Diplomarbeit Informatik

Thema:

Entwurf und Implementierung eines Systems zur Verwaltung elektronischer Publikationen an einem wissenschaftlichen Institut

Abgabe am : 21.12.2001

Bearbeitet von:
020855 Sebastian Lorr

1. Prüfer: Prof. Dr. Matthiessen
2. Prüfer : Dr. Ana Macario

Kurzfassung

In der vorliegenden Arbeit befasse ich mich mit der Aufgabe, Wissenschaftlern bei der Veröffentlichung von wissenschaftlichen Publikationen zu begleiten. Dieser Prozess der Veröffentlichung baut auf der Datenhaltung am Alfred Wegener Institut für Polar und Meeres Forschung auf. In dieser Arbeit wird der Prozess der Veröffentlichung strukturiert aufgezeigt, und eine Umsetzung anhand eines Objektorientierten Modells konzeptioniert. Das Konzept wird dann zu einem Prototyp implementiert, welcher diesen Arbeitsfluss unterstützt.

Vorwort

Diese Arbeit zur Erlangung des Grades eines Diplom- Informatikers ist nicht nur der Graduierung wegen erstellt worden, sondern sie stellt auch eine Dokumentation dar, mit der meine Arbeit am AWI nachvollzogen und weiterentwickelt werden kann. Da es sich um einen Teil eines Rahmenwerkes für LDAP- Intra/Internet Anwendungen handelt, ist eine gute Dokumentation von höchster Wichtigkeit für weitere Anwendungen bzw. für die Wartung von bestehenden Anwendungen.

Inhalt

Abbildungen

Tabellen

1. Aufgabenstellung

Im Rahmen dieser Diplomarbeit gilt es innerhalb eines bestehenden Informationssystems die Veröffentlichung von wissenschaftlichen Arbeiten mit Hilfe der EDV zu gestalten. So müssen bestimmte Funktionalitäten bereitgestellt werden, mit deren Hilfe Publikationen eingegeben, bearbeitet und gelöscht werden können. Dies soll natürlich mit bestmöglicher Performance geschehen.
Zusätzlich soll das Publikationssystem den Anwender bei dem gesamten Weg der Veröffentlichung seiner Arbeit unterstützen. Hierzu ist es notwendig den Weg durch verschiedene Instanzen herauszuarbeiten, und diesen gegebenenfalls durch die EDV zu vereinfachen, bzw. zu automatisieren.

2. Das Alfred-Wegener-Institut

Die Stiftung Alfred-Wegener-Institut für Polar- und Meeresforschung umfaßt vier Fachbereiche, die in Sektionen gegliedert sind: Klimasystem, Pelagische Ökosysteme, Benthische Ökosysteme und Geosystem. Zu den Themen: Tiefseeforschung und AUV-Entwicklung, Kohlenstoffflüsse und Solare UV-Strahlung arbeiten multidisziplinäre, zeitlich begrenzte Projektgruppen. Den wissenschaftlichen Bereich unterstützen Logistik, Verwaltung, Presse- und Öffentlichkeitsarbeit, Rechenzentrum und die Bibliothek [Abb 1].

1980 wurde das Institut in Bremerhaven als Stiftung des öffentlichen Rechts gegründet. Die Forschungsstelle Potsdam hat 1992 ihre Arbeit aufgenommen. Die Stiftung Alfred-Wegener-Institut, zu der auch die Biologische Anstalt Helgoland gehört, hatte 1999 einen Etat von 165 Mio. DM und beschäftigt rund 700 Mitarbeiterinnen und Mitarbeiter.

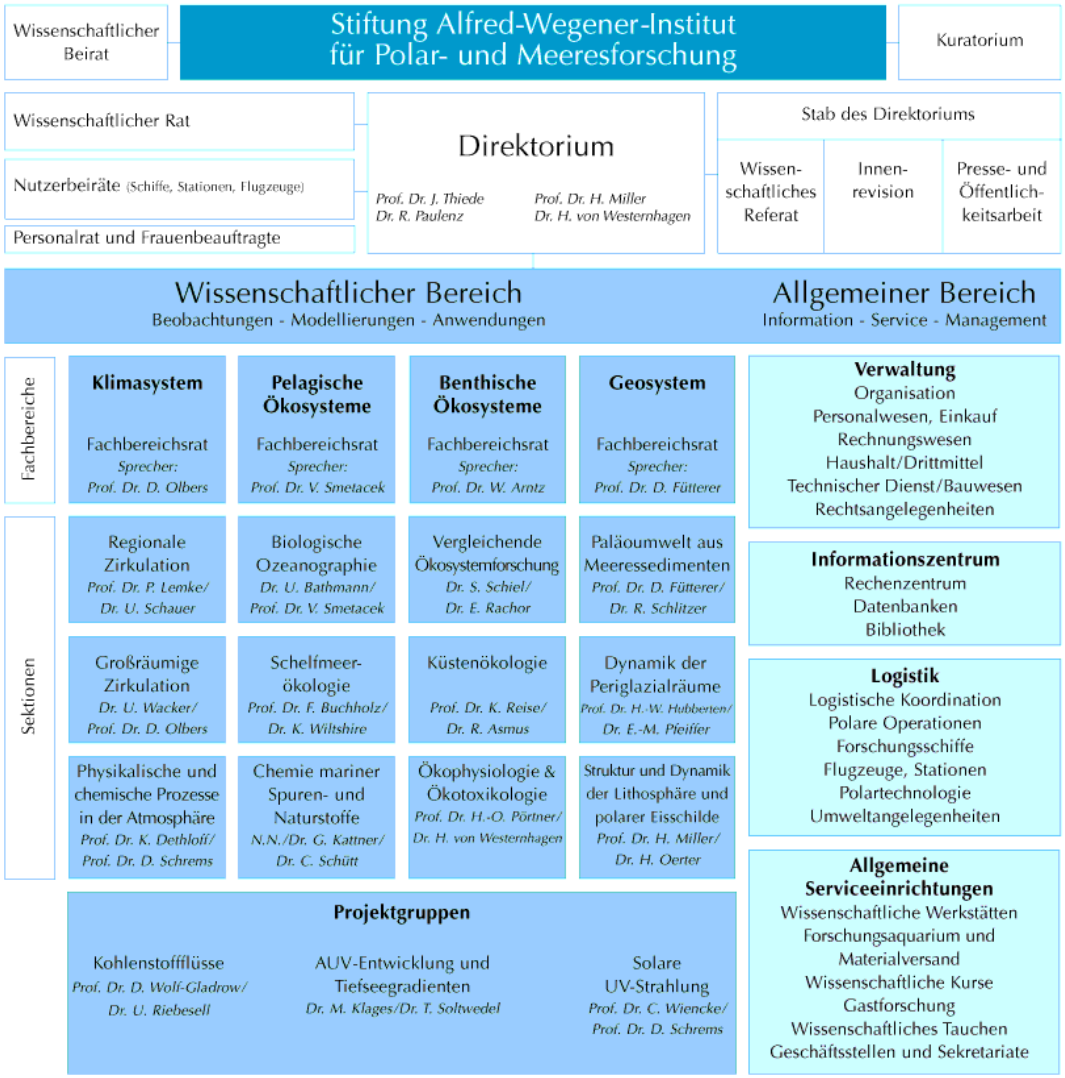


Abb. 1 Organigramm des AWI

Um die Abläufe die in dieser Arbeit beschrieben werden ist es unabdinglich, dass man die Struktur der Personen kennt. Diese Struktur wird in Abb. 2 aufgezeigt.

Jede Person ist mindestens einem Fachbereich zugeordnet. Sie kann weiterhin mindestens einer Sektion und /oder Projektgruppe zugeordnet sein. Jeder Fachbereich, Jede Sektion und jede Projektgruppe besitzen einen Leiter und einen Publikationsbeauftragten.

Autoren sind auch Personen und ordnen Ihren Publikationen Sektionen oder Gruppen zu. Diese sind dann immer einem Fachbereich untergeordnet und so wird auch der Fachbereich der Publikation zugeordnet.

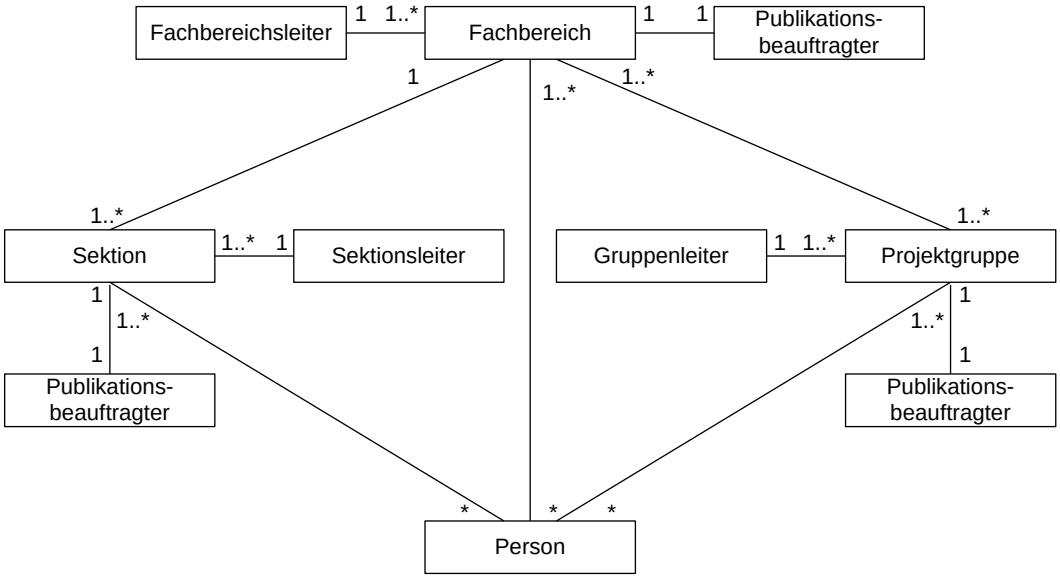


Abb. 2 Beziehung zwischen Organisationseinheit und Personen

3. Arbeits- bzw. Entwicklungsumgebung beim AWI

Das AWI besitzt eine Hard- und ‚Softwareumgebung, die ich im folgenden aufzählen und auf die einzelnen Elemente eingehen werde.

3.1 Hardwareumgebung

Das AWI besitzt mehrere Server von denen die „Enterprise 10000“ für mich von größter Bedeutung ist, da dort auf mehreren Domänen Webserver und Directoryserver installiert sind, welche die Hauptplattformen für die softwaretechnische Umsetzung meiner Arbeit bilden.
Der Entwicklungs- und Testserver ist dabei eine Domäne der E10000, welche mit 2 UltraSPARC Prozessoren mit jeweils 366 MHz und 2 GB Hauptspeicher ausgestattet.
Außerdem ist für meine Entwicklung von Interesse was für Plattformen den Mitarbeitern auf der „Clientseite“ zur Verfügung stehen, da die Software auf allen benutzten Plattformen lauffähig sein muss. Die verschieden Client- Konstellationen werden in folgender Tabelle aufgezeigt:

Rechner	Betriebsystem	Browser
Sun	Solaris 2.8	Netscape
PC	Windows NT/200	Netscape
PC	Windows NT/200	MS IE
MAC	MAC OS	Netscape
MAC	MAC OS	MS IE

Tab. 1 Client- Konstellationen

3.2 Softwareumgebung.

Wie hier schnell ersichtlich wird, muss man bei solch heterogenen Plattformen darauf achten, dass die Ausgabe für vielfältige Plattformen darstellbar ist. Deswegen wird als einfache und weitverbreitete Ausgabe HTML benutzt. HTML wird auf vielen Plattformen unterstützt und sieht fast auf allen Plattformen gleich aus. Als Datenspeichersystem wird ein sogenannter Directoryserver benutzt. Dieser ist auf einer Domäne der Enterprise 10000 installiert. Auf derselben Domäne ist als Webserver der iPlanet Webserver 4.1 sp6 installiert, welcher Java- Servlets unterstützt. Als Programmiersprache ist Java in der Version 1.2.2_8 installiert. Zusätzlich zum Standard JDK ist das Servlet API, das JAF API, das JAVAMAIL API sowie das Netscape LDAP SDK installiert. Des weiteren ist PERL installiert, was sich auch zum Erstellen von webbasierten Programmen über die CGI (Common Gateway Interface) eignet.

Um Klarzustellen weshalb man sich für diese bestimmte Umgebung entschieden hat werde ich nun im Folgenden genauer auf die Eigenschaften der im vorherigen Abschnitt erwähnten Produkte und Techniken eingehen.

3.2.1 Webserver

Als Webserver dient dem AWI der iPlanet Webserver in der Version 4.1_sp7. Diese Version unterstützt Schnittstellen wie z.B. CGI und auch den Einsatz von Java Servlets und JSP (Java Server Pages). Diese

Eigenschaften sind wichtig für die Entwicklung von webbasierten Anwendungen.

3.2.2 Perl- Programmierumgebung

Als CGI(Common Gateway Interface) Sprache wird Perl verwendet.
Die PerLDAP- Module von Netscape erlauben es Entwicklern, mit Hilfe der Perl Script-Sprache, LDAP fähige Applikationen schnell und einfach zu schreiben und zu verwalten. Der Quellcode für PerLDAP steht den Entwicklern umsonst zur Verfügung.
Diese Software besteht aus fünf grundlegenden Modulen:
Mozilla::LDAP::Conn
Mozilla::LDAP::Entry
Mozilla::LDAP::API
Mozilla::LDAP::LDIF
Mozilla::LDAP::Utils

„Conn“ ist das Hauptmodul und für das Eröffnen von Verbindungen zum Directory Server zuständig.
„Entry“ ist für das Bearbeiten von Einträgen zuständig
„API“ ist das eigentliche Grundmodul und bildet die Grundlage für alle anderen Module.
„LDIF“ ist für die Ausgabe von Datensätzen in einem speziellen Dateiformat zuständig.
„Utils“ stellt einige Zusatzfunktionen zur Verfügung.

3.2.3 Java- Programmierumgebung

Als Java Programmierumgebung wird das J2SE als Basis benutzt. Dies enthält Java Version JDK1.2.2_08. Zusätzlich hierzu bedarf es der Installation einiger APIs (Application Programming Interface). Als Editor benutze Ich den Borland JBuilder5 Personal. Dieser ist für nicht kommerzielle Zwecke frei verfügbar, und bietet einen erheblichen Komfort, wenn es um objektorientierte Programmierung mit Java geht.

3.2.3.1 LDAP- Zugriff

Man benötigt für den Zugriff auf den Directoryserver eine API, die LDAP (Lightweight Directory Access Protocol) unterstützt. Da wir einen Directoryserver von Netscape benutzen bietet sich das Netscape LDAP SDK in der Version 4.1 an. Alternativ hätte man von java.sun.com JNDI (Java Naming and Directory Interface) mit dem entsprechenden Provider für LDAP benutzen können, allerdings ist das Nestcape API besser auf den LDAP - Zugriff spezialisiert, während beim JNDI LDAP nur eine von vielen Funktionen ist, und somit auch nicht so umfassend von der Funktionalität ist.

3.2.3.2 Servlet

Auf java.sun.com findet man die Servlet- API in der Version 2.3, welche die Servlet- Spezifikation 2.3 sowie JSP(Java Server Pages) in der Version 1.2 unterstützt. Die Servlet API ermöglicht es Servlets zu erstellen, welche dann wiederum die Kommunikation zwischen Webclient und Webserver ermöglichen

3.2.3.3 Mails

Da die Applikationen auch Emails verschicken sollen, benötigt man eine Mail- API. Auf java.sun.com findet man eine solche API (JavaMail, version 1.2)als Provider vom JAF (JavaBeans Activation Framework).

3.2.4 Directoryserver

Das beim AWI verwendete Produkt ist der „iPlanet Directoryserver 4.1“. Ein Directoryserver stellt bestimmte Dienste zur Verfügung(Directoryservices). Er ist auf eine Datenbank aufgesetzt und bietet in einer bestimmten Art und Weise Zugriff auf diese Datenbank. Die Daten sind in einer hierarchischen baumartigen Struktur abgespeichert. Jeder einzelne Eintrag ist eindeutig identifizierbar durch seinen DN (Distinguished Name). Dieser DN ist zusammengesetzt aus der Position des jeweiligen Eintrages im Baum [Abb. 3]. Innerhalb jeden Astes ist jeder Eintrag durch seinen

RDN (Relative Distinguished Name) eindeutig identifizierbar. Im Abb. 4 ist der DN des Personeneintrages von slorr :
„uid=slorr,ou=people,o=awi-bremerhaven.de“
der RDN ist:
„uid=slorr“
Im Directory werden die Daten in „Attribut – Wert – Paaren“ abgespeichert. Es können allerdings auch einem Attribut mehrere Wert zugewiesen werden, was ein Vorteil den herkömmlich relationalen Datenbanken gegenüber ist. Der Directoryserver ist auf Lesezugriff optimiert. So geht man bei der Benutzung eines Directoryservers von einem Verhältnis von einem Schreibzugriff auf tausend Lesezugriffe aus. Die „Performance“ ist im Vergleich zu einem herkömmlichen relationalen System 5-10 mal so hoch.
Die Daten sind in einer baumartigen Struktur abgespeichert. Dabei gibt es eine Wurzel, von der verschiedene Äste ausgehen. Diese Äste enthalten Thematisch zusammengehörige Elemente, von denen wieder Äste abzweigen können. Beim AWI werden verschiedene Daten im Directoryserver gespeichert. So gibt es unter der Wurzel „o=awi-bremerhaven.de“ unter anderem folgende Zweige:

ou=People

In diesem Zweig werden Mitarbeiterdaten festgehalten.

ou=Publications

Dieser Zweig dient der Speicherung von wissenschaftlichen Veröffentlichungen.

ou=Units

Dieser Zweig spiegelt die organisatorische Struktur des AWI wider.

