Softwareentwicklung in der Wissenschaft

WaterGAP – A global hydrology and water use model

Seminararbeit

eingereicht bei
Prof. Dr. Thomas Ludwig
Arbeitsbereich Wissenschaftliches Rechnen
MIN Fakultät
Universität Hamburg

Betreuer:
Hermann Lenhart

von
Maral Chimed-Ochir

Inhaltsverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

BMBF:	Bundesministerium für Bildung und Forschung
CESR:	Center for Environmental Sytems Research
EEA:	European Environment Agency
EU:	Europäische Union
GIS:	Geoinformationssystem
GUI:	Graphic User Interface (Graphische Benutzeroberfläche)
UNEP:	United Nations Environment Programme
WaterGAP:	Water - Global Assessment and Prognosis

1 Zielsetzung

Das Ziel dieser Seminararbeit ist die Darstellung des Entwicklungsprozesses einer wissenschaftlichen Software in der Praxis anhand eines Beispiels.

Die Definition von wissenschaftlicher Software ist nach Kreyman et al eine Software mit großen rechenaufwendigen Komponenten, die physische Phänomene modelliert und Daten für die Entscheidungsfindung liefert [1]. Das Hauptziel der Entwicklung einer solchen Software ist es meist, Ergebnisse von Berechnungen eines wissenschaftlichen oder technischen Modells zu erhalten [2][3][4]. Wissenschaftliche Software hat meistens die Aufgabe, einen bestimmten naturwissenschaftlichen Prozess zu simulieren und bestimmte Fragen zu diesem Prozess zu beantworten.

Die Personen, die wissenschaftliche Software entwickeln, testen und benutzen, sind häufig Wissenschaftler und Ingenieure, von denen viele unter Segals Definition des „professional end-user-developer“ passen [5]. Diese sind nach Segal Personen, die in technischen Berufen und solchen, die große Sachkenntnis erfordern, ihre eigene Software entwickeln, um ihre persönlichen beruflichen Ziele voranzutreiben. Im Gegensatz zum normalen Endbenutzer-Entwickler sind sie keine professionellen Entwickler, sondern lediglich Profis in ihrem Fachgebiet. Deshalb haben sie häufig Probleme mit der Implementierung der Software [2].

Die Entwicklung einer wissenschaftlichen Software bringt einige Herausforderungen mit sich. Wissenschaftliche Software ist meistens rechenaufwändig. Sie wird meist in und für kleine Teams entwickelt und ist daher oft unverständlich oder nicht intuitiv verständlich für Außenstehende. Die Vorgehensweise der Entwicklung folgt oft nicht den Prinzipien traditioneller Entwicklungsmodelle, da Aspekte wie Benutzerfreundlichkeit und andere nicht im Vordergrund stehen. Auch die Testbarkeit stellen einige Herausforderungen dar, da z.B. das Ergebnis vorher noch nicht feststeht.

Um die Entwicklungsprozess einer wissenschaftlichen Software anschaulicher darzustellen zu können, wurde ein Interview mit einem Entwickler einer wissenschaftlichen Software durchgeführt. In den nächsten Kapiteln werden die wissenschaftlichen Fragestellungen über die Modellierung und Programmierung bis zur Analyse der Ergebnisse eines wissenschaftlichen Modells beschrieben.

1.1 WaterGAP allgemein

WaterGAP ist ein hydrologisches Modell zur Simulation der globalen Wassernutzung, Wasserverfügbarkeit und Wasserqualität. Das WaterGAP-Modell wurde in Center for Environmental Systems Research (CESR) an der Universität Kassel entwickelt und weiterentwickelt.

1.2 Interviewpartner

Der Interviewpartner ist ein wissenschaftlicher Mitarbeiter und Doktorand am CESR. Er ist beteiligt an der Weiterentwicklung des WaterGAP3-Modells (Version 3) und zurzeit mit der Entwicklung eines Bewässerung-Teilmoduls beschäftigt. Er besuchte einen dreiwöchcigen Blockkurs über die Programmiersprache C während seines Hydrologiestudiums in Freiburg. Da es einige Probleme in der Programmierung gibt, unterstützt die Chefprogrammiererin (Mathematikerin) am CESR ihn teilweise bei seiner Arbeit. Der Interviewpartner erhielt nie eine Ausbildung im Bereich des Software-Engineerings und hat keinerlei vorherige Erfahrung im IT-Bereich. Das Interview erfolgte durch E-Mail-Korrespondenz.

2 WaterGAP Einführung

WaterGAP ist eine kartografisch genaue Modellierung der globalen Wassernutzung, Wasserverfügbarkeit und Wasserqualität und die Simulation zukünftiger Entwicklungen. Dabei wird versucht, Antworten auf die Frage „Wann, wo und aus welchem Grund sind welche Wasserressourcen vorhanden bzw. genutzt?“ zu erhalten. Das Modell-Framework wird konstant weiterentwickelt, um klimatische, physiographische und sozioökomische Prozesse realitätsnah in einem Computermodell wiedergeben zu können. Das Modell kann den Einfluss von naturräumlichen und klimatischen Bedingungen (Niederschlag, Verdunstung usw.) auf die Wasserverfügbarkeit und die demographischen, ökonomischen und technologischen Veränderungen (z.B. Industriewasserentnahme) auf die Wassernutzung bis 2100 simulieren. Es deckt die gesamte Landmasse der Erde mit Ausnahme der Antarktis ab und hat eine räumliche Auflösung von 0.5°x0.5° (WaterGAP2) bzw. 5x5 Bogenminuten (WaterGAP3) mit Rasterzellgrößen von ca. 2000 – 3000 km² mit insgesamt 69896 Zellen bzw. 40 – 70 km² mit insgesamt 2,3 Mio. Zellen. Die zeitliche Auflösung hat eine Berechnungsschrittweite von einem Tag und die Ausgabe der Ergebniszeitreihe ist ein Monat (WaterGAP2 und WaterGAP3) bzw. ein Tag (WaterGAP3). WaterGAP3 ist das einzige räumlich so hoch aufgelöste Wassermodell.

Das WaterGAP-Modell ist nicht frei verfügbar, da es sehr komplex ist, keine GUI (graphische Benutzeroberfläche) und keine ausführliche Anleitung besitzt. Es ist aber auch keine kom-

merzielle Software. Die Software wird je nach Bedarf den Kooperationspartnern frei zur Verfügung gestellt und entsprechende Schulungen werden durchgeführt.

