Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES

BACHELOR-THESIS
im Studiengang Technische Redaktion

Dynamische Filterung von benutzerorientierten Inhalten auf Basis eines Content-Management-Systems

JAN OEVERMANN (30965)

BETREUER:
Prof. Dr. Wolfgang Ziegler
Prof. Martin Schober
Frank Ully

ZEITRAUM:
26.01. – 25.04.2013

BACHELOR-THESIS:
Dynamische Filterung von benutzerorientierten Inhalten auf Basis eines Content-Management-Systems
Dynamic filtering of user-oriented content based on a content-management-system

AN DER:
Hochschule Karlsruhe - Technik und Wirtschaft
Fakultät für Informationsmanagement und Medien
Studiengang Technische Redaktion

BETREUER:
Prof. Dr. Wolfgang Ziegler, Hochschule Karlsruhe
Prof. Martin Schober, Hochschule Karlsruhe
Frank Ully, Projektron GmbH

ORT:
Berlin

ZEITRAUM:
26.01. – 25.04.2013

ABSTRACT

In der Technischen Dokumentation mangelt es bisher an konkreten Konzepten zur dynamischen benutzerorientierten Aufbereitung von Inhalten. Doch gerade bei der Dokumentation komplexer Software werden Benutzer oft mit einer großen Menge an Informationen konfrontiert, die für sie nicht relevant sind.

Die vorliegende Arbeit beschäftigt sich mit der prototypischen Umsetzung eines Konzepts zur Erstellung, Publikation, Darstellung und Filterung benutzerorientierter Inhalte. Aufbauend auf einem studentischen Wettbewerbsbeitrag wird ein für die Software-Dokumentation gültiges Modell für die benutzer- und systemspezifische Klassifizierung von Inhalten entworfen.

Die konzeptionellen Ergebnisse werden in das Content-Management-System COSIMA go! implementiert auf dessen Basis es dadurch möglich ist, integriert in den Redaktionsprozess, benutzerorientierte Inhalte zu erstellen und zu publizieren. Zur dynamischen Darstellung und Filterung dieser Inhalte wird auf Basis von Web-Technologien ein Framework entwickelt. Die prototypische Anbindung des Frameworks an die Software Projektron BCS wird abschließend durch die Spezifizierung einer Schnittstelle vorgenommen.

INHALTSVERZEICHNIS

ABBILDUNGSVERZEICHNIS

TABELLENVERZEICHNIS

CODEAUSSCHNITTE

ABKÜRZUNGSVERZEICHNIS

API	Application Programming Interface
BCS	Business Coordination Software
CBT	Computer-Based-Training
CGO	COSIMA go!
CMS	Content-Management-System
CMP	Cross-Media-Publishing
CS	CoffeeScript
CSS	Cascading Style Sheets
CVM	Content-Variant-Management
DITA	Darwin Information Typing Architecture
DOM	Document Object Model
DTD	Document Type Definition
ERP	Enterprise-Resource-Planning
FPI	Formal Public Identifier
GUI	Graphical User Interface
HTML	Hypertext Markup Language
IETD	Interaktive Elektronische Technische Dokumentation
IETF	Internet Engineering Task Force
IO	Informationsobjekt
ISO	International Organization for Standardization
JS	JavaScript
OH3	Online-Hilfe 3.0
SGML	Standard Generalized Markup Language
SSP	Single-Source-Publishing
TD	Technische Dokumentation
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
XML	Extensible Markup Language
XSD	XML Schema Definition
XSL	Extensible Stylesheet Language
XSLT	XSL Transformation
XPath	XML Path Language

1 EINLEITUNG

1.1 MOTIVATION

Stellt man sich die Frage, warum Benutzer so selten einen Nutzen aus Dokumentationen und im Speziellen Online-Hilfen ziehen (vgl. Thiemann 2008:8), kommt man zu einem überraschenden Ergebnis: Die Information, die der Nutzer sucht, ist oft vorhanden – er findet sie jedoch nicht (vgl. Closs 2012b:50).

Verantwortlich dafür ist die steigende Zahl an Zielgruppen und Produktvarianten, die eine Dokumentation abdecken muss. Um die Informationsmenge einzuschränken, der ein Benutzer ausgesetzt ist, werden immer öfter Handbücher ausgeliefert, die automatisiert auf die passende Produktkonfiguration angepasst werden (vgl. Rath 2005). Ein Schritt, der sowohl dem Hersteller (niedrigere Druckkosten) als auch dem Anwender (bessere Bedienbarkeit) nützt.

Doch selbst mit einer auf das Produkt angepassten Dokumentation kann es noch zur „Informationsüberflutung" kommen (vgl. Belikan 2007). Oft müssen zu viele Zielgruppen und Nutzertypen bedient werden, was zu umständlichen Formulierungen und Zusatzbemerkungen im Anleitungstext führt. Und trotz steigendem Bedarf ist die angepasste Auslieferung von Hilfe-Inhalten an den Benutzer für die meisten Redakteure noch Zukunftsmusik (vgl. Siegel 2013:150).

Auch das Verhalten auf Nutzerseite hat sich in den letzten Jahren stark verändert. Ein steigendes Bedürfnis nach informationeller Selbstbestimmung ist im Alltag fest verankert und spiegelt sich im täglichen Umgang mit Medien und Internet wider (vgl. Hasebrink 2011:14 und 22).

Auf diese veränderten Mediengewohnheiten muss auch die Dokumentationsbranche reagieren und sich stärker am individuellen Nutzer orientieren (vgl. Oehmig 2011:147). Die technischen Voraussetzungen sind durch moderne Webtechnologien und weit entwickelte Content-Management-Systeme bereits gegeben.

Benötigt wird eine Lösung, die sowohl die Bedürfnisse der Redakteure nach Prozessintegration und Erstellungseffizienz als auch die der Nutzer nach Individualisierung und Selbstbestimmung erfüllt.

1.2 METHODIK

Die Struktur der vorliegenden Arbeit orientiert sich an ihrem Titel *Dynamische Filterung von benutzerorientierten Inhalten auf Basis eines Content-Management-Systems*. Die einzelnen Bestandteile werden nacheinander vorgestellt und ihre Verbindungen zueinander hergestellt.

Im Kapitel *Benutzerorientierter Content* werden für die Arbeit nötige Grundlagen erarbeitet. Zunächst werden zentrale Begriffe eingeordnet und Gründe für den Einsatz von benutzerorientiertem Content erläutert. Nach einer allgemeinen Bestandsaufnahme wird das Konzept Online-Hilfe 3.0 näher betrachtet. Abschließend wird auf Grundlage der vorangegangenen Erkenntnisse eine erweiterte Konzeption für ein benutzerorientiertes Modell erarbeitet.

Darauf folgt das Kapitel *Content-Management-Systeme als Basis*. Nach einer allgemeinen Betrachtung solcher Systeme wird das Content-Management-System (CMS) Docufy COSIMA im Hinblick auf Besonderheiten und seine Customizing-Eigenschaften betrachtet. Auf Basis dieser Ergebnisse werden anschließend die Ergebnisse aus Kapitel 2 implementiert.

Auf die *Dynamische Filterung des Contents* wird im nachfolgenden Kapitel eingegangen. Hier werden zunächst allgemeine Anforderungen beschrieben und das verwendete Basissystem wird erläutert. Im Anschluss daran werden, ausgehend von den Ergebnissen aus Kapitel 3, die benötigten Anpassungen konzipiert und anschließend technisch umgesetzt.

Nachdem in Kapitel 3 die Implementierung in das CMS und im Kapitel 4 in das Framework durchgeführt wurde, wird als letzter Schritt die *Anbindung als Hilfesystem* vorgenommen. Hierbei wird zunächst die Projektron Business Coordination Software (BCS) vorgestellt, die als Stammsystem eingesetzt wird. Im Anschluss wird eine Schnittstelle zwischen den Systemen definiert und prototypisch implementiert.

