Softwareentickwlung anfangen. Sobald der Plan bekannt ist, startet der Prozess. Die Phasen sind alle eng miteinander verknüpft.

Kernprozesse:

1. Planung
2. Analyse
3. Entwurf
4. Programmierung
5. Validierung und Verifikation

Unterstützungsprozesse:

6. Anforderungsmanagement
7. Projektmanagement
8. Qualitätsmanagement
9. Konfigurationsmanagement
10. Softwareeinführung
11. Dokumentation

[Foliensatz 3]

Teilgebiete

1. Planung

Bei der Planung werden folgende Phasen durchlaufen:

- Anforderungsanalyse
- Lastenheft
- Pflichtenheft
- Auffwandsschätzung
- Vorgehensmodell

Bei der Anforderungsanalyse versucht man die Anforderungen des Auftraggebers an das zu entwickelnde System zu ermitteln, strukturieren und zu entwickeln. Die Anforderungsanalyse selbst ist in der Informatik ein Teil des Systementwicklungsprozesses. Sobald man die Anforderungen der Anforderungsanalyse vollständig ermittelt hat , erfolgt im Prozess der nächste Schritt. Das Lastenheft beschreibt und beinhaltet die Gesamtheit der Anforderungen. Dazu ist das Lastenheft das Ergebnis einer Anforderungsanalyse und somit auch ein Teil des Anforderungsmanagements im Unternehmen. Das Lastenheft beschreibt somit, was und wofür etwas gemacht werden soll. Wurde im Prozess nun ein Lastenheft angenommen, folgt mit dem Pflichtenheft die nächste Phase im Prozess. Das Pflichtenheft beschreibt in der Regel wie und womit etwas realisiert werden soll. Genau aus diesem Grund ist die Reihenfolge auch so festgelegt, da die Anforderungen durch die Leistungen erfüllt werden.Das Pflichtenheft beschreibt konkret, wie wir oder der Auftragnehmer die Anforderungen des Auftraggebers zu lösen hat. Bei der Aufwandsschätzung wird geschätzt, wie viele Personen bzw. Mitarbeiter im Unternehmen- und wie viel Zeit für die einzelnen Schritte benötigt werden. Darüber hinaus wird auch noch darauf geachtet , welche Ressourcen gebraucht und verbraucht werden und wie viel Kosten auf das Unternhemen zukommt. Die Aufwandschätzung ist in der Softwareentwicklung ein wichtiger Baufaktor und Bestandteil bei der Planung eines Projektes.

Da allgemein Software ziemlich schwer zu erstellen ist, arbeiten Softwareentwickler mit einem Plan, um die Entwicklung der Software zu realisieren. Im Planungsprozess wird zunächst berücksichtigt, mit welchen vorgegebenen Mitteln unser Projekt bzw. Ziel erreicht werden kann und welche Mittel wir überhaupt anwenden können und müssen, um das Ziel zu erreichen. Diesen Schritt beschreibt man auch als “Vorgehensmodell“. Schon im Managementbereich eines Unternehmens werden im Planungsprozess die Vorstellungen und Planungen eines Projektes schriftlich festgehalten. Vorgehensmodelle spalten einzelne Aktivitäten auf verschiedene Phasen im Entwicklungsprozess auf und die werden einmal (Wasserfallmodell) oder mehrmals durchlaufen (Spiralmodell). Es gibt aber auch drei unterschiedliche Typen von Vorgehensmodellen (Softwareentwicklungsprozess, Softwarelebenszyklusmanagement und die Softwareentwicklungs – Philosophie). Das Vorgehensmodell hilft im allgemeinen dabei, den Ablauf eines Projektes zu strukturieren.

