Softwareentwicklung am CliSAP Arbeitsbereich Meereisfernerkundung

— Erkenntnisse eines qualitativen Interviews —

Arbeitsbereich Wissenschaftliches Rechnen
Fachbereich Informatik
Fakultät für Mathematik, Informatik und Naturwissenschaften
Universität Hamburg

Vorgelegt von:	Hagen Peukert
E-Mail-Adresse:	2peukert@informatik.uni-hamburg.de
Matrikelnummer:	6417586
Studiengang:	Mensch-Computer Interaktion (MCI)
Erstgutachter:	Hermann Lenhart
Zweitgutachter:	Julian Kunkel
Betreuer:	Hermann Lenhart

Hamburg, den 31.08.2015

Abstract

Der vorliegende Bericht fasst die Ergebnisse eines qualitativen Interviews zusammen. Ziel des Interviews war, an einem konkreten Beispiel herauszufinden, wie Softwareentwicklung in den Naturwissenschaften betrieben wird, d.h. ob in der alltäglichen Arbeit der WissenschaftlerInnen die Methodik aus der Informatik übernommen wird oder ob sich andere *best practices* herausbilden, die für die Modellierung und Simulation im Wissenschaftsbetrieb besser geeignet sind. Das Interview beschränkte sich auf die Exploration von vier Themenblöcken: Dokumentation, Versionierung, Umgang mit den Daten und den Prozess der Softwareerstellung.

Inhaltsverzeichnis

1. Einleitung

Das Einführungskapitel beschreibt die Ziele, Motivation, Methodik und den Aufbau dieser Arbeit.

Der Einsatz von Software ist in allen wissenschaftlichen Disziplinen heutzutage selbstverständlich. Dabei nehmen die Naturwissenschaften aufgrund ihrer stärkeren formalen Ausrichtung und experimentellen Methodik seit jeher eine Vorreiterrolle ein. Formale Modellierungen lassen sich oft in die Sprache des Computers übersetzen und so mittels Simulationen neue Erkenntnisse gewinnen, die nunmehr teure Experimente ergänzen oder manchmal auch zur Gänze ersetzen.

Dabei führt die inhärente Spezifizierung wissenschaftlicher Erkenntnisse nicht nur zur Diversifizierung der Wissenschaften an sich, sondern auch zur Diversifizierung der Erkenntnissgewinne und Methodik. Beim Einsatz von Softwareprogrammen bedeutet dies, dass eine Standardisierung für hoch spezielle Fragestellungen kaum mehr möglich; zudem nicht sinnvoll sein kann. Als Konsequenz daraus ergibt sich, dass die Erstellung von Software individualisiert wird, d.h. dem Wissenschaftler selbst obliegt nun die Verantwortung die auf seine spezifischen Bedürfnisse zugeschnittene Software eigenständig zu erstellen.

Das Ziel dieser explorativen Untersuchung ist es, den Prozess der Softwareentwicklung in einem naturwissenschaftlichen Arbeitsbereich näher zu beschreiben. Ich möchte erfahren, ob in diesem Arbeitsbereich andere, für den spezifischen Wissenschaftsbetrieb passendere, Softwareentwicklungstechniken Anwendung finden. Dabei soll nicht in Abrede gestellt werden, dass die Expertise für die Untersuchung und Erforschung geeigneter Softwareentwicklungsmethoden in der Informatik liegt und grundsätzlich nützlich ist. Teilgebiete der Informatik wie die Softwaretechnik oder die Softwaresystementwicklung haben bekanntermaßen ihren Untersuchungsgegenstand und ihr Erkenntnisinteresse in der korrekten Erstellung von Software definiert. Sie fokussieren sich auf die Erarbeitung einer möglichst allgemein anwendbaren Methodik zur Softwareentwicklung. Allerdings besteht auch die Möglichkeit, dass sich der spezielle Arbeitskontext, wie es in der alltäglichen wissenschaftlichen Arbeit der Fall sein könnte, von einer allgemeingültigen Methode der Softwareentwicklung zu weit abweicht, was dazu führen kann, dass die ideale Methode, wie sie in der Informatik vorgeschlagen wird, zu Ineffizienz in der Entwicklung in den Naturwissenschaften führen kann.

Um eine erste Idee von den tatsächlichen Softwareentwicklungsmethoden im naturwissenschaftlichen Bereich zu bekommen, bietet sich ein nicht standardisiertes, qualitatives Interview an. Es erlaubt, in einer gesprächnlichen Situation, die interessierendes Sachverhalte zu erfragen und gegebenfalls die Reihenfolge und Formulierung der Fragen der Gesprächssituation anzupassen.