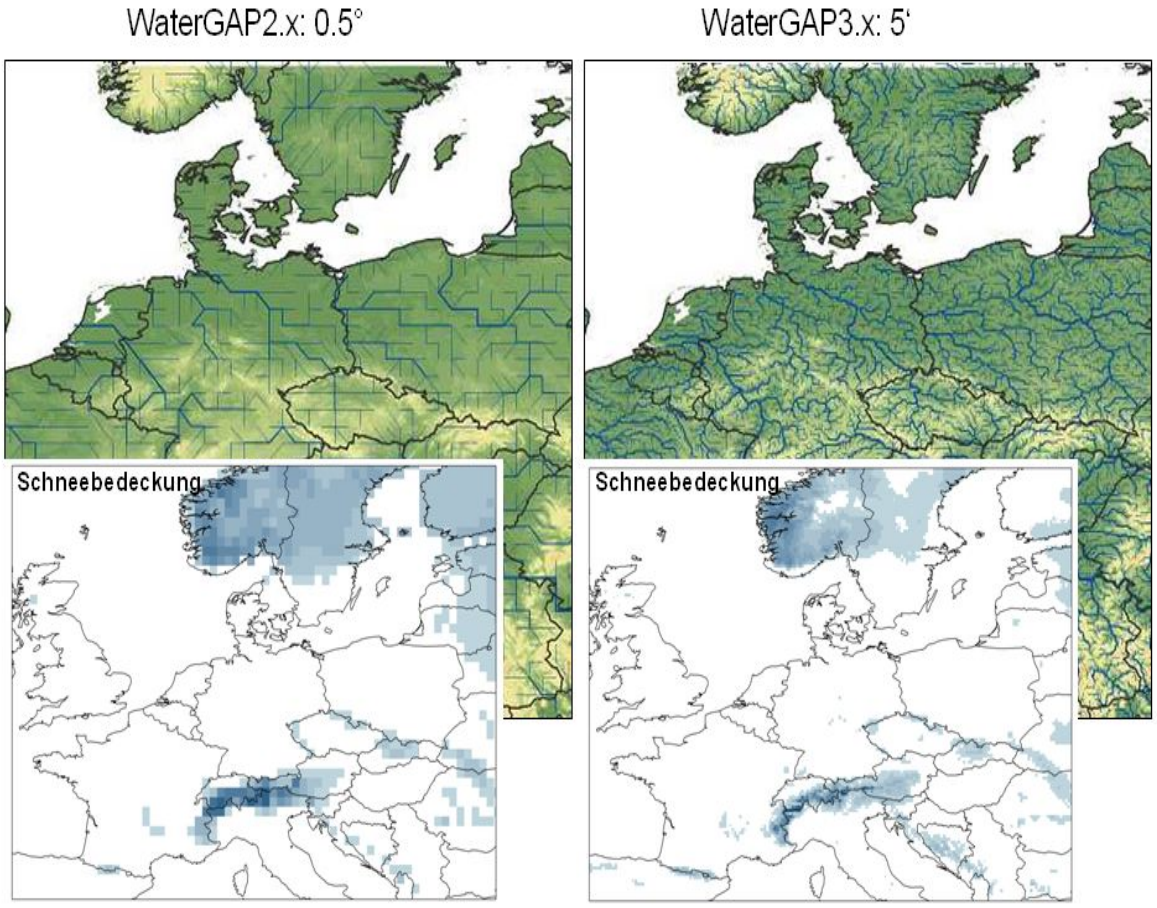

Abbildung 1: Visuelle Darstellung der Schneebedeckung mittels GIS für eine bestimmte Zeit (Quelle: Aus Der Beek, 2010)
links: Modellergebnisse von WaterGAP2, rechts: Modelergebnisse von WaterGAP3

2.1 Motivation

Es gibt bereits einige Wassermodelle im Einsatz. Allerdings decken diese nur ein bestimmtes örtliches Gebiet ab und deswegen konnten sie Fragen wie

- Welche Wassermengen stehen uns heute zur Verfügung und wie sind sie räumlich und zeitlich auf der Erde verteilt?
- Welche Rolle spielt der Einfluss des globalen Wandels (z.B. Klimaänderung, Bevölkerungsentwicklung) auf die Wasserverfügbarkeit und Wassernutzung?
- Wo tritt eine nicht nachhaltige Wassernutzung auf, wo bestehen noch Handlunsoptionen?

nicht beantworten. „Die Beantwortung dieser Fragen ist sowohl für politische als auch für wirtschaftliche Zwecke von großer Bedeutung“ [6]. Daraus hat sich die Notwendigkeit ergeben, ein Global-Skala-Modell zu entwickeln, um Antworten auf diese Fragen zu erhalten.

2.2 Historie

WaterGAP wurde erst 1996 am CESR entwickelt und war drei Jahre im Einsatz. In 2000 begann die Entwicklung von Version 2 (WaterGAP2), welche immer noch an der Universität Frankfurt mit fünf Entwicklern und am Geoforschungszentrum Potsdam mit zwei Entwicklern weiterentwickelt wird. Die Entwicklung von WaterGAP3 begann 2007 am CESR und zurzeit sind zehn Personen involviert.

2.3 Einsatz

Die Hauptbenutzer des Modells sind gleichzeitig die Entwickler selbst (ca. 15 – 20 Personen). Eventuell benutzen weitere Master- und Diplomstudierende im Rahmen ihrer Abschlussarbeit das WaterGAP-Modell. Der Ergebnisnutzergruppe ist aber erheblich größer. Die Modellergebnisse können über Web-GIS heruntergeladen und analysiert werden. GIS (Geoinformationssysteme) sind Informationssysteme zur Erfassung, Bearbeitung, Organisation, Analyse und Präsentation geografischer Daten. Viele Veröffentlichungen der Ergebnisse erscheinen in wissenschaftlichen Journalen, aber auch in den Reports von UNEP, EU, EAA usw.

3 Modellentwicklung

Das WaterGAP-Modell besteht aus einzelnen Teilmodellen:

- Hydrologie-Modell
- 5 Wassernutzungsmodelle (Bewässerung, Viehhaltung, Elektrizitätsproduktion, verarbeitendes Gewerbe, Haushalte)
- Wasserqualitätsmodell
- Metamodell

Diese Teilmodelle bilden das WaterGAP- Framework. Die sind konzeptionell mathematisch-physikalisch und werden durch loose Kopplung miteinander verknüpft. Die loose Kopplung bedeutet, dass der Modelloutput der Wassernutzungsmodelle direkt vom Hydrologiemodell eingelesen und verwertet werden können (oder umgekehrt).

In den folgenden Kapiteln werden die Teilmodelle Wasserqualitätsmodell und Metamodell nicht berücksichtigt, da sie im Zeitpunkt des Interviews gerade erst fertig gestellt waren und entsprechende Dokumentationen bzw. Informationen nicht vorhanden waren.

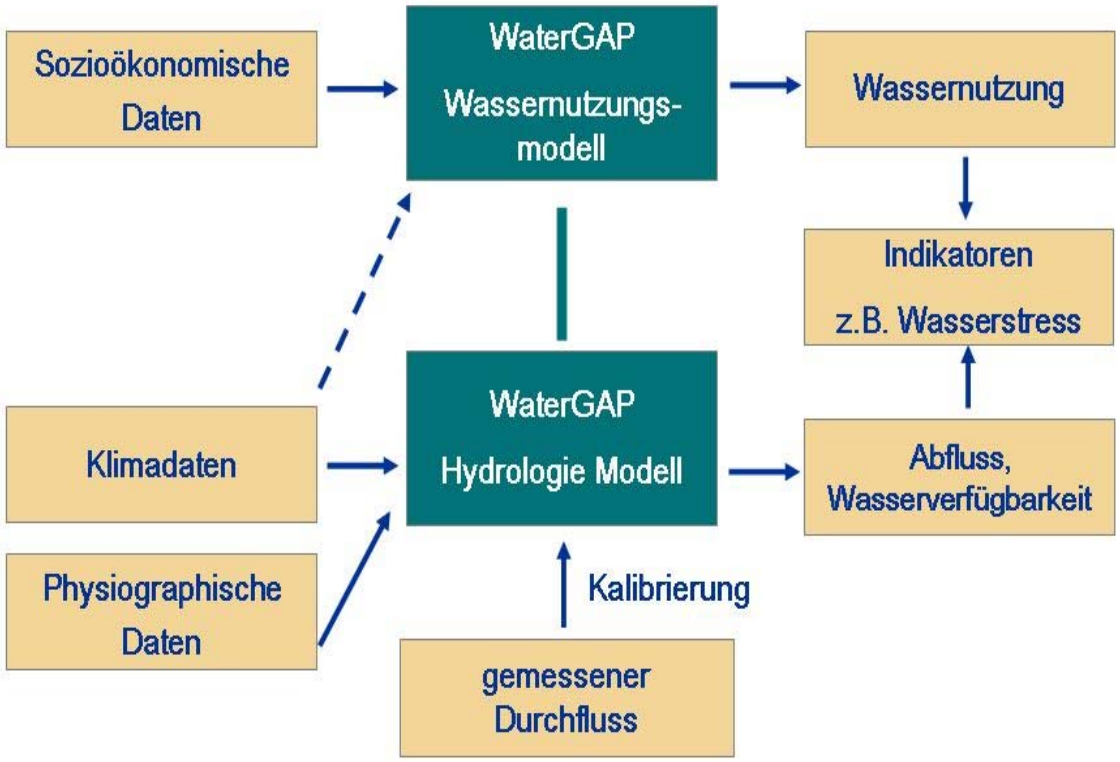

Abbildung 2: WaterGAP-Modellschema (Quelle: Aus Der Beek, 2010)

