Bachelorarbeit

Rechteverwaltung und Workflowmodellierung im CMS Plone

Neuhaus, Manja

geboren am 20. Juli 1986 in Rodewisch

Studiengang Informatik

Westsächsische Hochschule Zwickau
Fachbereich Physikalische Technik / Informatik
Fachgruppe Informatik

Betreuer, Einrichtung: Prof. Dr. W. Golubski, WH Zwickau

Abgabetermin: 26. Februar 2009

Autorenreferat

Kurzreferat

Die vorliegende Arbeit beschäftigt sich mit der Konzeption und Implementierung eines geeigneten Sicherheitskonzepts für ein hochschulinternes Portal zur Verwaltung von Modulhandbüchern auf der Basis des Content Management Systems Plone.
Im Mittelpunkt stehen einerseits die Berechtigungen, und andererseits der Einsatz der in Plone zur Verfügung gestellten Workflow-Technologie. Weiterhin wird die Einpassung in das vorhandene System behandelt.

Abstract

This bachelor thesis is about the definition and implementation of a security concept for an in-house project of the West Saxon University of Applied Sciences of Zwickau, which allows the online management of "Module Guides" based on the content management system Plone.
The focus of this thesis is on access rights, and on the application of the built-in workflow technology of Plone. Also, the integration into the existing system is covered.

Betreuende Einrichtung

Diese Bachelor-Arbeit entstand im Auftrag der
Westsächsischen Hochschule Zwickau
Dr.-Friedrichs-Ring 2A
08056 Zwickau
Kontakt: 0375 / 536 0

und Professor Dr. Wolfgang Golubski,
Kontakt: 0375 / 536 1531

www.fh-zwickau.de

Die Westsächsische Hochschule Zwickau ist eine staatliche Hochschule mit Standorten in Zwickau, Schneeberg, Reichenbach und Markneukirchen. Sie kann auf eine lange Tradition im Bereich des Ingenieurwesens zurückblicken und stellt damit einen wichtigen Faktor in der wirtschaftlichen Entwicklung der Region dar.

Inhaltsverzeichnis

Abkürzungsverzeichnis

CMF - *Content-Management-Framework*: Programmiergerüst zur Entwicklung eines CMS
CMS - *Content Management System*: ein Hilfssystem zur Verwaltung von Inhalten
EPK - *Ereignisgesteuerte Prozesskette*: Modell zur Darstellung von Workflows
ID - *Identification*: Identifikationsnummer eines Objekts
LDAP - *Lightweight Directory Access Protocol*: Anwendungsprotokoll für Verzeichnisdienste
TAL - *Template Attribute Language*: Template-Sprache zur Generierung von HTML- und XML-Seiten
TALES - *Template Attribute Language Expression Syntax*: Syntax zu TAL
UID - *UniqueID*: Interne ID eines Plone-Objekts
ZMI - *Zope Management Interface*: Die Administrationsoberfläche von Zope

Abbildungsverzeichnis

Tabellenverzeichnis

Anlagenverzeichnis

Inhalt der beigefügten CD

1. Digitale Ausgabe der Thesis (Format: Word 97-2003-Dokument, Adobe PDF)
2. Die Quelldateien für das Plone-Produkt CTModule
3. Die importfähigen Workflow-Dateien
4. Die Workflow-Skripte als Python-Dateien
5. Die angepassten Templates

1. Einleitung und Aufgabenstellung

1.1. Motivation

Im Zuge der Vereinheitlichung und Automatisierung des Informationssystems der Westsächsischen Hochschule Zwickau wurde von der Fachgruppe Informatik ein Projekt ins Leben gerufen, das die Verwaltung der Modulhandbücher und Prüfungsordnungen samt Anlagen erleichtern soll.
Basierend auf dem Content-Management-System Plone wird eine Plattform aufgebaut, die zukünftig die jeweils aktuelle und rechtsgültige Fassung dieser Dokumente, sowie ihrer Vorgängerversionen zur Verfügung stellen soll. Weiterhin bietet diese Plattform Dozenten die Möglichkeit, ihre Lehrveranstaltungen und Module zu aktualisieren, und die Änderungen automatisiert an die Verantwortlichen zur Genehmigung weiterleiten zu lassen.
Mit dieser Arbeit soll dazu beigetragen werden, das Projekt technisch abzusichern. Dies betrifft sowohl die Einhaltung der Genehmigungskette als auch die Wahrung der Datenintegrität durch ein feinstufiges Berechtigungssystem, das es nur den jeweilig Verantwortlichen erlaubt, Änderungen durchzuführen.
Ohne diese Absicherung wäre es möglich, dass fehlerhafte Daten im rechtskräftigen Modulhandbuch entstünden, zum Beispiel durch die unautorisierte Änderung einer schriftlichen Prüfung in eine mündliche - mit allen rechtlichen Konsequenzen.

1.2. Ist-Zustand

Die erste Version dieses Modulhandbuchportals auf der Basis des Content Management Systems (CMS) Plone wurde von Marcel Riedel entwickelt. Diese wies im Test jedoch einige Lücken auf, weshalb Mathias Schraps und der Autor dieses Dokuments beauftragt wurden, im Rahmen ihres Bachelor-Projekts eine verbesserte Version zu erarbeiten.
Mathias Schraps begann mit der Neukonzipierung der Modul-Dokumentenstruktur. Es entstand das Plone-Zusatzprodukt *CTModule*. Auf diese Arbeit wurde im Folgenden aufgebaut. Details zum bestehenden System können im Kapitel 3.1 *Analyse der früheren Version und Ist-Analyse* nachgelesen werden.

1.3. Zielstellung

Das Modulverwaltungsportal soll auf den Einsatz vorbereitet werden.
Es soll ein verfeinertes Berechtigungssystem entworfen werden, das eine benutzerspezifische Rechtevergabe zulässt, wobei jeder Benutzer nur mit minimalen Rechten ausgestattet sein darf.
Es soll ein System erarbeitet werden, das dieses Berechtigungssystem auf das Produkt CTModule umsetzt.
Der vorhandene Workflow muss verfeinert oder ersetzt werden, dazu gehört sowohl die Einführung weiterer Benutzer-Hierarchieebenen, als auch das Schützen der Workflowschritte gegen eine unberechtigte Ausführung. Weitere Workflows müssen für andere Inhaltstypen abgeleitet werden, unter Berücksichtigung der Kopplung der Inhaltstypen aneinander. Ferner müssen die Skripte zum Kopieren und Verschieben der Inhaltstypen an die neue Modulstruktur angepasst und dynamisiert werden.

1.4. Gliederung

Nach dieser Einleitung werden im zweiten Kapitel zunächst die Grundlagen behandelt um in den nachfolgenden Kapiteln auf dieses Wissen aufbauen zu können. Kapitel 3 entwirft das Konzept, auf dessen Grundlage in Kapitel 4 schließlich die Implementierung besprochen wird. Kapitel 5 dient zur Zusammenfassung. Abschließend wird im gleichen Kapitel ein Ausblick auf weitere Möglichkeiten in Zusammenhang mit diesem Projekt gegeben, die den Rahmen dieser Arbeit gesprengt hätten.
Die vollständigen Skripte und exportierten Workflow-Dateien befinden sich auf der beiliegenden CD.

2. Grundlagen

2.1. Content-Management und Plone

Populärwissenschaftlich erklärt ist ein Content-Management-System (CMS) eine Anwendung, über die Inhalte wie Texte und Bilder im World Wide Web oder im Intranet[1] veröffentlicht werden können, ohne dass sich der Autor selbst um die Programmierung oder das Seitenlayout kümmern muss. Die Konfiguration übernimmt der Administrator, die Erweiterung des CMS der Programmierer.
Das Open-Source-Projekt Plone ist ein solches CMS.

2.1.1. Plone in der Nussschale: Klärung Plone-spezifischer Begriffe

Plone ist modular aufgebaut. Es liefert eine Vielzahl an Werkzeugen mit, über die jeweils ein bestimmter Bereich des Plone-Systems konfiguriert werden kann. Über das Werkzeug *portal_quickinstaller* können zum Beispiel leicht Zusatzprodukte installiert werden.

Den wichtigsten Bestandteil in einem CMS bilden, wie der Name schon sagt, Inhalte.
Inhalte werden in Plone *Inhaltselemente* genannt. Je nach Art enthalten sie vordefinierte Felder oder können unterschiedliche andere Typen wie zum Beispiel Bilder enthalten. Diese Inhaltselemente werden als Instanzen von Klassen erzeugt, welche als *Inhaltstypen* bezeichnet werden.
Durch Plone vordefinierte Inhaltstypen sind unter anderem Dokumente, News, Links, Bilder, aber auch Ordner und Themen. Letzteres wird auch als intelligenter Ordner bezeichnet und stellt im Grunde gespeicherte Suchkriterien dar, zum Beispiel für Dokumente mit speziellen Eigenschaften. Eine Übersicht über alle verfügbaren Inhaltstypen findet sich im Plone-Werkzeug *portal_types*.
Inhaltselemente können andere Inhaltselemente enthalten. Im Plone-Jargon heißen sie deshalb *folderish objects*, also *ordnerähnliche Objekte*. Welche Inhaltselemente in andere hinzugefügt werden könnten, lässt sich in *portal_types* festlegen.

Die von Mathias Schraps erstellten Inhaltstypen basieren auf einer Erweiterung normaler Inhaltstypen, dem Archetypes-Framework. Jede im Modulhandbuch vorhandene logische Einheit wurde von ihm durch einen eigenen Inhaltstyp abgebildet - es gibt Inhaltstypen für Lehrveranstaltungen und Module, für Prüfungen, Prüfungsvorleistungen, Studiengänge und für

[1] Es gibt noch weitere CMS-Arten, die zum Beispiel Inhalte für Printmedien oder das Radio verwalten. Zur Abgrenzung wird manchmal der Begriff **Web-Content-Management-System (WCSM)** für Internetanwendungen verwendet

Schwerpunkte. Diese werden in Form von Python-Klassen als neues Produkt in Plone installiert und sind ebenfalls in *portal_types* konfigurierbar. Zu jedem Inhaltstypen gibt es Standard-Views, welche die Darstellung der gespeicherten Daten regeln. Sie können angepasst oder selbst definiert werden, dazu können mit Hilfe von TAL (Template Attribute Language) sogenannte TALES-Ausdrücke verwendet werden um innerhalb dieser Templates Zugriff auf Daten zu erhalten oder Bedingungen einzuführen. Standard-Views von Archetypes-Inhaltstypen sind View (Anzeigen), Edit (Bearbeiten), Properties (Eigenschaften), Folder Listing (Inhalte, also im *folderish object* enthaltene Elemente) und Sharing (Zugriff, also alle im Inhaltselement vorkommenden Rollen).
In der Bearbeiten-Sicht werden Daten über vor- bzw. selbstdefinierte Felder im jeweiligen Inhaltselement eingetragen und über Hilfselemente (Widgets) verwaltet; eine Zeichenkette zum Beispiel durch den *StringWidget,* oder eine Verknüpfung zur einem oder mehreren anderen Inhaltselementen (*RefenceField*) durch den *ReferenceBrowserWidget*. Somit kann zwischen Datenfeldern und Referenzfeldern unterschieden werden. Wird von der Bearbeiten-Sicht auf die Ansicht gewechselt, werden die Eingabefelder in HTML als Text angezeigt und alle referenzierten Inhaltselemente als Hyperlinks gelistet.
Eine detailierte Betrachtung zu diesem Thema führt Mathias Schraps in seiner Thesis zum Thema "Modellierung und Realisierung von Inhaltstypen im CMS Plone mit Hilfe des Archtypes-Frameworks" durch [SCH09].

2.1.2. Konfiguration

"Plone hat den Anspruch, komplexe Themen von den Benutzern fern zu halten" [Fri06, S. 159]

Die Web-Oberfläche des Plone-Portals, das *Plone-Control Panel*, bietet dem Administrator vereinfachte Administrationswerkzeuge, wie zum Beispiel Erweiterungen zu installieren oder die Daten des Mailservers einzutragen. Durch Anhängen von "/manage" an die aktuelle URL kann an jeder beliebigen Stelle des Portals auf die Maschinerie hinter der Oberfläche zugegriffen werden - das Plone Management Interface. Hier sind erweiterte Konfigurationen möglich, beispielsweise können neue Rollen definiert werden.
Unter dem Plone Management Interface liegt eine weitere Konfigurationsebene, das Zope Management Interface (ZMI). Da die Zope-Architektur die Basis für Plone bildet, ähneln sich beide Konfigurationsschnittstellen sehr stark. Oder anders ausgedrückt: Plone ist ein installiertes Produkt, das auf die Funktionalitäten vom CMF Zope zurückgreift.
In Zope lassen sich systemadministrative Tätigkeiten ausführen, beispielsweise der Neustart des Zope-Servers, das Anlegen des obersten Administratoren-Accounts oder das Sichern der Datenbank. Weiterhin kann über das ZMI auf Plone selbst zugegriffen werden. Diese Schnittstelle unterscheidet sich nur in wenigen Aspekten vom Plone-Management-Interface.

2.2. Benutzer und Gruppen

Plone behandelt Benutzer vergleichbar mit anderen Portalen: Standardmäßig kann sich jeder Besucher des Plone-Portals auf dieser Seite mit einer gültigen E-Mail-Adresse registrieren. Er bekommt dann ein eignes Profil, in dem er persönliche Daten wie Name und Wohnort speichern kann, und das für andere Benutzer sichtbar ist. Diese Daten werden in der Zope-Objektdatenbank gespeichert und im Plone-Werkzeug *portal_membership* verwaltet. Dort können wiederum Einstellungen getroffen werden, die alle Benutzer betreffen, zum Beispiel, ob beim Einloggen eines neuen Benutzers automatisch ein eigener Ordner für ihn erstellt werden soll, in dem er eigene Objekte anlegen kann.
Benutzer können zur leichteren Organisation in Gruppen eingeteilt werden. Gruppen allein stellen kein Werkzeug zur Rechteverwaltung dar, sondern ein reines Organisationswerkzeug.
Die Benutzer- und Gruppenverwaltung findet man im *Plone-Control Panel*, es ist direkt erreichbar über das Anfügen von */prefs_users_overview* an die aktuelle URL.
In diesem Werkzeug können neue Benutzer und Gruppen erstellt, sowie einander zugeordnet werden. Hier findet ebenfalls die globale Rollenverwaltung statt (Siehe auch Kapitel 2.3.2 *Rollen im Detail*).

Neue Benutzer können außerdem sowohl im ZMI als auch im Plone-Verwaltungsbereich angelegt werden, aber nur in Zope angelegte Benutzer können das Administrationszugriff auf Zope erhalten. Der interne Benutzerordner heißt *acl_users*, das entsprechende Plone-Werkzeug heißt *portal_registration*.
Ein Nutzer wird erst dann aktiv, wenn er sich zum ersten Mal auf der Plone-Seite einloggt. Erst dann wird der persönliche Ordner erstellt.

2.3. Das Rechtemanagement von Zope / Plone

Plone basiert auf dem Anwendungsserver Zope und dessen Content-Management-Framework. Aus diesem Grund baut auch das Rechtesystem von Plone auf Zope auf.

2.3.1. Rollen und Berechtigungen

Eine Rolle ist die Summe ihrer Berechtigungen.

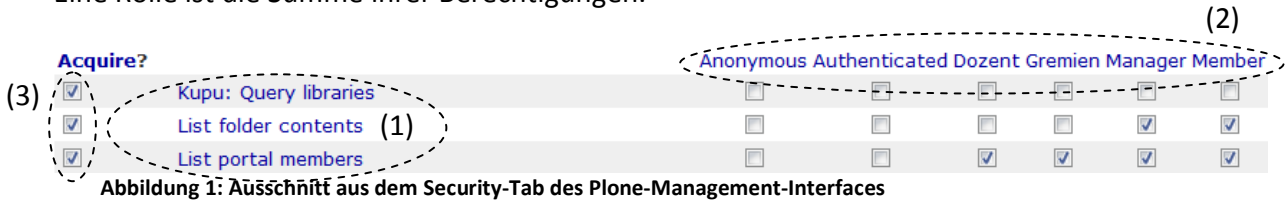

Abbildung 1: Ausschnitt aus dem Security-Tab des Plone-Management-Interfaces

Eine *Berechtigung* (1) (siehe Abbildung 1, Seite 5) ist die Erlaubnis, eine bestimmte Aktion auszuführen, zum Beispiel "List folder contents" - das Recht, den Inhalt von Ordnern anzeigen zu lassen. In Plone gibt es über 200 verschiedene Einzelberechtigungen, die per Mausklick zu einer *Rolle* (2) zusammengefasst werden können. Die obere Abbildung zeigt einen Ausschnitt von drei Berechtigungen. Ein gesetzter Haken bedeutet, dass einer Rolle genau dieses Recht gegeben wurde.
Rollen können, müssen aber nicht überall in Plone den gleichen Berechtigungssatz beinhalten. Tatsächlich können in jedem Ordner die Berechtigungen neu vergeben werden, was jedoch sehr schnell zu einem unübersichtlichen Sicherheitskonzept führt.
Die wichtigsten Berechtigungen für einen Dokumenten-Workflow sind: *View* (zur Anzeige eines Dokuments), *Access contents information* (zur Anzeige von Zusatzinformationen), *Modify portal content* (zum Bearbeiten), *List folder contents* (zur Anzeige von Ordner-Inhalten) oder auch *Use external editor* (die Erlaubnis, Dateien mit einem Editor außerhalb von Plone zu bearbeiten).
Weiterhin existiert eine Reihe von administrativen Berechtigungen wie *Manage users* und *Manage Groups* (das Recht, Benutzer bzw. Gruppen hinzuzufügen, zu bearbeiten und zu löschen) oder das Recht, das System zu erweitern mit *Add Python Scripts* bzw. *Change Python Scripts.*

2.3.2. Rollen im Detail

In Plone gibt es sechs vordefinierte *Standard-Rollen*: ANONYMOUS, AUTHENTICATED, MEMBER, OWNER, REVIEWER und MANAGER.
Unangemeldete Benutzer haben automatisch die Rolle ANONYMOUS inne. Auf diese Weise kann genau festgelegt werden, welche Inhalte für unangemeldete Benutzer zugänglich sind.
AUTHENTICATED (authentifiziert) ist das Gegenteil von ANONYMOUS, zu dieser Rolle gehören automatisch alle angemeldeten Benutzer.
Die *Zuweisung* aller übrigen Rollen kann sowohl an Benutzer als auch an Benutzergruppen erfolgen.
Einem Benutzer können beliebig viele Rollen zugeordnet werden. Wenn einer Rolle eine Berechtigung gegeben wurde, in einer anderen aber nicht, gilt die *gesetzte* Berechtigung.