[Foliensätze 4 und 5]

2. Analyse

"Eine Analyse ist eine systematische Untersuchung, bei der das untersuchte Objekt oder Subjekt in Bestandteile zerlegt wird und diese anschließend geordnet, untersucht und ausgewertet werden. Insbesondere betrachtet man Beziehungen und Wirkungen zwischen den Elementen" [allgemeine Definition nach Wikipedia]

Im Analyseverfahren macht man mehrere Analysen um zu gucken, wie man vorzugehen hat. Diese Analysen werden durchgeführt, um die Arbeitschritte und die Durchführung festzulegen. Die System, - Strukturierte und Objektorientierte Analyse sind die wichtigen Analysen, die im Kernprozess einer Softwareentwicklung durchgeführt werden. Die Systemanalyse ist die erste Phase im Entwurfsprozess. Die Systemanalyse hat das Ziel, die Umwelt ohne Maschine zu beschreiben
(Ist – Zustand). Davon ausgehend versucht man von diesem Ist – Zustand eine Maschine zu planen. Das Soll – Model jedoch zeigt, wie die Maschine aussehen soll. Durch die beiden Unterschiede soll klar gemacht werden, was die Maschine leisten soll. Bei der Systemanalyse wird nicht die Implementation der Maschine untersucht. Als Maschine hier unterscheidet man die Software und Hardware.

Die Strukturierte Analyse wurde erstmals im Jahre 1977 von Tom DeMarco veröffentlicht. Die Analyse ist eine Methode zur Erstellung einer formalen Systembeschreibung, die überwiegend in der Softwareentwicklung wieder gefunden wird. Die Strukturierte Analyse wird während der Analysephase eines Software – Projektes eingesetzt. Nach der Strukturierten Analyse soll ein Anforderungsdokument für Umfang und Inhalt der betrieblichen Anwendung entstehen, die im Softwaresystem realisiert werden soll. Die Analyse ist eine graphische Analysemethode, die sich an der Top – Down – Methode orientiert. Die Software für den Zentralrechner des Kampflugzeuges "Tornado" wurde mit Hilfe der Strukturierten Analyse realisiert. In der Praxis wurde Sie von der Objektorientierte Analyse abgelöst , jedoch wird sie aber immernoch in vielen Projekten eingesetzt.

Die Objektorientierte Analyse und Design (OOAD) sind beides objektorientierte Varianten. Im Entwicklungsprozess eines Softwaresystems beschäftigen sie sich mit der Anforderungsanalyse sowie dem Systementwurf. In der Objektorientierten Analyse geht es darum, die Anforderungen zu erfassen und zu beschreiben, die das zu entwickelnde Softwaresystem erfüllen soll. Man sucht und sammelt in der Analyse nach Punkten, hält sie schriftlich fest und überprüft sie. Das Ergebnis der Analyse ist das Produktmodell, welches auch Objektorientierte Analyse – Modell genannt wird. Das Modell beinhaltet Diagramme und wichtige darstellungen von Kontrollstrukturen. Darüber hinaus wird dann beim Objektorientierten Design das Modell weiterentwickelt und passend für den Systementwurf erstellt bzw. so angepasst, dass der Entwurf dadurch vereinfacht wird. Das erstellte Modell wird in einer Softwarearchitektur umgeformt. Das Modell enthält Information über die technische Umsetzung. Zusätzlich dient es auch als Vorlage für die Implementierung in einer bestimmten Programmiersprache.