Das einzige Problem ist, dass es nicht allzuviele Anwendungen gibt, mit denen man diese Daten nach bestimmten Mustern abfragen und einspielen kann. So ist der Nutzer eines solchen System bei der Entwicklung von Applikationen selbst gefordert. Zu diesem Zwecke stehen verschiedene SDK s zur Verfügung. So hat Netscape ein LDAP- SDK für Java entwickelt, welches viele Funktionalitäten zur Verfügung stellt. Dann gibt es auch noch JNDI (Java Naming and Directory Interface), welches

nicht nur LDAP Verbindungen anbietet, sondern auch Schnittstellen für NIS und weitere Dienste bietet.

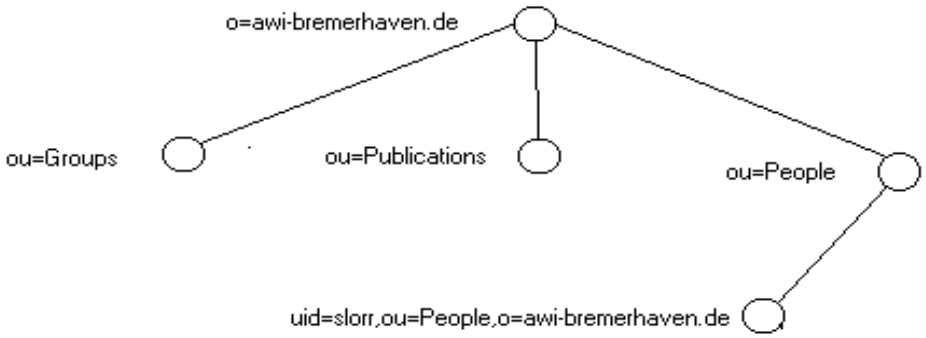


Abb. 3 Directory- Baumstruktur

Dieses Paket hat allerdings den Nachteil, das die Funktionalität nicht so speziell auf den LDAP Zugriff abgestimmt ist, und somit nicht die Funktionalität des Netscape LDAP SDK bietet. Als Perlmodul existiert Perldap, welches auch ein komfortablen Zugriff auf den Directoryserver bietet.

3.2.4.1 Eintrag (Objektklassen und Attribute)

Jeder Eintrag ist über Objektklassen und Attribute definiert. In einer Objektklasse sind die Attribute definiert, welche erforderlich sind und welche optional sind. Jeder Eintrag kann unterschiedliche Objektklassen ,und damit unterschiedliche Attribute zur Verfügung haben.

3.2.4.2 Sicherheit des Directory-Servers

In diesem Abschnitt möchte ich auf die Zugriffsrechte der Einträge und die Authentifizierung bei dem iPlanet Directory-Server eingehen.

3.2.4.2.1 Zugriffskontrollmechanismen

Die Zugriffskontrolle geschieht über sogenannte ACIs (Access Control Information). Diese ACIs entscheiden nach der Authentifikation über die Zugriffsrechte des angemeldeten Nutzers. Mir Hilfe der ACIs kann man das gesamte Directory, einzelne Äste, Einträge, oder einzelne –Attribute mit Zugriffsrechten versehen. Die Zugriffsrechte sind:

Read - Leserecht

Write – Schreibrecht (Neu, Modifikation)

Search – Ob der Eintrag gesucht werden darf

Compare – Ob der Eintrag verglichen werden darf

Selfwrite – Schreibrecht für den eigenen Eintrag

Add – Regelt die Erstellung von Einträgen

Die ACIs werden in einer ACL (Access Control List) aufgeführt. Man spricht bei der Vergabe von Rechten meist über die Gestaltung der ACL.

3.2.4.2.2 Authentifizierung

Der Directoryserver bietet 3 Möglichkeiten zu Authentifizierung:

1. Einfache, Passwortbasierte Authentifizierung
   Der DN (Distinguished Name) des Benutzer und das Passwort werden bei diesem Verfahren für gewöhnlich unnverschlüsselt an den Server versand.
2. Zertifizierte authentifikation
   Bei dieser Methode wird die Verbindung zum Directoryserver über SSL (Secure Sockets Layer) durchgeführt und der Client stellt ein Zertifikat zur Identifizierung.
3. Authentifizierung unter Benutzung von SASL Mechanismen
   SASL(Simple Authentication and Security Layer) ist ein Schema um eine Sicherheitsschicht für verbindungsbasierte Protokolle zu etablieren. In diesem Fall bei LDAP v3 die Verbindung zwischen dem Client und dem Directory Server. Dies setzt allerdings die Einbindung von Authentifizierungsmethoden externer Hersteller voraus.

Die 2. und 3. Methode benötigen eine Zertifizierungsstelle oder einen externen Authentifizierungsmechanismus. Die erste Methode ist sicher, wenn man mit SSL für eine Verschlüsselung sorgt, oder wenn man sich in einem abgeschlossenen Netzwerk befindet. [Wel2000, s.176]
Hier am AWI ist dies gegeben, da diese Anwendung nur fürs Intranet gedacht ist, welches für Zugriffe „von außen“ nicht zugänglich ist.

3.2.4.3 Das LDAP- Schema am AWI

Am AWI gibt es beim LDAP- Server mehrere Zweige. Diese nach Themen zusammengefasste Struktur nennt sich „Schema“. Die großen Datenzweige sind folgendermaßen strukturiert:
Eine Wurzel: o=awi-bremerhaven.de
Unterzweige: ou= People
ou= Publications
ou= Units
Es existieren natürlich noch weitere Zweige, welche aber nicht für die bestehenden noch für die neue Anwendung von Bedeutung sind.

4. Informationssystem am AWI

Die eben beschriebenen Komponenten bilden die Voraussetzung für das Informationssystem am AWI. Unter Nutzung dieser Basis existieren verschieden Applikationen, welche dem Nutzer verschiedenste Dienste zur Verfügung stellen. Als Plattform für die Applikationen existieren 2 Instanzen vom iPlanet Webserver; eine als Intranet, und eine als Internetpräsenz. Diese stellen wiederum Dienste für Mitarbeiter oder Außenstehende bereit.

4.1 Internet

Die Internetpräsenz besteht hauptsächlich aus Informationen rund ums AWI und beschreibt die einzelnen Forschungsgebiete. Hier existierten auch Telefonlisten und Publikationslisten für die einzelnen Fachbereiche, Sektionen und Projektgruppen. Diese Listen werden dynamisch über CGI-Skripts erstellt.

4.2 Intranet

Das Intranet beschäftigt sich hauptsächlich mit Informationen von und für Mitarbeiter. Hier existieren Veranstaltungskalender, Schwarze Bretter und ähnliche allgemeine Informationsmittel. Seit kurzem werden auch verstärkt Applikationen entwickelt. So existiert beispielsweise eine Anwendung zur Erstellung persönlicher Homepages für Benutzer, auf denen bestimmte Daten zur Verfügung gestellt werden. Diese Homepages werden dynamisch von einer Anwendung erstellt, welche Ihre Daten wiederum aus dem Directoryserver bezieht. Auf diesem Weg werden auch Telefonlisten erzeugt. Die Wartung der Daten wird dem Benutzer überlassen, da dies einen erheblichen Aufwand für eine Abteilung darstellen würde, und da jeder Nutzer auch Interesse an der Korrektheit seiner Daten hat.

Eine Besonderheit an diesem System ist, dass es unter niedriger „Manpower“ erstellt worden ist. Es existiert eine Webmasterin, während es

an anderen Instituten gleicher Größe üblich ist diese Stelle mit 2-4 Personen zu besetzen, was zur folge hatte, dass am AWI verstärkt zu Automatisierungen in Form von Anwendungen gegriffen wurde. Die Anwendungen wurden größtenteils mit Hilfe von studentischen Hilfskräften erstellt, und von der Webmasterin verwaltet und gewartet. Um dies zu unterhalten muss natürlich eine gute Dokumentation und eine Einheitlichkeit in der Programmierung gewährleistet sein. Ein erster Schritt in diese Richtung war die Diplomarbeit eines Studenten, welcher die persönliche Komponente, und damit den ersten Schritt zu einem einheitlichen Rahmenwerk entwickelt hat. So hat er Basisklassen erstellt, welche sich auch für meine Arbeit wiederverwenden und weiterentwickeln lassen.

4.3 Das bestehende Publikationssystem

Das bestehende System ist in der Programmiersprache „Perl “ implementiert. Das Publikationsprojekt wurde vor ca. 1,5 Jahren begonnen und die entsprechende Software wurde stetig weiterentwickelt. Es hat als Prototyp begonnen, und wurde ständig um Funktionalitäten erweitert.

Die Software wurde als Folge einer Datenbankumstellung von Sybase auf Directory-Server benötigt. Die Sybase Datenbestände wurden als Tabulatorseparierte Zeilen einer Textdatei exportiert, und dann mit Hilfe eines Perlskriptes in den Directoryserver importiert. Doch was fehlte war eine Anwendung, mit der man sich diese Einträge anschauen konnte. Dies war der Auftakt für das Publikationssystem. Was bald darauf folgte war der Bedarf, weitere Publikationen hinzuzufügen, bestehende Publikationen zu ändern, bzw. zu löschen. Auch diese Funktionalitäten wurden ständig erweitert und verbessert. Die Funktionalitäten brachten immer weitere Anforderungen mit sich, und so wurde der ursprüngliche Prototyp immer komplexer und komplizierter.

Die ursprüngliche Hauptseite war dementsprechend auch eine Suchseite, der immer weitere Optionen hinzugefügt worden sind. Es hat keine Re-Design gegeben, sondern eine ständige Erweiterung.
So sind die meisten Funktionalitäten fest mit der Suche an sich verknüpft. Direkt vom Suchergebnis gelangt man unabhängig von der Rolle zur Modifikationsseite, und ist so in der Lage auch Einträge zu bearbeiten, mit denen man nichts zu tun hat. Von der Detailansicht aus kann man Löschen. Das Löschen ist aber nur mit entsprechender Autorisierung möglich. Dies geschieht über eine Im Directoryserver erstellte Gruppe namens *Publicationgroup*. In dieser Gruppe sind alle Personen eingetragen, welche Für das Löschen autorisiert sind. Dies sind im Speziellen Bibliothekare und Publikationsbeauftragte

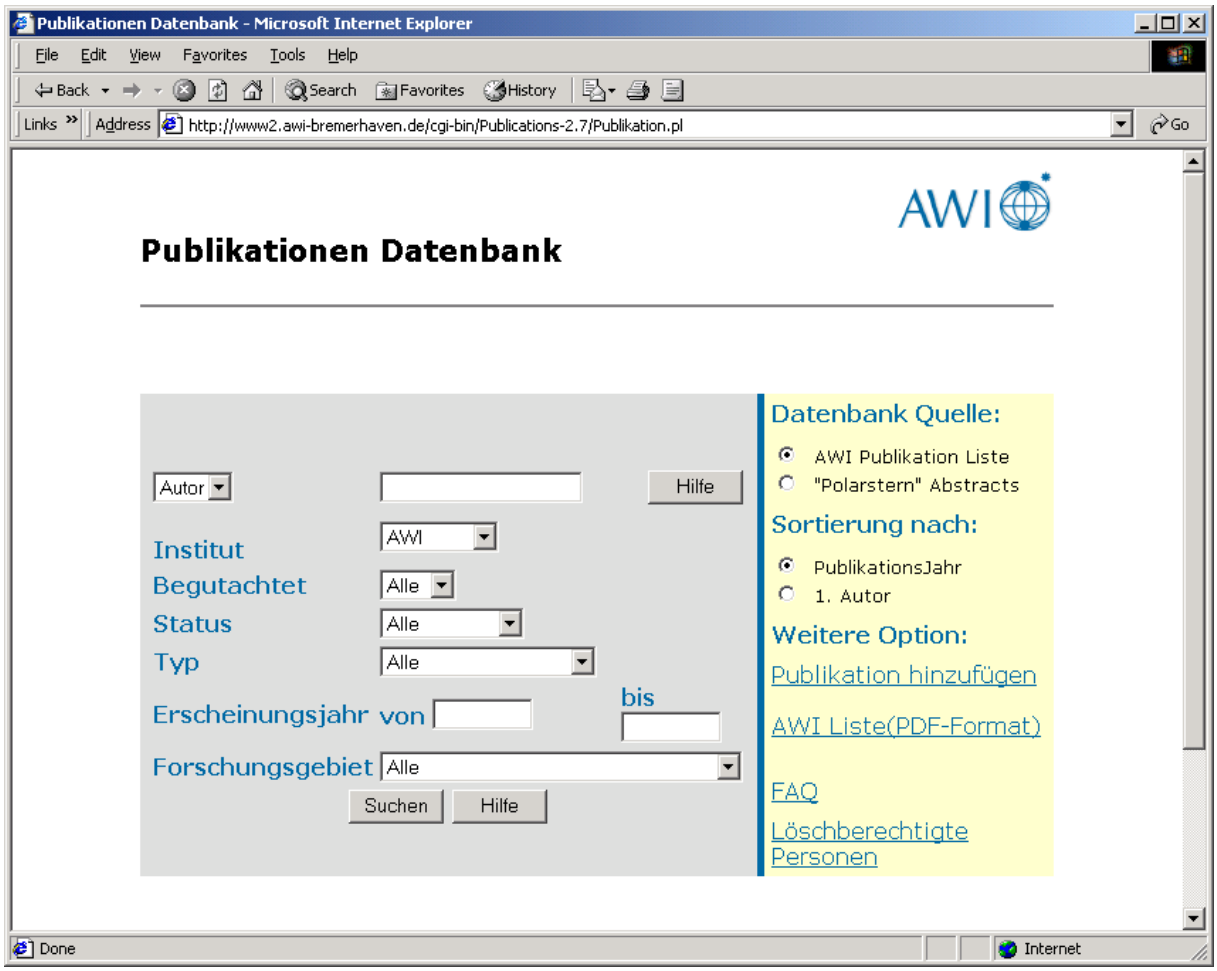


Abb. 4 Hauptmaske des bestehenden Systems

Ein weiterer Nachteil ist, dass durch ständiges „hinzuskripten“ der Code unübersichtlich, und nicht mehr zugänglich für weitere Modifikationen

wurde. Es ist kein sauber konzeptioniertes Programm, sondern ein ständig erweiterter Prototyp. Die Änderungen und der Anforderungen und Funktionalitäten wurde auch nicht dokumentiert, so dass die Funktionalitätsbeschreibung der Code selber ist. Und da Das Programm mittlerweile sehr Umfangreich geworden ist, besteht der Bedarf nach einer sauberen Konzeption, und Beschreibung der Anforderungen und Funktionalitäten, damit man ein Modell hat, welches erweiterbar ist.

Auch wurde der Ruf nach weiteren Anforderungen laut, die mit der derzeitigen Implementierung nicht mehr zu erfüllen waren, wie z.B. applikationsübergreifende Authentifikation und Internationalisierung.
Die Diplomarbeit von K. Stelling zeigte, dass diese Funktionalitäten mit Java umsetzbar sind. Es ist jedoch sehr aufwendig eine Applikation, die nicht dokumentiert ist, von der Pieke auf neu zu konzeptionieren, dokumentieren und zu Implementieren. Und dies ist das Thema dieser Arbeit.

5. Publikation

An dieser Stelle wird nun ausführlich auf die Publikation und deren Abspeicherung eingegangen. Wie schon erwähnt, ist jede Publikation im Directoryserver abgespeichert, und zwar als Zusammenstellung von Objektklassen und Attributen.

5.1 Objektklassen und Attribute

Es existieren Objektklassen, welche bestimmte Attribute fordern und bestimmte Attribute zulassen. Im Folgenden möchte ich auf diese Objektklassen und die dazugehörigen Attribute genauer eingehen. Hier stelle ich nun die für den Publikationseintrag bedeutsamen Objektklassen und die Attribute mit ihren Bedeutungen vor:

Objectclass top

Superior top

requires

- objectclass
  dient zur Definition von Objektklassen

allows

- aci
  Access Control Information

Objectclass person

Superior top

Requires

- sn
  Nachname
- cn
  Vor- und Zuname

Allows

- description
- seeAlso
  Verweis auf eine Internetadresse einer AWI- Seite des Arbeitsbereiches des Nutzers

- telephoneNumber
  Telefonnummer
- userPassword
  verschlüsseltes Passwort

Objectclass organizationalPerson

Superior person

Requires

Allows

- destinationIndicator
- facsimileTelephoneNumber
  Faxnummer
- internationalSDNNumber
- l
  (Kurzform für engl. „Location“) Ort
- ou
  Organisationseinheit (organizational Unit)
- physicalDeliveryOfficeName
- postOfficeBox
  Postfach
- postalAdress
  Anschrift
- postalCode
  Postleitzahl, oder ZIP- Code
- preferredDeliveryMethod
- registeredAddress
- st
- street
- teletexTerminalIdentifier
- telexnumber
- title
  persönlicher Titel (Prof., Dr.)

- x121Adress

Objectclass publicationclass

Superior top

Allows

- changetype
  - Keine Bedeutung -
- createtimestamp
  Zeitpunkt der Erstellung der Publikation
- creatorsname
  Der Ersteller wird anhand seiner uid festgehalten, aufgrund derer er eindeutig zu identifizieren ist
- modifiersname
  Hier wird derjenige festgehalten, welcher zuletzt eine Änderung am Eintrag vorgenommen hat.
- modifytimestamp
  Zeitpunkt der letzten Änderung
- ownersname
  Eigner des Eintrages; meist der Ersteller
- publicationauthor
  Der 1. Autor der Publikation. Die Konvention zum Eintragen ist <Nachname>, <erster Buchstabe des Vornamen>. Ist er auch Editor, muss durch die entsprechende Applikation ein (ed.) and den Autornamen angehängt werde.
- publicationawi
  Gibt an, ob eine Publikation am AWI erstellt worden ist. Gültige Werte sind „Yes“ und „No“.
- publicationbegutachtet
  gibt an ob eine Publikation „begutachtet“ worden ist. Gültige Werte sind „Yes“ und „No“.
- publicationcitation
  Hier wird die Zitation abgespeichert. Sie hat die Form:

„<Autor>,<1. Co- Autor>,<n. Co- Autor>,(<Publikationsjahr>). <Titel>. <Quelle>.“.