Die Rolle MEMBER ist vergleichbar mit AUTHENTICATED. Es ist die Standardrolle eines angemeldeten Benutzers; standardmäßig bekommt ein solcher Benutzer einen eigenen Ordner, dessen Besitzer er ist. Für diesen Ordner besitzt er die Rolle OWNER (dt. Besitzer). Dort kann er eigene Inhalte anlegen.
REVIEWER, oder auch Redakteure genannt, haben das Recht, Inhalte von anderen Benutzern zu bearbeiten und zu veröffentlichen.
Zur MANAGER-Rolle kommen noch administrative Rechte hinzu, zum Beispiel das Recht zum Hinzufügen und Löschen von Benutzern und das Verändern von Systemeinstellungen.
Im Prinzip reichen die von Plone vorgegebenen Rollen für redaktionelle Systeme aus. Zur feineren Abstufung können jedoch eigene Rollen definiert werden.

Am sinnvollsten ist es, eine neue Rolle im Wurzelverzeichnis von Plone zu definieren, da diese dann im ganzen Plone-System zur Verfügung stellt. Rollen müssen an zwei verschiedenen Stellen auf der Management-Oberfläche hinzugefügt werden, zum einen im Tab *Security* des (Wurzel-) Ordners, zum anderen in */acl_users/portal_role_manager*. Letzteres sorgt dafür, dass die Rollen in der Benutzer- und Gruppenverwaltung verwendet werden können. Diese ist über das *Plone-Control Panel* erreichbar.
Es ist möglich, neue Rollen an beinahe jeder Stelle im System zu definieren. Diese sind ab diesem Ordner für alle darin befindlichen Unterordner gültig. Dieses Vorgehen ist jedoch wenig systematisch und führt schnell zur Unübersichtlichkeit. Um dennoch Rollen örtlich begrenzen zu können, gibt es die sogenannten lokalen Rollen.

Diese lokalen Rollen ermöglichen es, Benutzern oder Gruppen erweiterte Rechte für ein bestimmtes Inhaltselement zu geben, anstatt wie bei globalen Rollen für alle Objekte eines Inhalts*typs*.
Wenn zum Beispiel MEMBER A ein Dokument erstellt und möchte, dass MEMBER B das Dokument direkt bearbeiten kann ohne dass er selbst seine OWNER-Rolle verliert, kann er MEMBER B für dieses Dokument die lokale Rolle OWNER zuweisen. Somit sind beide Besitzer des Dokuments.
Es können alle im System zuweisbaren Rollen verwendet werden, das heißt, alle Rollen außer AUTHENTICATED und ANONYMOUS. Lokale Rollen werden in der gleichen Weise wie globale Rollen an Unterobjekte vererbt (Siehe Kapitel 2.3.3 *Akquisition*).
Durch Anhängen von /manage_listLocalRoles an die URL des aktuellen Inhaltselements gelangt man auf die Plone Management-Seite mit der Übersicht über die lokalen Rollen des Objekts; dort kann ein Benutzername eingegeben werden, dem durch Mausklick eine oder mehrere der Rollen aus der Liste zugeordnet wird.

Eine weitere objektbezogene Rollenart sind *Proxy-Rollen*. Sie werden in Workflows benötigt, wenn der Benutzer für das Ausführen einer bestimmten Aktionen im Workflow nicht berechtigt ist, aber der Workflow dennoch ausgeführt werden soll ohne dem Benutzer erweiterte Rechte oder eine andere Rolle zu geben. Die Rolle wird also nicht an ein Inhaltselement, sondern an ein Workflow-Skript vergeben.

2.3.3. Akquisition

Im Normalfall werden Berechtigungen von einem Ordner an seine Unterordner vererbt. Dies wird in Plone als *Akquisition* (3) (siehe Abbildung 1, Seite 5) bezeichnet. Durch eine hierarchische Ordnerstruktur entsteht vom Wurzelordner bis zum tiefsten Unterordner eine Akquisitionskette.
Diese Akquisitionskette kann an beliebigen Stellen für einzelne Berechtigungen unterbrochen werden durch Entfernen des Häkchens bei "Acquire?". Die im aktuellen Ordner veränderten Berechtigungseinstellungen werden nun an alle darunter liegenden Ordner vererbt.
Um einen Überblick zu erhalten, für welche Rollen eine Berechtigung an der aktuellen Stelle gesetzt ist, genügt es, die entsprechende Berechtigung anzuklicken.

In Abbildung 1 (Seite 5) finden sich zwei Typen von Rollen: Zum einen die von Zope definierten Rollen ANONYMOUS, AUTHENTICATED und MANAGER, zum anderen die in Plone selbstdefinierten Rollen DOZENT und GREMIEN und die Plone-Standard-Rolle MEMBER.
Da diese Seite das Wurzelverzeichnis von Plone ist, werden hier erstmals die Berechtigungen für die in Plone neu hinzugekommenen Rollen gesetzt.
Die Zope-Rollen hingegen wurden im ZMI definiert und dort mit Berechtigungen versehen. Diese werden nun durch Akquisition an Plone vererbt. Deshalb müssen die Häkchen für diese Rollen in Plone nicht noch einmal gesetzt werden, außer in "Acquire?". Wie in Abbildung 1 zu sehen ist, sind bei der Zope-Rolle MANAGER trotz Akquisition standardmäßig Berechtigungen gesetzt worden. Dies sind die gleichen Werte, wie sie im ZMI zu finden sind. Das Setzen der Rechte an dieser Stelle ist somit redundant, kann aber aus Vorsichtsgründen geschehen um das unbeabsichtigte Entziehen der eigenen Rechte zu vermeiden.

2.4. Die Workflowtechnologe

Der Begriff *Workflow* kommt aus dem Englischen und bedeutet schlicht *Arbeitsfluss* oder *Ablaufplan*. In Plone steuert der Workflow den Weg, den ein Inhaltselement von seiner Erstellung bis zur Veröffentlichung nehmen kann. Er legt idealerweise genau fest, wer zu welchem Zeitpunkt welche Aktion auf welches Objekt eines Inhaltstyps ausführen darf, und in welchen Zustand das Inhaltselement dann übergeht.
Die Workflow-Technologie ist also eine zeitlich abhängige Komponente im Sicherheitskonzept.

2.4.1. Überblick

Am besten lässt sich die Funktionsweise von Workflows anhand eines Beispiels erklären.
Die folgende Darstellung bildet den Folder-Workflow, von Plone ab - also den Workflow, der standardmäßig alle Ordner steuert. Jedem Inhaltstypen kann gesondert ein Workflow zugeordnet werden. Dieser Workflow greift dann bei allen Inhaltselementen dieses Typs.

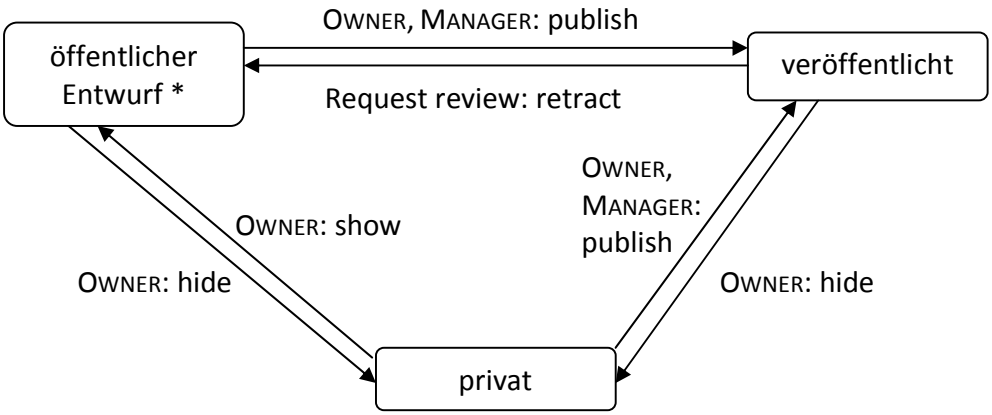

Abbildung 2: Graphische Darstellung des *Folder*-Workflows

Zu jedem Zeitpunkt befindet sich ein Plone-Objekt, wie in diesem Beispiel ein Ordner, in einem eindeutigen Zustand. Dieser sogenannte Status wird im Plone-Portal in der Statusleiste eines jeden Objekts angezeigt. In diesem Fall kann ein Ordner drei Zustände annehmen: Privat, veröffentlicht und öffentlicher Entwurf. Der Status regelt gleichzeitig die Zugriffsrechte auf den Ordner. Private Ordner können nur von ihren Besitzern oder Managern eingesehen werden, veröffentlichte Ordner hingegen von allen Nutzern. Dazwischen kann es je nach Bedarf verfeinerte Abstufungen geben. Bei der Implementierung des Workflows wird festgelegt, welchen Anfangszustand jeder neu angelegte Ordner annimmt. Hier ist dieser Zustand, wie in Plone, mit einem Stern gekennzeichnet.
Um von einem Status in einen anderen zu gelangen, werden sogenannte *Transitions* (Übergänge) definiert, hier dargestellt durch Pfeile. Um also einen Ordner vom Zustand "privat" in den Zustand "veröffentlicht" zu versetzen, muss der Übergang "publish" ausgelöst werden. Bevor ein Nutzer einen Übergang durchführen kann, wird durch Plone geprüft, ob der Nutzer die notwendige Berechtigung oder Rolle dafür besitzt. Dies lässt sich im entsprechenden Übergang festlegen.
Mit der von Plone bereitgestellten Technologie lässt sich also die gesamte Modulverwaltung der Westsächsischen Hochschule abbilden, inklusive Sicherheitsüberprüfungen und Benachrichtigungen.
Im Folgenden werden die Workflow-Funktionalitäten detailliert betrachtet.

2.4.2. Zustände (States)

Abbildung 3 zeigt einen Ausschnitt aus dem Standard-Ordner-Workflow im Werkzeug *portal_workflow*. Abgebildet ist der *States*-Tab. In diesem Bereich werden alle möglichen Zustände definiert, die ein Objekt im Laufe des Workflows annehmen kann. (Vergleich Abbildung 2, Seite 8)

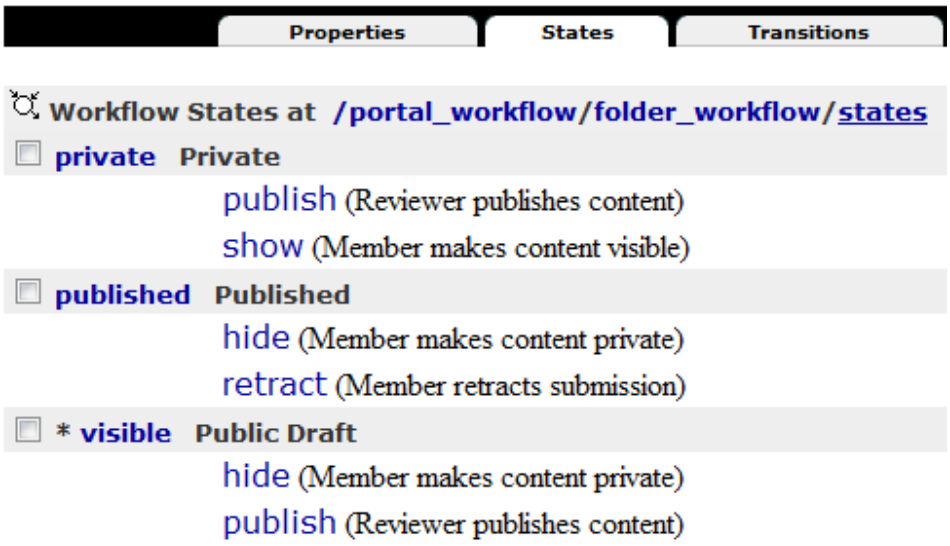

Abbildung 3: *States*-Tab des Ordner-Standard-Workflows

An erster Stelle neben jedem Kästchen steht die ID des Zustands, daneben der im Portal angezeigte Name des Zustands. Werden die originalen englischen Bezeichnungen verwendet, übersetzt Plone diese automatisch im Portal ins Deutsche. Nur auf diese Weise können die vorgefertigten Statusfunktionalitäten von Plone wie Berechtigungen und Farbkennzeichnung genutzt werden. (Siehe nachfolgende Tabelle)
Mit dem Stern ist der *initial state*, der Anfangszustand, gekennzeichnet - in diesem Fall *visible.* Wird ein neuer Ordner angelegt, erhält er automatisch diesen Zustand.

Tabelle 1: Standard-Zustände

ID	Titel	Übersetzung	Sichtbarkeit	Bearbeitungsrecht	Farbe
pending	Pending	Wartend	öffentlich	OWNER, MANAGER, REVIEWER	orange
private	Private	Privat	nicht öffentlich	OWNER, MANAGER	rot
published	Published	Veröffentlicht	öffentlich	MANAGER	blau
visible	Public Draft	Sichtbar	öffentlich	OWNER, MANAGER	grün

Neben den vier Grundzuständen *sichtbar, wartend, veröffentlicht* und *privat* können beliebig viele weitere Zustände angelegt werden um Zwischenschritte modellieren zu können. Dies wird nötig, wenn - wie in der Hochschule - ein Dokument durch mehrere Instanzen gereicht werden muss, bevor es veröffentlicht werden darf.
In seiner Funktion als redaktionelles System sieht Plone in seiner Grundeinstellung vor, dass nur bestimmte Personen den *veröffentlichten* Inhalt zusammenstellen dürfen. Für alle Objekte (außer Ordner) gibt es also einen Zustand *wartend*, in dem sie zwar öffentlich einsehbar sind, aber erst durch einen Redakteur freigeschaltet werden muss bzw. bearbeitet werden kann. Der Begriff "öffentlich einsehbar" ist hierbei im Gegensatz zu "privat" zu sehen. Im privaten Zustand ist ein Objekt nur durch seinen Besitzer oder einen Manager einsehbar und veränderbar.
Es gibt einen privaten Zustand und drei öffentliche Zustände, von denen jeder einen bestimmten Zweck erfüllt. Der oben erwähnte Zustand *wartend* dient speziell der Benachrichtigung von REVIEWERN, wenn ein Artikel von einem MEMBER fertiggestellt wurde. Dieser Artikel wurde im Zustand *sichtbar* angelegt, und durch Umschalten auf *wartend* signalisiert der Autor, dass der Artikel bereit zur *Veröffentlichung* ist.
Veröffentlichte Objekte können nur noch von Managern direkt bearbeitet werden, der Besitzer kann das Objekt nur noch indirekt bearbeiten: Er muss es erst in den Zustand *sichtbar* zurücksetzen, bevor er wieder einen *bearbeiten*-Tab erhält.

2.4.3. Übergänge (Transitions)

Jedem Zustand lässt sich eine beliebig große Menge von vordefinierten Übergängen zuordnen. Die Übergänge steuern somit den Workflow und bieten im definierten Rahmen Entscheidungsmöglichkeiten, welcher Zustand als nächstes eingenommen werden soll.

Vor und nach dem Übergang können durch den Workflow Skripte aufgerufen werden, in denen man auf Variablen des aktuellen Workflowschritts zugreifen, und auch in den Workflow selbst eingreifen kann, wie in Kapitel 4.3.1 im Zusammenhang mit unsichtbaren Übergängen gezeigt wird.