3. Entwurf

Der Entwurf einer Software ist im allgemeinen der “Entwurfsprozess“ und ist ein Teil des gesamten Softwareentwicklungsprozess. Bevor man aber mit dem Prozess anfangen möchte , sollte vorher immer eine Architektur vorhanden sein. Anhand dieser können Entwickler diesen Prozess Schritt für Schritt abarbeiten. In der Softwareentwicklung gibt es eigentlich keinen klar festgelegten und strukturierten Entwicklungsprozess. Denn es kann immer wieder zu einer Veränderung kommen. Sollte unser Programm fehler enthalten, müsste man diese Beheben und so den Code neu bzw. Teilweise schreiben. Allein durch solche, nicht vorher gesehenen Probleme gibt es keinen reibungslosen Ablauf. Der Entwurf einer Softwarearchitektur ist die Grobstruktur eines Softwaresystems , denn in der klassischen Softwareentwicklung repräsentiert die Architektur den frühesten Entwurf.Der Entwurfsprozess läuft meist iterativ und inkrementell ab. Dieser Entwurf wird durch bestimmte Aspekte wie Sicherheit oder auch Wiederverwendbarkeit bestimmt. Eine Softwarearchitektur ist verknüpft mit guten und wichtigen Faktoren des Softwareprojekts. Denn für die Benutzer und auch Entwickler der einzelnen Softwareprojekte dient eine gute Architektur für ein gutes Verständnis. Anhand dessen, kann die Entwicklung der Software basierend auf der Architektur aufgebaut bzw. entwickelt werden. Da wir nun festgestellt haben, dass die Softwarearchitektur der erste Schritt für den Entwurf ist, führen wir das mit dem Strukturierten Design fort. Denn das Strukturierte Design ist ein Entwurfsmuster in der Softwareentwicklung. Das Strukturierte Design verknüpft das Analyseverfahren und die Implementierung miteinander. Denn das Design stellt die inhaltliche Planung und Strukturierung einer Implementierung dar. Darüber hinaus werden auch die vorgesehenen Strukturen für das System bestimmt.

Um Software zu entwicklen ist es nicht sinnvoll ohne irgendeinen Entwurf, irgendwelche Planungen oder Vorbestimmungen zu beginnen. Es gibt natürlich bestimmte Kriterien, die erfüllt werden müssen, um einen guten Entwurf zu entwerfen, denn der Entwurfsprozess ist unterteilt in zwei Strukturen. Es gibt im Entwurfsprozess eine bestimmte Gliederung. Sie wird in der Praxis genauso angewendet. Zunächst einmal gibt es den sogenannten Grobentwurf (Gesamtstruktur). In diesem Entwurf werden Softwarearchitekturen sowie Schnittstellen und Subsystem – Spezifikation intensiv behandelt. Sollte dieser Entwurf nochmal verfeinert werden, geht man rüber zur Detailstruktur. Denn diese Struktur ist der Feinentwurf im Prozess. Algorithmen und bestimmte Datenstrukten werden hier entworfen um den Entwurf , wie schon oben erwähnt, einmal mehr zu verfeinern. Der Entwurf ist das wichtigste Element in der Softwareentwicklung. Alles basiert auf den Entwurf. Die Softwareentwickler haben anhand des Entwurfes eine bestimmte Vorgehensweise. Diese Vorgehensweise berücksichtigt alles. Die bestimmten Programmiermethoden sowie die Programmiersprache. Sie können auch sehen, ob das eher ein kleines oder großes Projekt sein wird. Mit dieser Informationen können die Entwickler sogar selbst festlegen, ob sie in einem Team programmieren oder eher alleine. Der Entwurf ist die wichtigste Schnittstelle in der Softwareentwicklung.

[Foliensatz 13]

4. Programmierung

Nachdem wir nun das Entwurfsmuster fertig gestellt haben, ist im Entwicklungsprozess die Programmierung der nächste Schritt. Die Programmierung bezeichnet die Tätigkeit, Programme bzw. Computerprogrammen zu erstellen. Die Programme werden mit bestimmten Programmiersprachen beschrieben und formuliert. Die weltweit am häufigsten eingesetzte Programmiersprache ist C sowie deren Weiterentwicklungen C++ und C#. Aber auch andere bestimmte und populäre Betriebssysteme sind teilweise in C verfasst wie z.B. Windows , Linux oder auch Mac. Die zweithäufigste Programmiersprache der Welt ist Java. Java wird vorallem im Internet und bei Spielen angewendet. Darüber hinaus gehört Java zu den weitverbreitesten Computersprachen und dient dort als Basis der Softwareentwicklung diverser Programme. Die Programmiersprache Java verwendet auch zur Ausführung einen eigenen Interpreteur undzwar die Java Virtual Machine. Mann muss hier aber auch erwähnen, dass die Java Virtual Machine in C++ verfasst ist.
In solchen Sprachen übersetzen die Entwickler bzw. Programmierer die Anforderungen und die bestimmten Algorithmen, die für das System notwendig sind.
Programmierer haben darüber hinaus noch die Aufgabe, ihr Programm auf Fehler zu testen. Dokumentationen und Behebungen von Programmcodes muss er ebenfalls erledigen. Der Programmierungsprozess ist ziemlich übersichtlich aufgebaut. Erstmal werden Programmmethoden und Sprachen vorgeschlagen. Je nachdem, wenn man sich geeinigt hat, bestimmt man die Programmiersprache und vorallem, wie man es Programmiert. Man setzt die Gruppen ebenfalls fest. Bei größeren Projekten werden immer Teams festgelegt.