Einzelne Bereiche, die sich im Gespräch als trivial herausstellen, können so schnell übergangen werden; andere, die sich entgegen der ursprünglichen Planung als wichtiger erweisen, können elaboriert werden. Die Entwicklung des Interviewleitfadens geschah mittels einer am Fachgebiet erstellten Schablone, welche im Allgemeinen die interessierenden Sachverhalte enthielt und auf die Bedürfnisse der Befragung am Arbeitsbereich Meereisfernerkundung angepasst werden musste.

Die zu untersuchenden Sachverhalte geben zugleich den Aufbau der vorliegenden Arbeit wider. Sie umfassen Fragen nach dem Umgang mit den Projektdaten und über das Versionierungsmanagement sowie der Dokumentation. Das Hauptaugenmerk lag auf dem Softwareentwicklungsprozess mit den klassischen Phasen der Architektur, des Designs, der Implementierung, des Testens und der Wartung. Um diese Kapitel in das Gesamtbild der Softwareentwicklung in den Naturwissenschaften besser verstehen und einordnen zu können, wird ein Theoriekapitel vorangestellt. Das Theoriekapitel dient der Definition der Voraussetzungen, auf welche die Ergebnisse des Interviews bezogen und eventuell Schlussfolgerungen angestellt werden können. Somit wird die Theorie nicht nur in die Problemstellung einführen, welche mit der Software gelöst werden soll, sondern auch kurz eine Theorie der Softwareentwicklung, wie sie im V-Modell vorgeschlagen wird, vorstellen.

2. Theoretischer Hintergrund

Das Kapitel zum theoretischen Hintergrund soll zum einen den idealtypischen Verlauf der Softwareentwicklung, wie er in der Standardliteratur zur Softwaretechnik dargestellt ist, aufzeigen. Zum anderen werden in Kürze die Forschungsfragen und Hypothesen des Arbeitsbereichs Meereisfernerkundung, die modelltheoretischen Annahmen und das Arbeitsfeld des Interviewpartners vorgestellt.

2.1. Der Softwareentwicklungsprozess in der Informatik

Nach Balzert [Bal98, 36] ist Softwareentwicklung definiert als die

> ... zielorientierte Bereitstellung und systematische Verwendung von Prinzipien, Methoden und Werkzeugen für die arbeitsteilige, ingenieurmäßige Entwicklung und Anwendung von umfangreichen Softwaresystemen.

Aus dem Blickwinkel der Softwaretechnik erscheint die Definition sicherlich für den allgemeinen Fall der Softwareentwicklung anwendbar. Allerdings deckt die Definition sicher nicht kleinere Projekte ab, in denen auch Software programmiert wird. Die Schlüsselwörter *arbeitsteilig*, *ingenieurmäßig* und *umfangreich* müssen bei Anwendbarkeit auf kleine Projekte hinterfragt werden, weil es sich ja auch bei Kleinprojekten um die Entwicklung von Software handeln muss, wenn man am Ende das fertige Produkt sieht.

Akzeptieren wir die Arbeitsdefinition mit den eben beschriebenen Einschränkungen bei kleinen Softwareprojekten, stellt sich als nächstes die Frage, wie und welche *Prinzipien* und *Methoden* systematisch verwendet werden. Auch dazu findet sich in der klassischen Literatur der Softwaretechnik eine Antwort, die im Grunde auf das Verarbeitungsmodell (V-Modell) hinausläuft (siehe Abb. 2.1). Mehr oder weniger direkt folgt aus der Definition der Softwareerstellung die Aufteilung des Softwareentwicklungsprozesses in sinnvoll abzugrenzende Bereiche, den Phasen oder Ebenen der Softwareentwicklung. Dazu zählen die Anforderungsdefinition gefolgt von dem Systemdesign und der technischen Konzeption über die Implementierung hin zu den einzelnen Tests und schließlich der Softwareeinführung.

Nach Abschluss einer jeden Phase wird überprüft, ob die Spezifikationen der Vorgängerphase noch vollständig erfüllt sind. In Abb. 2.1 werden diese *Feedback*-Schleifen dem Begriff der Verfikation zugeschrieben. Die Validation bezeichnet hier das Ergebnis des Tests, der die korrekte Implementierung der Spezifikation von genau dieser Phase sicherstellen soll. Scheitert ein solcher Test, beginnt der Entwicklungsprozess just auf

dieser Entwicklungsebene von Neuem. Auf diese Weise entstehen zwei ineinandergeschachtelte Zyklen: zum einen der Zyklus der Verifikation und zum anderen der Zyklus der Validation. Erst wenn alle Tests erfolgreich abgeschlossen sind, kann der Einsatz der Software beginnen. Im Normalfall wird eine weitere Phase – die Wartung und Pflege der Software – mit zum Entwicklungsprozess gezählt. Dieser kann dazu führen, dass der Gesamtprozess der Softwareentwicklung selbst als Zyklus begriffen werden muss, da sich aufgrund von anstehenden Veränderungen des Kunden oder technologischer Art, ein anderes Anforderungsprofil ergibt und so der Entwicklungsprozess von neuen beginnt.