- publicationcoauthor
  In diesem Attribut werden die Co- Autoren abgespeichert. Da dieses Attribut „mehrwertig“ ist, muss kein spezieller Separator eingeführt werden. Die Konventionen sind ansonsten die gleichen, wie beim 1. Autor. Diese Attribut wird allerdings momentan nicht genutzt. Statt dessen wird für Co- Autoren eine Abwandlung des Attributes *publicationauthor* benutzt: *publicationauthor;alternate*.
- publicationdataset
  Enthält einen Link auf die Datensätze, die für die Publikation benutzt worden sind.
- publicationemail
  Kontakt- Emailadresse für den Verantwortlichen der Publikation. Meist wird dieses Feld von der Emailadresse des Eintragenden belegt.
- publicationidentifier
  Hier wird eine ISBN oder ISSN eingetragen, anhand derer die Publikation, oder ihre Quelle identifizierbar ist.
- publicationlock
  Wird benutzt um zu überprüfen, ob die Publikation von der Bibliothek auf bibliographische Richtigkeit überprüft worden ist. Ist dieser Wert einmal belegt, können die bibliographischen Daten nicht mehr geändert werden
- publicationlockedby
  Enthält die uid desjenigen, welcher die Publikation für bibliographische Änderungen gesperrt hat.
- publicationpsabstract
  Wert wird auf „*yes*“ gesetzt , wenn die Publikation in der Reihe „Polarstern Abstracts “ erschienen ist.
- publicationpscruise
  die „Cruse“ (Fahrt der Polarstern) auf der die Publikation erstellt worden ist

- publicationpsleg
  der Fahrtabschnitt, auf dem die Publikation erstellt worden ist
- publicationsource
  Quelle, in der die Publikation erschienen ist
- publicationstatus
  Augenblicklicher Status der Publikation (Einreichung beantragt, eingereicht, im Druck, veröffentlicht)
- publicationtag
  Fachbereich, dem die Publikation zugeordnet ist. Hier sind auch mehrere Fachbereiche zulässig (Interdisziplinäre Arbeiten)
- publicationtitle
  Der Titel der Publikation
- publicationtype
  der Typ der Publikation: Artikel, Kapitel in einem Buch, Buch Konferenzbeitrag, Manual, Patent.
  Falls es sich bei der Publikation um eine Diplomarbeit, Dissertation oder Habilitation handelt, ist der Typ wie folgt: Diplomarbeit, Dissertation, Habilitation. Die ausstellende Hochschule wird dann im Feld „*thesisacceptedby*“ festgehalten.
- publicationurl
  Url zum vollen Text der Publikation
- publicationyear
  Jahr der Veröffentlichung
- thesisacceptedby
  Falls es sich bei der Publikation um eine Diplomarbeit, Dissertation oder Habilitation handelt, wird die ausstellende Hochschule dann in diesem Feld festgehalten.
- uid
  Dieses Attribut macht eine eindeutige Identifizierung möglich. Es setzt sich zusammen aus den ersten drei Buchstaben des Nachnamens vom 1. Autor, dem Publikationsjahr, und einem fortlaufenden Index. Bsp.: Lor1991a

Hierbei ist zu beachten, dass die Objektklassen „*Person*“ und „*organizationalPerson*“ nicht indizieren sollen, dass es sich bei einer Publikation um eine Person handelt. Es wurden bei der Konzeptionierung des Schemas einige Attribute aus diesen Objektklassen benötigt.

6. Die verschiedenen Stationen einer Publikation

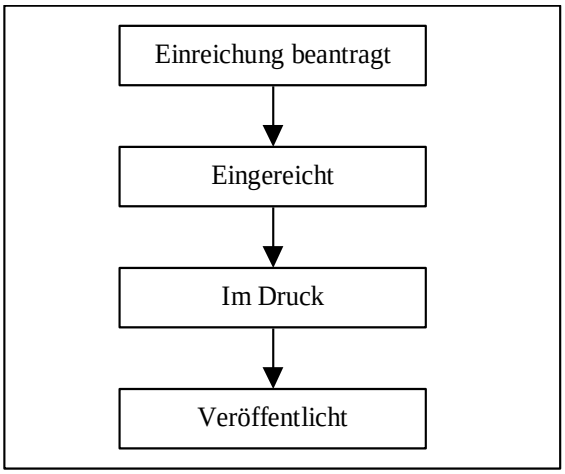


Abb. 5 Verschiedene Stati einer Publikation

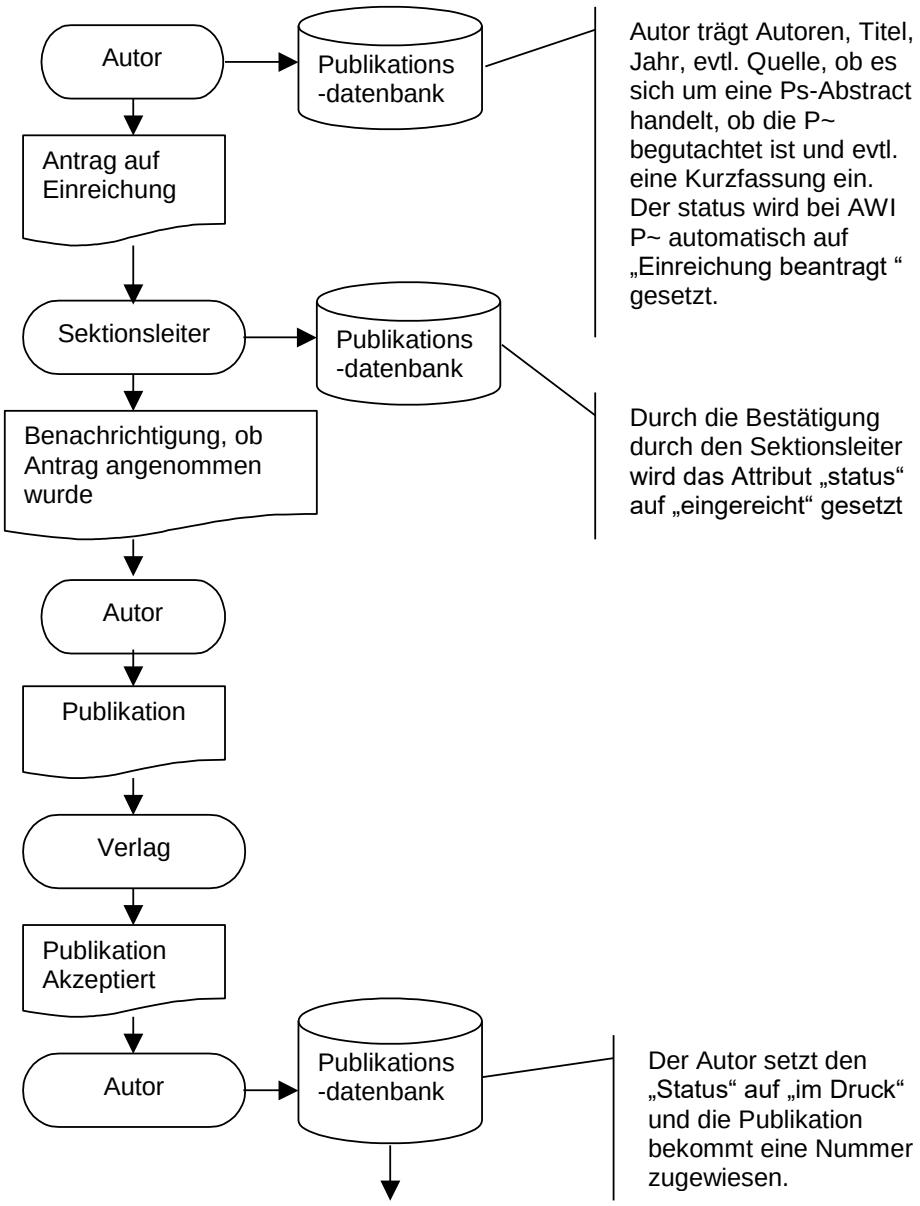
Bei Ihrer Erstellung durchwandert die Veröffentlichung mehrere Stationen. Dies momentane Status, sozusagen ihr Status, wird in einem Attribut im Directoryserver festgehalten. Es existieren 4 mögliche Werte:

- Einreichung beantragt
  Der Autor hat eine Vorstellung von einer Publikation und trägt diese ein. Es wird automatisch eine Nachricht an seinen Sektionsleiter gesendet, welcher dieser Einreichung zustimmen muss.
- Eingereicht
  Der Sektionsleiter erhält eine Nachricht, und muss dementsprechend reagieren; er kann die Einreichung akzeptieren, und damit den Status auf „eingereicht“ setzen, oder er kann die Einreichung mit einer Begründung ablehnen. Dies wird dem Autor dann per Mail zugesendet
- Im Druck
  Diesen Status muss wieder der Autor persönlich setzen und sich auch authentifizieren. Er bekommt die Bestätigung dass seine Veröffentlichung vom entsprechenden Verlag bestätigt worden ist, und trägt daraufhin die Änderung ein. Parallel muss er natürlich die Bestätigung des Verlages dem Fachbereichsleiter vorlegen. In diesem Stadium wird der Publikation eine AWI- interne Publikationsnummer automatisch zugeteilt.

- Veröffentlicht

  Dieses Attribut wird auch vom Autor gesetzt, welcher sich dafür wieder authentifizieren muss. Er bekommt die Bestätigung vom Verlag das seine Veröffentlichung fertig gedruckt, und damit veröffentlicht ist. Sobald das Attribut geändert wird, erhält die Bibliothek eine Mail, und kann die Veröffentlichung ein letztes Mal auf ihre bibliographische Richtigkeit überprüfen. Sobald dies abgeschlossen ist, werden die bibliographischen Daten schreibgeschützt.

Die Entstehung einer Publikation wird durch folgende Zeichnung nach DIN66001 verdeutlicht:



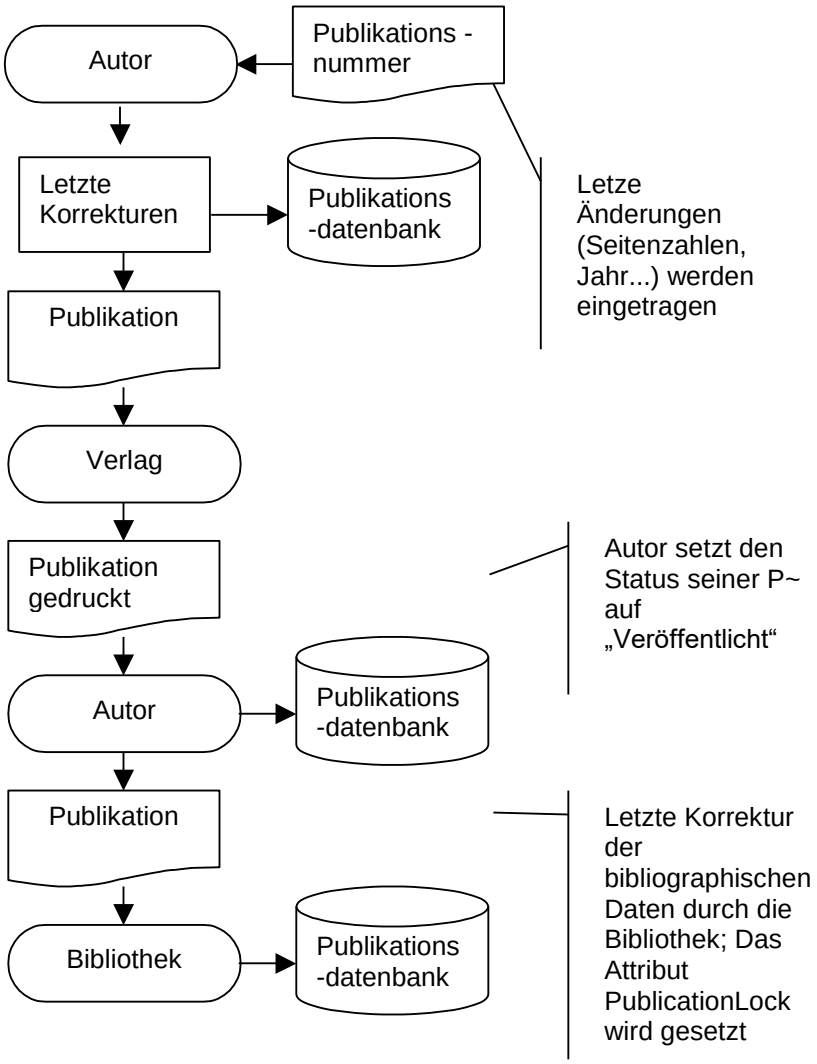


Abb. 6 Ablaufplan für die Veröffentlichung einer Publikation

7. Anforderungen an das Publikationssystem

Wie schon erwähnt existiert bereits ein Prototyp, welcher durch seine stetige Weiterentwicklung die Anforderungen an das Publikationssystem sichtbar gemacht hat. Dieser Prototyp ist allerdings nicht sehr sauber programmiert, und damit an weitere Anforderungen nicht mehr anpassbar. Die Idee des Publikationssystems ist es, Veröffentlichungen auf elektronische Art und Weise von statten gehen zu lassen. Dies erfordert eine Datenbasis zum strukturiertem Abspeichern von Daten, sowie Applikationen, welche die Instandhaltung und Ausdehnung dieses Datenbestandes gewährleisten. Als Datenbasis dient in diesem Fall der Directoryserver von iPlantet. Ausserdem muss eine Ausgabemöglichkeit zum Betrachten gewünschter Daten erfolgen.
Folgende Funktionalitäten soll das Publikationssystem erfüllen:
Suche von Einträgen
Korrigieren von Einträgen
Löschen von Einträgen
Neu Erstellung von Einträgen
Diese Funktionalitäten sind natürlich an dieser Stelle sehr abstrakt dargestellt. Es wird an späterer Stelle explizit darauf eingegangen, Wie die Suche auszusehen hat, wer was korrigieren und löschen darf, und was für verschieden Möglichkeiten zur Neueintragung von Publikationen umzusetzen sind.

7.1 Erweiterbarkeit

Nachteilig war an der bisherigen Version, dass es einfach aneinandergereihter Code war, welcher wenig modularisiert, und kaum objektorientiert war. Somit entstanden Schwierigkeiten neue Funktionalitäten zu implementieren, und bei Veränderungen von bspw. Attributnamen, musste es an mehreren Stellen geändert werden, was natürlich Fehleranfälliger ist.

7.2 Verständlichkeit

Da die Ressource „Manpower“ am AWI sehr knapp ist, muss der Code nicht nur sauber strukturiert, sondern auch leicht verständlich sein, damit Mitarbeiter sich schnell in den Code einarbeiten können und wenig Zeit für Korrekturen, und eventuelle Umstrukturierung brauchen.

7.3 Funktionalität

Es müssen bestimmte Funktionalitäten erfüllt werden. So muss in erster Linie der Workflow „Neueintragung einer AWI- Publikation“, auf den im Kapitel „Publikation“ genauer eingegangen wird, umgesetzt werden. Dies setzt eine genaue Unterscheidung von Personen nach Ihren Rollen voraus, da der Ablauf über mehrere Personen gehen muss, um einen „sauberen “ Datenbestand zu garantieren
Auch eine Suchfunktion ist zu implementieren, so dass der Datenbestand nach verschiedenen Kriterien zu durchsuchen ist, und die Ausgabe abgespeichert werden kann.

7.4 Benutzer

Sobald mehr als ein Nutzer auf eine Datenbasis zugreifen muss eine Zuteilung von Rechten für Zugriffe auf die Datenbasis erfolgen. Dies führt dazu, dass sich ein Nutzer vor der Ausführung von bestimmten Aktionen authentifizieren muss. Jeder Nutzer nimmt bei Prozessen des Publikationssystems eine bestimmte Rollen an welche bestimmte Rechte hat.

7.4.1 Anonymus

Der Anonyme Benutzer hat Leserechte auf Publikationen. Er braucht sich nicht einzuloggen, da er eh nichts verändern kann, und es nicht wichtig ist, wer wann welche Daten abgerufen hat.

7.4.2 Autor

Der Autor muss sich authentifizieren. Er hat Schreibrechte , und ist berechtigt, Veröffentlichungen neu einzutragen und zu modifizieren

7.4.3 Fachbereichs/Sektionsleiter

Auch der verantwortliche Fachbereichs, oder Sektionsleiter muss sich authentifizieren. Es obliegt seiner Verantwortung Anträge auf Publikationen entgegenzunehmen

7.4.4 Bibliothekar

Der Bibliothekar hat die Aufgabe die Veröffentlichung auf Bibliographische Richtigkeit zu überprüfen. So muss er Schreibrechte auf bestimmte Attribute erhalten, und sich folglich auch authentifizieren.

7.4.5 Publikationsbeauftragter

Ein Publikationsbeauftragter wird in jeder Sektion für diese Sektion festgelegt. Er darf Einträge sehen, verändern und löschen. Dazu muss er sich auch authentifizieren.

8. Pflichtenheft

Das Pflichtenheft hat die Aufgabe dien Umfang des Projektes zu erfassen, und dabei möglichst Abstrakt, d.h. einfach und nachvollziehbar zu sein. Es kommt nicht darauf an jede Funktionalität bis ins kleinste Detail zu erfassen, sondern das Projekt überschaubar zu machen [Ba1999].