Natürlich gibt es bestimmte Qualitätskriterien die ein Programmier erfüllen muss. Denn der Auftraggeber setzt bestimmte Kriterien vorraus. Die Qualität der Software steht im Vordergrund. Ein Programm muss die im Entwurf gemachten Vorgaben wiedergeben bzw. vorgeben und muss fehlerfrei sein. Jedoch ist kein Programm Fehlerfrei. Es kommen häufig zwei Arten von Fehlern vor (Syntax und Semantikfehler) Syntaxfehler sind meistens kleine Tippfehler. Semantische Fehler sind komplizierter zu beheben, denn das Programm arbeitet nicht so wie gewünscht. Die Umsetzung ist hier meistens das Problem. Das Programm sollte wie geplant korrekt ablaufen. Ein anderer wichtiger Punkt bei der Programmierung ist die Robustheit. Denn auch teure und vielfach geteste Software können Fehler enthalten. Ein Programm sollte so wenig Fehler wie möglich haben und eine Software gilt erst dann als Robust wenn fehler nur sehr selten auftreten. Damit unser Programm auch dauerhaft arbeitet, muss sie wertbar sein. Das Programm sollte Änderungen vornehmen können, ohne jedoch einen großen Aufwand zu haben. In der Softwareentwicklung stehen Programmen verschiedene Ressourcen zur Verfügung. Meistens sind dies Laufzeit und Speicherverbrauch. Entwickler mit guten Programmierkenntnissen können dazu beitragen, dass Ressourcen weniger verbraucht werden. Die Effizienz beim Programmieren spielt dabei eine enorm wichtige Rolle. Die Programmierung selbst sollte sich dem Entwurf anpassen, denn anhand des Entwurfes kann man beobachten wie man am besten vorzugehen hat. Programme können nicht alleine Programmiert werden, so setzt man in Teams bei größeren Projekten , die Arbeitsweise fest.

Ansätze werden mit Verfeinerungen verknüpft und so wird das Programmieren erleichtert.

5. Validierung und Verifikation

Die Validierung ist in der Informatik die Beweisführung, dass ein System oder Programm richtig Programmiert wurde und die Anforderungen in der Praxis auch erfüllen. Hier gibt es im wesentlichen nur eine Frage
(„Wird das richtige Programm entwickelt“?) die beantwortet werden muss. Die Verifikation jedoch ist der Nachweis, dass ein vermuteter Sachverhalt wahr ist
(„Ist das System richtig gebaut“?)
Bevor man jedoch aber sicherstellen kann , ob das richtige Programm entiwckelt und ob das System richtig entwickelt wurde, führt man Tests durch. Mit diesen Softwaretests prüft man die Qualität der Software, denn das Messen der Qualität ist das wesentliche Ziel des Softwaretestens. Die Testergebnisse werden schriftlich festgehalten und zusätzlich Dokumentiert. Das Testen soll Vertrauen in die Qualität der Software schaffen. Um die Software testen zu wollen, durchläuft man einen bestimmten Testprozess. In der Praxis wird der Prozess in der Regel eingehalten um die Software dementsprechend zu testen.