Im letzten Kapitel *Fazit und Ausblick* werden die Ergebnisse der Arbeit zusammengefasst, bewertet und ein Ausblick für weitere Entwicklungen wird formuliert.

Im *Anhang* finden sich ausgewählte Dokumente, auf die im Laufe der Arbeit verwiesen wird. Der komplette Anhang ist auf der beiliegenden CD in digitaler Form hinterlegt.

Jedes Kapitel beginnt mit einer kurzen Beschreibung der behandelten Inhalte (Advance Organizer). In der Randspalte neben dem Text befinden sich bei längeren Abschnitten kurze Inhaltsangaben der Absätze (Marginalien). In der digitalen Version können Querverweise, Literaturangaben, Abkürzungen und URLs angeklickt werden, um zur entsprechenden Stelle zu gelangen.

1.3 AUSGANGSSITUATION

Ein Konzept, das eine dynamische Filterung nach individuellen Benutzereigenschaften zulässt, wurde im Frühjahr 2012 mit dem 1. Platz des *Studentischen Innovationspreises der tekom - intro* ausgezeichnet (vgl. TEKOM 2012). Die Online-Hilfe 3.0 (OH3) ist eine Weiterentwicklung der klassischen Online-Hilfe, die den Nutzer den Inhalt durch Schieberegler dynamisch an seine Bedürfnisse anpassen lässt (vgl. SCHÄFER/OEVERMANN/MEIER 2012:45ff). Basis für die Filterung sind Inhaltselemente und -fragmente, die nach Benutzereigenschaften bewertet[1] worden sind.

Aufbauend auf der prototypischen Umsetzung des intro-Wettbewerbsbeitrags wurden eine Bachelor-Thesis (vgl. MEIER 2012) und eine Projektarbeit (vgl. OEVERMANN 2012) erstellt (beide in Abschnitt 2.4.3 erläutert).

1.4 PROBLEMSTELLUNG UND ZIELE

Die Arbeit hat eine erste prototypische Implementierung des OH3-Konzepts für den praktischen Einsatz in der Software-Dokumentation zum Ziel.

Basierend auf dem OH3-Konzept, seinen Erweiterungen und dem CMS Docufy COSIMA soll eine dynamische benutzerorientierte Online-Hilfe für die webbasierte Projektmanagement-Software Projektron BCS erstellt werden.

Die Umsetzung erfolgt innerhalb der Umgebung zwei spezifischer Systeme: Docufy COSIMA und Projektron BCS. Zugrundeliegende Konzepte und Methoden sowie Schnittstellen sollen aber auch so weit wie möglich allgemein gültig sein, so dass die Ergebnisse der Arbeit als Grundlage für weitere Implementierungen dienen können.

Teilziele der Arbeit sind somit:

- Entwicklung eines Metadaten-Konzepts zur benutzerorientierten Inhaltsbewertung
- Implementierung der Vergabe, Verwaltung und Ausgabe von Bewertungen in das CMS Docufy COSIMA
- Integrieren einer Anwenderunterstützung für die Vergabe der Bewertungen
- Entwicklung eines Frameworks zur benutzerorientierten Filterung und Darstellung von Inhalten
- Prototypische Anbindung an Projektron BCS und Spezifikation einer Schnittstelle zur Übertragung von Systemprofilen

1 Eine Bewertung wird in dieser Arbeit als eine Klassifikation einer Information nach Benutzereigenschaften verstanden.

Übergreifende Problemstellung der Arbeit ist die Beantwortung der Fragen:

- Mit welchem Aufwand ist die Implementierung des OH3-Konzepts verbunden?
- Wie gut lässt sich das OH3-Konzept in einen bestehenden Redaktionsprozess integrieren?
- Können die Erkenntnisse und Ergebnisse dieser speziellen Implementierung auch auf andere Systemkonstellationen übertragen werden?

1.5 ABGRENZUNG

Diese Arbeit konzentriert sich auf die Implementierung und Integration des Konzepts in den bestehenden zentralen Redaktionsprozess und die Bereitstellung einer Schnittstelle für den Endbenutzer.

Der Schwerpunkt liegt hierbei auf den technischen Aspekten des Konzepts. Methodische Grundlagen werden nur untersucht, falls dies für die Umsetzung erforderlich oder zielführend ist.

Nicht genauer betrachtet werden dabei angrenzende Aufgabengebiete und Disziplinen; diese werden in Kapitel 6.3 *Ausblick* aufgeführt. Bei Methodiken zur Ermittlung benutzerorientierter Faktoren wird auf vorhergegangenen Arbeiten aufgebaut.

Um den Umfang einer Bachelor-Thesis zu wahren, konzentriert sich die Arbeit auf die speziellen Anforderungen der Software-Dokumentation. Es lassen sich aber auch Schlüsse auf andere Branchen und Bereiche ziehen.

2

BENUTZERORIENTIERTER CONTENT

In diesem Kapitel werden grundlegende Begriffe und Methoden zu benutzerorientiertem Content vorgestellt. Gründe für dessen Einsatz werden genannt, bestehende Konzepte analysiert und eine Konzeption für allgemeine Anforderungen wird entworfen.

2.1 BEGRIFFE

2.1.1 *Informationsmodelle*

2.1.1.1 *Einführung*

Informationsmodelle werden auf der strukturellen Ebene der Standardisierung von Inhalten eingesetzt und geben vor, in welcher Struktur und in welchem Format Informationen erfasst werden (vgl. KRÜGER/ZIEGLER 2008:13). Diese Vorgaben können – je nach Modell – von der Makrostruktur bis auf Wortebene gelten.

generisch vs. semantisch

Die Modellierung der Informationen kann auf semantischen oder generischen Grundlagen erfolgen (vgl. DREWER/ZIEGLER 2011:381); auch Mischformen sind möglich. In der Technischen Dokumentation werden überwiegend Informationsmodelle eingesetzt, die stärker semantisch sind (vgl. DREWER/ZIEGLER 2011:381).

individuell vs. standardisiert

Je nach Einsatzgebiet kann ein eigenes Informationsmodell entworfen werden oder auf ein standardisiertes Informationsmodell zurückgegriffen werden. Bei eigenen Informationsmodellen handelt es sich oft um proprietäre Formate – manchmal auch, in Anlehnung an das Format der Quelle, „offenes XML“ (PELSTER 2011:56) genannt – die z.B. innerhalb eines Konzerns verwendet werden oder von CMS-Herstellern zur Verfügung gestellt werden (vgl. ZIEGLER/STEURER 2010:52). Standardisierte Informationsmodelle sind öffentlich zugänglich und werden von unabhängigen Organisationen gepflegt. Sie sollten in den meisten Fällen bevorzugt werden (vgl. KRÜGER/ZIEGLER 2008:15).

2.1.1.2 *Formate*

XML als Quellformat

Wird in einer Redaktion ein Content-Management-System (CMS) eingesetzt, so werden die Informationen darin sehr oft in der Extensible Markup Language (XML) erfasst (vgl. STRAUB/ZIEGLER 2008:204). Bei XML handelt es sich um eine Untermenge der Standard Generalized

Markup Language (SGML), einem ISO-Standard (ISO 8879) zur Auszeichnung von strukturierten Informationen aus dem Jahr 1986. XML wird voraussichtlich auch in den kommenden Jahrzehnten der Standard für die Erfassung von medienneutralen und systemunabhängigen Inhalten bleiben (vgl. KRÜGER 2012:108). Von verschiedenen Systemherstellern werden im Gegensatz dazu proprietäre Formate zur Speicherung von strukturierten Daten verwendet, z.B. in DTP-Software (vgl. DREWER/ZIEGLER 2011:297).