Jeder Übergang kann durch eine sogenannte *Wächterbedingung* (Guard) geschützt werden, sodass jene Übergänge nur von berechtigten Personen ausgeführt werden können. Dies kann eine Berechtigung, eine Rolle, eine Gruppe oder ein TALES-Ausdruck sein. [Fri06]

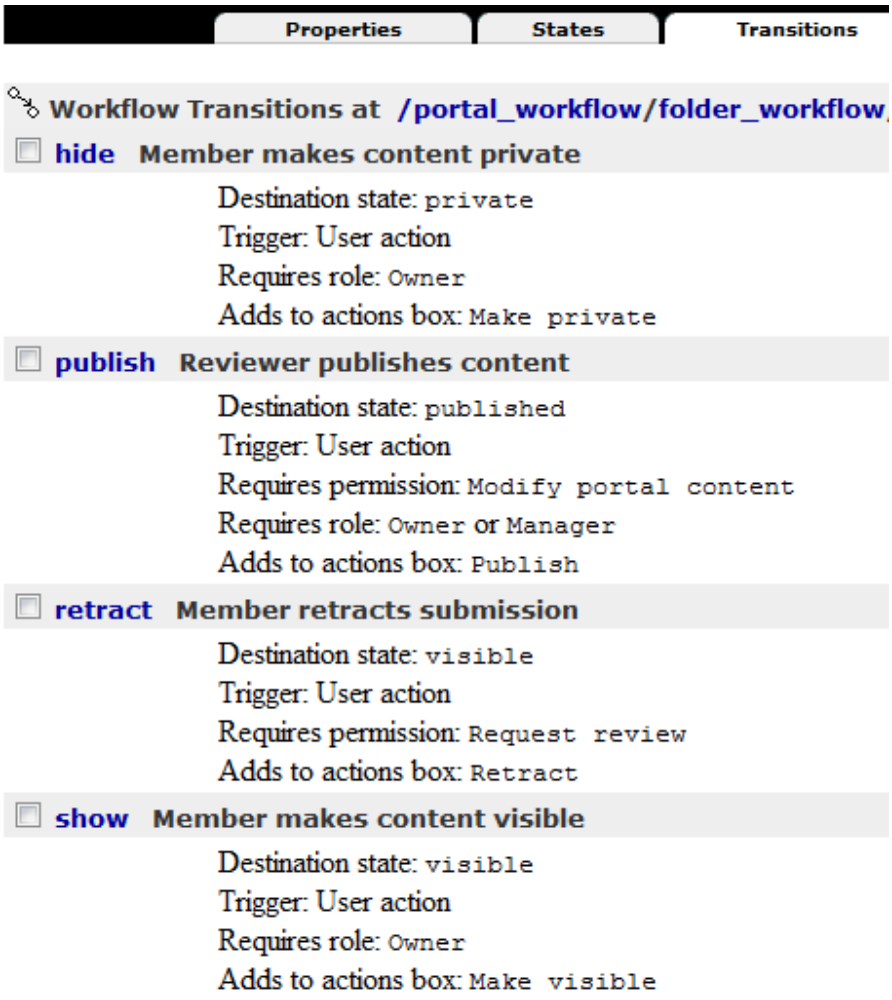

Abbildung 4: *Transitions*-Tab des Ordner-Standard-Workflows

Schließlich wird je Übergang genau ein Zustand festgelegt, welcher nach dem Übergang erreicht wird. Ebenso kann die Option gewählt werden, im aktuellen Zustand zu verbleiben. Auch davon wird im weiteren Verlauf Gebrauch gemacht.

Tabelle 2: Standard-Übergänge

ID	Übersetzung	Berechtigungen / Rollen
hide	privat schalten	OWNER
publish	veröffentlichen	Review portal content
reject	ablehnen	Review portal content
retract	zurückziehen	Request review
show	sichtbar machen	OWNER
submit	einreichen	Request review

2.4.4. Weitere Funktionalitäten

In *Variablen* sind Informationen über ein Objekt im Workflow speicherbar, beispielsweise das Datum der letzten Statusänderung oder der aktuelle Status im Workflow. Darauf kann in Skripten zugegriffen werden.

Eine *Worklist* ist eine Suchabfrage, die zum Beispiel alle Dokumente im Zustand "wartend auf Freigabe" zusammenstellt und den Redakteuren als *Portlet* - eine eingeblendete Box in Form einer Link-Liste - anzeigt. Der Plone-Standardworkflow implementiert die eben beschriebene Funktionalität als Worklist unter dem Namen "reviewer_queue".

Im Tab *Scripts* können Python-Skripte angelegt werden. Übergabeparameter ist normalerweise state_change, darin ist es Objekt selbst enthalten, auf das der Workflowschritt angewendet wird. Ob ein Skript vor oder nach dem Übergang ausgeführt wird, spielt eine Rolle: Wenn es zum Beispiel nach dem Übergang aufgerufen wird, wird sicher gestellt, dass der Übergang erfolgreich ausgeführt wurde. Wird das Skript jedoch vor dem Übergang aufgerufen, kann beispielsweise verhindert werden, dass ein Inhaltselement in einen anderen Zustand wechselt, wenn eine bestimmte Bedingung nicht erfüllt ist.

3. Konzeption

"Sicherheit ist schwer zu erreichen." - Jim Fulton, Chefentwickler von Zope [McK05, S. 269]

3.1. Analyse der früheren Version und Ist-Analyse

Die Vorgängerversion implementierte bereits einen Workflow, der das Workflow-Skript aus [MaK05, S. 262] verwendete um Module in den eigenen Ordner jenes Benutzers zu verschieben, der den Übergang "sperren und bearbeiten" ausgelöst hat. In diesem Ordner konnte das Modul bearbeitet und zur Freigabe eingereicht werden.
Das Rechtesystem war grobmaschig. Es gab keinerlei Zuordnung, wer tatsächlich für das Modul verantwortlich war, und nicht alle Workflowschritte waren mit Berechtigungen geschützt, somit konnte jeder Benutzer beliebige Module sperren und bearbeiten.
Das Rückverschieben der Module war statisch auf zwei Ordner beschränkt, die den Studiengängen Informatik Bachelor und Informatik Master entsprachen. Für jeden weiteren Studiengang wären zwei weitere Workflow-Skripte notwendig gewesen. Zusätzlich wäre die Liste der möglichen Übergänge auf der Anwenderseite sehr lang geworden, wenn das Vorgängersystem hochschulweit übernommen worden wäre.
Ein weiterer Kritikpunkt war, dass durch nur eine vorhandene Genehmigungsstufe der Workflow nicht dem realen Workflow der Westsächsischen Hochschule entsprach.

In der neuen Version des Modulhandbuchportals wurde die Struktur der Inhaltstypen grundlegend verändert und verfeinert. Neben Modulen existieren nun weitere Inhaltstypen wie Lehrveranstaltungen, Studiengänge und Prüfungen, die mit eigenen Workflows gesteuert werden müssen.

Strukturell gesehen ist die Lehrveranstaltung eines Dozenten das Kernelement des Modulsystems. Das Modul selbst wird als *folderish objekt* genutzt. Es kann beliebig viele Lehrveranstaltungen, Prüfungen und Prüfungsvorleistungen enthalten. Das Modul liegt mit allen enthaltenden Inhaltselementen in einem öffentlichen Modulordner, der jedoch nicht mit dem Containerformat des Moduls verwechselt werden sollte.
Auf gleicher Ebene zum Modulordner gibt es einen Studiengangordner, in dem Inhaltselemente vom Typ Studiengang gespeichert werden. Diese sind wiederum *folderish objekts* und können Schwerpunkte und Wahlpflichtkataloge enthalten.
Die Verknüpfung vom Studiengang zum Modul geschieht nicht wie bisher durch Unterordnung, sondern durch ein Referenzfeld.

3.2. Berechtigungen und Rollen

Das grundlegende zu lösende Problem war, wie in Plone ohne übermäßigen Konfigurationsaufwand die Rechte so gesetzt werden können, dass eindeutig festgelegt ist, welcher Benutzer Änderungen an welchen Dokumenten vornehmen darf.

Von der realen Struktur ausgehend sind vier Benutzertypen zu modellieren:

- Einen Dozenten, der für seine Lehrveranstaltung verantwortlich ist,
- Einen Modulverantwortlichen für jedes Modul und die dazugehörigen Prüfungsvorleistungen, Prüfungen und Lehrveranstaltungen
- Einen Studiengangsverantwortlichen für jeden Studiengang, und
- Einen Supervisor, der in letzter Instanz entscheidet, ob ein Inhalt zugelassen wird. Im Folgenden wird dieser Benutzer als "Gremien" bezeichnet.

Die Dozenten und Modulverantwortlichen müssen Bearbeitungsrechte für Inhaltselemente erhalten, für den Studiengangsverantwortlichen und die Gremien genügen Genehmigungs- und Freigabe-Berechtigungen im Workflow.
Um Benutzergruppen festzulegen, gibt es zwei Möglichkeiten: Gruppen und Rollen. Gruppen sind jedoch ein reines Organisationswerkzeug und somit ungeeignet für Berechtigungen. Es bleiben die verschiedenen Rollenarten. Globale Rollen würden nur wie in der Vorgängerversion für alle Inhaltselemente jedes Inhaltstyps Gültigkeit haben, das heißt, jeder Dozent könnte jede Lehrveranstaltung bearbeiten. Um jede Lehrveranstaltung spezifisch einem Dozenten mit dem entsprechend definierten Rechtesatz zuordnen zu können, werden lokale Rollen benötigt. Diese müssen zunächst als globale Rollen definiert werden um in der Zuordnungsliste Im ZMI verfügbar zu sein. Sie erhalten einen minimalen Standard-Berechtigungssatz, vergleichbar mit der Rolle MEMBER. Mit möglicherweise wachsenden Anforderungen an die Berechtigungsstruktur, wie zum Beispiel eine Druckberechtigung, könnten jederzeit weitere Einzelrechte eingeräumt werden.

Die Funktionsfähigkeit dieses Konzepts wurde durch einen einfachen Test bestätigt: Es wurde ein Modul mit zwei Lehrveranstaltungen angelegt, dem Modul und der *Lehrveranstaltung A* wurde *Benutzer A* als jeweils MODULVERANTWORTLICHER und DOZENT zugeordnet, der *Lehrveranstaltung B* den *Benutzer B* als DOZENT. Die Rollen DOZENT und MODULVERANTWORTLICHER erhielten das Recht "modify portal content". Das Ergebnis war, dass *Benutzer B* nur bei *Lehrveranstaltung B* die Bearbeiten-Sicht erhielt, welche durch selbiges Recht geschützt ist, während *Benutzer A* bei dem Modul und beiden Lehrveranstaltungen die Bearbeiten-Sicht erhielt. Durch Deaktivieren der Akquisition konnte erreicht werden, dass *Benutzer A* nur noch das Modul und die *Lehrveranstaltung A* bearbeiten konnte.

Mit der so erreichten Flexibilität entschied sich der Autor gegen die vordefinierten Rollen um die benötigten Benutzertypen abzubilden. Zusätzlich ergab sich dadurch die Möglichkeit, inhaltlich eindeutige Rollennamen zu verwenden. Dies sind konkret:

Tabelle 3: Rollenübersicht

Rollenname	Inhaltstyp
Dozent	CTLehrveranstaltung
Modulverantwortlicher	CTModul
Studiengangsverantwortlicher	CTStudiengang
Gremien	keiner, globale Rolle

Für die Gremien existiert kein Inhaltstyp, es ist einer Redakteur-Rolle, die nur zum Freigeben von vorgeprüften Inhalten dient.
Unabhängig davon wird weiterhin der Administrator in der MANAGER-Rolle gebraucht um Systemeinstellungen vornehmen zu können, und die OWNER-Rolle zum Bearbeiten von Inhaltselementen. Ebenso muss die Rolle ANONYMOUS erhalten bleiben, die alle Besucher des Portals ohne Benutzerkonto umfasst und somit den Zugriff der Studenten steuert, die das Portal benutzen können ohne Änderungen vornehmen zu dürfen.
Diese Rolle eröffnet zum Beispiel die Möglichkeit, bestimmte Bereiche des Portals auszublenden.

3.3. Klassenstruktur und Benutzer

Prinzipiell ist es nicht notwendig, eigene Inhaltstypen für Benutzer zu erstellen, da sämtliche vorher beschriebenen Berechtigungseinstellungen durch den Administrator über ZMI an den Inhaltselementen Modul, Lehrveranstaltung und Studiengang selbst vorgenommen werden können. Unter ergonomischen Aspekten ist eine eigene Anwendungsoberfläche jedoch sinnvoll. Dazu muss im Schema aller drei Inhaltstypen ein neues Feld für den verantwortlichen Benutzer eingefügt werden.
Es fehlt nun nur noch das Bindeglied von den Inhaltstypen zu den lokalen Rollen. Für diesen Zweck wurden die von Matthias Schraps generierten Klassen CTDozent und CTFachbereichsFolder übernommen, wobei CTFachbereichsFolder in CTDozentenFolder umbenannt wurde und CTFachbereich entfernt wurde. CTDozentenFolder ist ein Ordner für Dozenten. Zur logischen Organisation und leichteren Auffindung können innerhalb dieses Ordners weitere Ordner angelegt werden, zum Beispiel in Form von Fachgruppen- oder Fakultäten-Ordnern. Auf diese Weise entfällt die Notwendigkeit einer Organisation in Plone-Benutzergruppen vollständig.

Die folgende Abbildung zeigt einen reduzierten Ausschnitt aus dem Klassendiagramm des Projekts CTModule, erstellt mit ArgoUML[2]. Angegeben sind nur die im Rahmen dieser Thesis relevanten Datenfelder und Methoden. Die von Mathias Schraps modellierten Felder und Methoden wurden hier ausgespart um das Klassendiagramm übersichtlich zu halten.

[2] ArgoUML Projekt-Homepage: http://argouml.tigris.org/

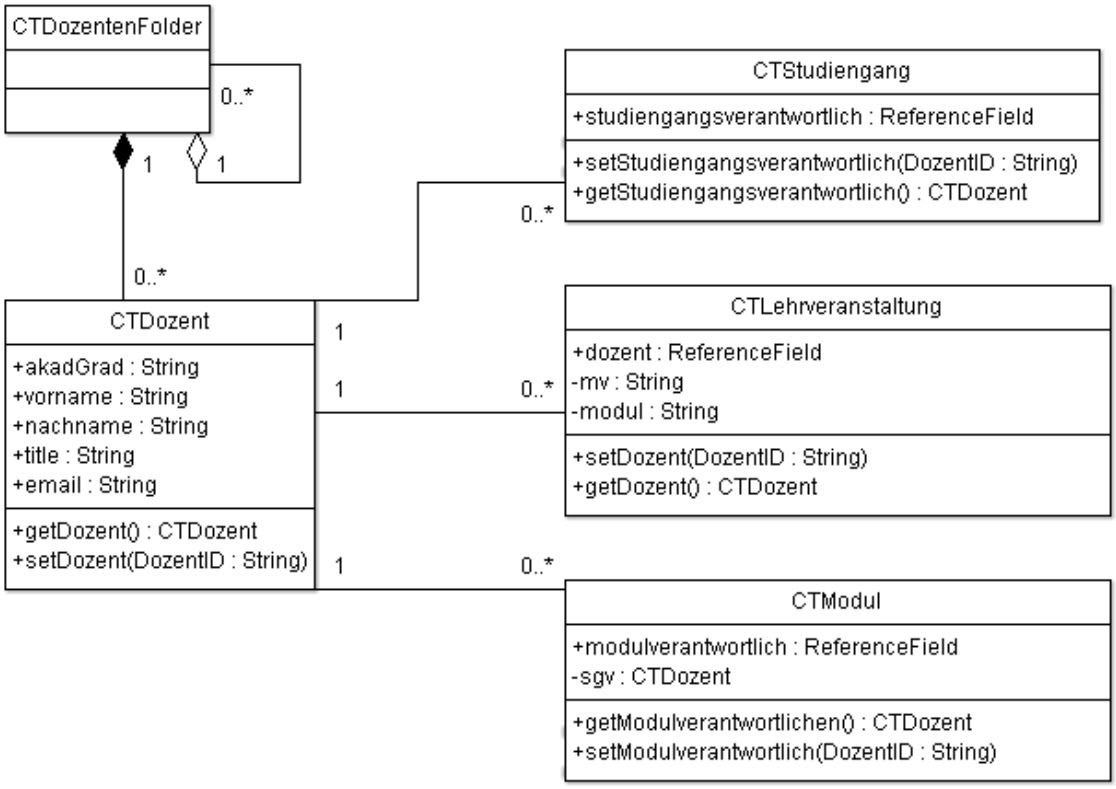

Abbildung 5: Relevanter Ausschnitt des Klassendiagramms

Der Inhaltstyp CTDozent nimmt die benötigten persönlichen Daten des Benutzers auf, dazu gehört Benutzername, realer Name, akademischer Grad und E-Mail-Adresse.
CTLehrveranstaltung enthält das Referenzfeld Dozent, welches die Referenz auf einen Dozenten speichert, CTModul enthält eine Referenz auf CTDozent mit dem Namen "Modulverantwortlich", CTStudiengang erhält einen *Studiengangsverantwortlichen*. Jeder Dozent kann beliebig viele Referenzen auf verschiedene Inhaltselemente erhalten, jedoch erhält jedes Inhaltselement nur einen direkt zugeordneten Benutzer. Es kommt zwar vor, dass eine Lehrveranstaltung von einer Gruppe von Dozenten gehalten wird, wie zum Beispiel *PTI001 Mathematik/Algebra* durch die Fachgruppe Mathematik, jedoch wurde in Absprache mit dem Betreuer dieser Arbeit festgelegt, dass es sinnvoller ist, in diesem Fall einen Benutzer mit der Bezeichnung "Fachgruppe Mathematik" einzusetzen.
Alle diese Klassen müssen ebenfalls Get- und Set-Methoden besitzen, um den Benutzernamen auslesen und setzen zu können. Weitere Methoden sind nicht nötig, da die gesamte Ablauflogik und Rechteverwaltung über den Workflow gesteuert wird. Die Inhaltselemente sollen nur die dafür notwendigen Daten bereitstellen.
Diese Daten sind zum Beispiel: Welcher Dozent ist in der Lehrveranstaltung xyz gesetzt, wie lautet die Bezeichnung des dazugehörigen Moduls und wer ist der Verantwortliche dieses Moduls? Da die Verfügbarkeit dieser Informationen nicht an allen Stellen des Workflows gewährleistet ist, sind weitere, versteckte Datenfelder nötig, die diese Verbindungen innerhalb der Inhaltselemente speichern.