Die soeben vorgestellten Phasen der Softwareentwicklung geben in weiten Teilen die Struktur des Interviews vor, da anfangs aufgrund der Beschreibung in der Standardliteratur angenommen werden durfte, dass sich die Softwareentwicklungsprozesse auf das V-Modell vollständig abbilden lassen.

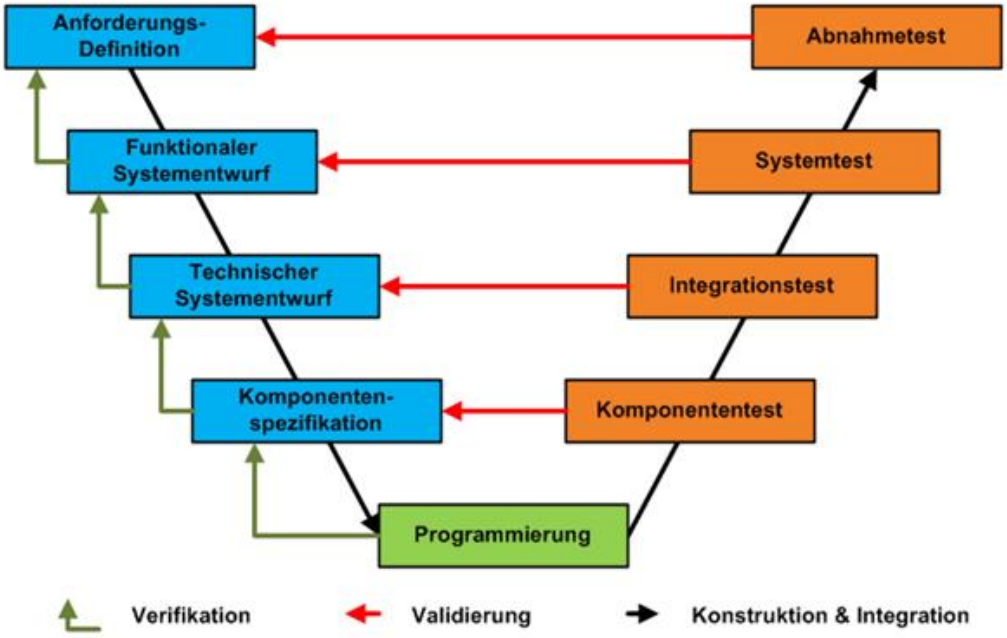

Abbildung 2.1.: Das V-Modell der Softwareentwicklung [Bal98, 99]

2.2. Meereisfernerkundung am CliSAP mit SMOS Satellitendaten

2.2.1. Modellierung und Simulation

Formuliert man die Forschungsfrage des Arbeitsbereichs Meereisfernerkundung des CliSAP kurz und allgemeingültig, dann geht es in erster Linie um die Vermessung der Eisdecke

an den beiden Polkappen. Dazu gehört zum Einen die Dicke und zum Anderen die Oberfläche oder die Größe, sodass man von der Ermittlung der Veränderung eines Volumens im Zeitverlauf sprechen kann. Die Ermittlung der Veränderung des Meereisvolumens ist ein komplexes Verfahren, das man mit unterschiedlichen Methodiken in unterschiedlicher Qualität erreichen kann. Eine sehr kostengünstige Möglichkeit besteht in der Nutzung des SMOS-Satelliten (Soil Moisture and Ocean Salinity). Dieser Satellit kann mittels seiner Sensoren sehr feine Unterschiede in Radiowellen messen. Einige Störfaktoren, die bei der Messung von Lichtreflexion (Bevölkung, Strahlung, andere Wetterbedingungen) auftreten, können zwar auf Basis vorhandener Modelle herausgerechnet werden, allerdings leidet durch die damit einhergehende höhere Variabilität der Meßdaten die Genauigkeit entweder der Modellvorhersage oder der Beschreibungsadäquatheit des Modells. Deshalb geht es bei der Verwendung der Satellitendaten nicht nur um die Verfeinerung, Erweiterung oder Falsifizierung von Modellen und den modelltheoretischen Annahmen, sondern zunächst auch um die Frage, wie sich aus den Satellitendaten die gleichen Messdaten ermitteln lassen, die bereits aus experimentellen Versuchen (z.B. Eisbohrkern) vorhanden sind und welche Validität und Repräsentativität die Satellitendaten haben.

Das Hauptproblem ist dabei dreischichtig: erstens nicht alle für die bestehenden Modelle notwendigen Daten können vom Satelliten erfasst und müssen ergänzt werden. Zweitens sind Informationen vorhanden (Rauschen), die nicht benötigt werden. Drittens lassen sich die benötigten Informationen nicht einfach von den nicht benötigten Informationen trennen.