3.1 Vorgehensweise der Entwicklung

Die Vorgehensweise der Softwareentwicklung entspricht nicht den traditionellen Modellen der Softwareentwicklung. Weder die Anforderungen an die Software noch die erwarteten Ergebnisse sind zu Anfang der Entwicklung bekannt. Dennoch scheint diese Vorgehensweise innerhalb dieses Kontextes erfolgreich zu sein [2]. WaterGAP nutzt ebenfalls diese Vorgehensweise. Das Ziel der Entwicklung ist es, einen Algorithmus zu finden, mit dem sich zukünftig gemessene Klimadaten möglichst genau vorhersagen lassen. Um dies zu verwirklichen, wird eine Zeitreihe ausgewählt, in der bereits Daten gemessen wurden. Anschließend wird ein Algorithmus entwickelt, der aus verschiedenen Parametern diese Werte möglichst genau zurückliefert. Durch Anpassung der Parameter des Algorithmus wird versucht, näher an die gemessenen Werte zu gelangen. Ist ein akzeptables Ergebnis vorhanden, kann mit der Validierung begonnen werden. Dafür wird der Algorithmus mit Parametern aus einer anderen

Zeitreihe verglichen. Unterscheiden sich die gemessenen Werte nur wenig von den Ergebnissen, funktioniert der Algorithmus akzeptabel. Andernfalls muss der Algorithmus und seine Parameter angepasst werden. Der Algorithmus ist nichts anders als eine mathematische Funktion, die in Software implementiert wird. Er wird als Methode bezeichnet.

Bei der WaterGAP-Entwicklung werden folgende Schritte durchlaufen:

1. Literaturrecherche zum Thema
2. Methodenauswahl (Welche mathematische Formel kann zur Berechnung der Modelldaten benutzt werden?)
3. Datenakquise (Sind die zu assimilierenden Eingangsdaten zur Bearbeitung der neuen wissenschaftlichen Fragestellung überhaupt vorhanden?
4. Planung der Implementierung
5. Implementierung
6. Test/Validierung des neuen Modelloutputs

Wenn die ausgewählte Methode nicht passend ist bzw. nicht kalibriert oder validiert werden kann, führt dies zurück zur Literaturrecherche und der Vorgang beginnt von Neuem.

3.2 Modellierung

Die Modellierung besteht aus drei Hauptbereichen. Im Mittelpunkt steht das Modell selbst. Es liest die Inputdaten, verrechnet diese und liefert die Outputdaten, die später analysiert und bewertet werden. Das Pre- und Post-Processing der Input- und Outputdaten ist ein Teil der Modellentwicklung und kann nicht separat betrachtet werden.

3.2.1 Inputdaten und Pre-Processing der Inputdaten

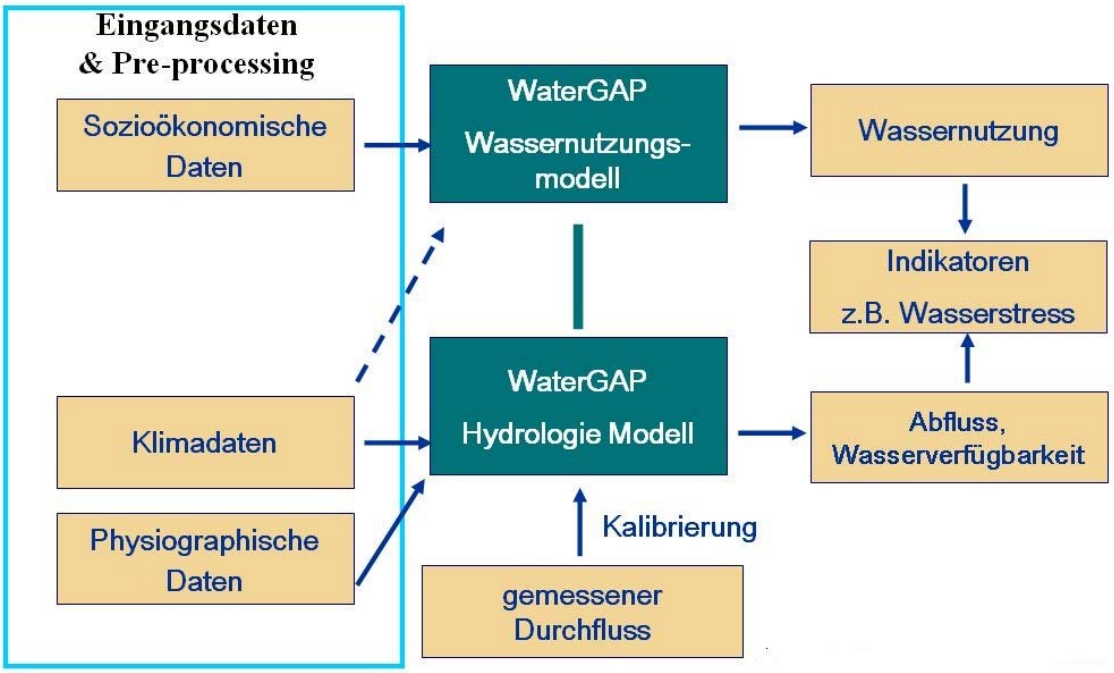

Abbildung 3: WaterGAP-Pre-Processing (Quelle: Tim aus der Beek, 2010)

Je nach Teilmodell werden verschiedene Inputdaten benötigt. Die Hauptinputdaten sind Klimadaten (z.B. Temperatur, Niederschlag, Verdunstung), Basisdaten (z.B. Basiskarten, Landnutzung, Bevölkerung) und sozioökonomische Daten (z.B. gemessene Abflüsse, Wasserentnahme der Industrien).

Da für jede der Rasterzellen (Grids) eine Berechnung stattfindet, müssen die Inputdaten nach räumliche und zeitliche Kriterien aufgeteilt werden. Dieser Schritt wird als Pre-Processing genannt. Er umfasst die:

Formatierung der Daten - Alle Inputdaten sollen entsprechend auf die Auflösung räumlich und zeitlich aggregiert werden. Hierfür wird eine mathematisch-physische Methode verwendet. Die Überprüfung dieser Daten erfolgt meist durch die Visualisierung mit Hilfe von GIS.

Homogenisierung der Dateiformate - Alle Inputdaten müssen anschließend ins binäre UNF-Format umgewandelt werden. Das Modell bearbeitet nur UNF-Dateien.