DTD vs. XSD als Regelvorgaben

Die Strukturregeln für XML-Dokumente werden in der Regel in einer Document Type Definition (DTD) oder einer XML Schema Definition (XSD) abgebildet. DTDs sind ein Relikt der SGML-Zeit, was sich daran erkennen lässt, dass sie nicht der XML-Syntax entsprechen (vgl. HAUSER 2010:34). Als Alternative dazu entwickelte das World Wide Web Consortium (W3C) die XSD, die auf XML basiert und im Vergleich zur DTD einen erweiterten Funktionsumfang besitzt. XSDs unterstützen z.B. die Unterstützung von Namensräumen und einfachen Datentypen (vgl. HAROLD/MEANS 2003:270f). Da XSDs auch bei kleinen Strukturen schnell komplex werden können und in einer DTD leichter abzubilden wären, bestehen beide Sprachen parallel.[2] Als eine alternative, bisher aber kaum verbreitete Schemasprache bietet sich *RelaxNG* an (vgl. HAUSER 2010:66).

2.1.1.3 *Individuelle Infomationsmodelle*

Viele CMS-Anbieter stellen ihren Kunden ein eigenes Informationsmodell zur Verfügung, mit dem das System in der Regel ausgeliefert wird (vgl. KRÜGER/ZIEGLER 2008:15 und STRAUB/ZIEGLER 2008:205). Diese Modelle sind für den entsprechenden Systemeinsatz optimiert und basieren auf Erfahrungen aus Kundenprojekten (vgl. DREWER/ZIEGLER 2011:383).

Gründe für individuelles Modell

Gründe, warum CMS-Anbieter ihre Systeme mit einem eigenen Informationsmodell ausliefern, können darin vermutet werden, dass standardisierte Modelle einer speziellen Branche entstammen und oft stark semantisch sind, was den möglichen Nutzerkreis einschränkt. Ein allgemeineres, eher generisches Modell ist unabhängiger von der Branche und lässt sich flexibler einsetzen.

Docufy-DTD

Auch der CMS-Hersteller Docufy liefert mit seinem System eine einsetzbare DTD aus, die für verschiedene Branchen geeignet sein soll. Genannt werden hier u.a. Maschinen- und Anlagenbau sowie Softwarehersteller und Medizintechnik-Unternehmen (vgl. KOTHES! GMBH 2008:3). Spezielle Anpassungen für Kunden werden nach Bedarf vom Hersteller vorgenommen.

2 Im weiteren Verlauf der Arbeit werden auf Grund von Einschränkungen des verwendeten CMS allerdings nur DTDs zum Einsatz kommen.

2.1.1.4 *Standardisierte Informationsmodelle*

Aus realen Anwendungen und Projekten heraus entstanden mit der Zeit standardisierte Informationsmodelle, die oft auf die Bedürfnisse einer bestimmten Branche abgestimmt sind.

Die Entwicklung und Pflege dieser Standards wird in den meisten Fällen von unabhängigen Gremien oder Organisationen übernommen (vgl. DREWER/ZIEGLER 2011:382).

Beispiele für standardisierte Informationsmodelle

Bekannte Vertreter mit der jeweils assoziierten Branche sind:

DITA
: *Software*. Bestehend aus Topics, die in Maps referenziert werden.

DOCBOOK
: *Software*. Generische Buchstruktur mit semantischen Elementen.

S1000D
: *Luftfahrt*. Stark komponentenorientiert mit strengen Strukturen.

PI-MOD
: *Maschinen- und Anlagenbau*. Freies semantisches Modell.

Eine Übersicht über die einzelnen standardisierten Informationsmodelle und die zugehörigen Gremien ist bei DREWER/ZIEGLER 2011 und KRÜGER/ZIEGLER 2008 zu finden.

Standardisierte Informationsmodelle haben gegenüber individuellen Lösungen u.a. den Vorteil, dass Daten austauschbar werden. So können beispielsweise Zulieferdokumentationen, die auf dem gleichen Modell beruhen, problemlos in die Gesamtdokumentation eingebunden werden (vgl. ZIEGLER/STEURER 2010:55).

2.1.2 *Metadaten*

2.1.2.1 *Einführung*

Beschreibende Metadaten (von griechisch μετα: zusammen mit, zu etwas hin) sind Informationen *über* Informationen.[3] Dazu gehört neben den physischen Daten in Systemen auch das Wissen der Mitarbeiter (vgl. ROCKLEY/KOSTUR/MANNING 2003:184, zitiert David Marco). In dieser Arbeit werden Metadaten, die außerhalb eines CMS auftreten oder gepflegt werden, nicht behandelt.

Metadaten von zentraler Bedeutung

In einem CMS sind Metadaten von zentraler Bedeutung, da durch sie das effiziente Auffinden und Wiederverwenden von Informationseinheiten wie Modulen erst möglich wird (vgl. DREWER/ZIEGLER 2011:361). Typische Meta-Attribute im Umfeld der Technische Dokumentation (TD) sind Sprache, Status und Gültigkeit einer Information

3 Im Gegensatz zu *Strukturierenden Metadaten*, die Informationen über die Informationsstruktur enthalten.

oder eines Objekts. Erfasser und Nutzer von Metadaten können sowohl Benutzer als auch das CMS selbst sein.

2.1.2.2 *Kategorisierung*

Durch ihre klassifizierende Eigenschaft bilden Metadaten die „Position" eines Objekt in einem „mehrdimensionalen Informationsraum" ab (vgl. DREWER/ZIEGLER 2011:327). Je genauer diese Position abgebildet werden kann, umso besser ist das Objekt auffindbar.

Kategorisieren lassen sich Metadaten nach DREWER/ZIEGLER 2011:364 in drei Hierarchie-Ebenen mit den Verzweigungen *Kategorie* (Klassifikation oder Lebenszyklus beschreibend), *Relation* (intrinsisch oder extrinsisch) und *Objekt* (produktbezogen [prod.] oder informationsbezogen [info.]), siehe Abb. 1:

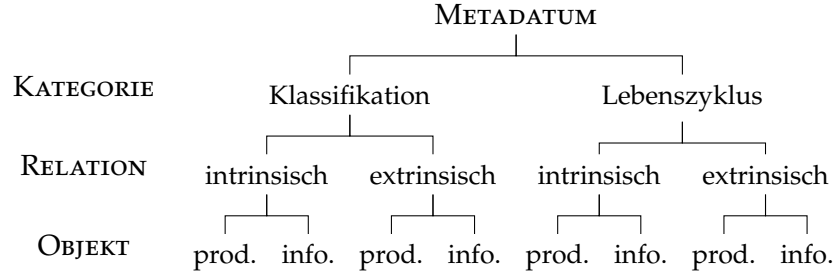


Abb. 1: Kategorisierung von Metadaten

Einordnungen von Metadaten

Können Metadaten nur bestimmte Werte annehmen, spricht man von *typisierten* Metadaten, bei beliebigen Werten von *untypisierten* (vgl. DREWER/ZIEGLER 2011:362). Metadaten können entweder *manuell* oder *automatisiert* vom System vergeben werden.

In einem CMS werden Metadaten in der Regel getrennt von den darin beschriebenen Daten verwaltet (z.B. in einer Datenbank). In diesem Fall spricht man von *externen* Metadaten. Will man Metadaten auch außerhalb des CMS weiterverarbeiten, können diese auch *intern* gespeichert werden, üblicherweise als Attribute an XML-Elementen (vgl. DREWER/ZIEGLER 2011:382).

2.1.2.3 *Vergabeprinzipien*

In einem Erstellungsprozess sollten Metadaten so früh wie möglich und wenn möglich automatisiert vergeben werden (vgl. ROCKLEY/KOSTUR/MANNING 2003:200). Müssen Metadaten manuell vergeben werden, sollte ein systematischer Vergabemechanismus vorgegeben sein (vgl. DREWER/ZIEGLER 2011:392). Die Vergabe sollte über Masken oder Formulare des CMS erfolgen (vgl. DREWER/ZIEGLER 2011:392) und wenn möglich vor dem (Wieder-)Einchecken der Information in das CMS (vgl. ROCKLEY/KOSTUR/MANNING 2003:200).