Es handelt sich also um ein klassisches Problem der Signalverarbeitung. Das erste Problem lässt sich durch die Berücksichtigung von vorhandenen Modellen lösen. Zum Beispiel spielen physikalische Größen, wie der Einfallswinkel des Lichts auf die Eisdecke, eine wichtige Rolle. Aus Strahlungs- und Transfermodellen können wichtige Einflussgrößen eingerechnet werden. So ist die Emissivität des Eises eine Funktion der Temperatur und seines Alters; zur Wassernähe nimmt die Temperatur ab, ebenso mit zunehmender Höhe des Eises über dem Meeresspiegel (Berge und Täler).

Das zweite und dritte Problem gehören gewissermaßen zusammen. Hier besteht die Lösung durch geeignete Filter möglichst nur die interessierenden Information aus dem Signal zu extrahieren. Die Konstruktion dieser Filter erfolgt durch eine Art des Experimentierens. Es werden schrittweise Informationen aus dem vorliegenden Signal entfernt und dann verglichen, ob das gefilterte Signal genauere Modellvorhersagen macht. Die idealen Vorhersagen sind die, die exakt mit den Modellvorhersagen übereinstimmen, die auf Daten eines Experiments oder aus präzisen Messungen vor Ort gewonnen wurden.

Das Projekt, das im Rahmen des hier geführten Interviews, beschrieben wird, hat das Ziel die Auswirkungen der *Rauhigkeit* des Eises auf die Vorhersagekraft der bestehenden Modelle zu ermitteln. Es wird am Forschungsgebiet B1 des CliSAP, Arktische und Permafrost Gebiete, unter Leitung von Professor Lars Kaleschke von Maciej Miernecki als Promotionsarbeit durchgeführt.

Dabei werden folgende modelltheoretische Annahmen gemacht. Die Eisfläche wird in ein Raster von gleich großen Quadraten (50 x 50 Kilometer) unterteilt, für die man annimmt, dass das Verhalten innerhalb eines jeden Quadrats homogen ist, das bedeutet hier insbesondere, dass die Dicke des Eises als konstant angenommen wurd. Weiterhin

wird unterstellt, dass sich die physikalischen Modelle widerspruchsfrei anwenden lassen und letztlich die Wirklichkeit sich auch nur auf diese Modelle beschränkt. Es werden also keine weiteren Einflussvariablen berücksichtigt, auch wenn aus anderen Modellen Einflüsse bereits bestätigt wurden. Man betrachtet demnach nur einen sehr kleinen Theorieausschnitt.

2.2.2. Projektmanagement

Am CliSAP haben alle Mitarbeiter Zugang zu einer Projektmanagementsoftware, die eigens für die Forschung an den unterschiedlichen Arbeitsbereichen entwickelt wurde. Über diese Software kann die Dokumentation und die Versionierung der eigenen Programme vollzogen werden. Ebenso übernimmt die Software das Management der validierten Daten, die mit dem SMOS Satelliten empfangen und dann mit dem eigenem Programm gefiltert wurden.

Über eine Webschnittstelle haben alle Mitarbeiter Zugriff auf die Programme, die andere Mitarbeiter geschrieben haben. Diese Programme können als Muster oder Schablone für eine Eigenentwicklung verwendet werden. Ebenfalls stehen validierte Daten zur Verfügung, die man gegebenfalls für eine ähnliche Fragestellung wiederverwenden kann. Für jedes Programm ist eine nicht standardisierte Dokumentation vorhanden.

3. Dokumentation, Versionierung und Umgang mit den Daten

In diesem Kapitel wird beschrieben, ob und wie das Forschungsprojekt zur Untersuchung der Rauhigkeit des Eises dokumentiert und versioniert wird. Außerdem wird ein kurzer Einblick in den Umgang mit den im Projekt anfallenden Daten gegeben.

3.1. Dokumentation

Die Dokumentation erfolgt überwiegend in der Projektmanagementsoftware des CliSAP. Die Software generiert diese Dokumentation auch auf die Wiki-Seiten des CliSAP-Webauftritts. Die Dokumentation kann ebenso über die Wiki der Webseite editiert werden. Darin werden insbesondere die Durchschnittsvariablen festgehalten, da diese für andere Klimaforscher am wichtigsten sind. Man kann festhalten, dass es bei dieser Form der Dokumentation des Programms eher darum geht, was das Programm im Allgemeinen tut, d.h. welche Variablen Verwendung finden und unter welchen Modellannahmen diese verarbeitet werden. Es wird weniger beschrieben, wie der Filter die Daten extrahiert oder wie die genaue Implementierung des Modells aussieht. Diese Dokumentationen erfolgen zudem nicht fortlaufend, sondern erst nach der vollständigen Fertigstellung eines Programms und werden zudem zugleich als Teil der Qualifikationsschrift verwendet.