Konzepte zur Organisation der Inputdaten - Die Klimadaten liegen mit täglichen und monatlichen Grids von 1901 bis 2002 als binäre Datei vor, die ca. 350 GB umfasst. Die Basisdaten bestehen aus statischen Grids und umfassen 2 GB. Die sozioökonomischen Daten betra-

gen ca. 300 GB. Zur Verwaltung dieser Dateien wird ein Datenbank-Managementsystem eingesetzt.

3.2.2 Implementierung

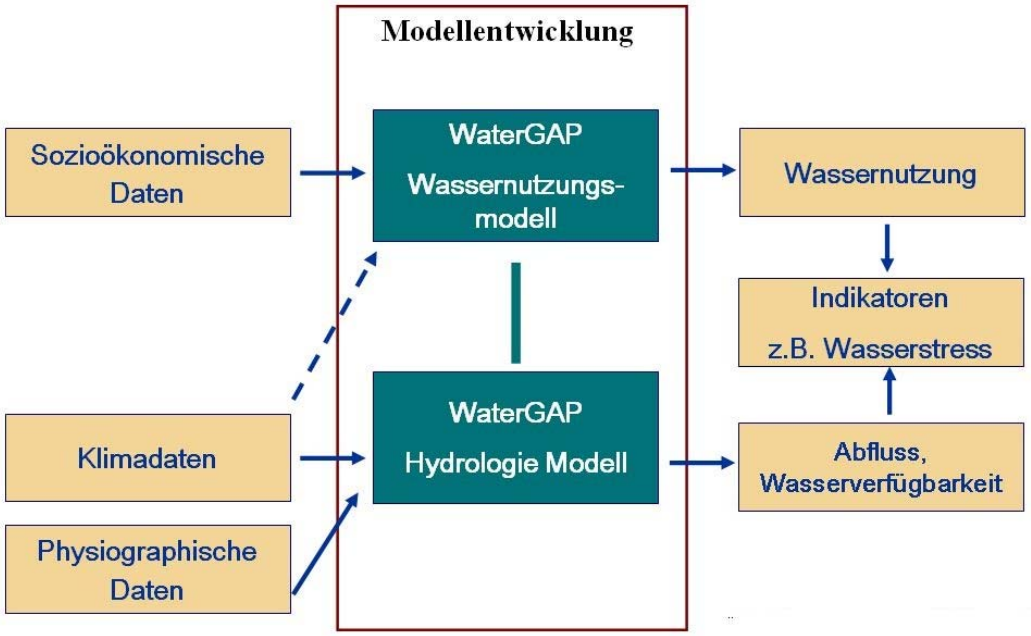

Abbildung 4: WaterGAP-Modellschema - Modellentwicklung(Quelle: Aus Der Beek, 2010)

Für die Implementierung eingesetzte Programmiersprache ist je nach Teilmodell unterschiedlich. Das Hydrologie-Modell ist fast komplett auf C++ geschrieben. Die Teilmodelle Wasserqualität, industrielle Wassernutzung und häusliche Wassernutzung sind nur in C++, die Tier-Wassernutzung ist in C und Bewässerungswassernutzung ist in C/C++ geschrieben.

Entwicklungswerkzeuge sind Eclipse und verschiedene Texteditoren. Eine MPI-Bibliothek wird zum Parallel-Processing, die NetCDF-Bibliothek wird zur Homogenisierung der Outputdaten und die MySQL++-Bibliothek wird zum Einlesen der Inputdaten aus der Datenbank verwendet. Der Aufwand der Implementierung beträgt ca. 20 000 Zeilen Code für das Hydrologiemodell, ca. 31 000 Zeilen für die Wassernutzungsmodelle und ca. 11 000 Zeilen für das Wasserqualitätsmodell. Das Modell besteht insgesamt aus über 62 000 Zeilen Code (keine Angabe zum Implementierungsaufwand des Metamodells).

Bibliotheken sind Sammlungen von Routinen und nützliche Datentypen, die in anderen Programmen verwendet werden können. Eine MPI-Bibliothek ist ein Framework für die Kommunikation zwischen den auf den verschiedenen Rechenknoten laufenden parallelen Prozessen. MPI wurde für höheren Datendurchsatz auf massiv parallelen Rechner und auf Workstation-Clustern konstruiert und allgemein als Standard angesehen [7]. Die Parallelisierung wird

zur Rechenzeitoptimierung eingesetzt und ist auf dem Cluster der Universität Kassel lauffähig. NetCDF ist ein standardisiertes binäres Dateiformat in der wissenschaftlichen Software.

3.2.3 Outputdaten und Post-Processing der Outputdaten

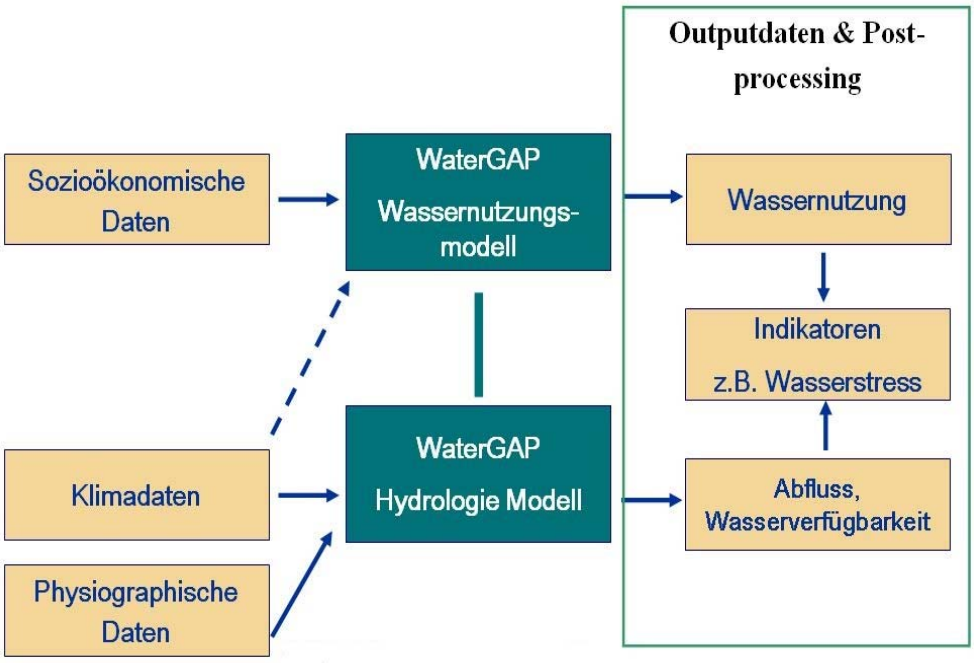

Abbildung 5: WaterGAP-Modellschema- Post-Processing (Quelle: Aus Der Beek, 2010)

Das Modell liefert die Outputdaten als täglichen und monatlichen Grid mit ca. 40 dynamischen Ausgabeparametern (Wasserverfügbarkeit, Grundwasserneubildung, usw.) als binäre Dateien im UNF-Format. Für einzelne Grids oder ganze Länder wird das Ergebnis als Textdatei geliefert. Diese binären Outputdaten werden anschließend in nutzbares Format umgewandelt für die Auswertung. Dieser Schritt wird Post-Processing genannt. Es werden ca. 15 verschiedene Programme benutzt, mit denen die Konvertierung (UNF-Datei zu NetCDF-Datei), Auswertung (z.B. Evaluierung der Modellgüte) und Formatierung (z.B. zeitlich räumliche Aggregierung) des Modelloutputs. Oft findet auch eine visuelle Überprüfung mittels GIS (z.B. ArcGIS 9) statt.