Pflichtenheft

1. Zielbestimmung

1.1 Muss- Kriterien

- Internationalisierung
- Implementierung des Workflows „Neue AWI- Publikation“
- Suche
- Korrektur
- Löschen
- Exportieren

1.2 Kann- Kriterien

- Endnote- Format beim Exportieren
- Dateikonvertierung beim Hochladen von Publikations- Volltexten

1.3 Abgrenzungskriterien

-

2. Einsatz

2.1 Anwendungsbereiche

- Intranet
- Internet

2.2 Zielgruppen

- Wissenschaftler (Autoren)
- Externe Betrachter/ Interessierte
- Publikationsbeauftragte
- Fachbereichs/Sektionsleiter
- Bibliothek

2.3 Betriebsbedingungen

- Eher seltene Nutzung durch Wissenschaftler
- Tägliche Nutzung durch Bibliothek
- Eher seltene Nutzung der Suchfunktion

3. Umgebung

3.1 Software

- Betriebssystem: SUN Solaris 8.0
- iPlanet Directory Server 4.1
- iPlanet Web Server 4.1 sp 7
- Java JDK 1.2.2_008

3.2 Hardware

- Server Plattform: SUN Enterprise 10k
- Client Plattform: Browser

3.3 Orgware

- LDAP- Schema
- Benutzer-, Organisations-, Publikationsdaten im Directory- Server

4. Funktionalität

Der aufwendigste Arbeitsablauf in diesem System ist die Neueintragung von Publikationen. In diesem Ablauf wird die Publikation nach verschiedenen Bedingungen modifiziert oder gelöscht.
Neben dieser Verwaltungsfunktion existieren noch separat die Möglichkeiten zu löschen oder zu modifizieren.
Die Suche geschieht nach der Auswahl verschiedener Kriterien, welche wiederum in den Attributen eines Publikationseintrages wiederzufinden sind (Autor, Titel, ...). Das Suchergebnis wird in einer Liste dargestellt. Man hat dann die Möglichkeit, einzelne Einträge anzuwählen und zu exportieren, bzw. einzelne Einträge in detaillierterer Form zu betrachten. Hier bietet sich dann auch die Möglichkeit zur Korrektur der Einträge.

5. Daten

Es werden alle Publikationen der am AWI beschäftigten Mitarbeiter erfasst; momentan: 4500-5000

Jährlicher Zuwachs: ca. 1000+

6. Leistungen

Die Suche soll möglichst performant sein, d.h. schnell Ergebnisse liefern; jeder Nutzer ist ungeduldig und erwartet das Ergebnis im „Nullkommanix“ auf seinem Bildschirm.

7. Benutzeroberfläche

Die Benutzeroberfläche wird als Webapplikation auf einem Browser dargestellt.

Da die Benutzung eher selten ist muss die Benutzerführung möglichst einfach und selbsterklärend sein. So sollten zusätzlich zu „Hilfe- Buttons“ erklärende Texte integriert werden. Außerdem muss die Oberfläche rollenspezifisch sein, und folglich für verschiedene Benutzergruppen anders aussehen.

8. Qualitätsziele

- Ausgabe muss auf verschiedenen Plattformen darstellbar sein
- Hohe einfache Änderbarkeit des Codes
- (Hohe Verfügbarkeit der Anwendung)
- (Korrektheit der Daten)
- (Hohe Sicherheit bei der Authentifikation)

9. Ergänzungen

-

9. OOA

Bei der OOA (Object Oriented Analysis) geht es darum, die Prozesse des Systems objektorientiert darzustellen.
Ich benutze dafür die Formalisierte Struktur nach [Ba1999] , nach der vorgegeben ist, eine Beschreibung des Prozesses, eine Klassendiagramm der involvierten Klassen und ein Sequenzdiagramm zu erstellen.
An dieser Stelle tauchen nicht unbedingt alle Klassen des Softwaresystems auf, sondern nur die, die auch direkt betroffen sind, denn es existieren auch Klassen die nur zur Steuerung der Software dienen. Deren Beschreibung findet im Kapitel „Umsetzung “ statt.
Die Authentifizierung wird an dieser Stelle auch erst einmal außen vor gelassen, um die Darstellung der Abläufe nicht unnötig zu verkomplizieren.
Die datenbankzugriff-spezifischen Klassen werden auch nur im Prozess „Neueintragen einer Publikation“ erwähnt, da sie an der Stelle das Verständnis fördern.

Die Benutzertypen tauchen in den Klassendiagrammen als Klassen auf, sind aber nicht explizit umgesetzt. Sie sind dennoch vom System eindeutig, durch verschiedene Attribute und Gruppen im Directoryserver, unterscheidbar. In den Sequenzdiagrammen tauchen sie dann als Akteure auf.
Auf das Publikationssystem wird von 6 verschiedenen Benutzertypen zugegriffen, die in folgenden Grafiken als Akteure dargestellt sind:

- Autor
- Publikationsbeauftragter
- Sektions-/ Fachbereichsleiter
- Sachbearbeiter Bibliothek
- Interner Betrachter (Intranet)
- Externer Betrachter (Internet)

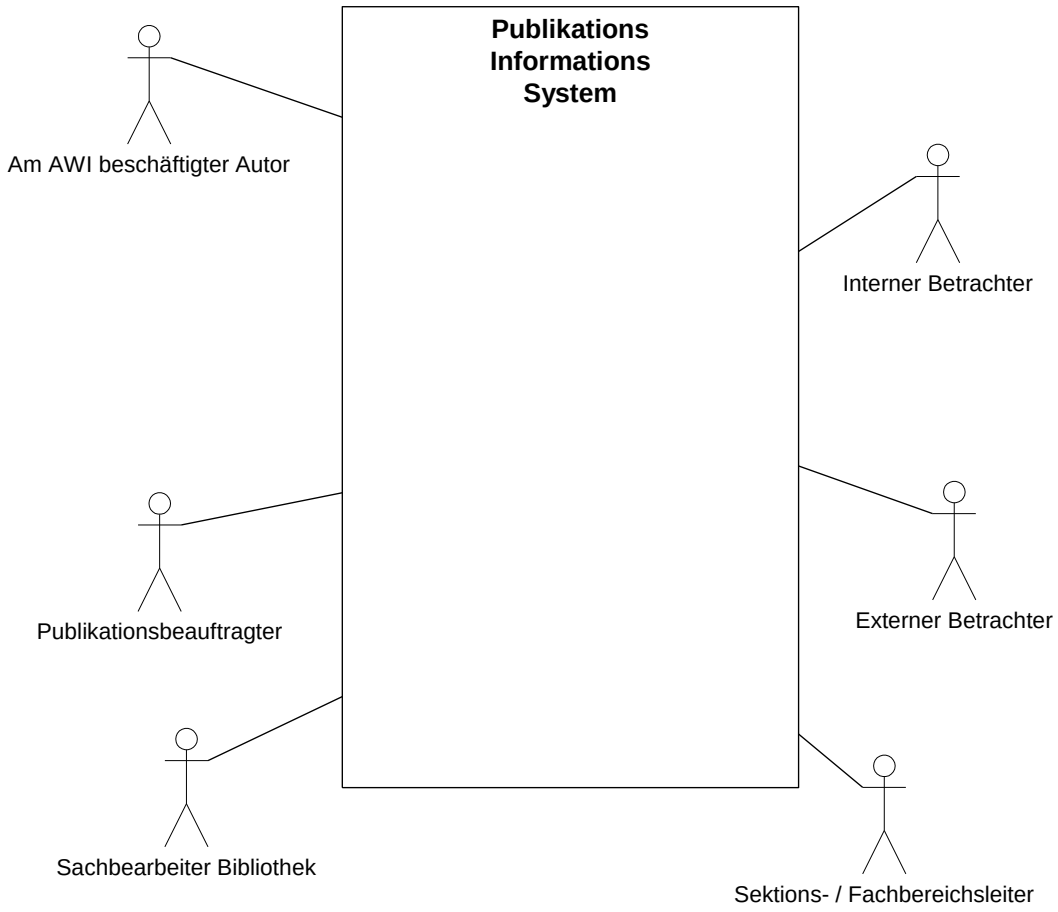


Abb. 7 Die Akteure beim Publikationsinformationssystem

Es existieren 4 grundlegende Geschäftsprozesse, auf die von den Benutzern zugegriffen wird. Man kann hierbei die internen und externen Betrachter als einen Akteur zusammenfassen, sowie den AWI- und den Nicht- AWI- Autoren. Der Sektions- / Fachbereichsleiter, der Sachbearbeiter Bibliothek dürfen zwar auch die Prozesse Löschen und Modifizieren anstoßen, allerdings nur im Zusammenhang mit dem Neueintragen einer Publikation. Somit führt keine Linie zwischen Ihnen und den Prozessen. Dasselbe gilt für den Autor und „Modifizieren“.

Die häufig auftauchende Operation „displayHtml()“ hat die Aufgabe eine Eingabemaske zur Verfügung zu stellen. Die Operation „performAction() “ verarbeitet die Eingaben, und gibt eine Meldung zurück, ob die Aktion erfolgreich war, fehlgeschlagen ist, oder ob man seine Eingaben noch

einmal überprüfen muss. Ist letzteres der Fall, wird die Option gegeben wieder zur Eingabe zurückzukehren.

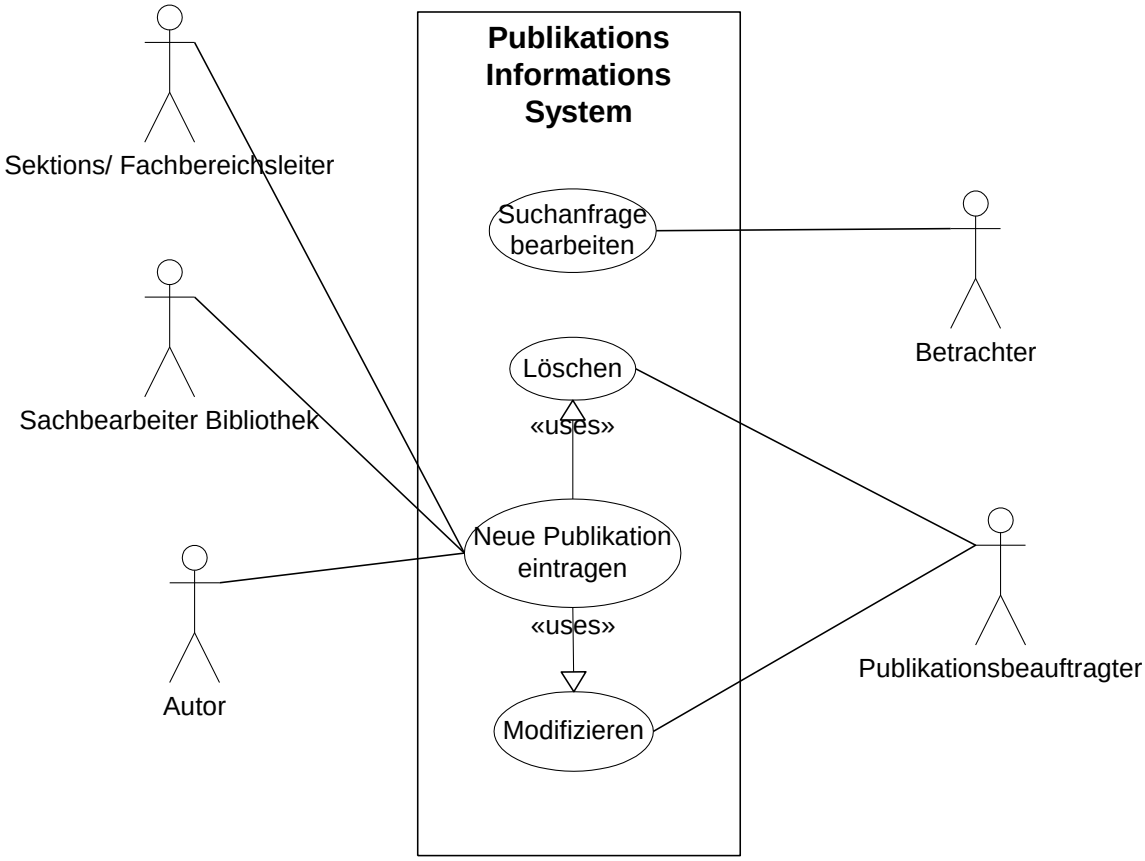


Abb. 8 Die 4 Geschäftsprozesse

9.1 Geschäftsprozesse

Geschäftsprozess: Suchanfrage bearbeiten
Ziel: Bereitstellung von Publikationsdaten
Vorbedingung:
Nachbedingung Erfolg: Ausgabe vom Suchergebnis; Möglichkeit zum Exportieren und Detailansicht von Einträgen
Nachbedingung Fehlschlag: kein passender Eintrag
Akteure: Betrachter
Auslösendes Ereignis: Betrachter stellt Suchanfrage
Beschreibung:

1. Suchkriterium eingeben
2. Suchfilter erstellen

3. Suche durchführen
4. Ergebnis darstellen

Erweiterungen:

4a Detailansicht eines Eintrages

4b Exportieren ausgewählter Einträge

Alternativen:

1a Ungültige Eingaben abfangen

In folgender Graphik wird der „Weg“ des Betrachters über die verschieden Uis (Benutzerschnittstellen)verdeutlicht. Auf die Vererbungsstrukturen der Klassen wird später genauer eingegangen.

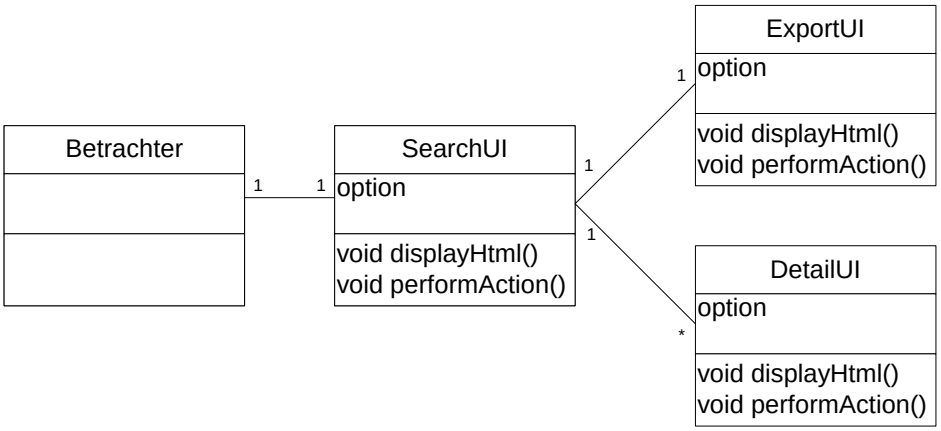


Abb. 9 Klassendiagramm „Suche“

In folgendem Sequenzdiagramm wird die Abfolge der Schritte des Ablaufes der Suche beschrieben.

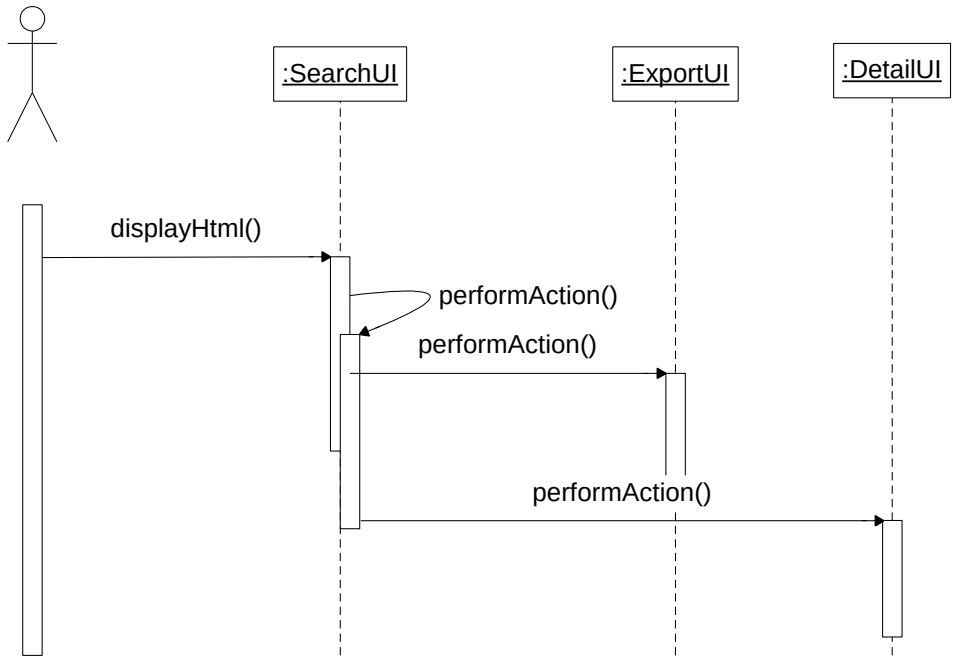


Abb. 10 Sequenzdiagramm „Suche“

Geschäftsprozess: Publikation Löschen

Ziel: Verwaltung von Publikationen

Vorbedingung: Datensatz existiert

Nachbedingung Erfolg: Datensatz wurde gelöscht

Nachbedingung Fehlschlag:

Akteure: Publikationsbeauftragter

Auslösendes Ereignis: Publikationsbeauftragter bekommt den Auftrag eine Publikation zu löschen

Beschreibung:

1. Suchanfrage
2. Authentifizieren
3. Löschen

Erweiterungen:

-

Alternativen:

1a Eintrag existiert nicht

2a Publikationsbeauftragter ist bereits Authentifiziert => authentifizierung entfällt

In diesem Klassendiagramm taucht die „löschberechtigte Person“ auf. Außer dem Publikationsbeauftragtem steh dieses Recht noch dem Bibliothekar zu.