Eine im informatischen Sinne systematische Dokumentation nach Paketen, Klassen und Methoden ist nicht vorhanden. Vereinzelt befinden sich in den Quellcodeskripten kurze Kommentarzeilen an Routinen oder einzelnen Datenstrukturen, die eine nicht sofort einsichtige Implementierung erklären sollen. In einem Ausnahmefall konnte ich ein Skript sehen, dass über jeder Funktion einen Kommentar enthielt.

Viel Wert wird auf die Einhaltung von Namenskonventionen gelegt. Diese sind durchgängig vorhanden und werden ausnahmslos befolgt. Der Grundsätzliche Aufbau und die Struktur des Programms werden in Form einer Metadatenbeschreibung in einem sogenannten *datafile* zur Verfügung gestellt. Diesen kann man sich auch über die CliSAP Webseite herunterladen und einsehen. Weitere Informationen zu Fehlermeldungen oder Ausgaben des Programms sind in einem *Log-File* vorhanden, dieser wird auch mit im Dateiordner abgelegt, ist aber nicht über die Webseite einsehbar.

3.2. Versionierung

Ein umfangreiches Versionierungsmanagement, wie es etwa mit *Git* umgesetzt werden kann, ist im vorliegenden Projekt nicht vorgesehen. Es existiert auch keine Versionierung während des Entwicklungsprozesses der eigenen Programme. Diese werden kontinuierlich in einer Datei weitergeschrieben. Der Interviewpartner hat jedoch angegeben, dass er sehr wichtige Meilensteine im Fortschreiben seines Programms als eine Extraversion lokal oder in seinem Anwenderverzeichnis auf dem Cluster ablegt. Zu dieser Version hat allerdings niemand Zugriff und wird nach Fertigstellung des Programms wieder gelöscht. Die Teilversionen stehen also keinen weiteren Mitarbeiter zur Verfügung.

Aus den lokalen Sicherheitskopien kann man weder den Zeitpunkt der Erstellung relativ zu anderen Versionen oder zu einzelnen Programmabschnitten erschließen. Man kann deshalb nicht von einer echten Versionierung oder Versionskontrolle sprechen. Nach Fertigstellung des Programms wird es in das Repositorium der Projektmanagementsoftware geladen und kann ab diesem Zeitpunkt von allen anderen Mitarbeitern verwendet oder auch weiterentwickelt werden. Im Falle von Abänderungen des Programms anderer Mitarbeiter wird das Programm unter anderem Namen in das Repositorium geladen. In der Metadatenbeschreibung des *datafile* ist dann noch ersichtlich aus welchen anderen Vorgängerversionen das Programm erstellt wurde. Im Repositorium befindet sich auch eine Toolbox von einzelnen Codeschnipseln, die als Vorlagen verwendet werden können.

3.3. Umgang mit den Daten

Der Zugriff auf die Satelittendaten ist strikt hierarchisch in einer Verzeichnisstruktur geregelt. Dabei nimmt sowohl die Informationsdichte als auch die Rechtevergabe an die Forscher mit zunehmender Verzeichnistiefe ab. In der obersten Ebene haben nur wenige Forscher Nutzungsrechte; die Daten haben hier den höchsten Informationsgehalt, wobei man anmerken muss, dass an dieser Stelle bereits eine Art der Vorverarbeitung der Satellitendaten stattgefunden hat. Die Rohdaten werden nicht den Verzeichnisses auf dem Cluster abgelegt und werden vermutlich gelöscht. Auf der Hierarchieebene, auf der sich die Daten für das hiesige Projekt befinden, können die Daten nach dem Ort (hier das Flächenquadrat von 50 x 50 Kilometern) oder den physikalischen Einheiten eingesehen werden. Die Rechtevergabe auf eine der Hierachieebenen richtet sich nach der zu bearbeitenden Forschungsfrage, also nach dem konkreten Bedarf am Informationsgehalt der Daten.

Die Daten, die in den Teilprojekten durch die Verarbeitung der selbst geschriebenen Programme entstehen, werden nach entsprechender Validierung veröffentlicht. Sie sind über die Website des CliSAP (http://www.icdc.zmaw.de) oft auch öffentlich zugänglich. Da gängige Standardformate verwendet werden, ist die Sichtung der Daten ohne Weiteres möglich. Der *headerfile* zur Metadatenbeschreibung liegt im xml-Format vor. Formate für die Darstellung von Bildern oder geographischer Koordinaten sind *tiff* beziehungsweise *qGis*. Weiterhin werden *hdf5*, *netcdf* und *h5* verwendet. Die aufbereiteten Daten liegen zudem binär oder als *ASCII* vor.

Unter Verarbeitung der Satellitendaten ist hier zu verstehen, dass die Satellitendaten aus der freigegebenen Hierarchieebene weiter bearbeitet werden. Sie können in ihrer Anordnung verändert oder mit entsprechenden Filtern komprimiert oder mit anderen Daten (Variablen) ergänzt werden, d.h. sie werden für eine spezifische Forschungshypothese aufbereitet.