3.2.4 Testen

Der Softwaretest ist definiert nach Myers die Ausführung eines Programms mit der Absicht, Fehler zu finden [8]. Das klassische Testverfahren ist in der wissenschaftlichen Softwareentwicklung schwierig zu verwenden, da das Testen einige Herausforderungen mit sich bringt. Die verwendete Methode kann leicht sehr komplex werden, wobei die Validierung sehr

schwierig wird. Außerdem ist es bei falschen Ergebnissen schwierig herauszufinden, welche Parameter falsch gewählt wurden.

In der wissenschaftlichen Softwareentwicklung spielen die Richtigkeit, Verifikation und Validierung der Berechnung eine Große Rolle. Die Richtigkeit oder die Genauigkeit einer Berechnung, ist ein kritischer Faktor für die Qualität meisten wissenschaftlichen Software. Wissenschaftliche Software erfordert oft ein hohes Maß an Korrektheit um seine Anforderungen zu erfüllen [5]. Die Verifikation ist die Sicherstellung, dass der Code die Gleichungen des Modells korrekt löst [3] oder nach Post und Votta die Feststellung, ob der Code das gewählte Modell richtig löst [9]. Allgemein ist sie die Antwort auf die Frage, ob der Code und der Algorithmus wie gewünscht funktioniert [4]. Die Validierung ist nach Kendall et al die Feststellung, ob das in den Code implementierte mathematische Modell das beabsichtigte physikalische Verhalten richtig imitiert [3] und nach Post und Votta die Bestimmung, ob das Modell die wesentlichen physikalischen Phänomene mit ausreichender Genauigkeit erfasst [9].

Das Problem bei der Verifizierung ist, dass sie sehr aufwendig ist und entsprechendes Fachwissen erfordert. Daher tendieren die Wissenschaftler zu Validierungstests. Auf andere Tests wie Integrations- oder Komponententest wird meistens verzichtet. Der Validierungstest befasst sich mit den von Ihnen erstellten Modellen und ist daher für sie leichter durchzuführen als andere Tests [5]. Auch eine Überprüfung durch externen Tester stellt ein Problem dar, weil die meisten Tests voraussetzen, dass der Tester die Arbeitsweise der Software versteht. Der Mangel an wissenschaftlichen Fachkenntnissen unter Software-Testern verhindert ihren Einsatz im Bereich der wissenschaftlichen Softwareentwicklung.

Bei der WaterGAP-Entwicklung werden keine Softwaretests nach bestimmten Verfahren durchgeführt, jeder Entwickler testet das Programm nach seiner eigenen Methode.

3.2.4.1 Debuggen

Wegen der Parallelisierung der Software wird klassisches Debugging kaum verwendet. Stattdessen werden die Zwischenwerte mit der Printf-Methode ausgegeben. Printf ist eine Funktion in C/C++, die formatierte Werte in der Standardausgabe stdout (normalerweise Bildschirm) zeigt. Falls keine Fehler in der Berechnung vorliegen, wird mit der Kalibrierung fortgefahren. Treten Fehler auf, werden die Zwischenergebnisse gesammelt und der Code wird analysiert.

3.2.4.2 Kalibrierung

Kalibrierung ist der Vergleich von Messungen eines Prüfmittels mit vorhandenen Werten nach einem dokumentierten Verfahren mit dem Ziel, Abweichungen zu erkennen und aufzuzeichnen. Dabei kann zusätzlich die Einhaltung vorgegebener Toleranzen überprüft und ggf. ein Abgleich durchgeführt werden. Der Begriff beinhaltet diesen Abgleich jedoch nicht zwangsläufig. Durch die Kalibrierung von Prüfmitteln wird deren Funktion und Genauigkeit überprüft und im Kalibrierschein dokumentiert [10]. Bei WaterGAP werden durch die Kalibrierung die Parameter der verwendeten Methode bestimmt. Wie in Abbildung 6 dargestellt, werden die Modellparameter variiert und die Simulationsergebnisse mit gemessenen Werten verglichen. Nichtübereinstimmung oder nicht plausible Ergebnisse führen zu Variation der Modellparameter bis sich die Ergebnisse an die gemessenen Werte annähern. Wenn die Kalibrierung nicht möglich ist, ist die gewählte Methode nicht passend gewählt, was wieder zurück zum Ausgangspunkt führt, der Literaturrecherche.

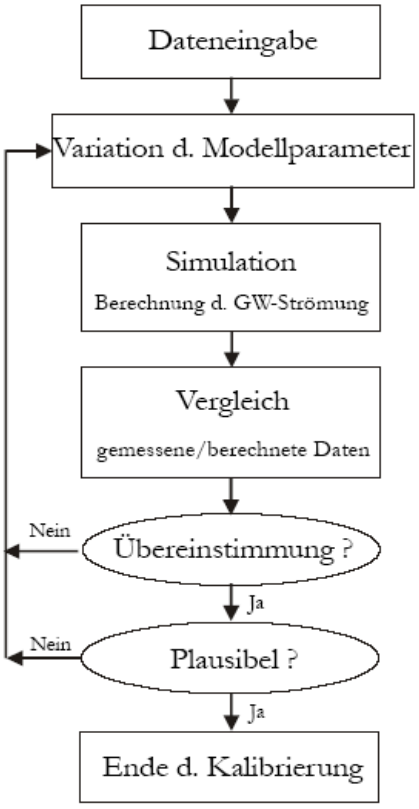

Abbildung 6: Kalibrierungsschritte von WaterGAP (Quelle: Aus Der Beek, 2010)

Die Abbildung 6 zeigt die Schritte der Kalibrierung des Modells. In Abbildung 7 ist deutlich zu sehen, dass eine eins-zu-eins Übereinstimmung nahezu unmöglich ist. Aber der Verlauf

der Ergebnisse ist ähnlich dem der gemessenen Werte.