3.4. Workflowgestaltung

Das Modulportal setzt sich aus einer Reihe von Inhaltstypen zusammen, die auf verschiedene Weise miteinander gelinkt oder ineinander geschachtelt sind. Dementsprechend müssen die Workflows der einzelnen Inhaltstypen ineinander übergreifen, um die Funktionalität abbilden zu können, die als Anforderung stand. In Worte gefasst, findet folgender Ablauf statt:

Der MANAGER legt die Grundlage für die Arbeit der Dozenten; er legt ein Gerüst aus Modulen, Prüfungen, Prüfungsvorleistungen und Lehrveranstaltungen im Zustand "privat" in einem öffentlichen Modulordner an. Sie enthalten im Normalfall nur eine ID und den Namen eines Verantwortlichen. Diese Module bleiben *privat*, solange sie nicht den vollständigen Workflow durchlaufen haben. Auf diese Weise sind unvollständige Dokumente von außen, das heißt durch unangemeldete Benutzer, nicht einsehbar.
Als nächstes legt der MANAGER einen Studiengang an und verlinkt die angelegten Module dort. Durch den an den Studiengang gekoppelten Workflow wird ein Skript aufgerufen, das in allen gelinkten Modulen die Rolle des Studiengangsverantwortlichen mit dem gewählten Benutzer verbindet und dem Benutzer selbst Bearbeitungsrechte für den Studiengang gibt.

Nun gibt der Manager ein Modul nach dem anderen zur Inhaltserstellung frei, indem er den Workflowschritt "initialisieren" aufruft. In diesem Schritt werden die entsprechenden Rechte an den Modulverantwortlichen und die Dozenten vergeben. Die Dokumente verbleiben im Zustand "privat", jedoch werden Kopien in den Mitgliedsordnern der jeweiligen Verantwortlichen im Zustand "sichtbar" abgelegt. Da sie OWNER ihres eigenen Verzeichnisses sind, wird durch das Prinzip der Akquisition das Besitzerrecht - und somit das Recht zum Bearbeiten - an die Kopien der Inhaltselemente vererbt, solange sie sich im Benutzer-Ordner befinden. Die Dozenten erhalten ihre Lehrveranstaltung, Modulverantwortliche das komplette Modul und extra dazu ihre eigene Lehrveranstaltung zur Bearbeitung. Die Verantwortlichen werden benachrichtigt und pflegen nun den entsprechenden Inhalt ein.
Nach der Fertigstellung ändern die Dozenten den Status ihrer Lehrveranstaltung in "einreichen". Die Lehrveranstaltung wird nun in den Ordner des Modulverantwortlichen verschoben und ihr Status auf "wartend" gesetzt. Wenn der Dozent ungleich dem Modulverantwortlichen ist, erhält der Modulverantwortliche die Nachricht, dass eine Lehrveranstaltung auf ihre Genehmigung wartet. Wenn er mit der Lehrveranstaltung unzufrieden ist, kann er sie *ablehnen*. Über den Punkt "erweitert" kann er eine Begründung eingeben, die in der E-Mail an den Dozenten weitergegeben wird. Der Dozent erhält wieder eine Kopie der Lehrveranstaltung und kann sie bearbeiten und erneut *einreichen*. Der Dozent kann die eingereichte Lehrveranstaltung auch selbst zur Bearbeitung zurückziehen, indem er den Übergang "abbrechen" wählt.

Ist der Modulverantwortliche jedoch mit der Lehrveranstaltung zufrieden, kann er sie genehmigen. Dies muss mit allen Objekten innerhalb des Moduls geschehen, wobei Prüfungen und Prüfungsvorleistungen allein in der Verantwortung des Modulverantwortlichen liegen und nur mit "ok" gekennzeichnet werden, wenn er sie fertig bearbeitet hat. Erst wenn alle

Unterobjekte auf "genehmigt" stehen, kann er das gesamte Modul an den Studiengangsverantwortlichen *einreichen*. Es befindet sich weiterhin im Mitgliedsordner des Modulverantwortlichen, da die höheren Instanzen keine Schreibrechte auf den Inhalt benötigen, sondern nur das Modul über den Workflow genehmigen oder ablehnen soll.
Der Studiengangsverantwortliche erhält eine Nachricht, kontrolliert das Modul und reicht es an die Gremien ein, wobei es im Ordner verbleibt. Wenn er es ablehnt, geht es nicht in den Ausgangsstatus zurück, sondern in den Zustand, in welchem es der Modulverantwortliche an den Studiengangsverantwortlichen einreichen kann. Hier kann der Modulverantwortliche nun anhand der Begründung des Studiengangsverantwortlichen selbst entscheiden, welche Lehrveranstaltung er noch einmal zur Bearbeitung an den Dozenten zurückkopiert, indem er den Übergang "ablehnen" im jeweiligen Dokument wählt. Nach eigenem Ermessen kann der Modulverantwortliche die Änderung auch selbst vornehmen, da er die vollen Rechte auf sein Modul besitzt.

Nach dem Studiengangsverantwortlichen sind die Gremien der nächste und auch letzte Schritt in der Genehmigungskette. Der Ablehnungsvorgang erbringt das gleiche Resultat wie bei der Ablehnung durch den Studiengangsverantwortlichen.
Sobald das Modul durch die GREMIEN abgesegnet wurde, wird das Modul in den öffentlichen Modulordner zurückgeschoben und erhält den Zustand "veröffentlicht". Nun ist es von außen einsehbar und es kann auf das Modul im Einzelnen verlinkt werden. Von diesem Zustand aus kann es nie wieder in den Zustand "privat" zurück gelangen. Auf diese Weise wird gewährleistet, dass Module nicht aus dem Handbuch verschwinden, es sei denn, sie werden vom Administrator über das ZMI gelöscht. Module im Zustand "veröffentlicht" können von Modulverantwortlichen zur Bearbeitung gesperrt werden. Das öffentliche Modul ist weiterhin sichtbar und erhält den Status "gesperrt", während der Modulverantwortliche eine Kopie des gesamten Moduls und jeder der Dozenten eine Kopie seiner Lehrveranstaltung erhält. Hier schließt sich der Kreis und der weitere Arbeitsablauf erfolgt wie vorher beschrieben.
Wenn ein Dozent seine Lehrveranstaltung bearbeiten möchte, reicht er einen Bearbeitungsantrag ein, das heißt, er verändert den Zustand seiner Lehrveranstaltung über "Inhalt bearbeiten". Der Modulverantwortliche erhält eine Nachricht und kann nun das Modul zur Bearbeitung sperren. Die Benachrichtigungen erfolgen über die bei der Registrierung angegebenen E-Mail-Adresse, zusätzlich stehen Worklists mit den Links zu den wartenden Dokumenten bereit.

Der MANAGER kann den Genehmigungs-Vorgang abkürzen, indem er das Modul von "privat" auf "veröffentlicht" setzt. In diesem Schritt werden ebenfalls alle benötigten Rollen gesetzt.

Das Kopieren und Verschieben der Dokumente ist einerseits sinnvoll, um ohne häufige Rollenänderungen Bearbeitungsrechte vergeben zu können, und andererseits notwendig, um im Modulordner immer die aktuell rechtsgültige Version der Module behalten zu können, während sich diese Module in Bearbeitung befinden.

4. Umsetzung und Ergebnisse

Nachdem in den vorangegangen Kapiteln sowohl die hinter Plone stehende Technologie als auch die Konzeption dieses Projekts besprochen wurde, widmet sich das aktuelle Kapitel der Implementierung.
Zunächst wird Anpassung der CTModule-Klassen beschrieben, um danach auf das implementierte Rechtesystem einzugehen. Den Hauptteil dieses Kapitels sowie des Projekts bildet die Workflow-Implementierung und im Zuge dessen das Erarbeiten einer Reihe von Skripten, deren Funktionsweise detailliert beschrieben wird.

Die vollständigen Skripte und Workflow-Dateien befinden sich auf der beiliegenden CD.

4.1. Anpassung der Klassen

Das Anpassen der im Kapitel 3.3 *Klassenstruktur und Benutzer* besprochenen Klassen erfolgte durch Editieren des vorhandenen Klassen-Schemas. Dabei wurde in den Klassen CTStudiengang, CTModul und CTLehrveranstaltung jeweils ein neues Referenzfeld hinzugefügt, welches den Verweis auf ein Inhaltelement vom Typ CTDozent als Inhalt zulässt:

Code: #1

```
ReferenceField(
    name='dozent',
    schemata="default",
    multiValued=0,
    required=1,
    relationship="CTLehrveranstaltung->CTDozent",
    allowed_types=('CTDozent'),
    widget=ReferenceBrowserWidget(
        label="Dozent",
        label_msgid='CTModule_label_dozent',
        i18n_domain='CTModule',
        startup_directory="/Plone/Dozenten/",
        allow_search=1,
        allow_browse=1,
        description="Geben Sie hier den Dozenten der Lehrveranstaltung ein.",
        description_msgid='CTModule_help_lvdozent',
    )
),
```

Die Eigenschaft *multivalued=0* legt fest, dass keine Mehrfachauswahl möglich sein soll, wie in der Konzeption begründet wurde. Durch *required=1* wird das Feld zum Pflichtfeld.

Mit *relationship* wird festgelegt, zwischen welchen Inhaltstypen eine Beziehung aufgebaut wird und allowed_types gibt alle Inhaltstypen an, die in dieses Feld gelinkt werden können.
Das ReferenceBrowserWidget erhält Angaben zum Auswahlfenster, mit dessen Hilfe die Dozenten verlinkt werden. Dieser Browser besitzt zwei Grundfunktionalitäten zur Auswahl von Inhaltselementen: Der Browser zur hierarchischen Auflistung aller Plone-Verzeichnisse und Inhaltselemente; dieser wird mit *allow_browse=1* eingeschaltet. Die andere ist eine Suchfunktion; das Suchfeld wird mit *allow_search=1* aktiviert. Die Eigenschaft *startup_directory* kennzeichnet Startpunkt im Verzeichnisbaum bei der Browsing-Funktion. Es muss darauf geachtet werden, dass dieses Verzeichnis existiert. Zusätzlich kann ein Beschreibungstext und Beschriftungen angegeben werden.

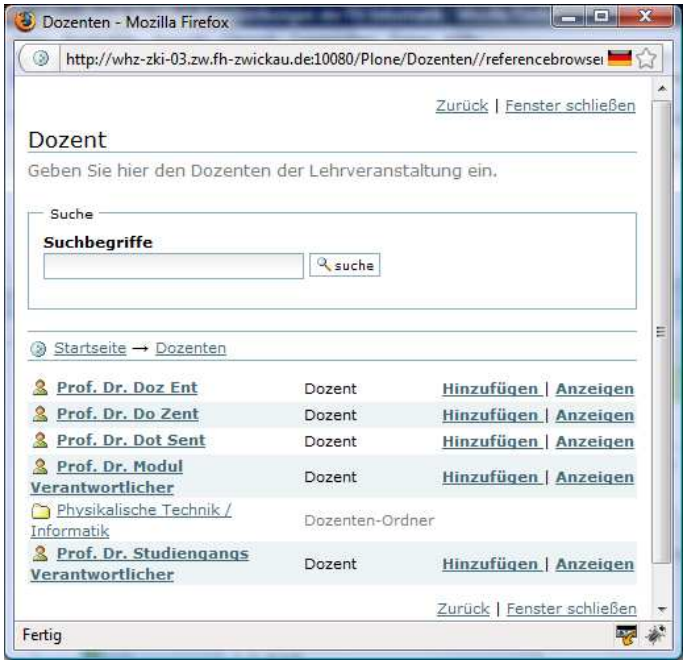

Abbildung 6: Der ReferenzBrowser aus CTLehrveranstaltung

Wie bereits in der Konzeption erwähnt wurde, ist es nötig, versteckte Zusatzinformationen in den Inhaltstypen zu speichern, zum Beispiel die ID des Moduls innerhalb der Lehrveranstaltung, da in einem Teil des Workflows die Lehrveranstaltung zur Bearbeitung vom Modul abgetrennt wird. Die versteckten Felder werden auf folgende Weise definiert:

Code: #2

```
StringField(
   name='modul',
   widget=StringWidget(visible={'view' : 'invisible', 'edit' : 'invisible'})
),
```

Auf sämtliche in Feldern gespeicherte Werte kann mit Get- und Set-Methoden zugegriffen werden, ohne dass diese Methoden expliziert definiert werden müssen.

In CTDozent waren die wesentlichen Felder schon vorhanden, es wurde jedoch das ID-Feld überschrieben um ein selbst definiertes Label anbringen zu können, da das Standard-Label "Titel" leicht mit dem Feld "Akademischer Titel" verwechselt werden konnte.
Das ID-Feld ist ein automatisch vorhandenes Feld und ist im Code von CTDozent nicht zu sehen. Indem ein neues Feld mit dem Namen "title" angelegt wird, beziehen sich die darin definierten Eigenschaften auf das ID-Feld des Inhaltstypen:

Code: #3

```
StringField(
    name='title',
    required=1,
    searchable=1,
    widget=StringWidget(
        label='Anmeldename',
        label_msgid='CTModule_label_id',
        i18n_domain='CTModule',
    )
),
```

Das ID-Feld sollte möglichst dem ZKI-Anmeldekürzel des Benutzers an der Hochschule entsprechen, um erstens eindeutige Benutzernamen zu haben, und zweitens die Zuordnung zu erleichtern.
Die Eigenschaft *required=1* kennzeichnet das Feld als Pflichtfeld, denn das Feld *ID* ist als Pflichtfeld definiert.
Durch *searchable=1* indiziert man Felder für die Suche in der Datenbank. Dies ist vor allem in Referenzfeldern von Bedeutung, da im ReferenceBrowserWidget nach allen Feldern gesucht werden kann, die so indiziert wurden. Im Fall von CTDozent sind dies Anmeldename, Vorname und Nachname.

Die eigentliche Funktionalität, wie das Anlegen des Benutzers, wurde nicht in der Klasse implementiert, sondern im dazugehörigen Workflow, der im Kapitel 4.3.2 *Der Dozenten-Workflow* besprochen wird. Auf diese Weise wird dem Anwender die Freiheit gegeben, selbst zu entscheiden, wann er eine gespeicherte Änderung ins Live-System überträgt. Zum Beispiel kann beim Anlegen des Benutzers die E-Mail-Adresse nicht bekannt sein; in diesem Fall muss das Erstellen des Inhaltselements nicht abgebrochen werden - die Adresse kann nachgetragen werden, sobald sie bekannt ist, um erst danach den Anmeldeprozess auszuführen.
Und nicht zuletzt bietet diese Variante größeren Komfort für den Programmierer, da bei jeder Änderung im Code der Server neu gestartet werden muss um das Ergebnis zu sehen, weil Plone auf kompilierten Python-Skripten aufbaut, die nur beim Neustart des Servers neu kompiliert werden. Workflow-Skripte hingegen werden direkt interpretiert.

4.2. Berechtigungen und Rollen

Im Berechtigungssystem wird zwischen öffentlichen Ordnern und Benutzer-Ordnern unterschieden. Im öffentlichen Modul-Ordner befinden sich sowohl ungenehmigte, unsichtbare Module als auch genehmigte, veröffentlichte Module. Auf diese Module besitzt nur der MANAGER Vollzugriff, das heißt alle anderen Benutzer können nur lesend darauf zugreifen.
Die Benutzer-Ordner sind von Plone angelegte Bereiche innerhalb des "Members"-Ordners. In diesem eigenen Ordner hat der jeweilige Benutzer Vollzugriff auf seinen Ordner und alle darin befindlichen Objekte, denn er besitzt in diesem Bereich die OWNER-Rolle.
Dieses Prinzip wurde sich hier zu Nutze gemacht: Ein Dozent erhält nicht die Berechtigung, im öffentlichen Ordner seine Lehrveranstaltung zu verändern, aber er kann beantragen, sie bearbeiten zu dürfen. Wird dem Antrag stattgegeben, erhält er eine Kopie seiner Lehrveranstaltung in seinen privaten Ordner, in welchem er automatisch durch Akquisition den Vollzugriff auf die Lehrveranstaltungskopie erhält und sie somit editieren kann. Beim Einreichen wird die Lehrveranstaltung in den Benutzer-Ordner des Modulverantwortlichen verschoben, in welchem der Dozent nur Leserechte besitzt. Somit kann er im Nachhinein keine Änderungen mehr an seiner Lehrveranstaltung durchführen.

Sämtliche Skripte in Workflowschritten, die von anderen Benutzern als dem Administrator ausgeführt werden können, wurden mit der Proxy-Rolle MANAGER versehen, da in den Skripten Funktionen verwendet wurden, die den Benutzern außerhalb des Workflows nicht zustehen sollen, dazu zählt etwa das Verschieben von Objekten in den öffentlichen Bereich.