4. Softwareentwicklung

Das Kapitel zur Softwareentwicklung beschreibt, wie am Arbeitsbereich für Meereisfernerkundung beim Erstellen der Programme zur Umsetzung der Forschungshypothesen vorgegangen wird. Das bedeutet, dass das Vorgehen kurz zusammenhängend beschrieben wird und dass wichtige Phasen des Entwicklungsprozesses an einem kurzen Beispiel beschrieben werden.

4.1. Entwicklungsphasen

Nach dem V-Modell werden vier Phasen, die dazugehörigen Tests und die Implementierung unterschieden. Die erste Phase der Anforderungsdefinition entspricht dem *use case* und kann im vorliegendem Beispiel mit der der spezifischen Forschungshypothese, wie sie oben beschrieben wurde, gleich gesetzt werden. Aus der Forschungshypothese leiten sich direkt alle Anforderungen an die Software ab, d.h. die Variablen und das zugrunde liegende Modell.

Die Phasen des funktionalen und technischen Systementwurfs lassen sich im klassischen Kreislauf der Softwareentwicklung unter *Design und Architektur* fassen. Während des Interviews wurde sehr schnell offensichtlich, dass diese Ebenen nicht explizit in die Planung des Projekts aufgenommen wurden. Dennoch könnte man im Nachhinein das Entwurfsmuster des Programms erkennen und die Gesamtarchitektur ließe sich ebenso ex post definieren. Der Unterschied besteht also in erster Linie darin, dass weder die Architektur der Software noch die einzelnen Entwurfsmuster wissentlich, also bewusst im Vorhinein, ausgesucht wurden und in Abwägung zu alternativen Möglichkeiten durchdacht werden konnten. Da das vorliegende Projekt jedoch nicht den Umfang eines großen Softwareprojekts hat, ist es ohnehin fraglich, ob der funktionale Systementwurf eine effiziente Lösung auch für erfahrene Programmierer wäre. Schließlich bietet die Überschaubarkeit der zu implentierenden Methoden keinen Anlass, das Projekt unnötig zu vergrößern.

Die Implementierung erfolgt in Python und der Doktorand arbeitet direkt auf dem Cluster. Methoden und Klassen sind nach den Grundsätzen der Objektorientierung aufgeteilt. Seine Vorgehensweise bei der Entwicklung erfolgt dem Prinzip von *einfach zu komplex*. Es wird zuerst eine einfache Funktion mit grundlegenster Funktionalität programmiert, die sofort auf Lauffähigkeit getestet wird. Diese Funktion wird dann schrittweise erweitert oder es werden weitere Funktionen hinzugefügt. Nach jeder Erweiterungsstufe wird das Programm wieder auf Lauffähigkeit und auf Plausibilität der Ausgaben getestet. Da das Promotionsprojekt auf eine Person ausgelegt ist, entfällt die Organisation im Team.

Die Tests, die nach jeder Änderung einer Funktion oder Funktionalität, durchgeführt werden, beruhen auf dem Prinzip des *Versuch und Irrtum*. Ist die Lauffähigkeit der Änderung gewährleistet, werden die Ergebnisse des eigenen Programms überprüft. Es gilt: Produziert das Programm valide Ergebnisse ist es korrekt. Dies kann auf zwei verschiedene Weisen erfolgen. Wenn aus experimentellen Studien bereits valide Ergebnisse und vergleichbare Ausgangsdaten vorliegen, wird verglichen, ob das Programm die gleichen Ergebnisse liefert. Ist dies der Fall, geht der Entwickler davon aus, dass das Programm korrekt funktioniert.

Sollten jedoch keine Ergebnisse oder ähnliche Messdaten vorliegen, gibt es nur noch die Möglichkeit, die Korrektheit des Programms aufgrund der Plausibilität der Ergebnisse zu *ermitteln*. Dabei wird verglichen, ob die Ergebnisse mit physikalischen Kennziffern im Widerspruch stehen oder enorme Abweichungen von den Vorhersagen hervorbringen. Diese Art des Testens kann dadurch gerechtfertigt werden, dass die Funktionen immer eine Implementierung des Modells beinhalten und keine zusätzlichen Annahmen gemacht werden dürfen. Natürlich darf man kritisch anmerken, dass die Modellimplementierung versteckte Annahmen enthalten kann, die man nicht bemerkt.

Eine Wartung des Programms ist vom Entwickler selbst nicht vorgesehen. Oftmals ist es aber der Fall, dass das Programm in einem Folgeprojekt oder einem Projekt mit ähnlicher Fragestellung genutzt wird, sodass durch die Übernahme eine Art der Wartung auf ein aktuelles Betriebssystem oder eine neuere Version der Programmierungsprache vollzogen wird. Auch hier gilt abermals, dass diese Phase implizit abläuft. Im vorliegenden Fall werden zahlreiche Probleme erwartet, wenn das Programm auf Python 3.0 oder höher umgestellt werden müsste. Es erscheint dann fraglich, ob sich Anpassungen überhaupt lohnen oder ob eine neue Entwicklung dann nicht profitabler wäre.