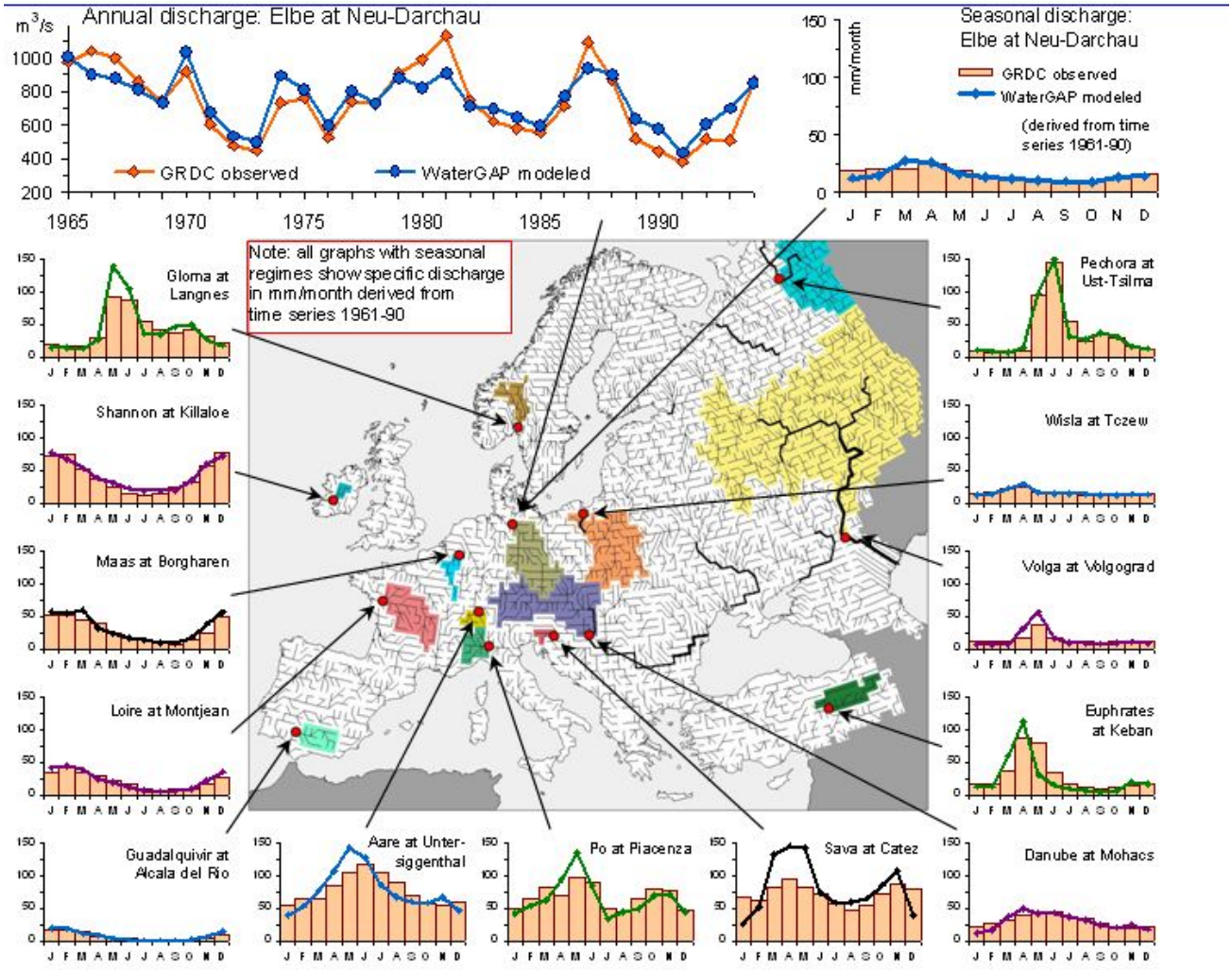

Abbildung 7: Kalibrierte Modellergebnisse in verschiedenen Flusseinzugsgebieten

(Quelle: Aus Der Beek, 2010)

Eine erfolgreiche Kalibrierung führt zum Schritt der Validierung.

3.2.4.3 Validierung

Bei WaterGAP ist die Validierung die Bestätigung, dass die Modellparameter richtig kalibriert wurden und die Simulation wie erwartet funktioniert. Das bedeutet, das Modell liefert zuverlässige Ergebnisse und die ausgewählte Methode ist die richtige.

Die Simulation wird mit den gemessenen Daten aus einer unabhängigen Zeitreihe und mit den durch Kalibrierung bestimmten Modellparametern durchgeführt. Diese Modellergebnisse werden mit den gemessenen Daten, sowie mit Statistiken und zusätzlich mit anderen hydrologischen Modellen verglichen. Notfalls werden die Ergebnisse durch Experten analysiert (siehe z.B. Journal of Hydrology 270 (2003): “A global hydrology model for deriving water availability indicators: model tuning and validation“, Döll et al.).

3.2.4.4 Qualitätssicherung

Die anschließende Sensitivitäts- und Unsicherheitsanalyse sind notwendig um die Zuverlässigkeit des Modells festzustellen. Sie gehören zum Schritt der Qualitätssicherung. Die Abbildung 8 stellt die Schritte zur Qualitätssicherung der Modellergebnisse dar.

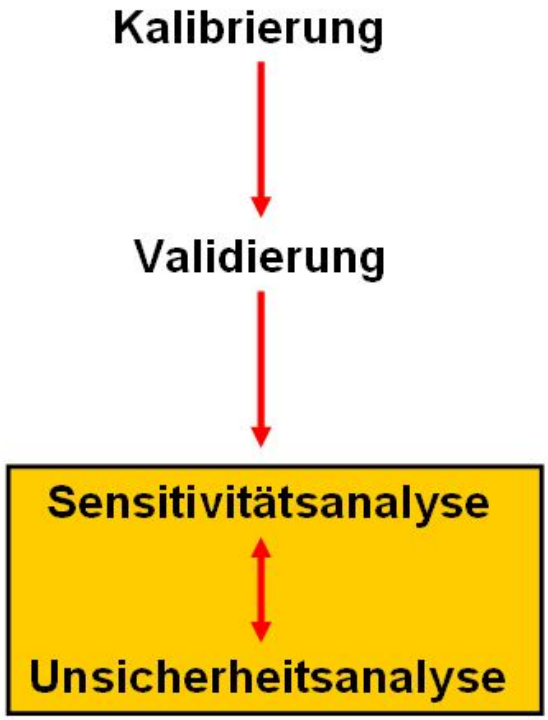

Abbildung 8: Schritte zur Qualitätssicherung der Modellergebnisse (Quelle: Aus Der Beek, 2010)

3.3 Entwicklungsschritte

3.3.1 Eingesetzte Hard- und Software

Das Hydrologie-Teilmodell wird am Cluster der Universität Kassel ausgeführt. Deshalb ist die Parallelisierung von WaterGAP3 zur Rechenzeitoptimierung erfolgt. Die restlichen Modelle werden auf einer Workstation mit einem AMD Opteron Prozessor mit 4 Kernen, 6,2 TB Festplatte und 32 GB Arbeitsspeicher ausgeführt. Das eingesetzte Betriebssystem ist Linux SUSE.

3.3.2 Releasemanagement/Versionskontrolle

Zur Releasemanagement wird die Versionierung verwendet. Ein bestimmtes Verfahren zum Releasemanagement und zur Versionskontrolle ist nicht festgelegt.

3.3.3 Mittel der Interaktion

Nach der Fertigstellung einer (Sub-)Version werden die interne schriftliche technische Berichte und Kurzanleitungen gefertigt. Diese Anleitungen werden in Microsoft-Word geschrieben und befinden sich auf dem Server des CESR. Der Einsatz von anderen Mitteln wie z.B. Wikis findet nicht statt.

3.3.4 Kollaboration

Aufgrund der Komplexität des Modells beschränkt sich die Benutzergruppe auf die Universitäten Kassel, Frankfurt und das Geoforschungszentrum Potsdam. Zurzeit sind ca. 15 – 20 Entwickler/Benutzer involviert.

3.3.5 Softwareeinführung

Die Softwareeinführung erfolgt durch involvierte Entwickler. Die Einarbeitung dauert mehrere Monate und die Schulung ist unbedingt notwendig.

3.3.6 Softwarelebenszyklus

Das Modell ist ständig weiterentwickelt bzw. erweitert. Eine neue Version 4 ist zurzeit nicht geplant. Eine neue Version ist nur dann sinnvoller, wenn die aktuell verwendete räumliche Auflösung verfeinert werden muss.

3.3.7 Wartung/Pflege

Der Programmcode wurde vollständig mit Kommentaren versehen und Releases werden in Subversionen veröffentlicht. Alle 5 – 6 Jahr werden die Workstations gewechselt. Dies stellte in der Vergangenheit keine Schwierigkeiten dar. Zuletzt erfolgte eine Migration des Systems zu einer 64 Bit-Architektur, welche ebenfalls keine Probleme verursachte.

3.3.8 Datenarchivierung

Nach Abschluss von Projekten werden sämtliche Modellergebnisse komprimiert und im Backup-System gespeichert.