Es wurden die Rollen DOZENT, MODULVERANTWORTLICHER und STUDIENGANGSVERANTWORTLICHER global definiert um dann durch Zuweisung in Workflowskripten lokal verwendet werden zu können. GREMIEN wird als globale Rolle im System verwendet und kann einem oder mehreren Benutzern zugeordnet werden; dies geschieht in der Benutzer- und Gruppenverwaltung, wie in Kapitel 2.3.2 *Rollen im Detail* nachzulesen ist. Zur Demonstration wurde der Benutzer "gremien" mit der Gremien-Rolle angelegt.[3]

Nun können nach Ermessen des Administrators weitere feinschichtige Berechtigungs-Einstellungen am live-System vorgenommen werden. Hierbei zu beachten ist, dass die Einstellungen für DOZENT, MODULVERANTWORTLICHER und STUDIENGANGSVERANTWORTLICHER nur lokal gelten. Das bedeutet zum Beispiel, dass dem MODULVERANTWORTLICHEN im Wurzel-Verzeichnis die Berechtigung "Modify portal content" gegeben werden könnte. Damit hätte der aktuelle Benutzer mit der Rolle des MODULVERANTWORTLICHEN das Recht, im öffentlichen Ordner seine eigenen Module direkt zu bearbeiten ohne den Workflow anstoßen zu müssen. Andere Module blieben für den Benutzer nicht bearbeitbar. Würde hingegen die GREMIEN-Rolle das Recht "Modify portal content" erhalten, könnte der Benutzer dieser Rolle systemweit alle Dokumente direkt bearbeiten. Diese Einstellungen wären jedoch unverantwortlich und sollten nur zur Verdeutlichung des Prinzips dienen.

[3] Standardpasswort dieses Benutzers: 00000

Andere Funktionen, wie das Nutzen des Mail-Servers, wurden sämtlichen Benutzern erlaubt, indem der Rolle MEMBER die Berechtigung "Use mailhost services" zugestanden wurde.
In ihrem eigenen Ordner besitzen sie als OWNER weitergehende Berechtigungen, die somit die Einstellungen aus den lokalen Rollen überschreiben. Der OWNER Rolle wurde zudem die Berechtigung "Delete objects" entzogen um zu vermeiden, dass Objekte von anderen Benutzern als dem Administrator gelöscht werden, um den Workflow nicht in unbeabsichtigter Weise zu unterbrechen.

Nun musste noch dafür gesorgt werden, dass unangemeldete Nutzer, d.h. Studenten, private Module nicht einsehen können, da diese noch nicht offiziell genehmigt sind. Dafür wird der Rolle ANONYMOUS im Workflow-Status *private* die *View*-Berechtigung genommen[4]. Die Akquisition muss dabei ausgeschaltet sein. Danach muss die *Security Update*-Funktion des Workflow-Werkzeugs verwendet werden um die bisher erstellten Dokumente an die neuen Einstellungen anzupassen.

Die vordefinierten Rollen wurden im System belassen, denn in [WaP07] wurde informiert: "Es ist nicht ratsam die Rollen Member und Reviewer zu löschen, da darauf die Plone-Workflows [...] aufbauen."

4.3. Workflowgestaltung

Die Workflows in diesem Projekt wurden auf dem vorhandenen Standard-Workflow von Plone aufgebaut. Diese Option bietet eine große Zeitersparnis, da Grundzustände, Variablen und Einstellungen schon vordefiniert sind und der Workflow somit ohne weitere Konfiguration verwendet werden kann, sobald er einem Inhaltstypen zugeordnet wird. Ein neuer Standardworkflow wurde nicht definiert, da die vorhandenen Inhaltstypen alle sehr unterschiedlich gesteuert werden müssen.

Nachdem im Kapitel 3.4 *Workflowgestaltung* eine verbale Zusammenfassung der Workflows und ihrer Verzahnung ineinander geliefert wurde, werden im Folgenden die Workflows einzeln beschrieben und mit einer graphischen Darstellung verdeutlicht. Die Darstellungsart wurde an das Zustandsdiagramm der UML angelehnt. In abgerundeten Rechtecken werden Zustände abgebildet, die dünnen Pfeile entsprechen Übergängen. Die Pfeilrichtung gibt an, von welchem Zustand der Übergang ausgeht. Wenn Pfeile parallel verlaufen, bezeichnet das direkt daneben stehende Feld den jeweiligen Übergang: Rechts in den unteren Zustand, links in den oberen Zustand. Die Übergangsbezeichnung setzt sich aus einer in Kapitälchen geschrieben Rolle oder einer Fremdworkflow-Bezeichnung, und dem Namen des Übergangs zusammen. Gestrichelte Pfeile bedeuten, dass ein Übergang von einem Fremdworkflow ausgeführt wird, und ausgegraute Pfeile deuten einen Kopiervorgang an. Weiterhin deuten breite Pfeile an, welche Workflow-Skripte in welchen Übergängen aufgerufen werden.

[4] /portal_workflow/WF_CTModul/states/private/manage_permissions

Jedem Workflowschritt wurden eine oder mehrere Rollen zugeordnet um ihn vor unberechtigter Ausführung zu schützen, dabei wurde konzipiert, dass alle Schritte ersatzweise von einem MANAGER vorgenommen werden können. Dies wird in den Grafiken der Übersichtlichkeit willens nicht dargestellt.
Als Werkzeug zur Benachrichtigung wurden sowohl Worklists als auch E-Mails verwendet. In welchem Übergang eine Benachrichtigungsmail verwendet wird, lässt sich am Brief-Symbol neben der Bezeichnung des Übergangs ablesen.

4.3.1. Unsichtbare Übergänge

Übergänge können sowohl vom Benutzer als auch per Skript aus dem aktuellen oder einem anderen Workflow heraus durchgeführt werden. Wenn ein Übergang nur für die Ausführung von einem Workflow-Skript vorgesehen ist, kann er vor dem Benutzer verborgen werden, indem bei seiner Definition die Felder im Abschnitt "Display in actions box" leer belassen werden. Somit wird der Übergang nicht in der Auswahlbox angezeigt und kann nicht vom Benutzer ausgelöst werden.
Dies wird zum Beispiel verwendet, um einem kopierten Objekt einen anderen Anfangszustand als dem originalen Objekt zu geben. Der Übersichtlichkeit halber werden unsichtbare Übergänge in den Skizzen dieser Arbeit nur verzeichnet, wenn sie für das Verständnis notwendig sind. Sie sind an einem gestrichelten Pfeil zu erkennen.

4.3.2. Der Dozenten-Workflow

Der Workflow WF_CTDozent kennt drei Zustände; angelegt wird jedes Element von CTDozent im Zustand "Entwurf". Um aus dem Inhaltselement einen Benutzer zu erzeugen, muss der Übergang "freigeben" durch den MANAGER ausgelöst werden. In diesem Schritt wird das Workflowskript "set_dozent_free" aufgerufen. Dieses Skript greift auf das portal_registration-Werkzeug zu, um den neuen Benutzer mit den Daten aus dem angelegten Dozenten-Dokument zu registrieren:

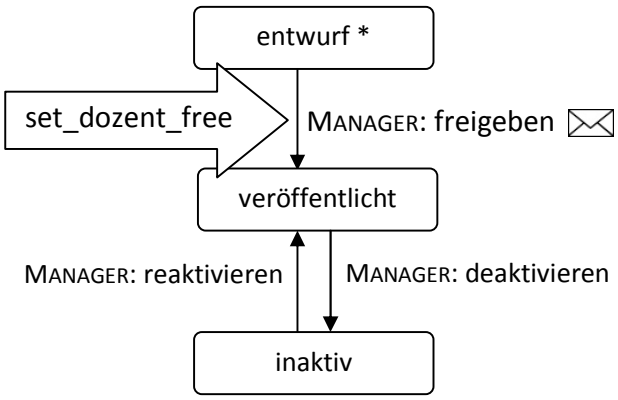

Abbildung 7: Der Dozenten-Workflow

Code: #4

```
context.portal_registration.addMember(
nickname, password, roles=(role,), domains = domainsj ,properties=None)
```

Die Variable *nickname* entspricht der ID des Dozenten, unter diesem Namen kann sich der Benutzer nach der Registrierung anmelden. Als Passwort wird eine fünfstellige Zahl generiert, im Fall dass eine E-Mail-Adresse angegeben wurde, an die das Passwort verschickt wird. Ist das Feld "email" leer, erhält der Benutzer das Standardpasswort 00000. Es wird jedoch empfohlen, die E-Mail-Variante vorzuziehen. Die im Skript vordefinierte Rolle ist MEMBER. Dadurch wird ein Grundsatz an Berechtigungen vergeben und der Benutzer erhält einen eigenen Bereich in Plone, seinen Nutzerordner.
Um die Daten wie die E-Mail-Adresse auch im Benutzerprofil sichtbar zu machen, werden sie noch über die Funktion *setMemberProperties()* als Nutzerattribute eingetragen:

Code: #5
```
context.plone_utils.setMemberProperties(member, email=email_obj)
```

Das Senden der E-Mail erfolgt über:

Code: #6
```
context.MailHost.send(messagetext , str(email), 'noreply@fh-zwickau.de', betreff)
```

Bei Tests mit dem Mailserver der Westsächsischen Hochschule wurde diese E-Mail nicht als Spam eingestuft, jedoch sollte im laufenden Betrieb sicher gestellt werden, dass es so bleibt. Falls überhaupt keine E-Mails versendet werden, sollte überprüft werden, ob der Workflow-Akteur die Berechtigung *Use mailhost services* innehält.

Die Sicherheit betreffend sei anzumerken, dass portal_registration von Haus aus mit Berechtigung *Add portal member* geschützt ist und von allen vorhandenen Rollen nur der MANAGER diese Berechtigung besitzt. Zusätzlich sind sämtliche Workflow-Schritte mit der MANAGER-Rolle abgesichert, somit können in der derzeitigen Konfiguration Inhaltselemente vom Typ CTDozent und dazugehörige Benutzer nur durch den Administrator angelegt werden.

Um den gesamten Vorgang der Nutzererstellung und E-Mail-Benachrichtigung wurde eine *try-catch*-Anweisung - eine Ausnahmebehandlung - eingefügt um zu verhindern, dass es zu einem Fehler kommt, wenn ein Nutzer das zweite Mal angelegt wird. Durch das Werfen der Exception verbleibt das Dozenten-Inhaltselement im Zustand "entwurf".
Um nur Dozenten im Zustand "veröffentlicht" in andere Inhaltselemente verlinkbar zu machen, musste ein Teil des Plone-Codes angepasst werden, genauer gesagt die View des *ATReferenceBrowserWidget*. In portal_skins/custom/referencebrowser_popup/manage_main wurde folgender TAL-Ausdruck für jeden Zustand eingefügt, der vom *ReferenceBrowser* ignoriert werden soll, wie hier *inactive*:

Code: #7
```
<tal:if condition="python: getInfoFor(item, 'review_state', None) != 'inactive'"> ... </tal:if>
```

Das TAL-Tag umschließt hierbei jene Codezeilen, die für das Anzeigen der Browse- beziehungsweise Suchergebnisse des *ReferenceBrowsers* verantwortlich sind.

4.3.3. Der Studiengangs-Workflow

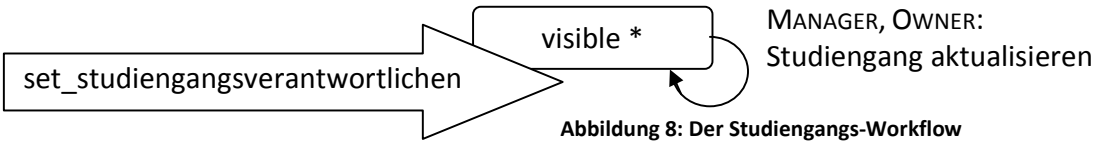

Abbildung 8: Der Studiengangs-Workflow

Der Workflow WF_CTStudiengang dient zum Neusetzen des STUDIENGANGSVERANTWORTLICHEN in den vom Studiengang gelinkten Modulen. Immer wenn neue Module in den Studiengang eingefügt werden, muss der Studiengang über den Workflow aktualisiert werden.

Der Studiengang wird vom Administrator angelegt und der Studiengangsverantwortliche eingetragen. Mit dem Aktualisieren des Studiengangs wird dem Studiengangsverantwortlichen die Rolle OWNER für dieses Inhaltselement zugesprochen. Damit erhält der Studiengangsverantwortliche ebenfalls direkte Bearbeitungsrechte für den Studiengang:

Code: #8

```
obj = state_change.object
sgv = obj.getStudiengangsverantwortlich()
obj.manage_setLocalRoles(sgv, ['Owner'])
```

Im Skript wird aus dem Übergabeparameter state_change das Studiengangsobjekt selbst gewonnen; dieses Objekt enthält wiederum den Inhalt des Felds "studiengangsverantwortlich", welches mit der automatisch vorhandenen Get-Methode des Felds ausgelesen wird. Es bleibt die ID des Inhaltselements von CTDozent, welche dem Namen des Benutzers entspricht, der als Studiengangsverantwortlicher ausgewählt wurde. In der dritten Zeile wird schließlich dem Studiengang die lokale Rolle hinzugefügt.

Der eigentliche Zweck des Skripts - das Setzen des STUDIENGANGSVERANTWORTLICHEN in allen Modulen - wird folgendermaßen realisiert:
Zuerst wird mit der Methode *getAlleModule()* aus dem Studiengang eine Liste mit allen verlinkten Modulen geholt, um dann in jedem einzelnen dieser Objekte die Rolle des STUDIENGANGSVERANTWORTLICHEN zu setzen. Diese Rolle wird im Modulworkflow als Teil der Genehmigungskette benötigt.

Das Setzen der Rollen ist jedoch nicht unproblematisch. Wenn zum Beispiel ein Benutzer zwei verschiedene Rollen erhalten soll, wird die vorher definierte Rolle von Plone überschrieben. Deshalb muss vorher abgeprüft werden, ob der Studiengangsverantwortliche gleich dem

Modulverantwortlichen ist, und im Falle der Gleichheit müssen zwei Rollen an diesen Benutzer übergeben werden:

Code: #9

```
modul.manage_setLocalRoles(mv, ['Modulverantwortlicher','Studiengangsverantwortlicher'])
```

Soll hingegen ein anderer Benutzer die lokale Rolle erhalten, müssen erst alle vorhandenen Rollen gelöscht werden, da es in dieser Plone-Version nicht möglich ist, sich gezielt eine Liste der gesetzten Rollen zurückgeben zu lassen. Die Funktion zum Löschen einer Rolle erfordert eine Benutzer-ID, aber zum Zeitpunkt der Neusetzung der Rolle existieren keine Informationen mehr über den vorherigen Benutzer, deshalb müssen für sämtliche Benutzer die Rollen-Informationen im Objekt gelöscht werden:

Code: #10

```
for user in mship.listMembers():
  memberID = mship.getMemberById(user.id)
  modul.manage_delLocalRoles([str(memberID)])
```

Bei einer hohen Anzahl von Benutzern geht dies zu Lasten der Performance. Dennoch ist dieses Vorgehen vertretbar, da der Prozess des Rollen-Setzens nur während des Erstellens und Überarbeitens von Inhaltselementen zur Anwendung kommt, und somit niemals von einer hohen Anzahl von Benutzern gleichzeitig ausgeführt wird.

Das Verlinken der Module im Studiengang und das Setzen des STUDIENGANGSVERANTWORTLICHEN muss vor der eigentlichen Initialisierung der Module geschehen, weil in der Überarbeitungsphase nur mit Kopien gearbeitet wird, die nicht im Studiengang verlinkt sein können. Deshalb werden im Studiengang unveröffentlichte Module angezeigt. Durch Anpassen des View-Templates für den Studiengang können jedoch in der Anzeigen-Ansicht die privaten Module ausgeblendet werden, während sie in der Bearbeiten-Ansicht schon gelinkt und sichtbar sind. Sobald der Modul-Workflow erfolgreich durchgelaufen ist, erscheinen die genehmigten Module im Studiengang automatisch als sichtbar.
Das Anpassen der View kann ähnlich geschehen wie das Ausblenden der deaktivierten Dozenten, wurde aber in der vorliegenden Arbeit aus Zeitgründen noch nicht berücksichtigt.

Im Moment gibt es keine Möglichkeit festzustellen, aus welchem Studiengang das Modul stammt, da die Module nicht kategorisiert sind. Somit erhält der Studiengangsverantwortliche auch für Fremd-Module aus einem anderen Studiengang das Recht, sie zu genehmigen.

Für die Inhaltstypen Wahlpflichtkatalog und Schwerpunkt war es nicht nötig, einen eigenen Workflow zu entwerfen, da sie nur als Unterobjekte des Studiengangs vorkommen und ebenso wie der Studiengang keine Genehmigungskette durchlaufen müssen. Sie können vom Studiengangsverantwortlichen frei definiert und umbenannt werden.