Am Anfang des Prozess steht der Testplan. Dieser Plan wird für das Projekt ausgearbeitet und soll den gesamten Ablauf des Testprozess darstellen. Dieser Testplan beinhaltet ziemlich wichtige Punkte. Zum einen die Teststrategie. In der Teststrategie werden Aspekte wie Testumfang und Risikoabschätzung intensiv behandelt. Beim Testumfang werden Art und Größe des Projektes besprochen und bei der Risikoabschätzung beobachtet man mögliche Risiken. Risiken die das Projekt mit sich bringen könnte. Testziele, Testbeginn und Ende sowie Testumgebungen werden hier auch noch vorher geklärt. Nach der Testplanung geht man zu der Vorbereitung hinüber. Bei der Testvorbereitung werden die festgelegten Aspekte, die in der Planung erfasst wurden, vorbereitet und zur Verfügung gestellt. Einzelne Dokumente zum Test werden hier bereitgestellt und Verfügbar gemacht. Bestimmte Werkzeuge die zu einem Testfall gehören , werden hier in den Testumgebungen aufgebaut. In der Testspezifikation werden nun alle Festlegungen und Vorbereitungen getroffen, um ein Test durchführen zu können. Es gibt immer einzelne Aktivitätsfälle wie z.B Beschreibung des Testfalles oder auch Vorbedingungen. Sobald nun alle Testphasen durchlaufen wurden sind, kommt es zur Testausführung. Nachdem die Entwickler das Programm entwickelt haben, erfolgt bei der Ausführung ein Test, welcher aber nicht von den Entwicklern getestet wird. Denn in der Regel wird Software von anderen Personen, die mit der Software nicht unbedingt in Verbindung stehen, getestet und auch ausgewertet. Denn Entwickler sehen meistens keine Fehler in ihren Codes, da Sie der Meinung sind, alles wäre fehlerfrei. Bei der Auswertung werden die Testergebnisse ausgewertet und überprüft. Das IST – Ergebnis wird hier dabei mit dem SOLL – Ergebnis verglichen und überprüft. Aus diesem Vergleich wird eine Entscheidung über das Testergebnis herbeigeführt. Bei Fehlern im Test bleibt der Testfall offen und man versucht die Ursache herauszufinden und zu gucken, wie man am besten diesen Fehler beheben kann bzw. könnte. Sollte der Testverlauf ohne Probleme ablaufen, gilt unser Testfall dann als “OK“. Für alle Tests werden Dokumentationen durchgeführt, d.h. alles wird Schriftlich festgehalten. Der Status zum Abschluss von Teststufen wird auch

kommuniziert. Entscheidungen werden getroffen und aufgelistet. Beim Testabschluss wird nach zwei Kriterien unterschieden (Ziele erreicht und Alternativen möglich).

[Foliensätze 19 und 20]

Unterstanding the High – Performance – Computing Community

A Software Engineers Perspective

Rechenbetonte Wissenschaftler, die Software für HPC Systeme entwicklen, stehen großen Softwaretechnikproblemen gegenüber. Seit mehreren Jahren hatten Softwareingenieure die Gelegenheit, die Entwicklung der Softwaresysteme zu beobachten. Es waren Wissenschaftler, meistens Geowissenschaftler, die diese Software programmiert haben und meistens ihren eigenen Code verwendeten. Viele Softwareentwickler haben ihre Grundausbildung von anderen Wissenschaftlern. Obwohl die Wissenschaftler Jahre lang software geschrieben haben, setzen Sie sich an die Ausbildung der Softwareentwickler fest. Viele der Codes sollen am Anfang nicht groß sein. Sie fangen klein an und wachsen dann auf der Grundlage von ihrem wissenschaftlichen Erfolg. Viele Entwicklungsteams verwenden ihren eigenen Codes. Mehrere SE Methoden arbeiten mit dem Gedanken, dass gute Ideen in anderen Entwicklungsumgebungen zu den Bedürfnissen der HPC Gemeinschaft ganz falsch angepasst werden. Kommen wir aber nun einmal zum Hintergrund des Papers.