Hierbei muss beachtet werden, dass, wenn zwischen zwei oder mehreren Metadaten Abhängigkeiten bestehen, diese auch durch das System abgebildet bzw. kontrolliert werden (vgl. DREWER/ZIEGLER 2011: 392). Sind diese Abhängigkeiten hierarchischer Art, spricht man von einer *Taxonomie*.

2.1.3 *Benutzerorientierter Content*

In dieser Arbeit wird die Bezeichnung *benutzerorientierter Content* für eine Inhaltsart verwendet, die folgende Merkmale aufweist:

1. Die Art der Inhaltsaufbereitung ist sehr stark am späteren Benutzer orientiert und die zielgruppengerechte Übermittlung der Informationen wurde schon in der Entstehungsphase berücksichtigt. Dies schließt alle Inhalte mit ein, die einer Technischen Redaktion entstammen.
2. Die Art der Inhaltsübermittlung ist am Benutzer und dessen Verwendung des übertragenden Mediums oder Systems ausgerichtet. Im Vordergrund stehen hierbei Gebrauchstauglichkeit und Interaktivität mit dem Benutzer (vgl. DIN EN ISO 9241-210 2011:6). Dies schließt alle Inhalte ein, die durch ein interaktives Softwaresystem dargestellt werden.

Die zweite Bedeutung von *benutzerorientiert* entstammt der Norm DIN EN ISO 13407 mit dem Titel *Benutzer-orientierte Gestaltung interaktiver Systeme* aus dem Jahr 2000. Diese wurde 2011 durch die neuere DIN EN ISO 9241-210 *Ergonomie der Mensch-System-Interaktion – Teil 210: Prozess zur Gestaltung gebrauchstauglicher interaktiver Systeme* abgelöst. Auch die Bezeichnung *benutzerorientiert* wurde mittlerweile durch das umfassendere *menschzentriert* ersetzt. Für die Arbeit mit zielgruppenspezifischen Inhalten bleibt *benutzerorientiert* aber weiterhin die treffendere Benennung.[4]

Unter Content versteht man alle Inhalte textueller und visueller Art, die in einem CMS in Form von Informationsobjekten verwaltet werden (vgl. DREWER/ZIEGLER 2011:297).

Benutzerorientierter Content sind demnach alle CMS-Inhalte, die darauf vorbereitet sind, in einem interaktiven System benutzerspezifisch dargestellt zu werden. Dazu müssen Informationen über die Benutzer als Metadaten am Content hinterlegt sein.

4 „In der Praxis werden diese Begriffe jedoch häufig synonym verwendet." (DIN EN ISO 9241-210 2011:6)

2.2 GRÜNDE FÜR DEN EINSATZ

2.2.1 *Zunehmender Bedarf*

Die zunehmende „Informationsüberflutung“ (Belikan 2007), mit der Benutzer in der heutigen Zeit konfrontiert werden, erschwert oder verhindert das Auffinden der gesuchten Informationen. So stellen Muthig/Schäflein-Armbruster 2012:18 fest:

> „Was dem Nutzer nutzt, ist gut, was ihm nicht nutzt, überflüssig; und wenn es ihn gar behindert, ist es schlecht.“

Gefordert wird von ihnen eine bessere Dosierung der Informationsmenge – zugeschnitten auf den jeweiligen Benutzer. Ziel soll sein, den Benutzer schneller an die eigentlich gesuchte Information zu führen. Diese Feststellung formuliert auch Closs 2012b:50:

> „Das schnelle, unkomplizierte Auffinden der passenden und gewünschten Information ist bis heute eine unbewältigte Aufgabe, deren Lösung in immer weitere Ferne zu schwinden droht [...].“

Großer Bedarf an Benutzerorientierung

Diesen Bedarf nach einer genauen Informationsdosierung fordert auch Siegel 2013:150 in seiner Vorstellung einer perfekten Dokumentation:

> „[...] eine Zusammenfassung von Informationen [...], die dem Anwender, basierend auf seinem Wissensstand, eine konkrete und richtige Unterstützung bietet.“

Auch eine dynamische Anpassung an die Benutzer und deren Bedürfnisse scheint eine bisher ungelöste Aufgabe zu sein. So fordert Oehmig 2011:142f auf,

> „[...] den Benutzer einfach mal nach seinem Vorwissen zu fragen und variabel auf sein Verhalten und seinen Wissenszuwachs zu reagieren [...].“

Auffällig ist hierbei die Aktualität der Veröffentlichungen, die sich mit diesem Thema beschäftigen und die zunehmende Menge in den letzten Jahren.[5] Dies lässt darauf schließen, dass mit einer voranschreitenden Digitalisierung auch der Bedarf nach einer besseren Informationsdosierung und Benutzerorientierung steigt.

2.2.2 *Wirtschaftlicher Nutzen*

Neben den Vorteilen, die sich für den Benutzer durch den Einsatz von benutzerorientiertem Content ergeben, erhoffen sich auch die bereitstellenden Unternehmen wirtschaftliche Vorteile.

5 vgl. Belikan 2007, Closs 2009, Closs 2012a, Closs 2012b, Meier 2012, Muthig/Schäflein-Armbruster 2012, Oehmig 2011, Siegel 2013

Die zugrundeliegende These lautet in diesem Fall:

> Bessere Dokumentation führt zu weniger Support-Anfragen.

Support-Anfragen als Kostenfaktor

Als Support wird hierbei eine problemlösungsorientierte Beratungstätigkeit bezeichnet, die z.B. dann anfällt, wenn die Dokumentation unvollständig oder nicht korrekt war; aber auch dann, wenn der Benutzer die gewünschte Information nicht finden konnte. Support-Anfragen bedeuten für Unternehmen immer Personalaufwand und damit verbundene Kosten. Eine bessere Dokumentation, die z.B. durch einen benutzerorientierten Ansatz erreicht werden kann, führt also langfristig zu geringeren Kosten.

In diesem Sinne urteilen auch Murthy 2010:

> „Better documentation means less support calls, more cost-effective translations [...], more sales and so on."

oder Closs 2006:

> „Professionelle Produktdokumentation bietet viele Vorteile: Sie unterstützt die Nutzer [...] und entlastet den Kundenservice."

Darüber hinaus wird diese These auch in der Norm zur „Gestaltung gebrauchstauglicher interaktiver Systeme"(DIN EN ISO 9241-210 2011) angeführt:

> „Die Kosten für die unterstützende Betreuung und Beratung von Kunden werden verringert, wenn Benutzer die Produkte ohne zusätzliche Hilfe[6] verstehen und nutzen können."

Dass sich eine gute Dokumentation auch durch ihre Nutzbarkeit auszeichnet, wird bei Oehmig 2011:142 deutlich:

> „Je mehr Sie in die Qualität und Nutzbarkeit Ihrer Online-Hilfe investieren, desto stärker entlasten Sie Ihre Hotline und die Serviceleute."

Weitere wirtschaftliche Vorteile

Neben dem verringerten Support-Aufwand ergeben sich aber auch noch weitere wirtschaftliche Vorteile aus einer benutzerorientierten Dokumentation: Möglichkeiten zur Textkürzung (vgl. Ebenhoch 2012: 45), aber auch eine erhöhte Attraktivität des Informationsprodukts (vgl. Schäflein-Armbruster 2008 und Closs 2012a) können sich positiv auf den Kostenfaktor auswirken. Auch auf die Kundenbindung und die Außenwirkung des Unternehmens kann sich eine gute Usability der Dokumentation auswirken (vgl. Krömker/Norbey 2012:111).

6 Die Dokumentation ist laut Maschinenrichtlinie Bestandteil des Produkts (vgl. EU-Maschinenrichtlinie 2006).

2.3 BESTEHENDE KONZEPTE

2.3.1 *In Informationsmodellen*

Einige Informationsmodelle haben bereits in Grundzügen einen benutzerorientierten Ansatz. Dieser kann sowohl *statisch* ausgeprägt sein, so dass das Modell an sich schon auf eine spezielle Benutzergruppe zugeschnitten ist oder auch *dynamisch*, so dass das Modell die Möglichkeit bietet, Modulen oder Fragmenten[7] bestimmte Benutzergruppen zuzuweisen.