4.3.4. Der Modul-Workflow

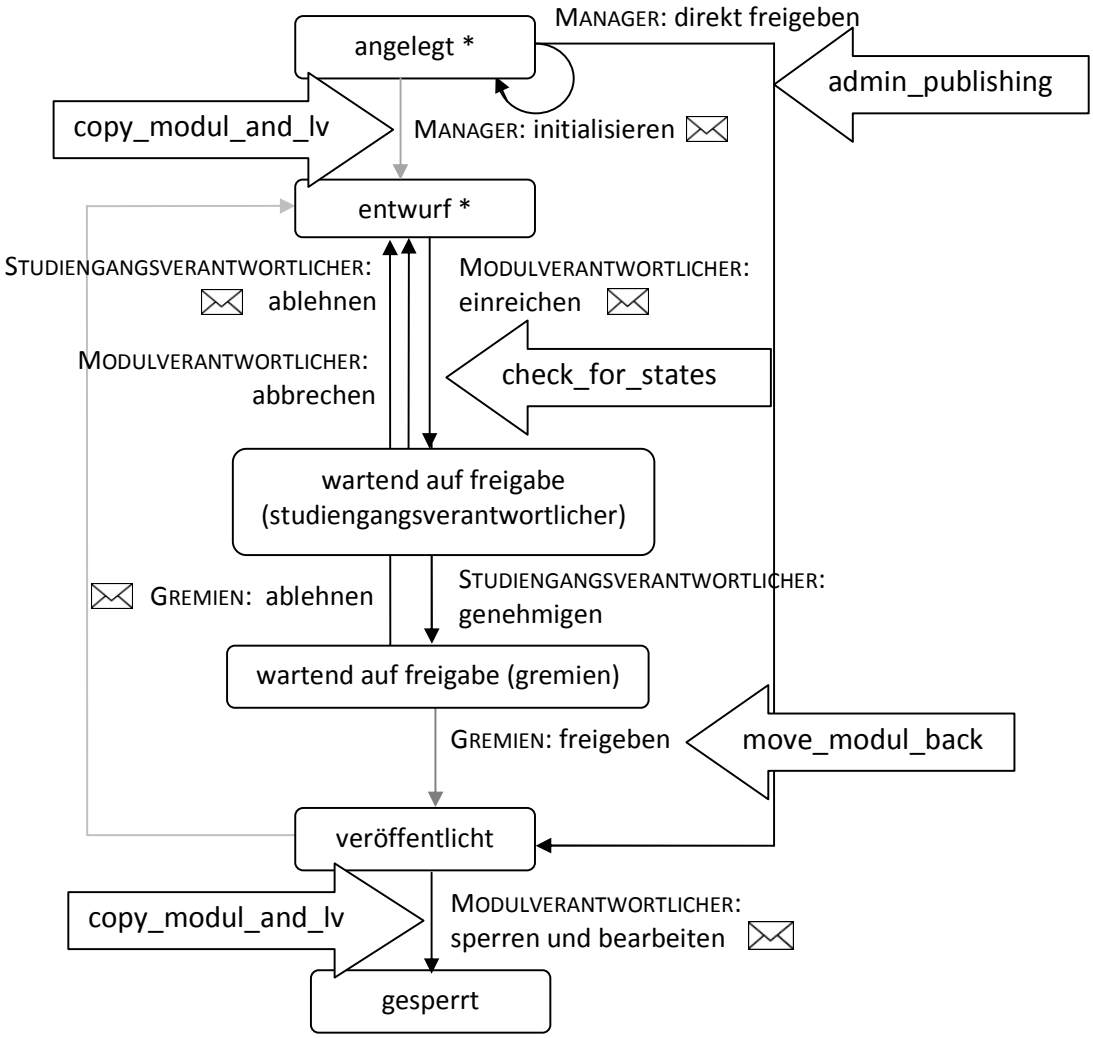

Abbildung 9: Der Modulworkflow

Der Workflow WF_CTModul steuert Inhaltselemente vom Typ CTModul und teilweise die untergeordneten Inhaltselemente vom Typ CTLehreranstaltung, CTPruefung und CTVorleistung über Workflow-Skripte. In Abbildung 9 wird nur der reine Modulworkflow abgebildet.

Der Workflow implementiert zwei verschiedene Arten der Initialisierung: Zu einen kann das Modul durch den Administrator selbst mit Daten gefüllt und freigegeben werden, dabei wird keine Kopie des Moduls angelegt, stattdessen ändert sich die Sichtbarkeit von *privat* auf *veröffentlicht*. Im aufgerufenen Skript admin_publishing werden auch alle Unterobjekte des Moduls auf *veröffentlicht* gesetzt. Zu diesem Zweck wird der unsichtbare Übergang "publish" aufgerufen:

Code: #11

```
alleItems = obj.contentItems()
for jedesItem in alleItems:
  context.portal_workflow.doActionFor(jedesItem[1], 'publish', comment='')
```

Im gleichen Skript werden die lokalen Rollen MODULVERANTWORTLICHER im Modul und DOZENT in den untergeordneten Lehrveranstaltungen zugewiesen. Da die Unterobjekte Pruefung und Vorleistung keine eigenen Rollen benötigen, werden die Lehrveranstaltungen aus der Menge der Unterobjekte herausgefiltert:

Code: #12

```
items = obj.contentItems(filter={'portal_type':['CTLehrveranstaltung',]})
```

Doch bevor ein Modul freigegeben werden kann, muss sicher gestellt werden, dass es im Studiengang verlinkt ist und die Rolle des STUDIENGANGSVERANTWORTLICHEN gesetzt ist. Die folgende Abfrage prüft, ob durch den Studiengangs-Workflow jenes unsichtbare Feld schon beschrieben wurde, das den Namen des Studiengangsverantwortlichen speichert, und die *raise*-Anweisung sorgt dafür, dass eine Exception geworfen wird, wenn das Feld leer ist. Der Übergang zu *veröffentlicht* wird somit nicht ausgelöst:

Code: #13

```
if sgv == "":
  raise """Das Modul muss erst im Studiengang verlinkt werden. Danach muss der
  Studiengang aktualisiert werden."""
```

Die andere Initialisierungsmöglichkeit besteht darin, dass durch den Administrator ein Gerüst angelegt wird, nur bestehend aus dem Modul mit ausgefüllten Pflichtfeldern wie der ID und dem Modulverantwortlichen, sowie das Gerüst der Lehrveranstaltungen mit ihren Verantwortlichen und wahlweise der Pruefung und Vorleistungen. Danach verlinkt er das Modul im Studiengang, aktualisiert den Studiengang und löst dann den *init*-Übergang aus.

In diesem Übergang wird das Skript copy_modul_and_lv aufgerufen, um Modul und Lehrveranstaltungen zum Bearbeiten in die Benutzer-Ordner zu kopieren. Im Skript wird zunächst geprüft, ob das Modul schon einmal initialisiert wurde, denn das Originalmodul bleibt so lange im öffentlichen Modulordner im Zustand *angelegt* bis es freigegeben wurde. In der Zwischenzeit könnte der *init*-Übergang ein weiteres Mal ausgelöst werden. Dies hätte eine zweite Kopie des Moduls zur Folge, deswegen wird es mit einer weiteren Exception abgefangen.
Nun wird noch geprüft, ob die Benutzer-Ordner aller Dozenten vorhanden sind, die im Modul oder den Lehrveranstaltungen verlinkt sind, denn Benutzer-Ordner werden nur angelegt, wenn sich der Benutzer im System zum ersten Mal einloggt. Es war nicht möglich, die Benutzer-Ordner mit Hilfe des Skripts selbst zu erstellen.

Um das Anlegen eines Moduls in Benutzer-Ordnern zu ermöglichen, musste im ZMI dem Inhaltstyp *Folder* die Inhaltstyp CTModul als erlaubten Typ zugewiesen werden. Analog wurde mit der Lehrveranstaltung verfahren.

Nachdem alle möglichen Fehler abgefangen wurden, wird der eigentliche Kopiervorgang ausgeführt. Dazu wird der übergeordnete Ordner des Modul-Objekts ausgewählt, dieser bringt die Methode zum Kopieren von enthaltenen Objekten mit. Mit Hilfe der ID des Modulverantwortlichen wird nun sein Benutzerordner als Zielordner bestimmt. Das Einfügen des kopierten Objekts geschieht über die Einfüge-Methode ("paste") des Zielordners:

Code: #14

```
id = state_change.object .getId()
quellOrdner = obj.aq_parent
nutzerID = obj.getModulverantwortlichen().getId()
nutzerOrdner = context.portal_membership.getHomeFolder(nutzerID)
nutzerOrdner.manage_pasteObjects(quellOrdner.manage_copyObjects([id,]))
```

Referenzfelder werden nicht automatisch mit kopiert, deshalb werden diese Felder aus dem alten Objekt ausgelesen und dem neuen explizit zugewiesen:

Code: #15

```
new_obj.setModulverantwortlich(obj.getModulverantwortlichen())
```

Nach dem erfolgreichen Kopiervorgang wird der Zustand des Moduls auf *sichtbar* gesetzt um eine Statusänderung durch andere Benutzer als den Administrator zuzulassen, genauer gesagt durch die entsprechend festgelegten Benutzer. Wenn nun einer dieser Benutzer in seinem privaten Bereich ein Modul anlegen sollte, gerät es durch diesen Mechanismus nie in Umlauf, denn niemand anderer als ein MANAGER kann den Anfangszustand eines Moduls verändern, somit wird es den Benutzer-Ordner nie verlassen.

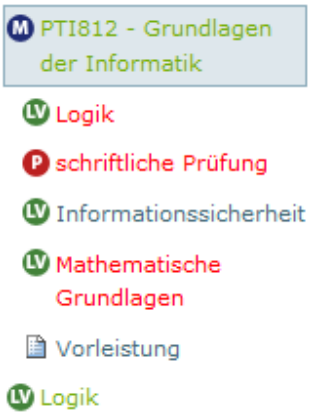

Abbildung 10: Beispielmodul

Das Skript kopiert nicht nur das Modul samt Unterobjekten zum Modulverantwortlichen, zusätzlich spaltet es das Modul auf. Dies funktioniert nach dem gleichen Prinzip wie das Kopieren des Moduls. Somit erhält ein Dozent nur seine Lehrveranstaltung ohne Modul, und der Modulverantwortliche eine komplette Modulkopie und für jede seiner Lehrveranstaltungen eine eigenständige Kopie, die außerhalb des Moduls in seinem Benutzer-Ordner liegt - wie im Bild links zu sehen ist. Der Sinn dieses Vorgehens liegt in der Rechtevergabe innerhalb der Benutzer-Bereiche begründet, wie in Kapitel 4.2 *Berechtigungen und Rollen* nachzulesen ist. Zusätzlich bringt es dem Modulverantwortlichen eine Übersicht, welche Objekte im Modul vorkommen, welche davon noch unbearbeitet sind, und welche davon noch genehmigt werden müssen.

Details dazu sind im in Kapitel 4.3.6 *Der Lehrveranstaltungs-Workflow* zu finden.

Im Modulworkflow geht es mit dem nächsten Schritt in der Genehmigungskette weiter: Das komplette Modul wird dem Verantwortlichen des Studiengangs zur Genehmigung vorgelegt. Um zu garantieren, dass das Modul komplett bearbeitet und alle Unterobjekte durch den Modulverantwortlichen genehmigt wurden, wird in diesem Übergang das Skript check_for_states aufgerufen. Dort wird geprüft, ob sich alle Unterobjekte des Moduls im Zustand "genehmigt" befinden:

Code: #16

```
objektStatus = context.portal_workflow.getInfoFor(modulObjekt, 'review_state')
if objektStatus != "genehmigt":
  alleBearbeitet = 0
```

Erst wenn diese Bedingung erfüllt ist, wird der Übergang "einreichen" ausgeführt.
Der STUDIENGANGSVERANTWORTLICHE und die GREMIEN haben zu keinem Zeitpunkt Schreibrechte auf das Modul, da es nicht ihre Aufgabe ist, darin Korrekturen anzubringen. Wenn Mängel festgestellt wurden, kann der Genehmigungsantrag mit einem Kommentar abgelehnt werden. Dies geschieht über die von Plone zur Verfügung gestellten erweiterten Workfloweinstellungen. Indem der Benutzer im Statusfeld nicht den Zustand ändert, sondern auf das unterste Feld "erweitert" klickt, erscheint ein Formular, in welchem unter Anderem ein Feld für Kommentare und ein Auswahlfeld für mögliche Übergänge vorhanden ist. Wenn der Benutzer den Zustand des Moduls über dieses Formular ändert, wird in der zugehörigen Mailbenachrichtigung der eingetragene Kommentar an den Modulverantwortlichen mit gesandt. Details zu Mailbenachrichtigungen folgen in Kapitel 4.3.8 *Benachrichtigungen*.
Durch die Ablehnung geht der bearbeitete Inhalt nicht verloren, und durch den Kommentar erhält der Modulverantwortliche alle Informationen, die er braucht, um das weitere Verfahren mit dem Modul entscheiden zu können, beispielsweise das Editieren bei einem Schreibfehler oder das Ablehnen einer Lehrveranstaltung zur Überarbeitung durch den Dozenten.

Angenommen, das Modul wurde sowohl durch den Studiengangsverantwortlichen genehmigt, als auch durch die Gremien für in Ordnung befunden - nun muss es wieder in den öffentlichen Modulordner zurück geschoben werden. Dies geschieht im Übergang "freigeben" durch den GREMIEN-Nutzer mit Hilfe des Skripts move_modul_back.
Zu Beginn muss das Modul im öffentlichen Ordner gesucht werden, das die gleiche ID wie das kopierte Modul besitzt. Die ID besteht aus Modulnummer und -name; somit entsprechen sich die IDs.

Code: #17

```
for modul in dstFldr.objectValues():
  if modul.getId() == aktuellesModulID:
    altesObjekt = modul
```

In der Variablen "altesObjekt" wird der Verweis auf das Modul im öffentlichen Ordner gespeichert; mit dieser Referenz kann genau wie mit dem Modul selbst gearbeitet werden. Zunächst wurde versucht, das gesamte Modul zu ersetzen. Dazu wurde auf die Ausschneid-Methode des Quellordners zugegriffen:

Code: #18

```
obj = srcFldr.manage_cutObjects([id,])
```

Es stellte sich jedoch heraus, dass beim Ersetzen des Moduls zwar die ID des Moduls stimmte, aber eine weitere, interne Identifikationsnummer (UID) existierte, die nicht mit dem neuen Modul überein stimmte. Aus diesem Grund entfernte Plone beim Ersetzen des öffentlichen Moduls selbstständig den Link aus dem Studiengang, in dem es verlinkt war, anstatt die Referenz zu ändern.
Dies machte es notwendig, das ursprüngliche Modul beizubehalten und sukzessive die Feldinhalte der genehmigten Modul-Kopie in das alte Modul zu übertragen:

Code: #19

```
...
altesObjekt.setSemester(aktuellesModul.getSemester())
altesObjekt.setEcts_punkte(aktuellesModul.getEcts_punkte())
...
```

Alle Unterobjekte können jedoch der Einfachheit halber ausgeschnitten und ins alte Modul eingefügt werden, nachdem dessen Unterobjekte gelöscht wurden:

Code: #20

```
altLvs = altesObjekt.contentItems()
for altLv in altLvs:
  idLv = altLv[0]
  altesObjekt.manage_delObjects([idLv,])
```

Code: #21

```
items = aktuellesModul.contentItems()
for item in items:
  idModulObjekt = item[0]
  objs = aktuellesModul.manage_cutObjects([idModulObjekt,])
  altesObjekt.manage_pasteObjects(objs)
```

Schließlich wird das alte Modul samt allen Unterobjekten auf "veröffentlicht" gesetzt und die Modul-Kopie gelöscht. Um dem Benutzer keine 404-Fehlerseite anzuzeigen, wird eine Exception ausgelöst, die ihn automatisch an die Stelle weiterleitet, zu der das als Parameter übergebene Objekt verschoben wurde:

Code: #22

```
new_obj = dstFldr[aktuellesModulID]
raise state_change.ObjectMoved(new_obj,new_obj)
```

In diesem Skript ist der Name des öffentlichen Modulordners definiert und muss beim Portieren auf einen anderen Server oder bei einer Neuinstallation eventuell angepasst werden:

Code: #23

```
dstFldr = context.module                    # Name des Modulordners: module
```

Hieraus lässt sich erkennen, dass es zu einem Fehler kommt, wenn sich Module in einem Unterordner befinden. Zur Lösung wurde versucht, den Ursprungspfad des Moduls mit Hilfe von Methoden wie "absolute_url" in einem unsichtbaren Feld des Moduls zu speichern. Dies scheiterte jedoch daran, dass sich aus dem gespeicherten Pfad nicht mehr das Pfad-Objekt gewinnen ließ, das zum Verschieben benötigt wird, oder der Pfad fehlerhaft abgespeichert wurde.
Somit sollte in dieser Version vermieden werden, Modulordner zu verschachteln, bis eine Lösung für dieses Problem - eventuell im Schema des Moduls - gefunden wird.

Nachdem das Modul nun zum ersten Mal den Modulworkflow komplett durchlaufen hat, ist es offiziell genehmigt, veröffentlicht und auch von unangemeldeten Nutzern einsehbar. Um nun noch Änderungen durchführen zu können, muss sich der Modulverantwortliche über den Übergang "sperren und bearbeiten" eine Kopie des Moduls ziehen. Dabei kommt das gleiche Skript zum Einsatz wie im Übergang "init". Im Gegensatz zur Initialisierungsrunde wird der Modulverantwortliche dabei nicht per E-Mail benachrichtigt; lediglich die anderen Dozenten erhalten jeweils eine E-Mail, dass ihre Lehrveranstaltung für sie zu Bearbeitung bereit liegt. Das Modul und alle Unterobjekte im öffentlichen Modulordner werden auf "gesperrt" gesetzt um zu verhindern, dass eine weitere Kopie gezogen werden kann, bevor die erste Kopie den kompletten Workflow durchlaufen hat. So wird Redundanz vermieden. Deshalb wurde auch - anders als in der Vorgängerversion - auf die Möglichkeit verzichtet, das Modul manuell entsperren zu können.