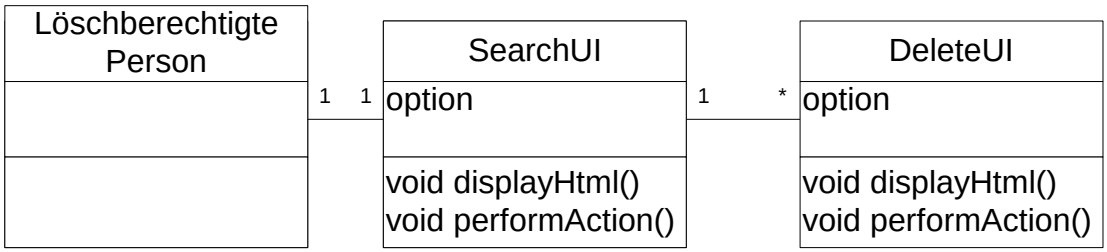


Abb. 11 Klassendiagramm „Löschen“

Das Sequenzdiagramm zeigt, dass der eigentlichen Löschung eine Suche vorangeht, deren Ergebnis dann null oder mehr Löschungen erlaubt.

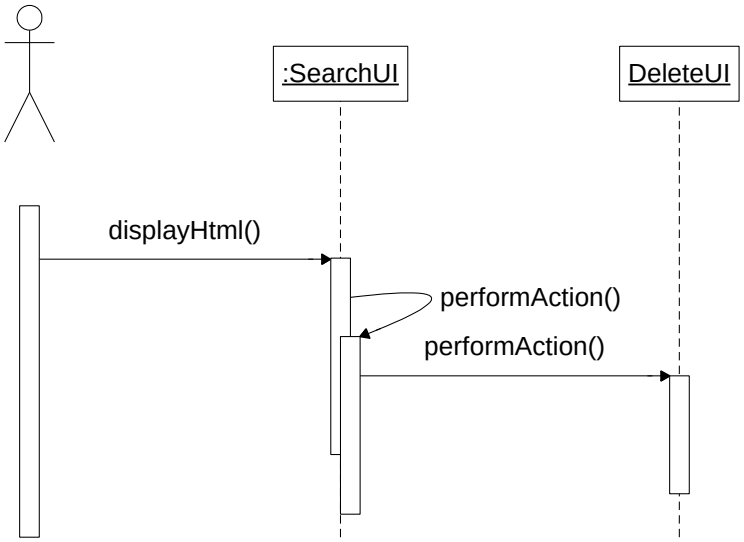


Abb. 12 Sequenzdiagramm „Löschen“

Geschäftsprozess: Neue AWI-Publikation eintragen

Ziel: Kontrollierte Eingabe von Publikationen

Vorbedingung: -

Nachbedingung Erfolg: Publikation veröffentlicht und in Datenbank eingetragen

Nachbedingung Fehlschlag: Publikation nicht veröffentlicht und in Datenbank eingetragen

Akteure: Autor, Sektions- /Fachbereichsleiter, Sachbearbeiter Bibliothek

Auslösendes Ereignis: Autor hat eine Idee für eine Publikation

Beschreibung:

1. Autor stellt einen Antrag auf Einreichung einer Publikation bei seinem Sektionsleiter
2. Sektionsleiter gibt Antrag statt
3. Autor stellt Publikation fertig und gibt sie an einen Verlag zum Drucken
4. der Verlag prüft die Publikation
5. der Verlag druckt die Publikation
6. Finaler Check des Datenbankeintrages der Publikation durch die Bibliothek

Erweiterungen:

Alternativen:

1a. Statt Sektionsleiter: stellvertretender Sektionsleiter, Fachbereichsleiter, stellvertretender Fachbereichsleiter

2a. Antrag wird abgelehnt => überarbeiten

5a. Verlag schickt die Publikation zurück

Die Zugriffe der verschieden Klassen aufeinander werden durch folgendes Klassendiagramm verdeutlicht:

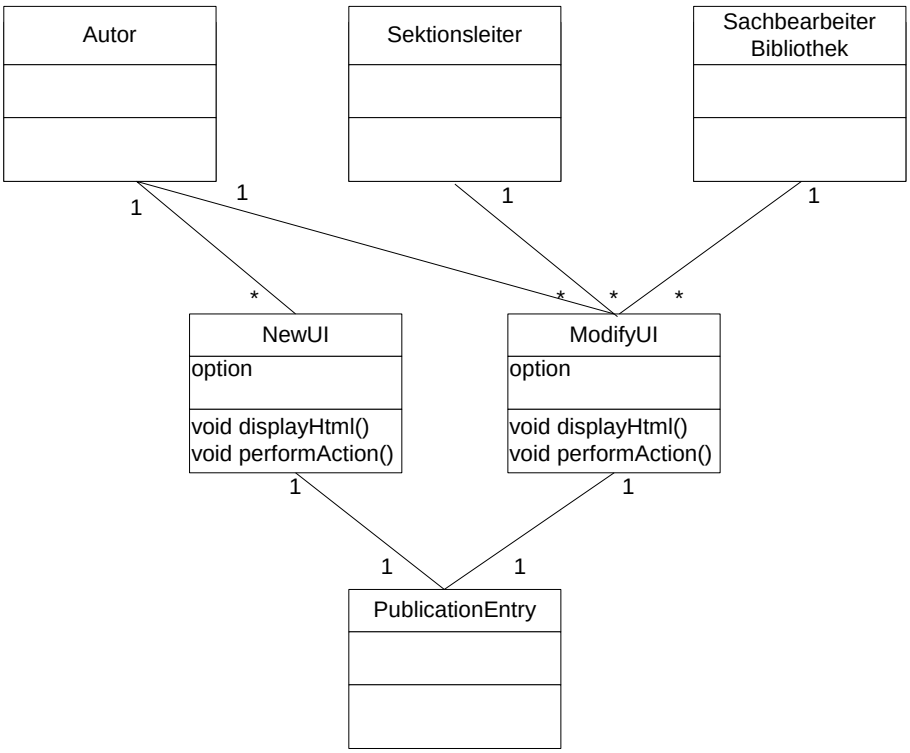


Abb. 13 Klassendiagramm Neueintragen einer Publikation

Durch das Sequenzdiagramm wird deutlich wer wann welches Objekt anstößt.

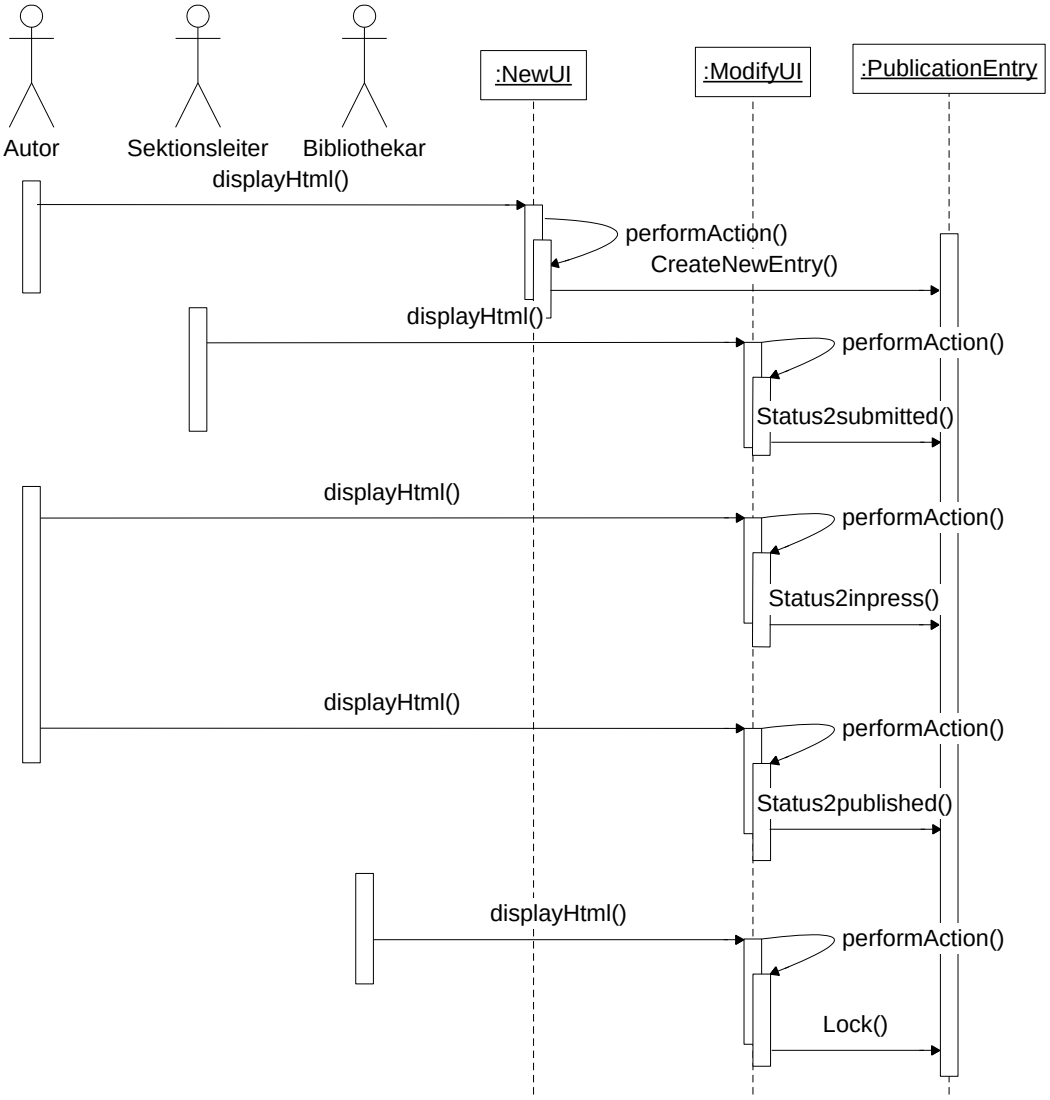


Abb 14. Sequenzdiagramm ”Neueintragung einer Publikation”

Geschäftsprozess: Publikation Modifizieren

Ziel: Verwaltung von Publikationen: Modifizieren

Vorbedingung: Datensatz existiert

Nachbedingung Erfolg: Datensatz wurde geändert

Nachbedingung Fehlschlag: Datensatz wurde nicht geändert

Akteure: Autor, Prozess „Neue AWI-Publikation“

Auslösendes Ereignis: Datensatz ist nicht aktuell

Beschreibung:

1. Suchanfrage
2. Authentifizieren
3. Korrektur

Erweiterungen:

1a. Detailansicht

Alternativen:

2a. Benutzer ist bereits authentifiziert

1a. Eintrag existiert nicht

Der Modifizierungsberechtigte kann in diesem Fall der Autor, der Publikationsbeauftragte, der Sektions-/ Fachbereichsleiter oder der Sachbearbeiter Bibliothek sein. Die letzten 3 können im Zuge der Neueintragung einer Publikation die Publikation modifizieren, und werden so in der Beschreibung des Prozesses nicht explizit erwähnt.

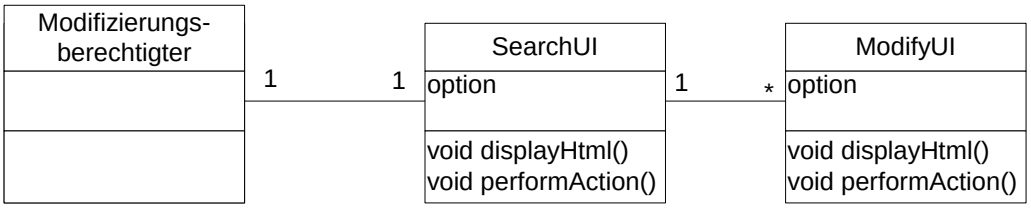


Abb. 15 Klassendiagramm Publikation Modifizieren

Das Sequenzdiagramm zeigt, dass dem Modifizieren eine Suche vorausgeht, deren Ergebnis dann die Modifikation ermöglicht.

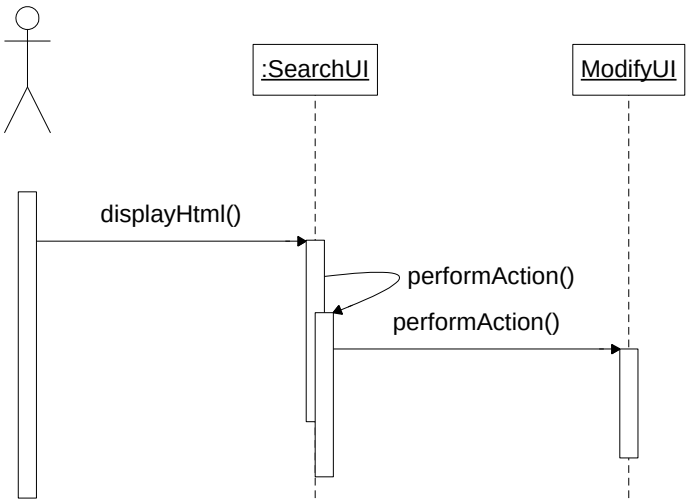


Abb. 16 Sequenzdiagramm Publikation Modifizieren

10 OOD (Object Oriented Design)

Das Benutzeroberflächensystem (GUI- System) ist bei einer Webbasierten Anwendung der Browser. Er sollte auf allen Plattformen gleiche Ausgaben erzeugen. Da aber entgegen gängiger Versprechen und Standards nicht auf allen Systemen alles gleich ausgegeben wird beschränke ich mich auf die Basis- HTML- Strukturen und verzichte weitgehend auf Java-Script und Java- Applets. So sind nur grundlegende Interaktionselemente verfügbar.
Maßgeblich für diese Entscheidung war das Abwägen von Erreichbarkeit und Komfort, da die Basis HTML Elemente in ihrer Funktionalität, beispielsweise Java Applets gegenüber, eher als begrenzt zu beschreiben sind, und im Gegensatz dazu die Basis- HTML- Elemente auf den gängigen Browsern ohne Modifikationen darstellbar sind.
Folgende Grafik beschreibt die Modellierung der Dialogstruktur mit Hilfe eines Zustanddiagramms.

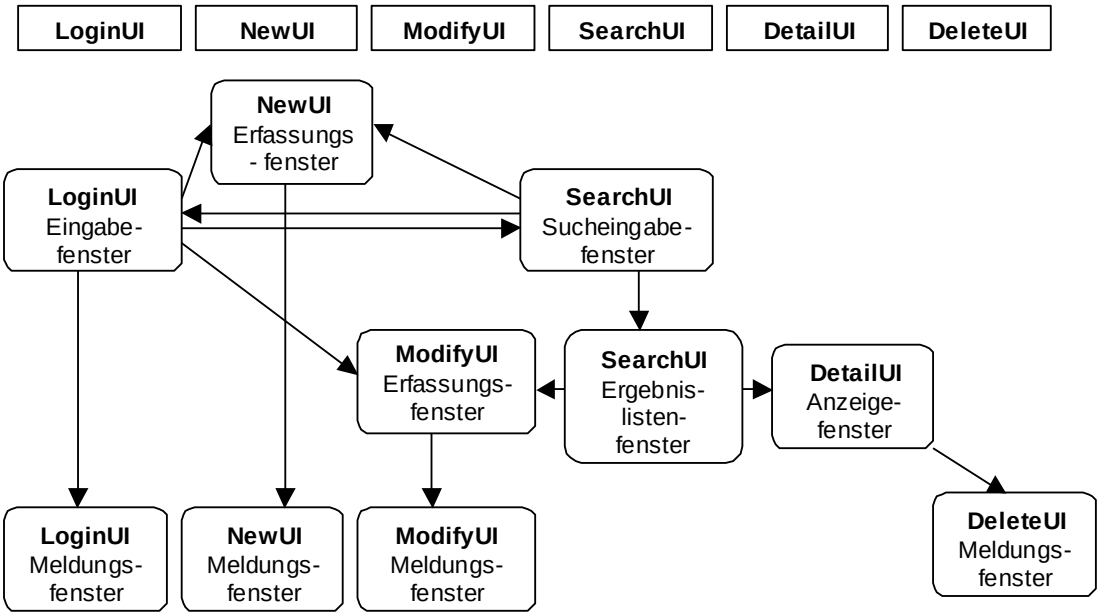


Abb. 17 Zustandsdiagramm Publikationsanwendung

Erläuterung:
Die rechteckigen Kästen stellen die Klassen dar, die Ausgaben erstellen können.

Die „Anzeigefenster“ und die „Meldungsfenster“ haben keine interaktiven Möglichkeiten, d.h. keine Eingabe-felder. Die Meldungsfenster dienen der Ausgabe von Erfolgs/Fehlermeldungen, die „Anzeigefenster“ dienen der Ausgabe von Daten. Die „Erfassungsfenster“ dienen der Eingabe von Daten. Dies kann sein Neueintragen, oder Modifizieren von Einträgen.
Zu beachten ist bei dieser Grafik, dass sie sehr Abstrakt ist, und so auf bestimmt Verfeinerungen nicht eingegangen wird, wie z.B. dass das Erfassungsfenster bei **ModifyUI** an verschiedene Rollen angepasst sein muss.

Die Oberfläche muss in Abhängigkeit von den Akteuren, und deren jeweiligen Aktionen angepasst sein. So ergeben sich folgende Masken:

- Suche (intern)
- Suche (extern)
- Fehlermeldung : Inkorrekte Suchkriterien
- Ergebnis der Suche (intern)
- Ergebnis der Suche (extern)
- Login- Seite
- Fehlermeldung : Authentifizierung fehlgeschlagen
- Neu (Alles, für bereits bestehende, aber noch nicht eingetragene Publikationen)
- Fehlermeldung : ungültige Authentifizierung
- Fehlermeldung : Inkorrekte Eingabe
- Erfolgsmeldung
- Neu (Nur die Daten, welche für eine Neueintragung einer AWI-Publikation notwendig sind)
- Änderung (Alles)
- Änderung (nur Status)
- Änderung (nur bibliographische Daten)
- Fehlermeldung : ungültige Authentifizierung
- Fehlermeldung : Inkorrekte Eingabe
- Erfolgsmeldung

- Löschen
- Fehlermeldung : ungültige Authentifizierung
- Erfolgsmeldung

10.1 Interaktionselemente

Um größtmöglichste Verfügbarkeit zu gewähren muss ich mich auf die bei der HTML gegebenen Interaktionselemente beschränken. Da HTML ein statusloses Protokoll ist, heisst das, das Eingabeformulare erst ausgeführt werden müssen, bevor Eingaben überprüft werden könne. Man muss also nach dem Abschicken eines Formulares überprüfen, ob die Eingabe gültige Werte enthält, und kann nicht während der Eingabe schon die Werte überprüfen.