Als Beispiel für eine statische Benutzerorientierung kann das standardisierte Informationsmodell S1000D dienen, das sehr stark an der späteren Anwendergruppe ausgerichtet ist, der Luft- und Rüstungsbranche (vgl. DREWER/ZIEGLER 2011:387).

Dynamische Benutzerorientierung

Im standardisierten Informationsmodell Darwin Information Typing Architecture (DITA) (siehe 2.1.1.4) ist es möglich, Topics mit Hilfe des Elements `audience` eine Zielgruppe zuzuweisen (vgl. CLOSS 2011:227). Dieser Zielgruppe kann wiederum eine Aufgabe (`job`) und der Stand der Vorkenntnisse (`experiencelevel`) zugewiesen werden. Bei einer anschließenden Publikation wird, z.B. durch das DITA Open Toolkit, die entsprechende Variante zielgruppenspezifisch gefiltert (vgl. KRÜGER/ZIEGLER 2008:28).

In herstellerspezifischen Informationsmodellen, wie dem von Docufy! COSIMA, ist oftmals keine Benutzerorientierung vorgesehen (vgl. SCHULZE 2013). Innerhalb des Gesamtkonzepts ist hier eine Filterung auf CMS-Ebene vorgesehen (vgl. DOCUFY GMBH 2012a:178).

2.3.2 *In Content-Management-Systemen*

Eine Benutzerorientierung kann auch außerhalb des Informationsmodells eingesetzt werden: im Variantenmanagement des CMS (vgl. DREWER/ZIEGLER 2011:343). Hierbei werden die nötigen Metadaten an das entsprechende Informationsobjekt[8] des Systems geschrieben und bei der Publikation wieder ausgelesen und geprüft.

Gültigkeiten für Varianten

Bisher beschränkte sich dieser Mechanismus aber eher auf ja/nein-Gültigkeiten, die z.B. eine produkt-, medien- oder länderspezifische Variantenfilterung ermöglichten (vgl. DOCUFY GMBH 2012a:178). So können Publikationen – je nach Art des eingesetzten Variantenmanagements – relativ genau auf eine Zielgruppe abgestimmt werden.

7 Nach DREWER/ZIEGLER 2011:339 ist ein Fragment „[...] eine eigenständige, wiederverwendbare Unterstruktur eines Moduls".

8 Informationsobjekte sind neben dem eigentlichen Content auch weitere Daten, die im CMS verwaltet werden, z.B. Übersetzungsaufträge oder Stylesheets, *inklusive* ihrer Metadaten (vgl. hierzu DREWER/ZIEGLER 2011:298 und DOCUFY GMBH 2012c:44).

Da die Filterung aber während des Publikationsprozesses greift, sind die entstehenden Dokumente statisch und im Nachhinein nicht mehr dynamisch zu verändern.

2.3.3 *In Content-Enhancement-Systemen*

Das Berliner Software-Unternehmen paux GmbH stellt mit *paux* ein sog. Content-Enhancement-System her, das es erlaubt, Inhalte semantisch und sozial anzureichern und auch bis auf Wortebene miteinander zu verknüpfen. Die Idee wurde auf der tekom Frühjahrestagung 2011 (vgl. DREUSICKE 2011) vorgestellt und später in einem Artikel vertieft (vgl. DREUSICKE/LIESKE/SASAKI 2012)[9].

Filterung je nach Nutzergruppe

Mit Hilfe des *PAUX Editors* ist es auch möglich, Content bzw. *Microcontent [sic]* mit Informationen zur Nutzergruppe anzureichern, welche später in einer Webdarstellung durch ein Drop-Down-Menü gefiltert werden kann. Ähnlich dem Konzept *Online-Hilfe 3.0* kann Text damit zielgruppenspezifisch reduziert oder erweitert werden (vgl. PAUX TECHNOLOGIES GMBH 2013b).

Innerhalb des Content-Enhancement-Systems werden (Mikro-)Inhalten Metadaten in bis zu 7 Kategorien zugeteilt: Nutzergruppe, Vorwissen, Schwierigkeit, Relevanz, Qualität, Anmerkung und Beschreibung (vgl. DREUSICKE 2013). Die entscheidenden Faktoren zur Filterung stellen das Vorwissen, der Ort der Inhaltskonsumierung und der Informationskanal dar (vgl. DREUSICKE 2013).

Einsatz im E-Learning-Bereich

Als Anwendungsgebiete nennt der Hersteller u.a. die (interaktive) Lehrbucherstellung, Agentur- und Verlagsarbeit sowie den Einsatz in Bildungseinrichtungen. Online-Hilfe-Anwendungen werden nicht erwähnt (vgl. PAUX TECHNOLOGIES GMBH 2013a).

2.3.4 *In Ausgabeanwendungen*

Multimedia-Anwendungen

Zwar gibt es gerade im Bereich der Multimedia-Lernanwendungen viele interaktive Systeme zur Darstellung von instruktiven Inhalten, doch diese lassen sich in der Regel nicht ohne Weiteres aus den vorhandenen CMS-Quelldaten heraus publizieren und sind in der Produktion teuer (vgl. OEHMIG 2011:142). Darüber hinaus fehlt auch sehr interaktiven Medienformen, wie Online-Tutorials, noch die Möglichkeit, benutzerorientiert und entsprechend des Vorwissens vorzugehen (vgl. CLOSS 2006).

Nischenlösungen sind zum Beispiel in der Luftfahrt im Einsatz. Dort wird aus CMS, die mit dem Informationsmodell S1000D arbeiten, die

9 Dieser Artikel erschien zeitgleich mit der Vorstellung des sehr ähnlichen Konzepts *Online-Hilfe 3.0* (vgl. TEKOM 2012). Es handelt sich um unabhängige Entwicklungen.

sog. Interaktive Elektronische Technische Dokumentation (IETD) erzeugt (vgl. DREWER/ZIEGLER 2011:387). Die Systeme zur Darstellung von IETD sind allerdings nur in dieser Branche anzutreffen und deshalb für den Einsatz in einem breiteren Umfeld (bisher) ungeeignet.

Online-Hilfen zählen zu den wenigen Ausgabemedien in der TD, die sowohl eine Interaktivität mit dem Nutzer erlauben als auch aus nahezu allen CMS publiziert werden können (vgl. STRAUB/ZIEGLER 2008:215ff). Ihre Verbreitung ist dementsprechend hoch, die Akzeptanz bei Anwendern allerdings verschwindend gering. So wurde in Usability-Studien festgestellt, „[...] dass die wenigsten Kunden überhaupt wissen bzw. wahrnehmen, dass es eine Online-Hilfe gibt, und kaum einer sie jemals nutzt" (CLOSS 2006). Zum gleichen Schluss kommt auch OEHMIG 2011:142: „Der Ruf von Online-Hilfen ist noch schlechter als der von Papierdokumentationen [...]".

Interaktion in Online-Hilfen

In den heute eingesetzten Online-Hilfen beschränken sich die Interaktionsmöglichkeiten mit dem Nutzer bisher auf Navigation, Retrieval-Funktionen[10] und (Hyper-)Links (vgl. KNOPP 2000:117-147). Innovative Veränderungen des Konzepts gab es in den letzten 20 Jahren nicht (vgl. CLOSS 2006 und CLOSS 2012a:38); die Technik, die von bisherigen Hilfeplattformen eingesetzt wird, ist größtenteils veraltet (vgl. CLOSS 2012a:39).

Keine Filterung nach Zielgruppe

Die Idee, eine zielgruppenspezifische Filterung von Inhalten in Online-Hilfen zu integrieren, gab es zu Beginn der digitalen Hilfe[11], bisher findet sie allerdings kaum Anwendung, was auch dem befürchteten redaktionellen Aufwand geschuldet ist (vgl. KNOPP 2000:113-116). Ausgereifte Konzepte für dynamische und interaktive Modelle zur Benutzerorientierung sowie Erfahrungen in Gestaltung und Handhabung entsprechender Systeme fehlen bisher fast vollständig (vgl. CLOSS 2009).