4.3.5. Der Prüfungs- und Prüfungsvorleistungs-Workflow

Die Inhaltstypen CTVorleistung und CTPruefung haben einen gemeinsamen Workflow, *WF_Pruefung_Vorleistung*. Sie sind beide Unterobjekte eines Moduls und im Verantwortungsbereich des Modulverantwortlichen.
Privat angelegt genügt es einem einfachen OK durch den Modulverantwortlichen um zu kennzeichnen, dass der Inhalt der Elemente für ihn in Ordnung ist.
Der mit gestrichelten Pfeilen dargestellte Übergang findet nur indirekt im Workflow WF_Pruefung_Vorleistung statt, denn im Genehmigungsprozess wird immer das vollständige Modul betrachtet; Statusänderungen werden nur am Modul vorgenommen, während die Unterobjekte des Moduls ihren Status mit dem Modul ändern.

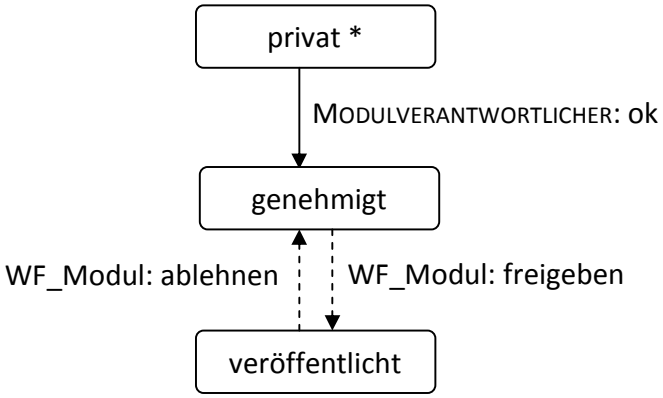

Abbildung 11: Der Prüfungs- und Prüfungsvorleistungs-Workflow

So wurden zwei unsichtbare Übergänge definiert, welche die Zustandsänderung von *genehmigt* zu *veröffentlicht* und zurück ermöglichen. Es ist nicht nötig, den Zwischenschritt Genehmigung durch den STUDIENGANGSVERANTWORTLICHEN mit zu modellieren, da der angezeigte Status zu dieser Zeit ebenfalls "genehmigt" ist. Auf diese Weise geschieht das Ausführen des Übergangs etwas schneller.

4.3.6. Der Lehrveranstaltungs-Workflow

Der Workflow WF_CTLehrveranstaltung greift immer dann, wenn Lehrveranstaltungen unabhängig vom Modul betrachtet werden, dies tritt dann auf, wenn aus dem Modul heraus Lehrveranstaltungen in die Benutzer-Order der Dozenten kopiert werden.
Angelegt werden Lehrveranstaltungen zunächst innerhalb ihres Moduls im Zustand "privat". Ihr Zustand kann nicht manuell verändert werden, da die Übergänge unsichtbar sind. Beim Kopieren des Moduls bleiben sie innerhalb es Moduls privat und dienen dort als Platzhalter.

Lehrveranstaltungen benötigen kein eigenes Initialisierungsskript, da die Initialisierung der Lehrveranstaltungen komplett durch den Modul-Workflow gesteuert wird. Gleichermaßen werden durch das Modul sämtliche Rollen gesetzt.
Durch das Skript copy_modul_and_lv aus dem Modul-Workflow wird von jeder Lehrveranstaltung zusätzlich eine Kopie gezogen und in den Benutzer-Ordner des verantwortlichen Dozenten kopiert, danach setzt das Skript den Anfangszustand dieser Kopie auf "entwurf". Dieser Zustand kennzeichnet unbearbeitete Lehrveranstaltungen.
Die folgende Skizze bildet den Lehrveranstaltungs-Workflow ab und deutet mit gestrichelten Pfeilen die Schnittstellen zum Modul-Workflow an.

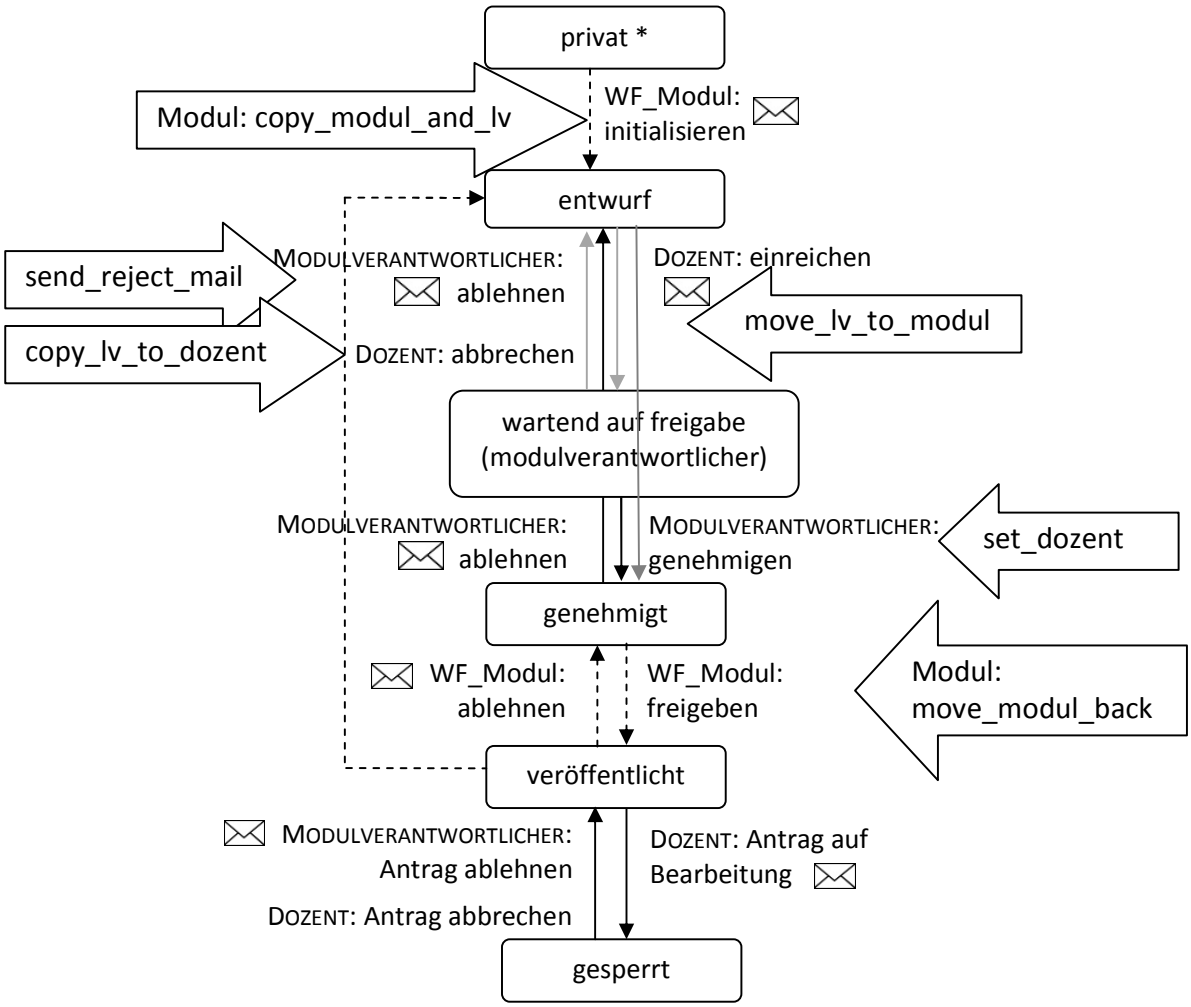

Abbildung 12: Der Lehrveranstaltungs-Workflow

Nachdem der Dozent seine Lehrveranstaltung mit den entsprechenden Daten gefüllt hat, kann er sie an den Modulverantwortlichen einreichen. An den grauen Pfeilen ist zu erkennen, dass hier ein Kopiervorgang stattfindet, und zwar einmal in Genehmigungsrichtung mit Hilfe des Skripts move_lv_to_modul und das andere Mal in die Gegenrichtung zurück zum Entwurf, worauf später eingegangen wird.
Das move_lv_to_modul-Skript funktioniert nach dem gleichen Prinzip wie das Zurückschieben der Lehrveranstaltungen im Modul: Das Modul wird im Zielordner gesucht - in diesem Fall der Benutzerordner des Modulverantwortlichen, der aus dem versteckten Feld "mv" der Lehrveranstaltung ausgelesen wird. Der Ordner wird ermittelt über:

Code: #24

```
ndstFldr = context.portal_membership.getHomeFolder(mv.getId())
```

In einem weiteren, unsichtbaren Feld ist die Information gespeichert, zu welchem Modul die Lehrveranstaltung gehört; dieses Feld wird mit obj.getModul() ausgelesen. Nun werden wieder die Modul-IDs miteinander verglichen um das richtige Modul ausfindig zu machen.
Die Platzhalter-Lehrveranstaltung im Modul wird gelöscht und durch die bearbeitete Lehrveranstaltung ersetzt. Es verbleibt keine Kopie im Ordner des Dozenten.
Dies funktioniert ebenfalls, wenn der Modulverantwortliche gleich dem Dozenten ist: Die Lehrveranstaltung wird aus dem Ordner des Modulverantwortlichen in das Modul verschoben, das sich ebenfalls in diesem Ordner befindet, das bedeutet, es wird eine Ebene tiefer geschoben, wenn man das Modul als Ordner betrachtet.

Die Lehrveranstaltung erhält im Modul den Zustand "wartend auf Freigabe durch den Modulverantwortlichen", falls sie von einem Dozenten eingereicht wurde, und muss nun vom Modulverantwortlichen genehmigt werden. Im Fall dass der Modulverantwortliche den Einreichen-Übergang aufgerufen hat, wird direkt der Übergang "genehmigen" ausgelöst.
Dazu musste wieder ein unsichtbarer Hilfsübergang erstellt werden, denn des strikte Rechtemanagement von Plone verbietet dem Modulverantwortlichen, den Übergang "genehmigen_Modulverantwortlicher" auszulösen, wenn es aus einem Skript heraus geschieht, das als Dozent aufgerufen wird - selbst wenn es die gleiche Person mit beiden Rollen ist.

Hier ist ebenfalls eine Besonderheit in der Mail-Benachrichtigung: Der Modulverantwortliche erhält für jede zur Genehmigung ausstehende Lehrveranstaltung eine Mail, wenn es nicht seine eigene ist; außerdem wird bei der letzten eingereichten Lehrveranstaltung die Bemerkung eingefügt, dass das Modul nun komplett ist, das heißt wenn keine weitere Lehrveranstaltung im Modul im Zustand "privat" ist.

Nun ist es die Aufgabe des Modulverantwortlichen, die Lehrveranstaltungen der anderen Dozenten in seinem Modul zu kontrollieren, die noch im Zustand "wartend auf Freigabe durch den Modulverantwortlichen" stehen. In diesem Zustand gibt es - neben der Möglichkeit sie zu genehmigen - die Übergänge abbrechen und ablehnen. Dies sind zwei unterschiedliche Sichten auf den gleichen Vorgang: Das Zurücksetzen des Moduls auf den Zustand "entwurf", wobei durch die entsprechenden Rollen im Workflow der Übergang "abbrechen" nur dem DOZENTEN angezeigt wird, und "ablehnen" nur durch den Modulverantwortlichen durchgeführt werden kann.
Aber selbst nach der Genehmigung durch den Modulverantwortlichen kann er seine Meinung ändern und die Lehrveranstaltung noch ablehnen. Der Übergang ist aber hauptsächlich dafür gedacht, dass der Modulverantwortliche nach der Ablehnung des gesamten Moduls durch höhere Instanzen Entscheidungsfreiheit hat, welche Lehrveranstaltung noch einmal zur Überarbeitung an den Dozenten zurück gegeben werden soll, falls das überhaupt geschehen soll. Dies ist insofern sinnvoll, dass Dozenten ihre Lehrveranstaltung nicht noch einmal erhalten, wenn keine Änderung notwendig ist und stellt somit eine Zeit- und Belastungsersparnis für die Dozenten dar.

Wieder werden die Zwischenschritte nicht in den Workflow einbezogen, da es nach außen hin keine Rolle spielt, in welchem Zustand sich die Lehrveranstaltung zwischen "genehmigt durch den Modulverantwortlichen" und "veröffentlicht" befindet.
Dozenten und Modulverantwortliche können selbst einen Nachfolge wählen; diese Änderungen beeinträchtigen den Workflow nicht und werden erst wirksam, wenn das Modul durch die Gremien ist. Diese Bearbeitungsmöglichkeit ließe sich jedoch auch in dem jeweiligen Bearbeiten-View mit einem Administrationsrecht schützen.

Nach der Veröffentlichung des Moduls kann der Dozent keinen Einfluss mehr auf das Modul nehmen. Wenn er dennoch seine Lehrveranstaltung bearbeiten möchte, kann er einen Veröffentlichungsantrag an den Modulverantwortlichen stellen. Dies geschieht wie in allen anderen Genehmigungsschritten über eine generierte E-Mail. Die Lehrveranstaltung geht derweil über in den Zustand "gesperrt". Falls der Modulverantwortliche nicht einverstanden ist, kann er die E-Mail ignorieren, oder die Lehrveranstaltung entsperren, indem er den Übergang "antrag ablehnen" auslöst. Bei Einverständnis hingegen löst er über den Modul-Workflow den Übergang "sperren und bearbeiten" des ganzen Moduls aus, sodass jeder der beteiligten Benutzer seinen Teil des Moduls erhält und die Chance hat, Änderungen daran vorzunehmen. Wünscht er dies nicht, kann er seine Lehrveranstaltung ohne Änderungen einreichen.

4.3.7. Ordner-Workflows

Der Workflow WF_CTFolder wird von den Inhaltstypen CTDozentenFolder und CT_ModulFolder verwendet.
Durch die Kopiervorgänge sind Module mehrfach im System vorhanden. Um im ReferenceBrowser nicht versehentlich das Modul aus dem Benutzer-Ordner zu verlinken, wurden bereits verschiedene Zustände von der Verlinkung ausgenommen. Nun wurde bewusst mit einem weiteren Workflow die Sichtbarkeit jener Ordner verändert, die Module und Dozenten enthalten. Auf diese Weise wird die Navigation mit dem ReferenceBrowser erleichtert, da nur der öffentliche Modulordner und der Dozenten-Ordner angezeigt werden, sowie alle darin enthaltenen Unterordner. Der Folder-Workflow basiert auf dem vordefinierten Folder_Workflow; es wurde darin lediglich der Anfangszustand von *visible* auf *published* gesetzt. Dieser Workflow wurde detailliert im Einführungskapitel zu Workflows beschrieben (Kapitel 2.4.1 *Überblick*).

4.3.8. Benachrichtigungen

E-Mail-Benachrichtigungen wurden nicht in jedem Workflowschritt eingesetzt, sondern grundsätzlich immer dann, wenn sinnvoll erschien. Im Prinzip ist das, wenn die Aktion eines Benutzers erforderlich ist: Bei Genehmigungsanfragen und bei der Ablehnung eines Moduls oder einer Lehrveranstaltung. Eine Übersicht, in welchen Skripten welche Benachrichtigungen

implementiert sind, bietet die folgende Tabelle. Weiterhin zeigen die Brief-Icons in den Workflow-Diagrammen, in welchen Schritten es zu Benachrichtigungen kommt.

Tabelle 4: Übersicht der Mail-Benachrichtigungen

Skript-Bezeichnung	Zweck
check_for_states	Anforderung der Genehmigung des Studiengangsverantwortlichen.
copy_modul_and_lv	Beim Initialisieren: Benachrichtigung der Dozenten und des Modulverantwortlichen. Ausgelöst durch den Modulverantwortlichen: Benachrichtigung der Dozenten
mail_antrag_abgelehnt	Nachricht an den Dozenten, dass der Modulverantwortliche im Moment keine Änderungen am Modul genehmigt.
mail_modul_abgelehnt	Der Studiengangsverantwortliche oder die Gremien haben das Modul abgelehnt, Nachricht an den Modulverantwortlichen
move_lv_to_modul	Der Dozent hat seine Lehrveranstaltung eingereicht und wartet auf die Genehmigung durch den Modulverantwortlichen, Bedingung: Der Dozent ist nicht gleichzeitig der Modulverantwortliche. Oder: Benachrichtigung des Modulverantwortlichen, wenn das Modul mit dem Einreichen dieser Lehrveranstaltung vollständig ist.
send_lock_mail	Benachrichtigt Modulverantwortlichen, dass ein Dozent seine Lehrveranstaltung im veröffentlichten Modul bearbeiten möchte.
send_reject_mail	Nachricht an den Dozenten, dass seine Lehrveranstaltung abgelehnt wurde.
set_dozent_free	Registrierungs-E-Mail an den Benutzer

Die Skripte zur E-Mail-Benachrichtigungen wurden in die bestehenden Workflow-Skripte integriert, da einem Übergang nur zwei Skripte zugeordnet werden können: Eins vor und eins nach der Ausführung, und nur nach der Ausführung des Workflow-Schritts kann das Kommentar-Feld ausgelesen werden:

Code: #25

```
comments = wf_tool.getInfoFor(obj, 'comments')
if (comments != ""):
  comments = "Bemerkung: " + comments
```

Wenn vorhanden, wird der Kommentar an den Nachrichtentext angefügt.
Zudem sollte die E-Mail nur dann gesendet werden, wenn die Ausführung erfolgreich war.