4.2. Beispiel: Salzprofil

Betrachten wir zur besseren Veranschaulichung der Vorgehensweise die Entwicklung eines Teils des Programms, nämlich zur Ermittlung des Salzprofils im Meereis. Die grundsätzliche Frage lautet hier, ob das aus der Theorie bekannte Modell zur Salzprofilberechnung die Vorhersagen mit den Satellitendaten liefert, die zu erwarten, also üblich, sind. Mögliche Fehler des Programms ergeben sich just aus diesem Prozess. Gibt es Ergebnisse, die nicht den möglichen Werten des Salzprofils im Meerwasser entsprechen, dann weiss man, dass das Programm falsch ist. Übersteigt der Salzgehalt des Wassers die Werte, die in Arktis oder Anarktis jemals gemessen wurden, liegt höchstwahrscheinlich ein Implementierungsfehler des Modells vor. Wie oben bereits beschrieben wurde, ergibt sich aus der Programmentwicklung vom Einfachem zum Komplexem auch die Komplexität der Tests. Im vorliegenden Fall wird zuerst das Salzprofil für dünnes Eis bestimmt, erst dann für dickere Eisschichten, da mit zunehmender Eisdicke die Anzahl der relevanten Einflussgrößen (Variablen), die auch zwingend berücksichtigt werden müssen, zunehmen. Dabei sind alle Inputsatellitendaten auf einer Hierarchiebene zwar vorhanden, werden aber bei den einfachen Modellen nur zum kleinen Teil genutzt. Nach jeder Erhöhung der Eisdicke (Änderung bzw. Hinzunahme von Inputsatellitendaten und

von Modellvariablen), wird überprüft, ob die Aussagen noch realistisch sind oder ob es *Brüche* (sogen. *cuts*), Widersprüche, in den homogenen Feldern eines Modells gibt.

5. Zusammenfassung und Fazit

Das Schlusskapitel fasst die wichtigsten Erkenntnisse des qualitativen Interviews zusammen und präsentiert zwei wesentliche Schlussfolgerungen daraus als ein Fazit.

Zusammenfassend lässt sich hinsichtlich der vier untersuchten Bereiche – Umgang mit Forschungsdaten, Dokumentation, Versionierung und Softwareentwicklungsprozess – folgendes festhalten. Die Satellitendaten werden in verschiedener Detailtiefe archiviert. Auf dem Cluster sind die mit selbsterstellten Programmen erzeugten Outputdaten in einer Verzeichnisstruktur abgelegt und nach Validierung auf der Projektwebsite zugänglich.

Dort und im Wiki sind auch Teile der Dokumentation vorhanden. Eine Metadatenbeschreibung liegt in einem *datafile* vor. Weitere Dokumentationen befinden sich direkt im Quellcode.

Eine echte Versionierung wird nicht durchgeführt. Das Einstellen der Programme in das vorhandene Softwarerepositorium der Projektmanagementsoftware erfolgt nach vollständiger Fertigstellung. Die Versionierung ist streng genommen nicht Teil des Entwicklungsprozesses.

Im Interview wurde mehrfach betont, dass der Softwareentwicklungsprozess *ergebnisorientiert* abläuft. Einige der klassischen Phasen in der Softwareentwicklung laufen unterbewusst ab. Dies betrifft definitiv die Phase der Architektur und des Entwurfs. Das Testen beruht auf Plausibilitätskriterien. Eine kontinuierliche Verbesserung und systematische Wartung findet nicht statt.

Aus diesen Ergebnissen ergeben sich zwei wesentliche Schlussfolgerungen als Fazit. Zunächst spielen Fragen, die in der Informatik im Vordergrund stehen, etwa die Laufzeit, Effizienz oder Lesbarkeit des Programms, eine untergeordnete Rolle. Die Laufzeit des Programms dauert derzeit ungefähr 13 Stunden. So ist zwar im vorliegenden Fall eine Parallelisierung angedacht, die Planung und Umsetzung bleiben aber unspezifisch und sollen nur dann durchgeführt werden, wenn am Ende des Promotionsprojekts noch Zeit übrig bleiben sollte.