3.4 Qualitätskontrolle nach der Norm ISO/IEC 9126

Die Norm ISO/IEC 9126 ist ein Modell zur Sicherstellung der Softwarequalität. Sie umfasst sechs wichtige Qualitätsmerkmale: Funktionalität, Zuverlässigkeit, Benutzbarkeit, Effizienz, Änderbarkeit und Übertragbarkeit. In folgenden Unterkapiteln wird das WaterGAP-Modell nach dieser Norm durch den Interviewpartner bewertet.

3.4.1 Funktionalität

Die Definition der Funktionalität ist beschrieben durch das „Vorhandensein von Funktionen mit festgelegten Eigenschaften. Diese Funktionen erfüllen die definierten Anforderungen“ [11]. Sie umfasst Angemessenheit, Richtigkeit, Interoperabilität, Sicherheit und Ordnungsmäßigkeit.

Angemessenheit ist die „ Eignung von Funktionen für spezifizierte Aufgabe, zum Beispiel aufgabenorientierte Zusammensetzung von Funktionen aus Teilfunktionen“ [11]. Der Interviewpartner konnte zu diesem Punkt keine Angaben machen.

Richtigkeit ist das „Liefern der richtigen oder vereinbarten Ergebnisse oder Wirkungen, zum Beispiel die benötigte Genauigkeit von berechneten Werten“ [11]. Es wird angenommen, dass das Modell richtige Ergebnisse liefert, ansonsten würde es nicht eingesetzt werden. Die Modellergebnisse schließen aber „right for wrong reasons“ [6] nicht aus. Auch die empirisch bestimmten ausgewählten Methoden stellen eine Fehlerquelle dar.

Interoperabilität ist die „ Fähigkeit, mit vorgegebenen Systemen zusammenzuwirken.“ [11]. Der Interviewpartner konnte zu diesem Punkt keine Angabe machen.

Sicherheit ist die „Fähigkeit, unberechtigten Zugriff, sowohl versehentlich, als auch vorsätzlich, auf Programme und Daten zu verhindern.“ [11]. Die Daten und der Programmcode liegen auf dem internen Server des CESR und das Programm ist nicht online erreichbar. Sofern der Server nicht kompromittiert wird, ist es als sicher angesehen.

Ordnungsmäßigkeit besteht dann, wenn eine Software Merkmale besitzt, „die bewirken, dass die Software anwendungsspezifische Normen oder Vereinbarungen oder gesetzliche Bestimmungen und ähnliche Vorschriften erfüllt“ [11]. Der Interviewpartner konnte zu diesem Punkt keine Angabe machen.

3.4.2 Zuverlässigkeit

Zuverlässigkeit ist die „Fähigkeit der Software, ihr Leistungsniveau unter festgelegten Bedingungen über einen festgelegten Zeitraum zu bewähren.“ [11]. Diese Norm umfasst vier Merkmale: Reife, Fehlertoleranz, Wiederherstellbarkeit und Konformität.

Reife ist definiert durch „Geringe Versagenshäufigkeit durch Fehlerzustände.“ [11]. Das Modell ist ausgereift genug um stabil zu laufen.

Fehlertoleranz ist die „Fähigkeit, ein spezifiziertes Leistungsniveau bei Software-Fehlern oder Nicht-Einhaltung ihrer spezifizierten Schnittstelle zu bewahren“ [11]. Einen Programmabbruch gibt es nur bei schwerwiegenden Fehlern wie fehlende Eingangsdaten.

Wiederherstellbarkeit ist die „Fähigkeit, bei einem Versagen das Leistungsniveau wiederherzustellen und die direkt betroffenen Daten wiederzugewinnen. Zu berücksichtigen sind die dafür benötigte Zeit und der benötigte Aufwand“ [11]. Das Programm wird durch Neustart wiederhergestellt.

Konformität ist der „Grad, in dem die Software Normen oder Vereinbarungen zur Zuverlässigkeit erfüllen“ [11]. Da zur Zuverlässigkeit keine Angaben gemacht werden konnten, kann auch keine Angabe zur Konformität gemacht werden.

3.4.3 Benutzbarkeit

Benutzbarkeit ist der „Aufwand, der zur Benutzung erforderlich ist, und individuelle Beurteilung der Benutzung durch eine festgelegte oder vorausgesetzte Benutzergruppe“ [11]. Diese Norm umfasst fünf wichtige Eigenschaften: Verständlichkeit, Erlernbarkeit, Bedienbarkeit, Attraktivität und Konformität einer Software.

Verständlichkeit ist der „Aufwand für den Benutzer, das Konzept und die Anwendung zu verstehen“ [11]. Um das Programm zu verstehen muss der Benutzer erst die komplexen hydrologischen Prozesse verstehen. Für den normalen Endbenutzer stellt dies ein großes Problem dar.

Erlernbarkeit ist der „Aufwand für den Benutzer, die Anwendung zu erlernen“ [11]. Die Erlernbarkeit ist sehr aufwendig, da die Einarbeitung mehrere Monate dauert.

Bedienbarkeit ist der „Aufwand für den Benutzer, die Anwendung zu bedienen“ [11]. Die Bedienbarkeit ist eingeschränkt, da es keine graphische Benutzeroberfläche (GUI) gibt.

Attraktivität ist die „Anziehungskraft der Anwendung gegenüber dem Benutzer“ [11]. Das Programm besitzt kaum eine Attraktivität gegenüber normalen Benutzern.

Konformität ist der „Grad, in dem die Software Normen oder Vereinbarungen zur Benutzbarkeit erfüllt“ [11]. Da zur Zuverlässigkeit keine Angaben gemacht werden konnten, kann auch keine Angabe zur Konformität gemacht werden.

Das Modell wird hauptsächlich von Entwicklern genutzt, deshalb stellt die Benutzbarkeit nur ein geringes Problem dar.

3.4.4 Effizienz

Effizienz ist das „Verhältnis zwischen dem Leistungsniveau der Software und dem Umfang der eingesetzten Betriebsmittel unter festgelegten Bedingungen“ [11]. Diese Norm umfasst drei wichtige Eigenschaften: Zeitverhalten, Verbrauchsverhalten und Konformität. Das WaterGAP hat erhöhten Bedarf an Rechenzeit und Speicher.

Zeitverhalten beschreibt die „Antwort- und Verarbeitungszeiten sowie Durchsatz bei der Funktionsausführung“ [11]. Zur Optimierung der Rechenzeit wurde eine Parallelisierung des Programmcodes für den Cluster durchgeführt. Ein Modelldurchlauf dauert ca. 1,5 Tage auf dem Cluster.

Verbrauchsverhalten beschreibt die „Anzahl und Dauer der benötigten Betriebsmittel bei der Erfüllung der Funktionen. Ressourcenverbrauch , wie CPU-Zeit, Festplattenzugriffe usw. „ [11]. Das Programm besitzt erhöhten Bedarf an Ressourcen. Der Arbeitsspeicher mit 32GB RAM ist zurzeit zufriedenstellend, aber der Zielspeicher (Festplatte) ist zu klein (6TB) für die Outputdaten.

Konformität bedeutet „Grad. In dem die Software Normen oder Vereinbarungen zur Effizienz erfüllt“ [11]. Da zum Effizienz keine Angaben gemacht werden konnten, kann auch keine Angabe zur Konformität gemacht werden.

Allgemein ist die Software als effizient anzusehen.