10.2 Konzept der Benutzungsoberfläche

Bei dem Prototyp der Anwendung war die Benutzungsoberfläche sehr allgemein gehalten, und durch unvollkommene Anwendungslogik wurde dem Benutzer Aktionen erlaubt, zu denen er eigentlich nicht berechtigt sein sollte. Die Autorisierung war daher auf Selbstkontrolle angewiesen, was natürlich immer wieder zu fehlerhaften Eingaben führte. Dies wird nun durch eine Rollen-orientierte Benutzerführung verbessert werden. Sie gewährleistet, dass der Benutzer nur Aktionen durchführen kann, zu denen er durch das Publikationssystem formalisierte Vorgänge berechtigt ist. Auf die Einzelnen Elemente dieser Benutzerführung soll im Folgenden, durch eine genaue Beschreibung der Funktionen der Masken, eingegangen werden. Die Reihenfolge sei die Präsenz der Funktionalitäten auf der linken Seite der Hauptmaske, geordnet nach den Rollen.

10.2.1 Suchmaske

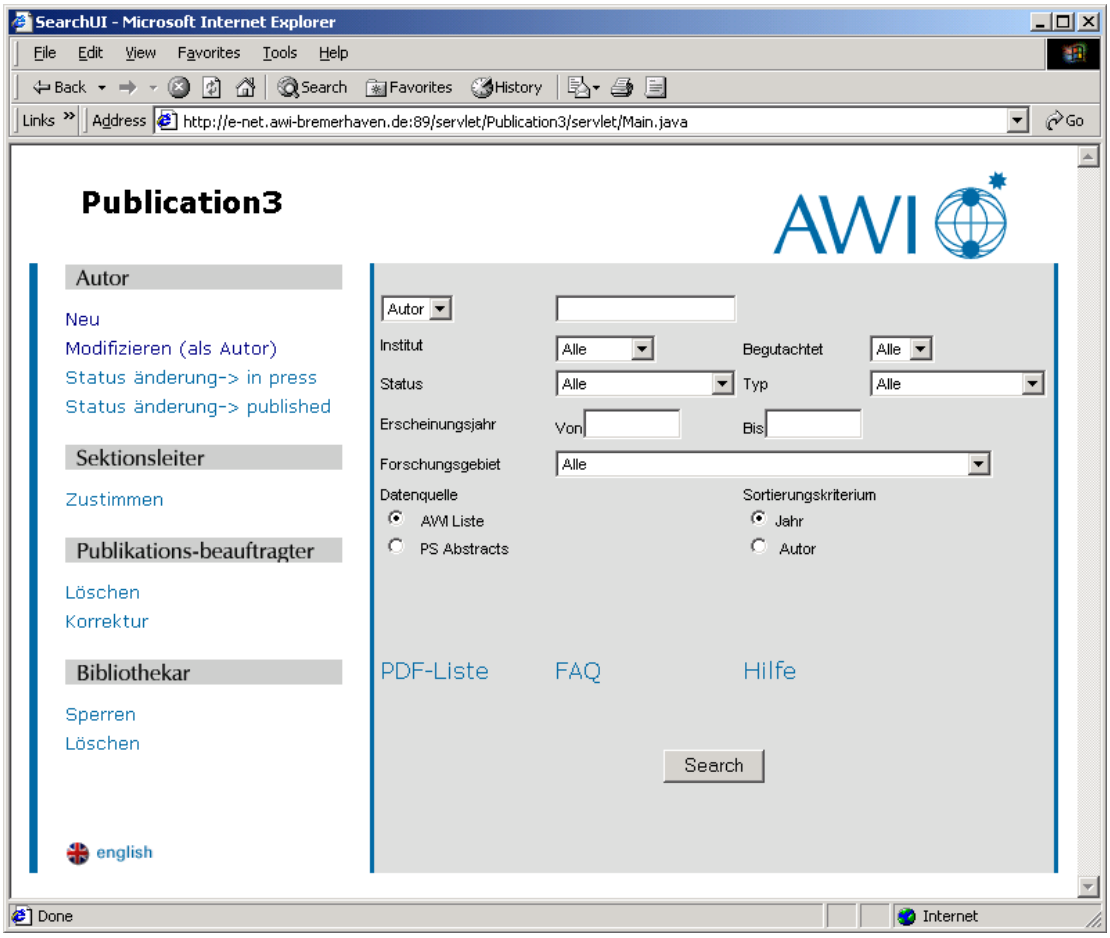


Abb. 18 Suchmaske

Die Suchmaske ist auch gleichzeitig die Hauptmaske der Applikation. Von hier aus gelangt man zu den anderen Funktionalitäten. Prägend für alle Masken ist der Name der Applikation oben links und das AWI-Logo oben rechts.

Das Fenster ist in zwei Bereiche unterteilt: Rechts die Suchmaske, und links die Möglichkeiten zum Bearbeiten von Publikationseinträgen. Die Rechte Seite ist wiederum nach Rollen gegliedert, und bietet so für jede Rolle die spezifischen Funktionalitäten.

Sobald man eine der Bearbeitungsfunktionalitäten auswählt, gelangt man zu einem Login Fenster, und wird von dort aus zur entsprechenden Funktionalität weitergeleitet. Sollte man bereits eingeloggt sein, gelangt man sofort zu der entsprechenden Funktionalität.

10.2.2 Loginmaske

Abb 19 Login - Maske

In der Login Maske muss man seinen Benutzernamen und sein Passwort eingeben. Da die Authentifikation über den Directoryserver läuft, sind hier die Daten die gleichen, wie für das Mailsystem, und die „Personal3“ Anwendung. Im Zuge der Umstellung auf Java rückt ein „Single Sign On“ immer näher, da durch Java ermöglicht wird, dass die Nutzerdaten in einer Session, und somit für alle auf dem Server ausgeführten, webbasierten Javaprogramme zugänglich sind.

10.2.3 Auswahl einer einzutragenden Publikation treffen

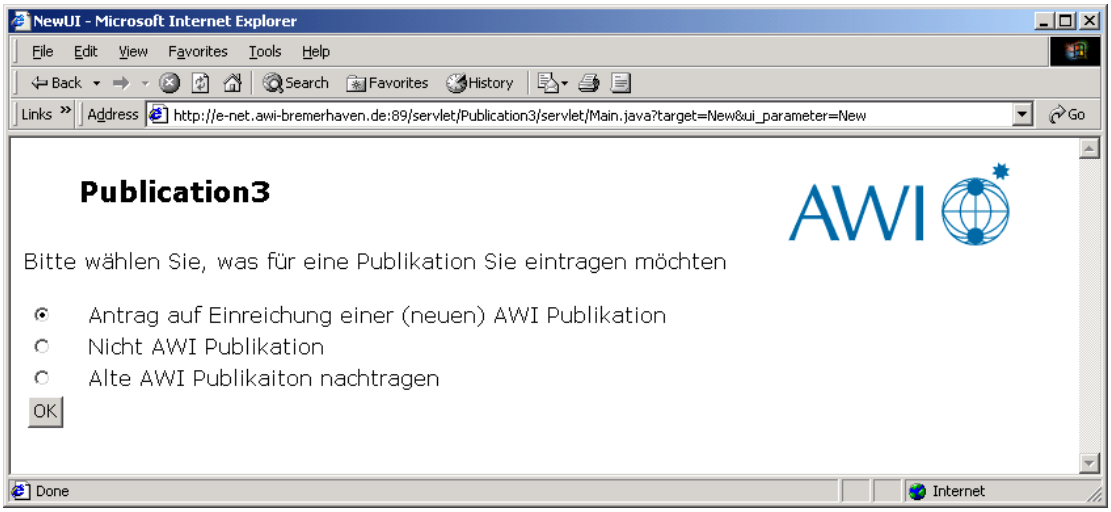


Abb. 20 Auswahl der Neuen Publikation

Bevor man eine Publikation eintragen kann muss man eine bestimmte Art der Publikation auswählen. Diese Wahl hat eine entscheidende Bedeutung für den Prozess des Eintragens von Publikationen, da je nach Entscheidung bestimmte Werte automatisch gesetzt werden, bzw. bestimmt Mechanismen in Gang gesetzt werden.

10.2.4 Publikation eintragen

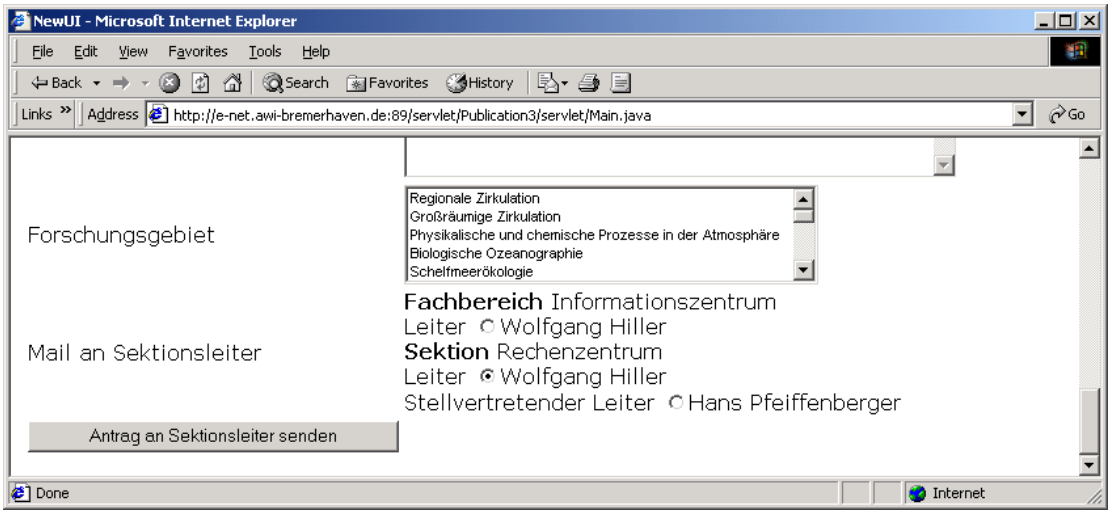


Abb. 21 Ausschnitt vom Neueintragen

In diesem Ausschnitt wird der Unterschied deutlich, den eine der Oben getroffenen Wahlen bewirken:

Beim Neueintragen einer AWI- Publikation muss der Sektions- , oder Fachbereichsleiter benachrichtigt werden.

10.2.5 Modifikationsmaske

An dieser Stelle zeige ich die Modifikationsmaske am Beispiel einer Modifikation durch eine Autoren. Zuerst gelangt man zu einer Auswahlmaske, bei der man die zu modifizierende Publikation auswählen kann. In diesem Fall sieht man noch den LDAP- Suchfilter, welcher deutlich macht, dass ein sauberer Datenbestand von höchster Wichtigkeit ist. Hält man sich nicht an die vorgegebenen Konventionen bei der Eintragung des Autoren, findet die Suche nicht alle Einträge.

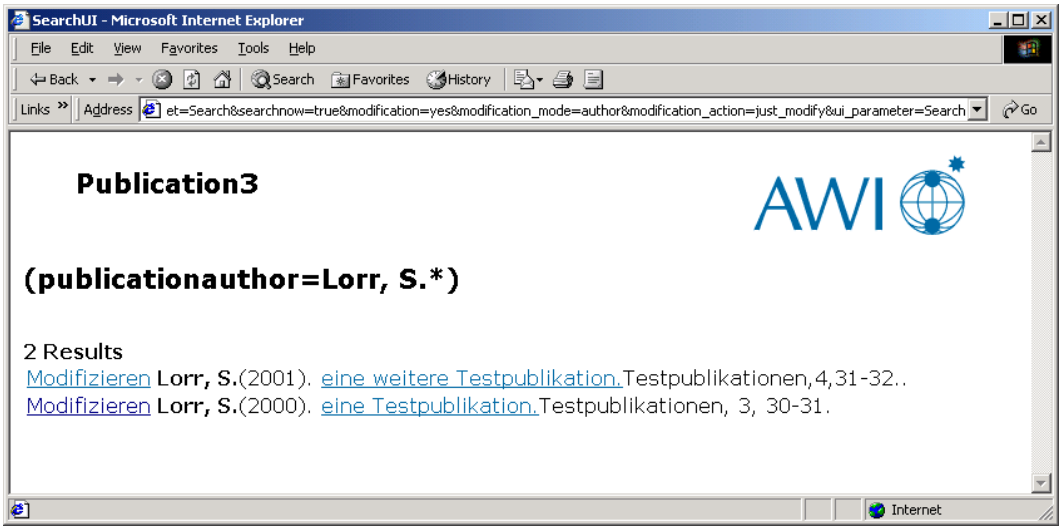


Abb. 22 Auswahl der zu modifizierenden Publikation

Von der Auswahlmaske gelangt man dann zu Modifizierungsmaske

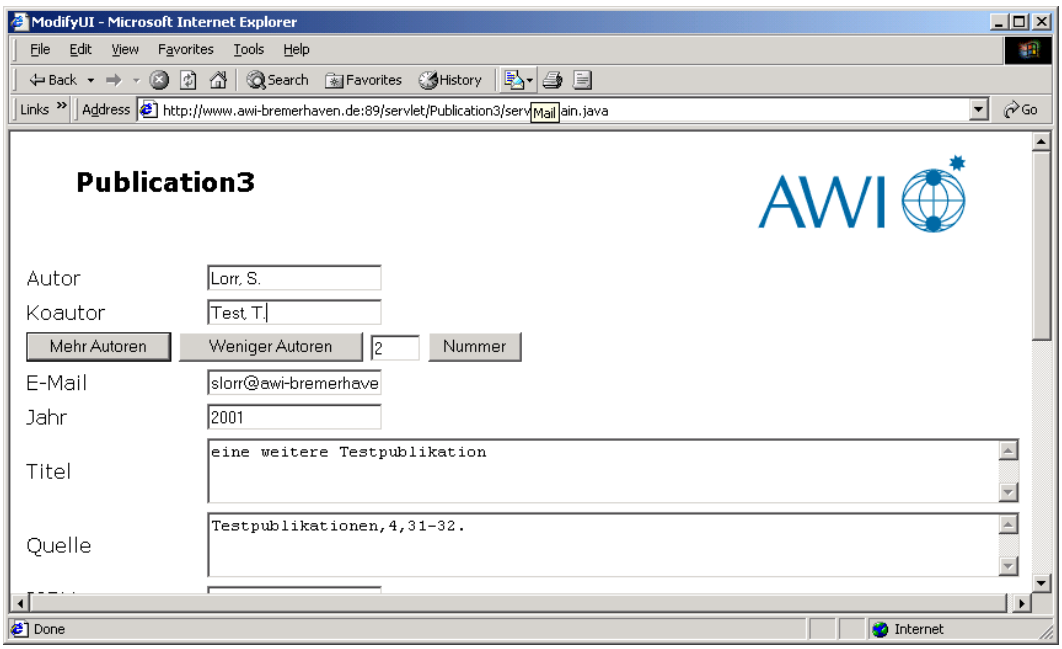


Abb. 23 Ausschnitt aus der Modifizierungsmaske

In dieser Maske ist es möglich die Werte eines existierenden Datensatzes zu ändern. Der Autor hat allerdings nicht die Möglichkeit ALLE Werte zu ändern. Dies bleibt dem Publikationsbeauftragten vorbehalten.

10.2.6 Löschmaske

Die Löschmaske besteht nur aus einer Auflistung von Publikationen, ähnlich der Auswahlmaske für die Modifikation. Die Liste ist genau für die bearbeitende Person zugeschnitten. D.h. dass der Publikationsbeauftragte eines Fachbereiches ein Liste der Publikationen seines Fachbereiches bekommt. Klickt er dann auf „Löschen“ wird die entsprechende Publikation gelöscht.

11. Bestehendes Rahmenwerk

Durch Karsten Stellings Diplomarbeit wurde bereits ein Rahmenwerk geschaffen, an das ich meine Arbeit adaptieren werde.
Dieses Rahmenwerk umfasst einige Programmiertechniken und Pakete, auf welche ich im folgenden Abschnitt eingehen werde.
Der grundsätzliche Aufbau von dem bereits bestehenden Programmen ist wie folgt:
Es existieren Servlets zur Kommunikation via HTTP zwischen Client und Server. Diese Servlets greifen zur Darstellung von Benutzeroberflächen auf die Klassen mit der Endung „UI“ zu. „UI“ steht für User Interface und kennzeichnet so diese Klassen als Benutzerschnittstellen. Diese „UI“s greifen wiederum auf Klassen zu, welche den Datenbankzugriff steuern. Somit wird hier das 3 Schichten- Modell (3 tier model) praktiziert; Die Client-Schicht wird über die Ausgabe der „UI“s an einen Webbrowser abgedeckt, die Server- Schicht über die Servlets und die UIs, und die Datenschicht über die entsprechenden Klassen, welche auf die Datenbank zugreifen.