Dabei sind vielerlei Ausnahmen zu beachten, wie sie teilweise schon in der Beschreibung der einzelnen Workflows erwähnt wurde. Ein weiteres Beispiel ist, dass die Änderungsantrags-Mail im Skript send_lock_mail nicht gesendet werden soll, wenn der Übergang vom Skript ausgelöst wird:

Code: #26

```
akteur = wf_tool.getInfoFor(obj, 'actor')
if(email != "" and akteur == dozentId):
  context.MailHost.send(messagetext , str(email), 'noreply@fh-zwickau.de',
  'Modulportal: Aenderungsantrag')
```

Denn der Übergang zum Zustand "gesperrt" wird sowohl ausgelöst, wenn der Dozent seine Lehrveranstaltung bearbeiten möchte, als auch wenn alle Lehrveranstaltungen durch das Modul-Skript copy_modul_and_lv auf "gesperrt" gesetzt werden. Indem auf den Akteur geprüft wird, kann herausgefunden werden, ob tatsächlich der Dozent den Übergang ausgelöst hat.

Als E-Mail-Adresse kann eine beliebige, auch nicht-existente Adresse angegeben werden. Oft ist es notwendig, Variablen auf String zu casten um keine ASCII-Dekodierungsfehler zu erhalten. Unicode-Unterstützung ist prinzipiell in Plone realisiert, funktioniert aber in Verbindung mit E-Mails nicht, deshalb wurde auf Umlaute verzichtet.

Worklists wurden vor allem zur erleichterten Auffindbarkeit dort eingesetzt, wo die betreffenden Dokumente nicht im eigenen Ordner des jeweiligen Benutzers liegen, und zur Erinnerung, welche Dokumente noch bearbeitet oder genehmigt werden müssen. Diese sind dann in einer einzigen Liste gesammelt.
Hier war es sinnvoll, die Berechtigungen für das Anzeigen der Revisionsliste zu ändern. Die ursprüngliche Berechtigung 'Review portal content' war sinnvoll, da nur Redakteure Revisionslisten angezeigt bekommen sollten. Mit dem Ändern in 'View' kann sich jeder Benutzer seine Revisionsliste vollständig anzeigen lassen, wo vorher nur eine leere Seite angezeigt wurde.
Konkret wurden folgende Worklists definiert:

Tabelle 5: Übersicht der Review-Listen

Rolle	Zustand	Workflow
Gremien	wartend auf freigabe (gremien)	WF_CTModul
Studiengangsverantwortlicher	wartend auf freigabe (studiengang)	WF_CTModul
Modulverantwortlicher	veröffentlicht	WF_CTModul
Dozent	veröffentlicht	WF_CTLehrveranstaltung

Der GREMIEN-Benutzer erhalten keine E-Mails zur Benachrichtigung, da es sich bei diesem um keinen aktiven Benutzer handelt, sondern um ein Komitee, das nicht häufig zusammentritt.

Somit wird eine Sammlung der Module benötigt. Diese Funktion ist der Hauptzweck einer Revisionsliste. Somit erhält der GREMIEN-Benutzer beim Einloggen eine Übersicht, welche Module auf Freigabe warten.
Der Studiengangsverantwortliche erhält ebenfalls einer Revisionsliste, neben der E-Mail-Benachrichtigung. Zwar erhält die E-Mail einen Link zum entsprechenden Modul, aber zur erleichterten Auffindung innerhalb des Portals erschien hier eine Revisionsliste sinnvoll.
Aus dem gleichen Grund wurden Worklists für Modulverantwortliche und Dozenten definiert; allerdings nicht um sie zu informieren, wo die Module und Lehrveranstaltungen liegen - denn sie liegen in den eigenen Benutzerordnern - sondern zur leichteren Auffindung der Module und Lehrveranstaltungen im öffentlichen Ordner, für die sie verantwortlich sind. Wenn ein Modul zum Überarbeiten zurückgezogen wurde, verschwindet es aus der Revisionsliste.
Aufgrund des lokalen Charakters der Rollen werden jedem Benutzer nur die Dokumente angezeigt, für die ihm die Rolle zugeteilt wurde.

4.4. Systemeinstellungen

Um dieses Projekt auf einen anderen Server portieren zu können, sind eine Reihe von Systemeinstellungen im ZMI zu treffen[5]. Sie sind im Folgenden tabellarisch aufgelistet.

Tabelle 6: Systemeinstellungen

http://whz-zki-03.zw.fh-zwickau.de:10081/Plone...	**Aufgaben**
/acl_users/portal_role_manager/manage_roles /manage_access	Rolle Gremien, Modulverantwortlicher, Studiengangsverantwortlicher und Dozent erstellen und aktivieren (Siehe Kapitel 2.3.2. Rollen im Detail)
/portal_workflow/manage_selectWorkflows	Den Inhaltstypen die gleichnamigen Workflows zuordnen und abspeichern
/manage_access	Berechtigungen für Member setzen: Set own password, Use mailhost services
/manage_access	Owner Berechtigung "Delete objects" entziehen
alternativ: /portal_actions	Feld deaktivieren bei cut, copy, paste, delete, folderContents
/portal_types/Folder/manage_propertiesForm	CTLehrveranstaltung und CTModul als erlaubten Typ wählen
/portal_skins/custom/ referencebrowser_popup/manage_workspace	Nicht zu verlinkende Zustände definieren (siehe Kapitel 4.3.2 Der Dozenten-Workflow)
portal_skins/custom/ full_review_list/manage_workspace	Ändern der Berechtigung in von 'Review portal content' in 'View'
/prefs_mailhost_form	E-Mail-Einstellungen vornehmen (hier: mailhost.fh-zwickau.de, Port 25)
nach eigenem Ermessen: /manage_access	Der Rolle OWNER die Berechtigungen entziehen: *View management screens*: Verbietet Anhängen von /manage *Manage properties*: Schaltet den Zugriffs-Tab aus
/manage_access	ANONYMUS die Berechtigung *Add portal member* entziehen um zu verhindern, dass sich Nutzer selbst über /join_form anmelden können

[5] Es wird dabei angenommen, dass das Produkt CTModule bereits erfolgreich installiert ist

5. Zusammenfassung und Ausblick

5.1. Offene Fragen

Bei der Zusammenführung der beiden Projekte kam es zu kleineren Inkonsistenzen, die vor der Inbetriebnahme des Projekts angepasst werden müssten, dazu zählt, dass in der Bearbeitungsphase ein Warnhinweis angezeigt wird, dass das Modul in keinem Studiengang verlinkt sei; der Zugriffs-Tab des Moduls zeigt die gesetzten Rollen oft fehlerhaft an und es wurde nie ganz geklärt, wer das Recht zum Generieren des Modulhandbuchs als PDF-Datei bekommen soll. Der entsprechende Tab kann mit einer Berechtigung, aber keiner Rolle geschützt werden. Bei Bedarf kann dafür ein eigenes Recht erstellt werden. Aktuell wurde es so gestaltet, dass Benutzer mit den Rollen GREMIEN und MANAGER mit Hilfe des Rechts *Access inactive portal content* diesen Tab sehen und aufrufen können.

Die Arbeit mit Templates war im Grunde nicht als Bestandteil dieser Thesis gedacht, da dieses Thema recht komplex werden kann, deshalb wurde an einigen Stellen verzichtet, tiefer in diese Materie einzudringen. Im Text erwähnt wurde diesbezüglich das Problem, dass private Module nur in der Bearbeiten-Ansicht des Studiengangs angezeigt werden sollten, und nicht in der Anzeigen-Ansicht.

Weiterhin besteht noch das Problem, dass es aufgrund einer fehlenden Kategorisierung der Module nicht möglich ist zu bestimmen, ob ein Modul dem aktuellen Studiengang gehört oder ob es sich um ein Fremdmodul aus einem anderen Studiengang handelt. Der Studiengangsverantwortliche wird derzeit mit der Aktualisierung des Studiengangs für jedes Modul neu gesetzt. Zur Behebung dieses Problems erscheint der Gedanke logisch, nach Fachbereichen zu gliedern und den Studiengangsverantwortlichen nur zu aktualisieren, wenn der Fachbereich überein stimmt. Dabei sollte geklärt werden, inwieweit Module überhaupt auf diese Weise kategorisiert werden können, da beispielsweise zum Fachbereich Physikalische Technik / Informatik (PTI) drei Studiengänge gehören, die alle zum überwiegenden Teil Module aus der PTI nutzen.

Bei der Übertragung des Projekts auf den Plone-Server der Westsächsischen Hochschule wurde festgestellt, dass dort die IDs der Inhaltselemente durch Plone selbst generiert werden statt die Information aus dem Titel-Feld zu nutzen. Diese Einstellung konnte nicht verändert werden. Um die Funktionstüchtigkeit des Dozenten-Registrierungsskripts dennoch zu gewährleisten, wurde in der Plone-Konfiguration die Einstellung "Kurzname der Artikel anzeigen?" auf "ja" gesetzt und im Benutzer-Profil des Administrators die Einstellung "Das Bearbeiten der Kurznamen erlauben" angekreuzt.
Wenn nun ein Dozenten-Inhaltselement angelegt wird, muss der Kurzname mit dem Anmeldenamen übereinstimmen um einen korrekten Anmeldenamen für den Benutzer zu erhalten. Bei anderen Inhaltstypen wurden keine Probleme festgestellt.

5.1.1. LDAP-gestützte Authentifizierung

Plone bietet die Möglichkeit, Benutzer aus einer LDAP-Datenbank auszulesen, wie sie auch hochschulintern an der WHZ und beispielsweise auch im Bildungsportal Sachsen[6] genutzt wird. Durch Anwendung dieser Technik könnte die Akzeptanz des Modulhandbuchportals erhöht werden, da auf diese Weise die im ZKI vergebenen Login-Daten verwendet werden würden. Allerdings wäre dies schwierig in das bisherige System einzuarbeiten, weil das Anlegen von Benutzern allein nicht ausreicht, sie mit den verwendeten Inhaltstypen zu verlinken. Außerdem findet sich in der LDAP-Datenbank ein hoher Anteil an Benutzern, für die das Modulhandbuchportal nicht vorgesehen ist, namentlich sind dies Studenten und Mitarbeiter. Somit ist die praktische Relevanz im Vergleich zu den Faktoren Realisierbarkeit und Nutzen gering.

5.1.2. Versionierung

Mit Hilfe einer Versionsverwaltung können die Änderungen an einem Inhaltselement zum Vergleich mit der Vorgängerversion gespeichert und bei Bedarf für den Nutzer hervorgehoben werden. Ein solches System könnte den Verantwortlichen die Genehmigungsentscheidung erleichtern.

Es existiert eine Reihe vorgefertigter Zusatzprodukte zur Versionierung in Plone. Als "Produkt der Wahl" gilt *CMFEditions*[7], das ab Plone-Version 3.0 standardmäßig in Plone integriert ist. Die vorliegende Plone-Version ist 2.5.5. Der Versuch, die aktuelle *CMFEditions*-Version 1.1.7 als Zusatzprodukt zu installieren, schlug fehl. Schließlich wurde *CMFEditions* in dieser Arbeit in der Version 1.0rc1 getestet. Die getestete Version bringt laut Hersteller[8] nur grundlegenden Support für Archetypes-basierende Inhaltstypen, doch selbst dies konnte bei den Inhaltstypen des Produkts CTModule nicht festgestellt werden. Im Test boten nur Plone-eigene Typen wie Bilder die Option an, das Element als Version abzuspeichern.

Es wurde weiterhin *StagingAddOn*[9] Version 1.7.1 getestet, ein Zusatzprodukt, das physikalisch Sicherheitskopien anlegt. Jedoch greift dieses Produkt in die selbst-definierten Workflows ein, erstellt nicht automatisch Sichrungen und arbeitete bei Tests nicht mit den selbst erstellten Workflow-Skripten zusammen. Das Produkt *iterate*[10] steht im Moment nicht zum Download zu Verfügung, und CMFDiffTool[11] war nicht installierbar.

[6] https://bildungsportal.sachsen.de
[7] http://plone.org/products/cmfeditions
[8] http://plone.org/products/cmfeditions/releases/1.0rc1
[9] http://plone.org/products/stagingaddon
[10] http://plone.org/products/iterate/
[11] http://plone.org/products/cmfdifftool

Es ist zu bezweifeln, dass eine Versionierung in Verbindung mit den Verschiebevorgängen innerhalb der Module funktionieren würde, denn wie schon in Verbindung mit den Verlinkungen innerhalb der Studiengänge festgestellt wurde, ändert sich durch das Löschen eines Elements die UID, wodurch es intern als anderes Element betrachtet wird. Abhilfe könnte wieder das Kopieren der Feldinhalte mit dem anschließenden Löschen der Kopien schaffen.

Wenn es darum geht, ganze Studiengänge zu versionieren, zum Beispiel für den Übergang vom Bachelor-Studiengang Informatik 2005 zu 2006 wäre die einfachste Lösung, einen zweiten Studiengang anzulegen, in welchem an den Stellen andere Module gelinkt werden, an denen sich der Studiengang verändert hat. Eine Funktionalität zu implementieren, die es beim Verlinken in den Studiengang möglich macht, die Version eines Moduls auszuwählen, ist nicht trivial und kann wahrscheinlich durch kein existierende Zusatzprodukt abgedeckt werden.

5.2. Ergebnisse und Schlusswort

Ein Rollenkonzept wurde entwickelt und umgesetzt, sowie mit dem bestehenden, Archetypes-basierenden Produkt zusammengeführt.
Der geforderte Workflow wurde in Form von vielen Einzel-Workflows umgesetzt, da die verschieden Inhaltstypen eine unterschiedliche Steuerung erforderten. Dabei musste der Zusammenhang der Inhaltstypen zueinander beachtet werden. Durch das Verschieben der Module existiert immer eine aktuell gültige Fassung im öffentlichen Modulordner. Unbearbeitete, private Module können nicht von unangemeldeten Nutzern eingesehen werden.
Der Administrator hat volle Kontrolle darüber, welche Benutzer das Modulportal benutzen dürfen, eine eigenständige Registrierung ist nicht möglich. Er kann mit geringem Aufwand Benutzer erstellen, kategorisieren und Inhaltselementen zuordnen.
Um ein neues Modul zu erhalten, kann der Administrator das Modul vollständig selbst erstellen, oder ein Modul-Gerüst durch die Verantwortlichen mit Inhalt füllen lassen.
Der Arbeitsplan des Administrators in letzterem Fall ist: Das Anlegen des Moduls und der Lehrveranstaltungen mit ihren Verantwortlichen, das Aktualisieren oder Erstellen des Studiengangs mit dem Festlegen des Studiengangsverantwortlichen, das Aktualisieren der verlinkten Module über den Workflow des Studiengangs und schließlich das Wählen des Übergangs "initialisieren". Damit wurde das gesamte Rechtesystem initialisiert und kann im Folgenden von allen beteiligten Personen genutzt werden; so kann ein Benutzer mit der entsprechenden Rolle im Inhaltselement alle Übergänge auslösen, für die die Rolle prinzipiell berechtigt ist. Die Genehmigungs- und Ablehnungsvorgänge wurden mit E-Mail-Benachrichtigungen versehen, und Revisionslisten dienen zur besseren Auffindung der Inhaltselemente im Portal.

Quellenverzeichnis

Verwendete Literatur

[Fri06] FRIEDRICH, HANS JÖRG: *Content Management mit Plone - Gestaltung, Programmierung, Anwendung und Administration*. Springer-Verlag, Berlin und Heidelberg, 2006.

[WaP07] WALEROWSKI, PETER: *Plone 2.5. Umfassender Einstieg in Plone - Zope, Python und CMF*. Bonn: Galileo Press, 1. Auflage, 2007.

[WaZ01] WALEROWSKI, PETER: *Zope - Content-Management-& Web-Application-Server*. Heidelberg: dpunkt, 2001.

[McK05] MCKAY, ANDY: *Plone - Leitfaden für Administratoren und Entwickler*. München: Addison-Wesley, 2005.

[RRZNP] DIVERSE AUTOREN: RRZN-Handbuch Python - *Grundlagen, fortgeschrittene Programmierung, Praxis.* Leibniz Universität Hannover, 2008

[SCH09] SCHRAPS, MATHIAS: Bachelor-Thesis (in Vorbereitung) - *Modellierung und Realisierung von Inhaltstypen im CMS Plone mit Hilfe des Archtypes-Frameworks*, Westsächsische Hochschule Zwickau, 2009

Thesen

1. Es musste ein Sicherheitskonzept ausgearbeitet werden, das es ermöglicht, objektspezifische Berechtigungseinstellungen vorzunehmen

2. Dieses Berechtigungssystem musste die vorhandene, Archetypes-basierte Dokumentenstruktur eingearbeitet werden, u. A. durch einen eigenen Inhaltstyp für Benutzer

3. Es musste ein Gesamtworkflow entsprechend der Anforderungen der Westsächsischen Hochschule Zwickau implementiert werden

4. Verschiedene Workflows musste die unterschiedlichen Inhaltstypen und ihrer Verbindung untereinander berücksichtigen

5. Die Workflows wurden um eine Reihe von Skripten ergänzt, um eine komfortable Nutzung und flexible Steuerung der Workflows zu ermöglichen

6. Geeignete Benachrichtigungswerkzeuge mussten gefunden und umgesetzt werden