3.4.5 Änderbarkeit

Änderbarkeit ist der „Aufwand, der zur Durchführung vorgegebener Änderungen notwendig ist. Änderungen können Korrekturen, Verbesserungen oder Anpassungen an Änderungen der Umgebung, der Anforderungen oder der funktionalen Spezifikationen einschließen“ [11]. Diese Norm umfasst vier Eigenschaften: Analysierbarkeit, Modifizierbarkeit, Stabilität und Testbarkeit.

Analysierbarkeit beschreibt den „Aufwand, um Mängel oder Ursachen von Versagen zu diagnostizieren oder um änderungsbedürftige Teile zu bestimmen“ [11]. Die Änderungen im Programm sind sehr aufwendig.

Modifizierbarkeit beschreibt den „Aufwand zur Ausführung von Verbesserungen, zur Fehlerbeseitigung oder Anpassung an Umgebungsänderungen“ [11]. Da die Analyse sehr aufwendig ist, ist auch die Modifikation sehr aufwendig.

Stabilität beschreibt die „Wahrscheinlichkeit des Auftretens unerwarteter Wirkungen von Änderungen“ [11]. Diese Eigenschaft ist nicht voraussehbar.

Testbarkeit bedeutet „Aufwand, der zur Prüfung der geänderten Software notwendig ist“ [11]. Tests sind bei WaterGAP sehr aufwendig.

3.4.6 Übertragbarkeit

Übertragbarkeit ist die „Eignung der Software, von der Umgebung in eine andere übertragen werden zu können. Umgebung kann organisatorische Umgebung, Hardware- oder Software-Umgebung sein“ [11]. Diese Norm umfasst fünf Eigenschaften: Anpassbarkeit, Installierbarkeit, Koexistenz, Austauschbarkeit und Konformität.

Anpassbarkeit ist die „Fähigkeit der Software, diese an verschiedene Umgebungen anzupassen“ [11]. Das Programm ist nur unter Linux SUSE lauffähig. Es gibt es keine Angabe zur Kompatibilität mit anderen Linux-Distributionen.

Installierbarkeit beschreibt den „Aufwand, der zum Installieren der Software in einer festgelegten Umgebung notwendig ist“ [11]. Bei WaterGAP ist keine Installation notwendig.

Koexistenz ist die „Fähigkeit der Software neben einer anderen mit ähnlichen oder gleichen Funktionen zu arbeiten“ [11]. Zur Fähigkeit der Koexistenz mit anderer Software oder Funktionen ist nichts Genaueres bekannt.

Austauschbarkeit ist die „Möglichkeit, diese Software anstelle einer spezifizierten anderen in der Umgebung jener Software zu verwenden, sowie der dafür notwendige Aufwand“ [11]. Es gibt kein anderes Modell, das auf der Global-Skala rechnet. Ein Austausch mit einer anderen Software ist daher nicht möglich.

Konformität bedeutet „Grad, in dem die Software Normen oder Vereinbarungen zur Übertragbarkeit erfüllen“ [11]. Zur Konformität kann keine Angabe gemacht werden.

3.5 Besonderheiten

Es sind keine dauerhaften Stellen zur Programmierung und Pflege von WaterGAP vorhanden. Die Entwicklung von WaterGAP wird durch Drittmittelprojekte (z.B. BMBF, EU) finanziert, welche keine solche Stelle vorsehen. Deswegen sind ausschließlich Naturwissenschaftler in der Entwicklung involviert. Lediglich die Parallelisierung wurde von einem externen Informatiker übernommen. Die Chefprogrammiererin am CESR unterstützt die Entwickler teilwei-

se bei ihrer Arbeit. Die Entwickler am CESR arbeiten im selben Gebäude und jede Woche werden im Team-Meeting die Fortschritte bzw. Ergebnisse diskutiert.

Aus finanziellen Gründen wird hauptsächlich Freeware (z.B. Eclipse, MySQL) eingesetzt.

4 Zusammenfassung

4.1 Zusammenfassung des Interviews

Der Interviewpartner bezeichnet Softwareentwicklung in der Wissenschaft als „notwendig um natürliche und sozioökonomische Prozesse realitätsnah in einem Computermodell wiederzugeben und damit weitergehende Fragestellungen bearbeiten zu können. In Projekten wird oftmals das Ziel verfolgt, das Modell weiter zu entwickeln um neue wissenschaftliche Fragestellungen zu beantworten: z.B. Wie wirkt sich der Klimawandel auf die Wasserressourcen aus? Außerdem ist sie unterstützend beim Pre- und Post-Processing der Input- und Outputdaten“ [6]. Der Interviewpartner konnte während des Interviews softwaretechnischen Fragen gar nicht oder nur teilweise beantworten. Wahrscheinlich liegt der Grund dafür darin, dass der Interviewpartner die formalen Aspekte der Softwareentwicklung nicht kennt.

4.2 Fazit

Es ist schwierig für Informatiker in das Projekt einzuspringen und die Implementierungsrolle zu übernehmen. Da das Modell sehr komplex ist, ist das naturwissenschaftliche Fachwissen unbedingt erforderlich. Der Einsatz von Software-Testern ist auch deshalb sehr schwierig. Eine Hilfstätigkeit ist daher sinnvoller z.B. zur Parallelisierung, Automatisierung der Modelldurchläufe, die Implementierung bestimmter Funktionen oder die Pflege der Software.

Die traditionellen Softwareentwicklungsprinzipien müssen nicht an erster Stelle stehen, da sich Vorgehensweise meist sehr von der traditionellen Softwareentwicklung unterscheidet. Es ist jedoch von Vorteil, einige Werkzeuge wie verbesserte Mittel zur Dokumentation (z.B. eigene Wiki) und ein verbessertes Datenmanagement (Backup-System, Versionsmanagement, Releasemanagement) einzusetzen.

Literaturverzeichnis

Quellen:

Aus der Beek, Tim (2010): Geodätisches Oberseminar Universität Stuttgart

J.Alcamo, P.Döll, T.Henrichs, F.Kaspar, B.Lehner, T.Rösch, S.Siebert (2003): Development and the testing of the WaterGAP2, Hydrological Sciences-Journal-des Sciences Hydrologiques 48(3) June 2003

P.Döll, F.Kaspar, B.Lehner (2001): A global hydrology model for deriving water availability indicators: model tuning and validation, Journal of Hydrology 270 (2003) 105-134

Referenzen:

[1] K.Kreyman, D.L.Parnas, S.Qiao (1999): "Inspection Procedures for Critical Programs that Model Physical Phenomena". CRL Report No.368. McCaster University

[2] J.Segal (2004): Models of scientific software development

[3] R.P.Kendall, D.E. Post, J.C. Carver, D.B. Henderson, D.A.Fisher (2007): A proposed Taxonomy for Software Development Risks for High-performance Computing (HPC) Scientific/Engineering Applications. Technical Notes. Carnegie Mellon University.

[4] D.E.Stevenson (1999): A Critical Look at Quality in Large-Scale Simulations. Computing in Science and Engineering, (May-Jun. 1999), 53-63

[5] D.Kelly, R.Sanders (2008): The Challenge of Testing Scientific Software, Dissertation on CAST (2008)

[6] T.Aus der Beek (2011): Interview zum WaterGAP-Modell

[7] http://www.mcs.anl.gov/research/projects/mpi

[8] G.J.Myers (2004): The Art of Software Testing, 2nd Edition John Wiley & Sons Inc.

[9] D.E. Post, L.G.Votta (2005): Computational Science Demands a New Paradigm. Physics Today (2005), 35-41.

[10] http://quality.de/lexikon/kalibrierung.htm

[11] ISO/IEC 9126