2.3.5 *Im Usability-Engineering*

Die Forschungen im Bereich des Usability-Engineerings haben zwar interessante Modelle ergeben, die Benutzereigenschaften zusammenfassen (z.B. den *User Cube* bei NIELSEN 1994:44), liefern aber keine Vorgaben für die dafür notwendige Vorbereitung des Contents.

10 Mit Retrieval-Funktionen sind hier analog zu KNOPP 2000 zusätzliche Zugangsmöglichkeiten wie Inhaltsverzeichnis, Glossar und Suche gemeint.

11 KNOPP verweist auf Quellen, die bis ins Jahr 1988 zurückreichen.

2.4 KONZEPT ONLINE-HILFE 3.0

2.4.1 *Entstehung*

Gewinner des intro-Preises

Das Konzept OH3 wurde als Wettbewerbsbeitrag für den *Studentischen Innovationspreis der tekom - intro* von Eva-Maria Meier und Jan Oevermann ausgearbeitet. Nach der Bewertung durch eine Fachjury und einer anschließenden Online-Abstimmung konnte es sich gegen die Konkurrenz durchsetzen und wurde mit dem ersten Platz ausgezeichnet (vgl. TEKOM 2012:11).

2.4.2 *Überblick*

Übergreifende Idee des Konzeptes ist eine dynamische Darstellung von benutzerorientiertem Content. Diese Darstellung wird beim Aufruf auf Basis vorhandener Informationen voreingestellt und kann im Nachhinein noch vom Nutzer nachjustiert werden. Grundlage für die zielgruppenspezifische Bewertung des Contents ist ein submodulares Variantenmanagement auf Basis extrinsischer Metadaten (vgl. auch MUTHIG 2012:13 und SCHÄFER/OEVERMANN/MEIER 2012:45ff.).

Web-Technologien als Basis

Die Darstellung und Filterung basiert auf Web-Technologien. Das Nutzerverhalten wird analysiert und Profil- sowie Contentbewertungen werden dynamisch verbessert (vgl. BERTRAM 2012:15).

Das Konzept wurde am Beispiel einer webbasierten Software erarbeitet, kann aber prinzipiell auch für andere Branchen und Anwendungen verwendet werden. Seit der Entstehung wurde das Konzept weiterentwickelt, siehe dazu Abschnitt 2.4.3.

2.4.2.1 *Content-Bewertung*

Bewertung in Dimensionen

Inhalte können bei der OH3 von Modul- bis auf Fragmentebene bewertet werden. Eine Bewertung erfolgt in drei Dimensionen mit jeweils fünf Abstufungen (vgl. OEVERMANN/MEIER 2012a:12).

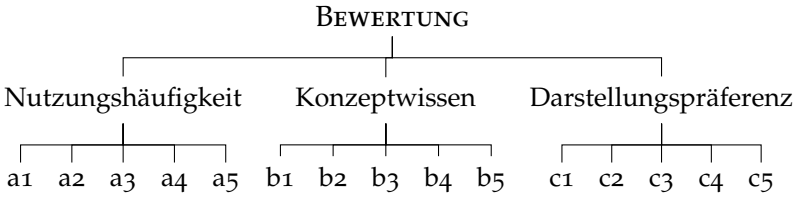


Abb. 2: Aufbau der OH3-Bewertung

Die Bewertung, die in der Regel vom Inhaltsersteller vorgenommen wird, gibt mögliche Ziel- bzw. Benutzergruppen für den Inhalt an.

Die Extremwerte der Dimensionen stellen kontrastive Benutzergruppen dar (z.B. Anfänger vs. Experten), Werte dazwischen entsprechende Übergänge (z.B. Fortgeschrittene).

Die Kurznotation für einen bestimmten Wert in einer Dimension entspricht dem Buchstaben der Dimension und dem zugeordneten Wert (z.B. b4).

Bewertungen sind bei der Vergabe voneinander unabhängig, werden aber bei der Filterung unterschiedlich gewichtet. Ein Inhalt kann in einer oder mehreren Dimensionen bewertet werden. Wird keine Bewertung vergeben, ist der Inhalt für alle Benutzergruppen gültig.

2.4.2.2 *Methodische Grundlage*

Die Inhaltsbewertungen entsprechen klassifizierenden extrinsisch-informationsbezogenen Metadaten, da sie die Zugehörigkeit zu einer Benutzergruppe klassifizieren. Die Inhaltsbewertungen werden intern gespeichert (als XML-Attribute) und sind typisiert, da sie nur Werte innerhalb des Werterahmens annehmen können (siehe auch Abschnitt 2.1.2.2).

Submodulares Variantenmanagement

Durch die Möglichkeit, auch Fragmente mit Bewertungen auszuzeichnen, handelt es sich bei der Filterung um ein submodulares Variantenmanagement, das auf einer Maximalvariante je Modul beruht (nach Drewer/Ziegler 2011:348: „Variantensammlung“).

2.4.2.3 *Darstellung und Filterung*

Der Content wird webbasiert in einem Browser dargestellt. Zur Filterung wird eine Kontrollleiste eingeblendet, deren zentrale Steuerelemente drei Schieberegler sind (vgl. Oevermann/Meier 2012a:8).

Schieberegler zur Nutzereingabe

Der Benutzer kann über diese Schieberegler den Inhalt entsprechend seines Profils dynamisch anpassen. Die Ausgangsstellung der Schieberegler ist auf die Informationen voreingestellt, die bereits über den Nutzer vorliegen. Jeder der drei Schieberegler ist einer Dimension zugeordnet.

Technische Grundlagen für das Konzept sind HTML5[12] und JavaScript in Verbindung mit offenen Bibliotheken (z.B. jQuery). Als Austauschformat ist XML vorgesehen.

Progressive Filterung

Die Filterung verfolgt einen progressiven Ansatz, das bedeutet, dass z.B. ein Inhalt, der die Bewertung *Fortgeschrittene*(`a3`) trägt, automatisch auch für die Benutzergruppe *Anfänger*(`a1`) sichtbar ist. Dies spiegelt sich am verwendeten Filtermechanismus wider:

12 Zur Verwendung der Bezeichnung HTML5 vgl. Schober 2012:36f.

```
for (var i = 1; i < a; i++){
  var cl = ".a" + i;
  $(cl).fadeOut(1000);
  }
for (var i = a; i <= 5; i++){
  var cl = ".a" + i;
  $(cl).fadeIn(500);
  }
```

Code 1: Filterung im OH3-Prototyp (JavaScript)

Die Filterung erfolgt über `for`-Schleifen. `a` entspricht der Reglerstellung *Nutzungshäufigkeit* zwischen 1 und 5. Alle Elemente, die eine Bewertung geringer als `a` haben, werden ausgeblendet (`fadeOut()`). Elemente mit einer Bewertung, die größer oder gleich `a` ist, werden oder bleiben eingeblendet (`fadeIn()`).

2.4.2.4 *Soziale Netze*

Im OH3-Konzept ist auch die Einbindung von Nutzerprofilen aus sozialen Netzwerken oder ähnlichen Plattformen angedacht. Dabei wird dem Nutzer die Möglichkeit gegeben, sich mit dem Profil eines Netzwerkes anzumelden, aus dem dann nutzerspezifische Daten ausgelesen werden. Diese Lösung ist angedacht für Online-Hilfen auf Websites oder in webbassierten Anwendungen. Benutzereigenschaften wie Vorwissen oder Darstellungspräferenz könnten so z.B. aus öffentlichen Informationen eines Facebook-Profils ausgelesen werden (vgl. Dambeck 2013). Bei Unternehmenssoftware könnten solche Informationen beispielsweise aus einem sozialen Firmen-Intranet stammen.