Als zweites Fazit muss man anmerken, dass der hier beschriebene Softwareentwicklungsprozess den Grundsätzen und Prinzipien der *Agilen Softwareentwicklung* sehr viel ähnlicher ist als das V-Modell, das als Theorierahmen am Anfang des Projekts gewählt wurde. Im hiesigen Projekt wie auch in der Agilen Softwareentwicklung fallen kurze Entwicklungszyklen an, in denen schnelle Änderungen ohne großen Aufwand möglich sind. Alle Teillösungen werden in der Agilen Programmierung wie auch in dem hier beschriebenen Promotionsprojekt sofort durchgeführt. Auf diese Weise entstehen viele kleine und ähnliche Entwicklungszyklen. Andere Prinzipien, wie zum Beispiel die Grundsätze zur Entwicklung im Team, sind nicht zutreffend. Dennoch ergibt sich aus diesen Erkenntnissen in der Gesamtbetrachtung, eine neue Hypothese, der man in einem

gesonderten Forschungsarbeit nachgehen müsste. Die Hypothese könnte lauten: Der SE-Prozess in den (Natur)Wissenschaften bildet sich selbstorganisatorisch heraus und strebt nach einem effizienzoptimalen Punkt der Softwareentwicklung, der den Prinzipien der Agilen Softwareentwicklung stark ähnelt.

Literaturverzeichnis

[Bal98] Helmut Balzert. *Lehrbuch der Software-Technik*. 1998.

Abbildungsverzeichnis

Anhänge

A. Interviewpartner

Institut:	Institut für Meereskunde, Zentrum für Marine und Atmosphärische Wissenschaften (ZMAW), Integrated Climate System Analysis and Prediction (CliSAP)
Interviewpartner:	Maciej Miernecki Bundesstraße 53, R019 maciej.miernecki@zmaw.de
Position:	Doktorand
Lehrstuhlinhaber:	Prof. Dr. Lars Kaleschke

B. Interviewleitfaden

Dieser Interviewleitfaden wurde mit Hilfe des im Seminar vorgeschlagenen Leitfadens erstellt.

1. Kurze Vorstellung und Grund des Interviews
   a) Sensibilisiert für Problematik aus eigener Erfahrung
   b) Hintergrund des Interviewpartners
   c) SE-Prozess verstehen, unterschiedliche Erkenntnisgewinne
2. Kennenlernen des Untersuchungsgegenstandes
   a) Was untersuchen Sie i.a. (Wie schnell schmelzen die Polkappen)? wie ist ihre F-Hypothese (genauer)?
   b) Welche Theorie steht dahinter? Welche Besonderheiten gibt es?
   c) Wie sehen die modelltheoretischen Annahmen aus?
3. Software- und Hardwareeinsatz
   a) Welche Software setzen sie ein (DB, numerisches Modell)
   b) Entwickeln Sie selbst? (oder Nutzer/Anwender)
   c) In welcher Programmiersprache entwickeln Sie?
   d) Welche Tools nutzen Sie für die Entwicklung?
   e) Greifen Sie auf vorhandene Softwarebibliotheken zurück?
   f) Werden Neuerungen fortlaufend gespeichert, gibt es eine Versionierung (z.B. Git)
   g) Wie stellen Sie sicher, dass „Nachfolger“ die Software schnell verstehen (geg. die Befristung der Stellen)?
   h) Welche Hardware benutzen Sie (Cluster, Mainframe, lokal)?
4. Daten
   a) Wie werden die Daten archiviert (Eingabe sowie Ausgabe)?
   b) Wie sieht der Output des Programms aus: werden Standardformate verwendet (Nachhaltigkeit)
   c) Wie gehen Sie grundsätzlich mit den erzeugten (Ausgabe)Daten um? (Öffentlich, unter Verschluss)

d) Wie müssen die Eingabedaten beschaffen sein? Müssen sie speziell für die Software aufbereitet werden?

5. Softwareentwicklung

a) Entwickeln Sie Software im Team? Wie werden die Aufgaben verteilt? In welche logischen Bereiche wird unterteilt (gibt das Modell Bereiche vor oder der organisatorische Entwicklungsprozess?)

b) Wie sieht die Kommunikation in der Entwicklergruppe aus?

c) Welche Softwareentwicklungsphasen werden typischerweise durchlaufen (Erstellung eines Use Case sicher nicht? Design, Implementierung, Testen, Warten)

d) Wird die Software auch in der Implementierungsphase dokumentiert? Wie ist die Vorgehensweise?

e) Wie wird getestet?

f) Wie werden Änderungen am Programm umgesetzt?

g) Wie wird die Software gewartet (z.B. bei Updates)?

Erklärung

Ich versichere, dass ich die Arbeit selbstständig verfasst und keine anderen, als die angegebenen Hilfsmittel – insbesondere keine im Quellenverzeichnis nicht benannten Internetquellen – benutzt habe, die Arbeit vorher nicht in einem anderen Prüfungsverfahren eingereicht habe und die eingereichte schriftliche Fassung der auf dem elektronischen Speichermedium entspricht.

Optional: Ich bin mit der Einstellung der Bachelor-Arbeit in den Bestand der Bibliothek des Fachbereichs Informatik einverstanden.

Hamburg, den 31.08.2015 .....