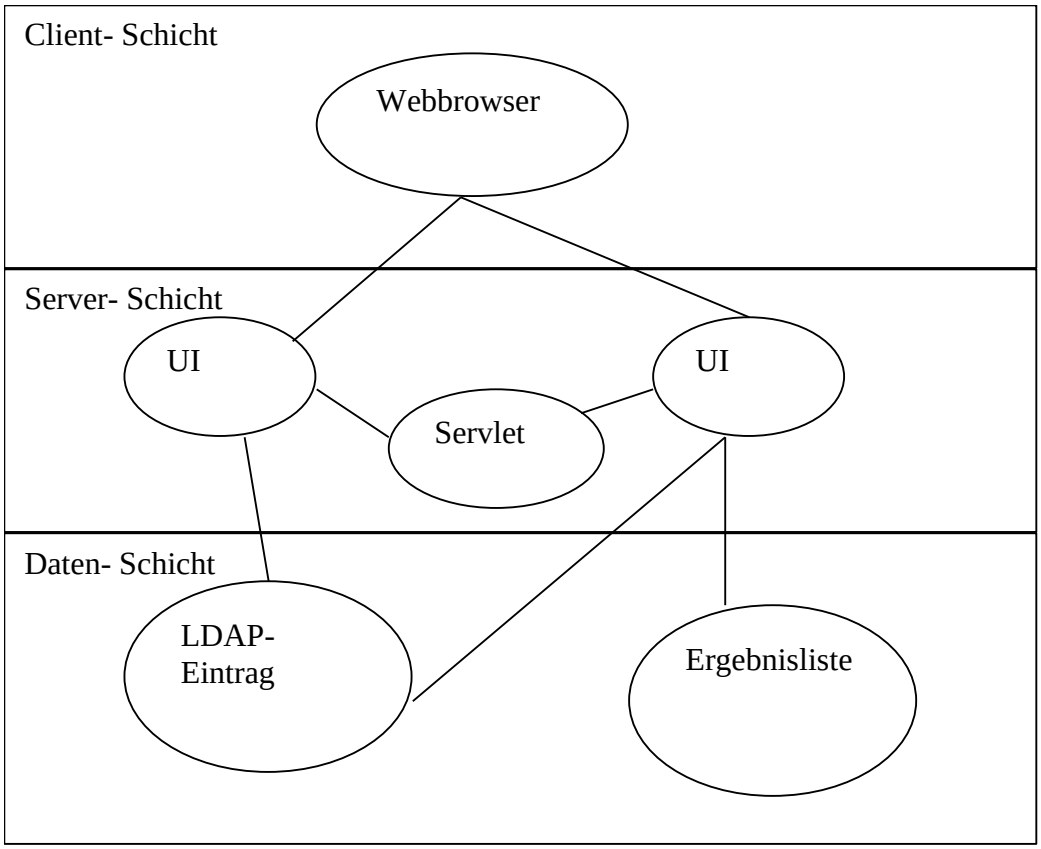


Abb. 24 3Schichten Architektur der bisherigen Anwendung

11.1 Struktur von „Personal3“

Das Programm ist so aufgebaut, dass es 3 Servlets gibt, welche eine Kommunikation zwischen Client und Server gewährleisten: Login, Menu und Mitarbeiter.
Die Ausgabe ist in Klassen die auf UI („User Interface“ engl. für „Benutzer Schnittstelle“) enden ausgelagert. Jedes Servlet greift für die Ausgabe auf ein oder mehrere UI’s zu. Die UI’s greifen wiederum auf Klassen zu , welche den Datenbankzugriff durchführen.
Sämtliche Klassen die über die Servlets hinausgehen sind im Packet „awi“ zusammengefasst, welches wieder in mehrere Paket- Komponenten unterteilt ist. Ein wichtiger Bestandteil ist das Paket „awi.personal“, da hier die Applikation betreffende Klassen liegen. Verdeutlicht wird dies durch folgende Grafik:

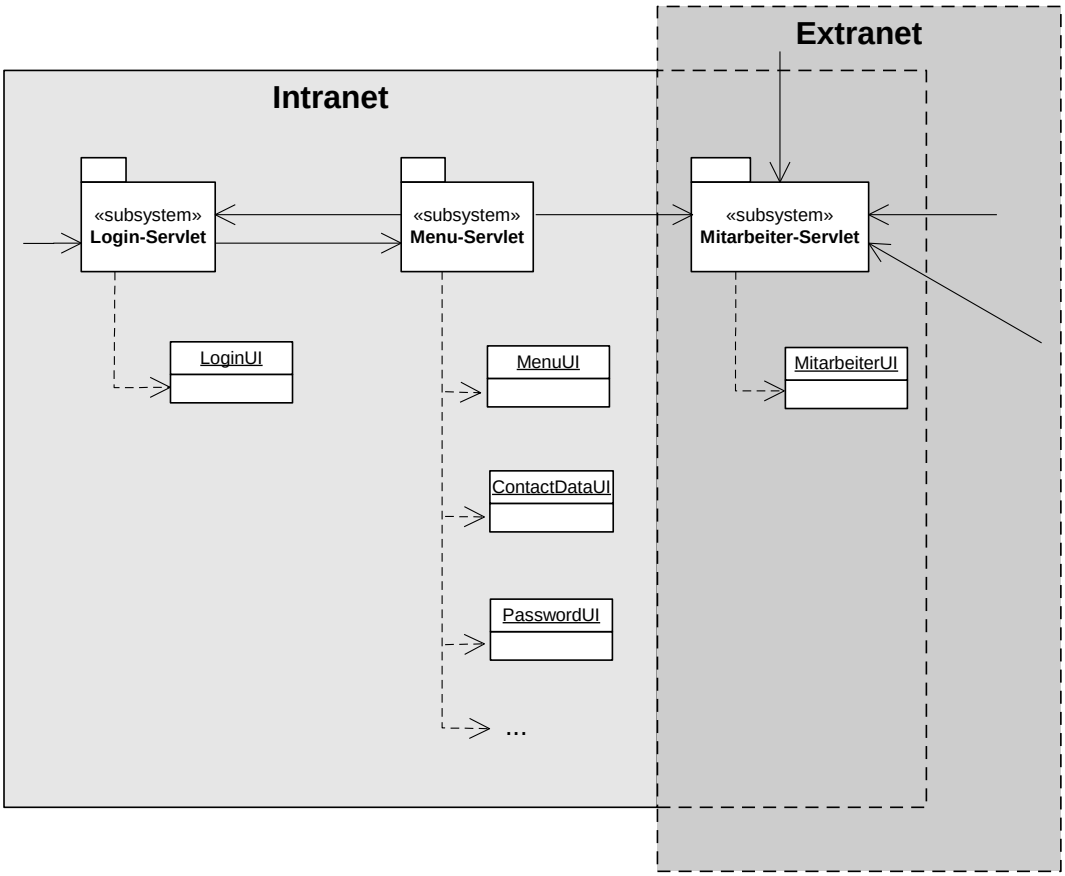


Abb. 25 Personenbezogene Komponente des Informationssystems [Ste2000 ,s. 54]

11.1.1 Auslagerung von Parametern und Internationalisierung

Die Auslagerung von Parametern und die Internationalisierung haben eins gemeinsam: Die Daten liegen in Textdateien mit der Dateiendung „.properties“. Diese Dateien sind mit Attribut-Wert-Paaren gefüllt, und lassen sich einfach editieren. So müssen wichtige Änderungen nicht im Code vorgenommen werden, sondern können zentral durch Modifikation dieser Dateien erreicht werden.
Das besondere an Dateien die mit Internationalisierung zu tun haben, ist dass Ihre Namen mit dem Sprach und dem Ländercode enden. Durch diese Normung sind die Dateien von der Anwendung nach Sprachen unterscheidbar. Man kann dann einfach ein durch einen Parameter bestimmten Wert ein Objekt vom Typ „Locale“ erzeugen, und mit Hilfe dieses Objektes ein Resourcebundle erzeugen, welches durch einen Hashtable Zugriff auf die lokalisierten Werte gibt.

11.1.2 Reflection

Um die Anwendung so flexibel wie möglich zu machen wird das Prinzip „Reflections“ mit den dazugehörigen Klassen benutzt. Es ermöglicht zur Laufzeit die Einbindung von Klassen deren Namen während des Kompilierens nicht bekannt sind. Die Klassen werden dann während der Laufzeit geladen und instanziert. Mit Hilfe der Reflection-API kann man so eine Plug In Lösung für Erweiterungen schaffen.
Durch dieses Prinzip ist es bei Personal3 besonders einfach Erweiterungen zu erschaffen und anzubinden. [Ste2000, s. 45]

11.1.3 Sessions

Zur Speicherung des aktuellen Status wird eine sogenannte „Session“(Sitzung) eröffnet. Mit Hilfe dieser Session können Daten über mehrere HTML- Seiten hinweg abgespeichert werden. Da HTTP ein statusloses Protokoll ist, kann man normalerweise keine Daten HTML-Seiten unabhängig abspeichern. Der Mechanismus des Sessionshandling erlaubt dies jedoch. Die Daten werden allerdings Serverseitig

abgespeichert, und beim Client wird nur die ID der Session in einem Cookie gespeichert.
In einer Session werden folgende Attribute abgespeichert:

- **IsAuthenticated** – Boolescher Wert, welcher gesetzt wird, sobald sich jemand einloggt.
- **PersonToEdit** – Objekt der Klasse „Person“, welches die Daten des Directory- Eintrages des augenblicklichen Nutzers enthält.
- **Mode** – Modus in dem sich der Nutzer befindet; mögliche Werte sind : User, Admin
- **AdminPerson** – Objekt der Klasse „Person“, welches gesetzt wird, sobald sich jemand als Administrator einloggt. Dieser ist dann der Editor, während **PersonToEdit** zu editierten wird.
- **Resources** – Hashtable mit Lokalisierten Beschriftungen
- **peopleNames** -
- **locale** – Ein Objekt der Klasse „Locale“, in dem der Sprach- und der Ländercode abgespeichert werden

Über diese Daten kann man Applikations-unabhängig den Benutzer betreffende Daten abspeichern. Diese Daten können dann von anderen Javaprogrammen auf dem Server benutzt werden, um z.B. die jeweiligen Ländereinstellungen des Benutzers zu benutzen, oder um sich zu authentifizieren.

Die Paketstruktur des Rahmenwerkes wird in Abb. 26 dargestellt.

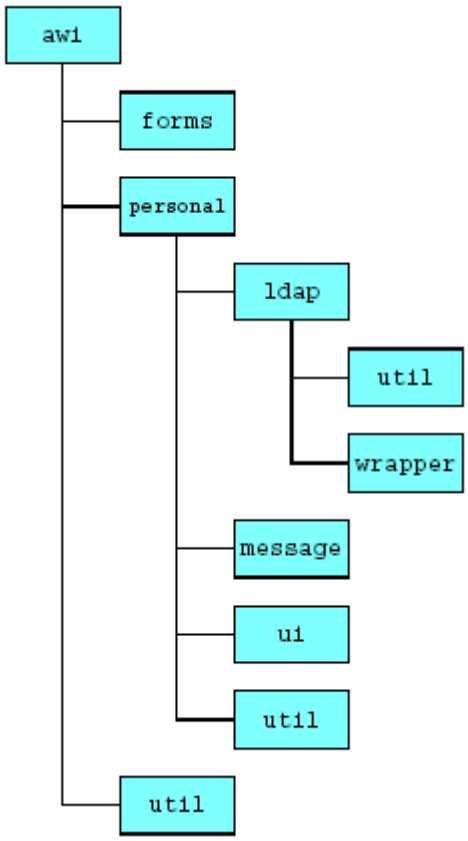


Abb. 26 Paketstruktur des Rahmenwerkes [Ste2000, s. 80]

12. Implementierung

Meinen Umsetzung des Publikationssystems knüpft an die Arbeit von K. Stelling an. So verwende ich auch Java mit den entsprechenden APIs für den Zugriff auf den Directoryserver, die Benutzung von Servlets und den Versand von Mails.

12.1 Grundsätzliche Unterschiede zur Arbeit von K. Stelling

Der augenscheinlichste Unterschied zwischen „Personal3“ und „Publication3“ liegt wohl in der Zahl der Servlets: „Personal3“ besteht aus 3 Servlets, während „Publication3“ nur eins besitzt.
Der Grund hierfür ist, dass „Personal3“ grundlegende Anwendungselemente in eigene Servlets aufsplittet, während bei „Publication3“ die ganze Anwendungslogik zentral über ein Haupt– Servlet gelenkt wird.

Wie schon in der Einleitung beschrieben lege ich bei meiner Implentation viel Wert auf die Einfachheit des Codes. In diesem Rahmen habe ich auf das Prinzip der “Reflection“ total verzichtet. Die Reflections hatten den Zweck die Einbindung neuer Uis einfacher, ohne Veränderung des bestehenden Codes zu ermöglichen. Bei „Personal3“ wurde dies im „Menu- Servlet“ benutzt, um Funktionalitäten in form von Uis in der Navigationsleiste aufzuzeigen. Dies setzt voraus, dass die Reihenfolge nicht besonders wichtig ist, da man ja schließlich die Reihenfolge in der Anwendung festlegen müsste, ohne zu Wissen was noch alles eingebunden werden soll.
Um die Reihenfolge besser kontrollieren zu könne empfiehlt es sich daher die Reihenfolge oder besser gesagt die Reihenfolge der Aufrufe der Uis im Code fest zu verdrahten. So behält man die Übersichtlichkeit, und erhält eine höhere Performance.
Auf der Herstellerseite vun SUN- Java wird in einem Tutorial davon abgeraten das Reflection-API für normale Anwendungen zu nutzen, da es

eigentlich dafür gedacht ist Entwicklungstools zu erschaffen, wie z.B. Klassenbrowser und Debugging Werkzeuge. [Web 5]

12.2 Erweiterungen des Rahmenwerkes

Durch das bestehende Rahmenwerk wird mir eine Vorgehensstruktur bis zu einem gewissen Maße vorgegeben. Nun gilt es diese Struktur um passende Elemente zu erweitern.

12.2.1 Neue Paketstruktur

Die Publikationen betreffenden Klassen habe ich neu geschrieben. Bei der Anwendung „Personal3“ liegen diese Klassen im Paket *awi.personal.ldap.wrapper.* Sie sind soweit ausgebaut, wie die Applikation es benötigt. Für die Publikationsanwendung benötige ich allerdings volllen Zugriff auf alle Attribute. Desweiteren Iste es für ein Rahmenwerk besser solche Klassen nicht an einer Applikation festzumachen. Deswegen erschuf ich ein neues Paket: *awi.ldap.wraper.* Es ist besser für die Erstellung weiterer Applikationen vorteilhaft. wenn man den Paketbaum nicht in den applikationsspezifischen Verästelungen nach Klassen für den Datenbankzugriff durchforsten muss. Einer der nächste Schritt besteht also darin die personenspezifischen Klassen auch in das neutrale *awi.ldap.wraper* – Paket zu packen, und die bestehende Anwendung dementsprechend anzupassen.
So ergibt sich somit folgende Paketstruktur:

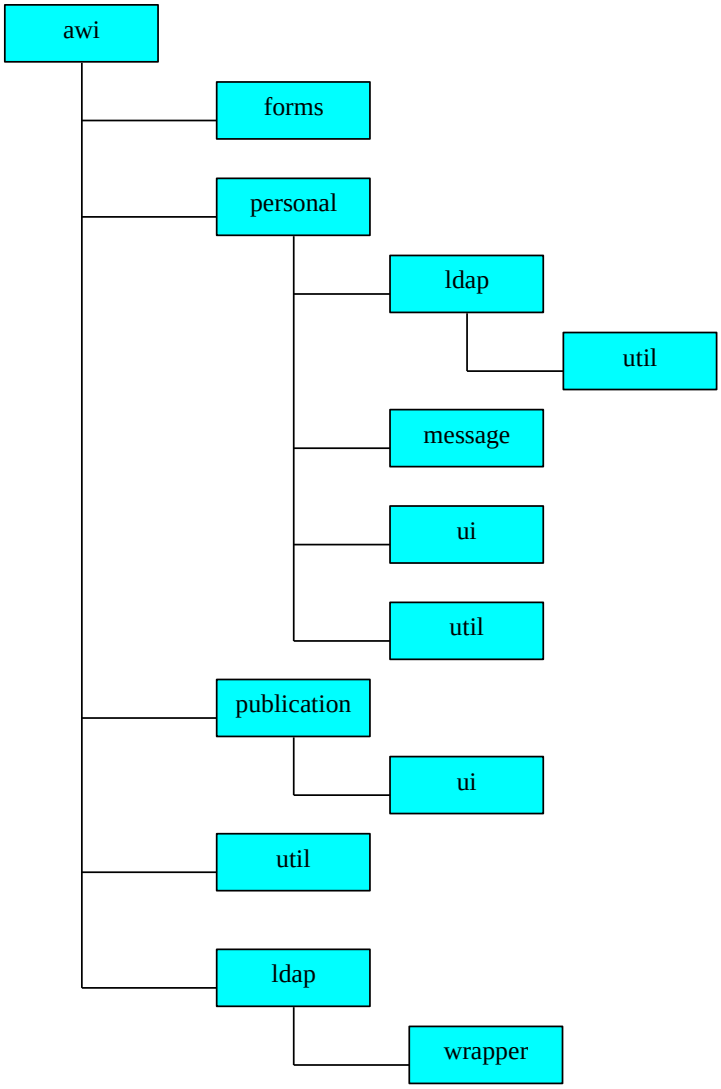


Abb. 27 Neue Paketstruktur

12.2.2 Erweiterung der Session

Publication3 setzt beim ersten Aufruf eine Session ohne Authentifizierung. Es wird einfach nur ein „locale“- Objekt erzeugt, um die Länderund Spracheinstellung festzulegen. Standardmäßig wird deutsch eingestellt. Die Personal- Anwendung erwartet jedoch Authentifizierungsdaten in der Session, welche deswegen auch durch Publication3 gesetzt werden müssen.

Dies birgt allerdings die Schwierigkeit, dass man sich für die Nutzung von Publication3 nicht notwendigerweise authentifizieren muss. Also muss Personal3 so geändert werden, dass es auch eine Session akzeptiert, welche nur Lokalitätsdaten enthält.