2.4.2.5 *Dynamische Lernfunktion*

Bewertungen passen sich an

Eine im Konzept vorgestellte Idee, die aber technisch nicht weiter ausgeführt wird, ist die automatische Anpassung von Inhaltsbewertungen und Benutzerprofilen auf Basis von Verhaltensbeobachtung (vgl. Oevermann/Meier 2012a:15). Dabei wird die aktuelle Position im Text beobachtet[13] und kontrolliert, ob der Nutzer seine Profileinstellungen über die Schieberegler nachkorrigiert. Anschließend wird dieses Verhaltensmuster mit dem anderer Nutzer abgeglichen. Korrigieren an dieser Stelle viele Benutzer ihre Einstellung, so wird die Bewertung der Information angepasst; handelt es sich eher um einen Einzelfall, wird die Profileinstellung des Nutzers angepasst.

13 Scrollposition innerhalb einer Seite.

2.4.2.6 *Umsetzung*

Prototypische Online-Demo

Als Teil des Wettbewerbsbeitrags wurde bereits eine prototypische Umsetzung (vgl. OEVERMANN/MEIER 2012b) entwickelt, die anhand eines festen Beispiel-Moduls die Funktionsweise der Filterung zeigt (vgl. TEKOM 2013). Die Meta-Informationen zu den Bewertungen sind als Klassen-Attribute an den entsprechenden Content-Elementen gespeichert. Wird einer der Schieberegler bewegt, wird der Inhalt der aktuellen Stellung angepasst, so dass nur Inhaltselemente angezeigt werden, die für den Nutzer relevant sind. Die Funktionalitäten zur Einbindung sozialer Netzwerke und die selbstlernende Bewertungsanpassung wurden im Prototyp nicht umgesetzt.

2.4.3 *Erweiterung*

2.4.3.1 *Methodische Grundlagen*

Im Rahmen ihrer Bachelor-Thesis (vgl. MEIER 2012) beschäftigte sich Eva-Maria Meier mit den theoretischen Grundlagen des OH3-Konzepts („Nutzerzentrierte Kommunikation in Online-Hilfen“) und der Ermittlung des Informationsbedarfs spezifischer Nutzergruppen.

Untersucht wurden unter anderem die möglichen Dimensionen, in denen Hilfetexte bewertet werden können, und eine geeignete Abstufung für die Bewertung von Inhalten. Hierbei wird speziell auf die Besonderheiten und Bedürfnisse der Softwarebranche eingegangen. Auf eine mögliche technische Umsetzung geht die Arbeit bewusst nicht weiter ein.

MEIER kommt abschließend zu dem Ergebnis:

> „Die Analyse des intro-Konzepts[14] hat gezeigt, dass die wichtigsten Einflussfaktoren bereits berücksichtigt werden. Allerdings wurde auch deutlich, dass eine Weiterentwicklung notwendig ist, um dem Anspruch der Nutzerzentrierung gerecht zu werden.“ (MEIER 2012:65)

Neue Dimension Informationsanliegen

Als Weiterentwicklung wird die Einführung eines neuen Bewertungskriteriums empfohlen, um auch auf den Kontext des Hilfe-Aufrufs reagieren zu können: Das Informationsanliegen des Nutzers. Als mögliche Werte werden *Erklären lassen*, *Anleiten lassen* und *Überblick verschaffen* genannt.

Die Bewertungsdimensionen *Produktwissen*[15] und *Konzeptwissen*, die auch schon im OH3-Konzept zu finden waren, wurden weitgehend bestätigt. MEIER schlägt hier jedoch eine Reduzierung der Bewertungsabstufungen von fünf auf drei Schritte vor. Auf die Fragestellung, ob

14 Wird bei MEIER synonym zu Online-Hilfe 3.0 verwendet.
15 Entspricht *Nutzungshäufikgeit* aus OEVERMANN/MEIER 2012a

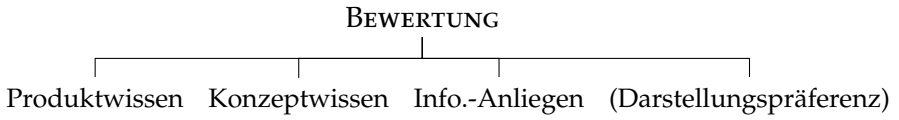


Abb. 3: Benutzerorientierung nach MEIER

die Dimension *Darstellungspräferenz* für eine vollständige Benutzerorientierung notwendig ist, konnte keine eindeutige Antwort gefunden werden.

2.4.3.2 *Entwicklung eines Frameworks*

Projektarbeit zu Framework

Im Rahmen der Veranstaltung *Multimedia Programmierung* im Bachelor-Studiengang Technische Redaktion der Hochschule Karlsruhe entstand die Projektarbeit „Webbasiertes Framework zur Anzeige und Filterung von XML-Inhalten“ unter Betreuung von Prof. Martin Schober (vgl. OEVERMANN 2012).

Die Arbeit beschäftigt sich mit Konzeption und Umsetzung eines Frameworks, das benutzerorientierte Inhalte nach dem OH3-Konzept darstellen und filtern kann. In Aufbau und Funktionsweise orientiert sich das Framework am OH3-Prototyp, kann aber im Gegensatz zu diesem beliebige XML-Inhalte einlesen, so lange ein entsprechendes XSL-Stylesheet zur Umwandlung in Hypertext Markup Language (HTML) hinterlegt ist. Die Transformation erfolgt dabei clientseitig im Moment des Aufrufs.

Weiterhin ist es möglich, Benutzerprofile zu hinterlegen und DITA-ähnliche Topicmaps einzubinden, in denen verwandte Module zusammengefasst werden können und so für den Benutzer auswählbar werden (vgl. OEVERMANN 2012:4).

Die technische Umsetzung erfolgte mit HTML5 als Markup- und CoffeeScript als Scriptingsprache. Für Quell- und Konfigurationsdaten kommt XML zum Einsatz. Für die Transformation der Quelldaten in das Ausgabeformat wird die Extensible Stylesheet Language (XSL) verwendet.

2.4.3.3 *Anforderungsanalyse*

Das ursprüngliche OH3-Konzept wurde auch an reale Anforderungen angepasst. Die Ergebnisse dieser Anforderungsanalyse basieren auf Gesprächen mit Firmen und Personen.

Bei den Befragten konnte ein breites Spektrum an Sichten auf die Problematik abgedeckt werden: Führungskräfte eines großen TD-Dienstleisters (GOLDMANN/NITSCHE 2013), Technische Redakteure (KLEMT/KUTTER/MÄSER 2013) und Experten im Bereich von benutzerorientiertem Content (DREUSICKE 2013).

Anpassung an Anforderungen

So ist die Anbindung an soziale Netzwerke weitgehend nicht von Firmen erwünscht (vgl. GOLDMANN/NITSCHE 2013) und ist nicht mehr fester Bestandteil des Konzepts. Auch die automatische Bewertungsermittlung, die in Teil 2.4.2.5 beschrieben wird, ist technisch nur mit sehr viel Aufwand zu realisieren und datenschutzrechtlich bedenklich.

Größte Anforderung an das Konzept war eine möglichst einfache und schnelle Bewertung der Inhalte (vgl. KLEMT/KUTTER/MÄSER 2013), die auch für den Redakteur keine erweiterten Technikkenntnisse, etwa im XML-Bereich, erfordert (vgl. GOLDMANN/NITSCHE 2013). Auch der Wunsch nach einer automatisierten Bewertungsvergabe wird geäußert (vgl. GOLDMANN/NITSCHE 2013).

2.4.4 *Möglichkeiten und Grenzen*

Als „lebendes" Konzept hat sich die Online-Hilfe 3.0 seit ihrer Entstehung zwar weiterentwickelt (vgl. Teil 2.4.3 und 2.4.3.3), sie existiert bisher jedoch nur prototypisch und als theoretisches Konstrukt. Ziel muss es nun sein, auf Basis der schon gewonnenen Erkenntnisse einen Rahmen zu schaffen, um das Konzept auch praktisch anwenden zu können.