Eine Liste der 500 schnellsten Supercomputer aus dem Jahre November 2007 zeigt dass das stärkste System 212,992 Prozessoren hatte. Obwohl eine gegebene Anwendung alle diese Prozessoren nicht alltäglich verwenden würde, würde sie regelmäßig einen hohen Prozentsatz von ihnen für einen einzelnen Job verwenden. Das Verwenden von Zehntausenden von Prozessoren auf einem einzelnen Projekt wird als normal in dieser Gemeinschaft betrachtet. Wir haben uns für Codes interessiert, die nichttriviale Kommunikation unter den individuellen Prozessoren während der Ausführung verlangen. Obwohl HPC Systeme in vielen Bereichen sehr nützlich sind, soll eine allgemeine Anwendung physische Phänomene wie Erdbeben, globale Klimaveränderung oder Kernreaktionen simulieren. Diese Codes müssen geschrieben werden, um den Parallelismus von HPC Systemen ausführlich anzuspannen. Obwohl viele Parallele programmierende Modelle bestehen, ist das dominierende Modell MPI, eine Bibliothek, wo der Programmierer ausführlich die ganze Kommunikation angibt. Fortran(Programmiersprache) bleibt weit verwendet, um neue HPC Software zu entwickeln, wie C und C ++ tun. Oft vereinigt ein einzelnes System vielfache Programmiersprachen. Wir haben sogar mehrere Projekte gesehen, wo dynamische Sprachen wie Pytho verwendet wurden sind, um verschiedene Module zu verbinden, die in einer Mischung von Fortran, C, und C ++ geschrieben sind.

2004 hat Darpa das Hohe Produktivitätsrechensystemprogramm gestartet, um neue HPC Technologie bedeutsam vorzubringen, indem hier Verkäufer-Anstrengungen unterstützt werden um die Systeme der folgenden Generation zu entwickeln, indem Sie sich auf die Hardware und Softwareproblemen fokussieren. Außerdem hat Darpa auch Forscher finanziell unterstützt. Unsere anfängliche Rolle sollte bewerten, wie kürzlich vorgeschlagene Sprachen die Programmierer-Produktivität betreffen. Darüberhinaus hat einer von uns geholfen, eine Reihe von Fallstudien zu führen, die auf die Betonung von der Ausführungszeit bis zur Zeit zur Lösung, die sowohl Entwicklung als auch Ausführungszeit vereinigt. Wir haben diese Forschung begonnen, indem wir kontrollierte Experimente geführt haben, um den Einfluss von verschiedenen par-allel-programmierungs Modellen zu messen. Weil die vorgeschlagenen Sprachen noch nicht verwendbar waren, haben wir

verfügbare Technologien wie MPI, OpenMP, UPC , Co-Reihe Fortran und Matlab*P mit Studenten in Parallele programmierenden Kursen von acht verschiedenen Universitäten studiert. Um diesen Spielraum dieser Forschung breiter zu machen, haben wir "Volkskunde" — d. h. die stillschweigende, unformalisierte Ansicht der Gemeinschaft davon gesammelt, was wahr ist. Wir haben es zuerst durch eine Fokus-Gruppe von HPC Forschern gesammelt, indem wir HPC Praktiker überblickt haben, die am HPCS Programm beteiligt waren, indem sie eine Stichprobenerhebung von Praktikern einschließlich akademischer theoretischer Forscher, Technologen interviewert haben, die neue HPC Systeme und Projektmanager entwickeln. Schließlich haben wir Fallstudien von Projekten sowohl an der US-Regierung als auch an den akademischen Laboratorien geführt. Das Ziel der Wissenschaftler ist es , wissenschaftlich zu arbeiten und dabei nicht die Software zu benutzen.