Auf die Session der Personal3- Anwendung lässt sich problemlos durch ein entsprechendes Session- Objekt zugreifen. Diese Session erweitere ich um das Element „Pubmode“. Dies ist der Benutzermodus für die „Publication3“- Anwendung. Mögliche Werte sind:

- „author“ – Einfacher Autor
- „librarian“ – Bibliothekar
- „sectionhead“ – Sektionsleiter
- „docmaster“ – Publikationsbeauftragter

12.3 Klassenstruktur von Publication3

Die Klassenstruktur wird durch folgende Grafik vereinfacht dargestellt.

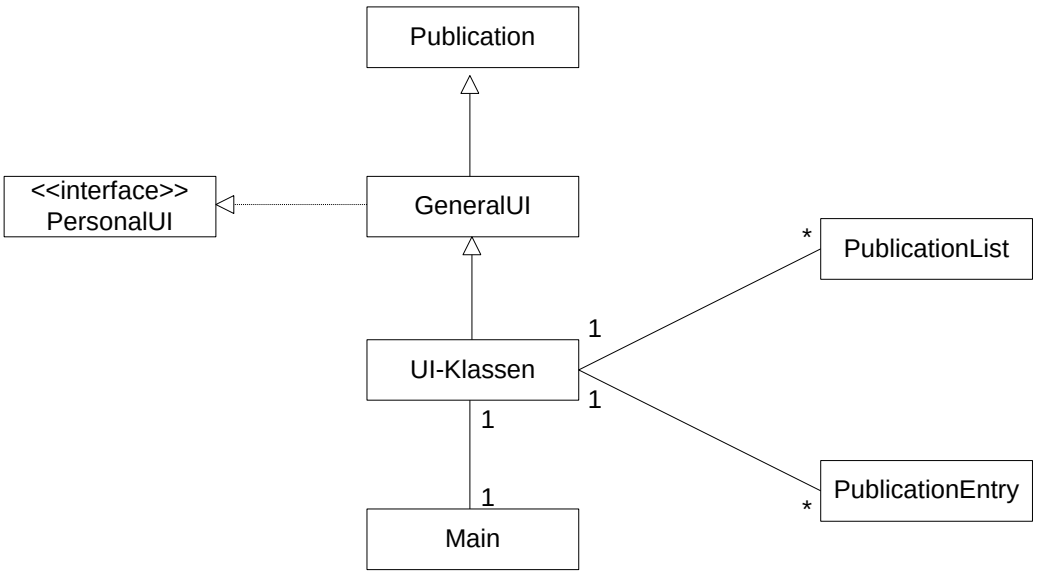


Abb. 28 Klassenstruktur

In folgendem Abschnitt werden die Klassen erläutert.

Publication

Dies ist sozusagen die Steuerungsdatei. Hier wird die Deklaration der globalen Variablen aus der *publication.properties* durchgeführt. Da diese Klasse diese Werte vererbt, sind sie auch für alle anderen Klassen zugänglich. Sollte sich ein Wert ändern, muss man die Änderung nur in der *publication.properties* vornehmen. Sollte man eine Wert hinzufügen muss man ihn nur in der .properties-Datei und in der publication- Klasse

hinzufügen, damit er allen anderen ,die Applikation betreffenden ,Klassen zur Verfügung steht

GeneralUI
In dieser Klasse wird das generelle Design festgelegt. Hier werden auch abstrakte Klassen definiert, die dann von den eigentlichen Uis implementiert werden sollen So werden hier auch in den Methoden grundlegende Designelemente hinterlegt, welche dann an die Uis mitvererbt werden.

PersonalUI
Dies ist eine Schnittstelle, welche von K. Stelling erstellt worden ist. Sie legt die abstrakten Klassen für eine UI- Klasse fest. Es war nicht notwendig eine eigene „PublicationUI“ zu erstellen, da die PersonalUI allgemeingültig ist, und nichts Personal – spezifisches enthält.

UI- Klassen
UI steht hier für „User Interface“, also die Benutzerschnittstelle. Das bedeutet, in diesen Klassen findet die Interaktion mit dem Benutzer statt. Dies Uis im speziellen sind:

- LoginUI
- SearchUI
- ModifyUI
- NewUI
- DeleteUI

Hier werden Sucheingaben verarbeitet und die für die Eingabe von Datenbankeinträgen notwendigen Masken erstellt. Die UI- Klassen haben in Ihrem Konstruktor ein Objekt der Klasse „Locale“ als Parameter. So wird die Sprache und das Land festgestellt, was dann wiederum die internationalisierte Ausgabe ermöglicht.
Die Uis sind recht unabhängig. Man kann sie theoretisch in andere Systeme einbauen wobei in diesem Fall die SearchUI eine entscheidende Rolle für die Anwendungslogik spielt; Sie ist der Dreh und Angelpunkt der

Applikation. Von hier aus wird implizit oder explizit der Zugriff auf alle anderen Uis ermöglicht.

PublicationList
Dies ist die Erstellung einer Liste von Publikationseinträgen nach verschiedenen Kriterien in eine Klasse gekapselt. Sie wird jedes Mal benutzt, wenn es gilt eine Suche durchzuführen

PulbicationEntry
Dies ist die Abbildung eines Publikationseintrages in eine Klasse. Ihre Attribute sind denen eines Directoryserver – Eintrages gleich. Die Methoden lassen es neben den üblichen Verwaltungsfunktionen(get, set) zu, neue Einträge zu schreiben, sowie bestehende zu editieren oder zu löschen. Diese Klasse ermöglicht also einen einfacheren Zugriff auf die bestehenden Publikationseinträge, ähnlich der „PeopleEntry“- Klasse von K. Stelling. Somit wird hier auch wieder ein Schritt in Richtung „Rahmenwerk“ für den Zugriff und die Verwaltung von Directory- Einträgen gemacht.

Main
Dies ist die eigentliche Verwaltungsdatei der Anwendung. Hier wird gesteuert, welche UI aufgerufen werden soll. Diese UI wird dann mit einem „Locale“ Objekt initiiert, welches wiederum aus einer http- Session stammt. Existiert keine Session, wird eine neue Session erstellt, welche ein „Locale“ – Objekt für Deutschland enthält. Somit wird standardmäßig die deutsche Sprache geladen.

12.4 Suchen

Für eine Suche nach Daten in einem LDAP-Server müssen immer folgende Daten zu Verfügung stehen:

- LDAP- Server
- Der Pfad im Server, ab dem gesucht werden soll
- Suchfilter

Der Filter besteht aus einem Attributname gefolgt von dem Operator und dem gesuchten Wert .
Möchte man gerne nach mehreren Werten suchen, so muss man zuerst die logische Verknüpfung und dann die Filter nennen.
Z.B. Alle Einträge die als Autor Lorr haben:

(publicationauthor =Lorr)

Das Attribut welches den Autor enthält ist in diesem Fall „publicationauthor“
Alle Einträge die als Autor Lorr haben und aus dem Jahr 2000 stammen:

(&(publicationauthor=Lorr)(publicationyear=2000))

In diesem Beispiel kommt das Jahr mit dem Attribut “publicationyear“ hinzu.

Die Suchseite stellt die Hauptseite der Anwendung dar. Sie wird standardmäßig geladen, auch wenn keine Parameter gesetzt sind
Diese Seite hat einerseits die Funktion auf die anderen Funktionalitäten zu verweisen, und andererseits Einträge nach bestimmten Kriterien zu finden. So gibt es mehrere Private definierte Operationen zur Erstellung verschiedener Suchfilter und Ergebnislisten. Diese Privaten Operationen erstellen auch Listen, die auf Rollen zugeschnitten sind. So wird auch zum Beispiel die Liste für den jeweiligen Autor zum Modifizieren erstellt. In diesem Fall werden durch den Link zur Modifikation Parameter übergeben, welche die entsprechende Methode der ModifyUI auslösen. Aus dem „Session“- Objekt erhält die Anwendung dann Daten über den Autor, wonach dann ein Bestimmter Suchfilter erstellt werden kann.

12.5 Neueintragen einer Publikation

Das Neueintragen einer Publikation ist ja schon im Kapitel „Publikation“ beschrieben worden. Hier folgt nun der technische Teil , die Umsetzung des ‚Konzeptes.

Die Neueintragung wird von der NewUI durchgeführt. Ihre displayHtml() Operation erstellt zunächst eine Auswahlseite, auf der die verschiedenen

Arten von Publikationen zur Auswahl stehen. Je nach Wahl wird dann eine Eingabeseite erstellt.
Die performAction() Operation erstellt dann einen neuen Eintrag in Abhängigkeit von den eingegebenen Werten und verschickt ggf. eine Email an den zuständigen Sektionsleiter.

Es gibt 3 verschiedene Möglichkeiten, eine Publikation einzutragen:
Als
Neue AWI- Pulbikation
Diese Art des Eintragens verläuft nach dem im Kapitel „Publikation“ vorgestelltem Ablauf. Hier wird die Idee einer Veröffentlichung über das entsprechende Onlineformular dem zuständigen Fachbereichs- oder Sektionsleiter vorgestellt.

Bestehende AWI- Publikation nachtragen
Diese Option ermöglicht es dem Nutzer, bereits veröffentlichte Publikationen nachzutragen; der Sinn ist, dass so Publikationen in den Datenbestand aufgenommen werden könne, deren Veröffentlichung vor oder während der Einführung des Softwaresystems stattgefunden hat.

Nicht- AWI- Publikationen
Bei diesen Publikationen handelt es sich um Veröffentlichungen, welche nicht am AWI erstellt worden sind, welche der entsprechende Nutzer aber gerne bspw. Auf seiner Persönlichen Homepage eintragen möchte. An dieser Stelle wird das Zusammenspiel der verschiedenen Applikationen deutlich: Mit dieser Applikation wird Die Publikation in die Datenbank eingetragen, und mit der „Personal“ – Applikation wird auf diesen Datenbestand (sogar über die gleichen Java- Klassen) zugegriffen

Der kleine datentechnische Unterschied sieht so aus, dass bei den verschieden Eintragungsmöglichkeiten verschieden Werte automatisch gesetzt werden können. Zur Verdeutlichung dient folgende Tabelle:

Publikationstyp	Automatisch gesetzte	mögliche Attribute

	Werte	
Neue AWI-Publikation	uid=(generierter wert aus den ersten Anfangsbuchstaben des 1. Autoren, dem Jahr und einem Fortlaufendem Index) publicatiostatus=proposed publicationawi=yes	publicationauthor, publicationauthor;alternate, publicationemail, publicationtitle, publicationsource, publicationidentifier, publicationpsabstract, publicationpscruise, publicationpsleg, publicationbegutachtet, publicationtype, thesisacceptedby, publicationtag
Alte AWI-Publikation	uid=(s.o.) publicationawi=yes	s.o. publicationawinumber, publicationstatus
Nicht- AWI-Publikation	uid=(s.o.) publicationawi=no	Wie bei Neu-AWI publicationstatus

Tab. 2 Attribute bei Neueintragung einer Publikation

Zusätzlich zu diesen Unterschieden muss man beim Neueintragen einer AWI- Publikation einen Fachbereichs/ Sektionsleiter aus einer automatisch erstellten Liste auswählen, welcher dem Antrag stattgeben muss. Dieser bekommt dann eine Email, die ihn darüber informiert, dass er die neu eingetragene Publikation bearbeiten muss.

12.6 Modifikation einer Publikation

Alle Modifikationen werden von der ModifyUI durchgeführt. Ihre displayHtml() Operation erstellt je nach parameter eine Eingabeseite die auf die verschiedenen Rollen zugeschnitten ist.
Die performAction() Operation führt dann die Modifikationin Abhängigkeit von den eingegebenen Parameter durch.

Es sind 2 Möglichkeiten zur Änderung von Publikationseinträgen implementiert:
1. Durch direkten Aufruf der performAction –Operation
Diese Methode wird bei Änderungen benutzt, bei denen nur ein oder zwei Attribute auf einen bestimmten Wert gesetzt werden sollen. Dies ist der Fall, wenn der Status verändert werden soll, oder wenn die Publikation durch einen Bibliothekar gesperrt werden soll.
2. Durch eine Eingabe von Daten in einer Eingabemaske
Im folgenden gehe ich auf die verschieden Rollen ein und beschreibe, welche Attribute auf den Änderungsmasken ihnen zur verfügbar sind und warum genau diese und andere nicht:

12.6.1 Autor

Der Autor ist derjenige der für seine Publikationen verantwortlich ist. Er muss dafür sorgen, dass die Daten auf dem aktuellen stand sind und richtig eingetragen sind. Aus Tabelle 3 wird deutlich, dass er nicht alle Attribute ändern darf. Das liegt daran, dass es Felder gibt in denen er falsche Abgaben machen könnte, um in Listen aufzutauchen in denen er gerne auftauchen möchte, allerdings nicht hingehört. Diese Aktionen würden den Datenbestand verunreinigen. So hat er keinen Zugriff auf die Attribute
publicationawi, welches festhält, ob es sich um eine AWI- Publikation handelt
*publicationawinumbe*r, die vom AWI vergebene Nummer einer Publikation
publicationstatus, den Status [vergl. Tab. 3]

Diese Attribute werden automatisch nach einer bestimmten Verhaltensweise und unter festgelegten Bedingungen verändert. Es wird eine Liste der Publikationen vom Autor mit einem bestimmten Status angezeigt. So erfordert das Ändern des Status, dass ein bestimmter Status gesetzt ist, da die Publikation sonst nicht zum Ändern des Status zugänglich ist.

12.6.2 Sektionsleiter

Der Sektionsleiter hat keine wirkliche Eingabemaske zum ändern von einträgen. Ihm wird einfach eine Liste mit Publikationen seinen Bereiches vorgelegt, welche den Status „Einreichung beantragt“ haben. So kommt er nicht in die Verlegenheit eine falsche Eingabe zu machen, und es ist übersichtlicher für ihn.

12.6.3 Publikationsbeauftragter

Eine Besondere Rolle in diesem System nimmt der Publikationsbeauftragte ein. Jede Sektion hat mindestens einen. Er ist ein Sektionsweiter Administrator für Publikationen. Er kann die Publikationen seiner Sektion löschen und beliebig bearbeiten.

12.6.4 Bibliothekar

Der Bibliothekar ist ein Mitarbeiter mit herausragenden bibliographischen Kenntnissen. So kann auch er entscheiden, ob eine Publikation gelöscht werden darf.

Folgende Tabelle zeigt, welche Attribute für welche Rolle zur Änderung zur Verfügung stehen

Attribut	Autor	Publikations-beauftragter	Bibliothekar
Publicationawi		X	
publicationawinumber		X	
Publicationauthor	X	X	X
publicationauthor;alternate	X	X	X
Publicationemail	X	X	
Publicationyear	X	X	X
Publicationtitle	X	X	X
Publicationsource	X	X	X
Publicationidentifier	X	X	X
publicationpsabstract	X	X	

publicationpscruise	X	X	
Publicationpsleg	X	X	
publicationbegutachtet	X	X	
Publicationtype	X	X	X
publicationstatus		X	
thesisacceptedby	X	X	X
Publicationtag	X	X	

Tab. 3 Attribute auf verschiedenen Masken

12.7 Löschen einer Publikation

Das Löschen vom Programmieraufwand ein sehr einfacher Vorgang. Man öffnet eine Verbindung zum Directoryserver, authentifiziert sich mit DN (Distinguished Name) und Passwort, und ruft die Deletefunktion mit dem DN des Publikationseintrages auf.

Das wichtige hierbei ist, dass die Rolle des Benutzers abgefragt wird, damit es keine unautorisierten Löschzugriffe gibt. Zur Löschung von Publikationseinträgen sind nur 2 Rollen berechtigt: Der Bibliothekar, der alle Einträge löschen darf und der Publikationsbeauftragte, der Löschrechte für Publikationen hat, die seinem Fachbereich zugeordnet sind.

13. Fazit und Ausblick

In der Vorangegangenen Arbeit wurde das Systems zur Verwaltung von elektronischen Publikationen von der Basis auf konzeptioniert, implementiert und dokumentiert.

Es hat sich herausgestellt, dass das bestehende Rahmenwerk gut erweiterbar war, da es eine ausführliche Dokumentation gab. Durch diese Arbeit wurde das Rahmenwerk um die Komponente „Publikation“ erweitert

Und der Prozess der stetigen Weiterentwicklung ist noch nicht beendet; durch die derzeitige Implementierung und Dokumentation ist eine Basis für weitere Funktionalitäten geschaffen worden.

14 Glossar

ACI	Access Control Information
ACL	Access Control List
API	Appliation Programming Interface
CGI	Common Gateway Interface
DIN	Deutsches Institut für Normung
DN	Distinguished Name
LDAP	Lightweight Directory Access Protocol
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
JDK	Java Development Kit
JSP	Java Server Pages
OOA	Object Oriented Analysis
OOD	Object Oriented Design
RDN	Relative Distinguished Name
SASL	Simple Authentication and Security Layer
SDK	Software Development Kit
SSL	Secure Sockets Layer
UI	User Interface

15 Quellen

[Web 1] http://www.awi-bremerhaven.de

[Web 2] http://www.teamone.de/sefhtml/

[Web 3] http://www.perldap.org

[Web 4] http://www.perl.com

[Web 5] http://java.sun.com/docs/books/tutorial/reflect/index.html
Java Tutorial , Trail: Reflection

[Ste2000] Stelling, K.(2000). Entwurf und Implementierung der personenbezogenen Komponente eines Informationssystems in einer Forschungseinrichtung zur Nutzung im Intra- und Internet

[Ba1999] Balzert, Heide(1999). Lehrbuch der Objektmodellierung, Analyse und Entwurf

[Wel2000] Weltman, R., Dahbura, T. (2000). LDAP Programming with Java. Addison Wesley Longman.