Ein mögliches Maß der Produktivität sind die wissenschaftlich nützliche Ergebnisse im Laufe der Kalenderjahre. Das bezieht genügend simulierte Zeit und Entschlossenheit. Darüber hinaus die genügende Genauigkeite der physischen Modelle und Algorithmen.

Am Anfang haben wir geglaubt, dass die Wissenschaftler höchste priorität an der Leistung der zu Entwickelnden HPC Syteme gelegt haben. Jedoch, nach eingehenden Interviews, haben wir herausgefunden, dass sich wissenschaftliche Forscher darauf konzentrieren, Öffentliche-Ergebnisse zu erzeugen. Das Schreiben von Codes, die auf die Leistung der Effizienz auf den HPC Systemen beruhen. Es ist ein Mittel zu einem Ende, jedoch nicht an einem Ende an sich. Obwohl dieser Punkt offensichtlich klingen könnte, finden wir, dass viele in der HPC Gemeinschaft es überblicken können. Die Absicht eines Wissenschaftlers ist es, neue wissenschaftliche Kenntnisse zu erzeugen. Also, wenn Wissenschaftler ihre rechenbetonten Simulationen mit der Zeit und den Mitteln durchführen können die ihnen auf dem HPC System zugeteilt sind, sehen sie kein Bedürfnis oder Vorteil bzw. Nutzen darin, die Leistung zu Optimieren. Sie sehen keine Notwendigkeit nach der Optimierung, wenn sie die Simulation an der gewünschten Genauigkeit mit den zugeteilten Mitteln bereits vollenden können. Wenn Optimierung notwendig ist, ist es häufig breit, nicht nur einschließlich traditioneller Informatik-Begriffe, der Codeeinstimmung und Algorithmus-Modifizierung, sondern auch des Umdenkens der zu Grunde liegenden mathematischen Annäherungen und potenziell im Wesentlichen der Änderung der Berechnung. Also Technologien, die sich nur auf die Codeeinstimmung konzentrieren, gelten in dieser Gemeinschaft als ein beschränktes Dienstprogramm.

Fazit: Die Entscheidungen der Wissenschaftler basieren auf die Maximierung der Wissenschaftlichen Outputs , jedoch aber nicht der Programmleistung.

Die Wissenschaftler benutzen bzw. verwenden zur Zeit Codes , die eine bestimmte Laufzeit haben. Wissenschaftler würden nur dann, einen neuen Code entwickeln wollen, wenn Sie sicher sind , dass der Code einen längeren Lebenszyklus hat. Wissenschaftler können selber garnicht richtig Programmieren , denn die nötigen IT – Kenntnisse fehlen. Sie eignen sich ihr Wissen von anderen Wissenschaftlern an. Die Zusammenarbeit von Wissenschaftlern und Softwareentwicklern sollte verbessert werden, denn es gibt immer mehr Projekte wo die Wissenschaft sich mit der Softwareentwicklung verbindet. Die Wissenschaftler haben weder Lastenheft noch Pflichtenheft angegeben und setzen sich an keine Anforderungen. Hier sollen die Softwareentwickler mit ihrem Wissen behilflich sein und den Wissenschaftlern helfen. Eine Andere Methode zur Programmierung von Projekten bzw. Codes wäre Pair Programming. Ein Wissenschaftler setzt sich mit einem Entwickler hin, um ein Programm zu

schreiben. Welches natürlich sinnvoll ist, da der Softwareentwickler bessere Programmierkenntnisse mit sich bringt, als der Wissenschaftler.

[Foliensätze 23 und 24]

Quellen

- Enzyklopädie Wikipedia
- http://www.software-saxony.de
- https://www.tu-braunschweig.de/Medien-DB/isf/sse/vl5_entwurf.pdf
- http://www-sst.informatik.tu-cottbus.de/~rust/ws1999-2000/tools/Paulus.pdf
- http://www-st.inf.tu-dresden.de/Lehre/WS00-01/st2/vorlesung/st2k4a-extra.pdf
- *Understanding the High – Performance – Computer Community / Paper*