Keine vollständige Klassifikation

Grenzen weist das Konzept bei der Vollständigkeit des Klassifikations- und Bewertungsmodells auf. Hier werden nur benutzereigene Faktoren konkret behandelt. Fremdbestimmte Faktoren wie Rollentyp oder Systemkonfiguration werden zwar aufgegriffen, jedoch nicht weiter behandelt (vgl. OEVERMANN/MEIER 2012a:10):

> „Alleine über die Information, in welcher Rolle der Nutzer die Software verwendet und welche Rechte er hat, können verschiedene Inhalte ausgeschlossen werden. Über Bereiche, die der Nutzer aufgrund seiner Rolle nicht benutzen kann, braucht er auch keine Informationen."

An dieser Stelle muss ein allgemeines Modell für Klassifikationen bzw. Bewertungen gefunden werden, das auch die oben genannten Faktoren neben den Benutzerbewertungen einbezieht.

2.5 KONZEPTION FÜR ALLGEMEINE ANFORDERUNGEN

2.5.1 *Motivation*

Das OH3-Konzept und die erfolgten Weiterentwicklungen sind für eine vollständige Benutzerorientierung einer Dokumentation eine sehr gute Basis (vgl. MEIER 2012:65). Allerdings entscheiden nicht nur benutzereigene Faktoren darüber, ob eine Information für einen Anwender, der die Hilfe konsultiert, relevant ist. Auch das spezifische Pro-

dukt, das er einsetzt, oder die Rolle und Kompetenzen, die ihm zugeteilt sind, können Einfluss auf den Relevanzgrad einer Information haben.

Externe Faktoren miteinbeziehen

Für eine umfassende Benutzerorientierung müssen diese externen Faktoren berücksichtigt werden. Erst dann kann eine effektive Filterung erfolgen, die die Relevanz der angezeigten Informationen für den Nutzer sicherstellt.

Ein Beispiel aus der Anwender-Dokumentation *Navigation* der Projektmanagement-Software Projektron BCS lautet (vgl. PROJEKTRON GMBH 2012a:23):

> „[...] Klicken Sie dazu mit der rechten Maustaste auf den Link der gewünschte Ansicht und wählen Sie im Kontextmenü den Eintrag *Link in neuem Tab öffnen* (kann je nach Webbrowser variieren).
>
> *Tipp*: Falls Sie Projektron BCS unter Mac OS X benutzen, verwenden Sie zum Öffnen einer Ansicht in einem neuen Tab Ihres Webbrowsers die Tastenkombination `cmd + shift + rechte Maustaste`."

Produkt beeinflusst Relevanz

Der dargestellte Textteil enthält Informationsteile, die sich je nach eingesetztem Browser unterscheiden können („[...] kann je nach Webbrowser variieren") und für Anwender des Betriebssystems Windows nicht relevant sind („Falls Sie Projektron BCS unter Mac OS X benutzen [...]"). Bestimmt wird die Informationsrelevanz also nicht durch eine Benutzer-, sondern durch eine Produkteigenschaft.

Eine dynamische Informationsfilterung, die auf Produkteigenschaften basiert, wird immer häufiger eingesetzt, um große Informationsmengen (z.B. im Automobilbereich) zu verwalten (vgl. RATH 2005), eine Filterung unter Einfluss von Benutzereigenschaften erfolgt hierbei aber nicht (vgl. GOLDMANN/NITSCHE 2013).

Daraus entstehen Anforderungen an ein kombiniertes Modell.

2.5.2 *Modell*

Zentrale Anforderung für eine effektive Filterung ist ein umfassendes Metadatenmodell, das sowohl Produkt- als auch Benutzereigenschaften aufgreift und die Relevanzfaktoren einer Information abbildet.

Achsen im Informationsraum

Diese Relevanzfaktoren sind Klassifikationen, in die eine Information – möglichst eindeutig – eingeordnet werden kann. Die Klassifikatoren eines solchen Modells dienen als Achsen innerhalb eines mehrdimensionalen Informationsraums, anhand deren die Position einer Information innerhalb des Informationsraums bestimmt werden kann (vgl. DREWER/ZIEGLER 2011:369).

Nach dem OH3-Konzept müssen einer Information nicht alle Bewertungen zugeordnet werden (vgl. OEVERMANN/MEIER 2012a:12). In

den meisten Fällen ist es sogar so, dass nur eine der Bewertungen *Produktwissen* oder *Konzeptwissen* zugeordnet wird (vgl. KLEMT/KUTTER/MÄSER 2013). Werden nun in einem n-dimensionalen Raum z.B. $n-2$ Bewertungen vergeben, spannt sich eine Ebene auf, das heißt die Position der Information im Raum ist nicht mehr eindeutig bestimmt; sie überspannt einen bestimmten Bereich.

Position des Benutzers im Informationsraum

Im Gegensatz dazu sind im Benutzerprofil immer alle drei Werte belegt.[16] Das heißt, die Position des Nutzers im Informationsraum ist zu einem festgelegten Zeitpunkt eindeutig. Überlagert sich nun diese Position mit dem Wertebereich einer Information, so ist diese für den Benutzer relevant.

Um Informationen eindeutig zu klassifizieren, müssen diese Relevanzfaktoren trennscharf und – so weit wie möglich – unabhängig voneinander sein, so dass jeder Relevanzfaktor eine Dimension (Achse) des Informationsraums repräsentiert.

2.5.3 *Relevanzfaktoren*

Nach Analyse der vorhandenen Software-Dokumentation der Projektron GmbH und Gesprächen mit Redakteuren (vgl. KLEMT/KUTTER/MÄSER 2013) sowie Dokumentationsdienstleistern (vgl. GOLDMANN/NITSCHE 2013) wurde eine Sammlung von wichtigen Faktoren zusammengestellt, die darüber entscheiden, ob eine Information für einen Nutzer relevant ist.

Allgemeine Relevanzfaktoren

Allgemein und branchenübergreifend betrachtet lassen sich diese Faktoren in vier Kategorien einordnen, von denen jeweils zwei fremdbestimmt und zwei durch den Benutzer selbst beeinflusst werden können (siehe Abb. 4).

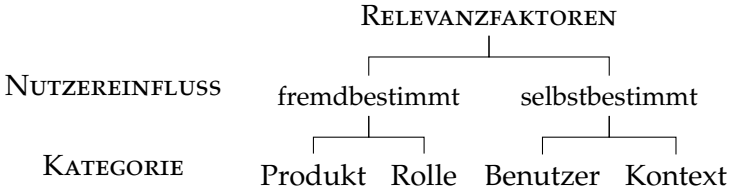


Abb. 4: Allgemeine Relevanzfaktoren

2.5.3.1 *Kategorien*

Die vier Kategorien umfassen folgende Bereiche:

16 Verändert der Nutzer seine Einstellung nicht durch die Schieberegler, werden die Standardwerte verwendet (Mittelstellung).

PRODUKT
: Alle Eigenschaften, die bei einem Produkt zur Variantenbildung führen können und eine direkte Auswirkung[17] auf die Interaktion mit dem Benutzer haben.
Beispiel: Ein Auto wird je nach Ausführung *mit* oder *ohne* Schiebedach ausgeliefert.

ROLLE
: Die Rolle, Funktion oder Lizenz, die einem Nutzer von einer höheren Institution (z.B. dem Arbeitgeber) zugewiesen wird und ihm den Rahmen für sein Handeln vorgibt.
Beispiel: In einer Software hat die Rolle *Benutzer* eingeschränkte Rechte gegenüber der Rolle *Administrator*.

BENUTZER
: Alle Eigenschaften, die nach dem OH3-Konzept Auswirkungen auf Informationsverständnis und -relevanz haben.
Beispiel: Jemand, der eine Maschine das erste Mal bedient, benötigt zunächst eine Übersicht über die Bedienelemente. Ein langjähriger Anwender benötigt diese mit hoher Wahrscheinlichkeit nicht.

KONTEXT
: Alle situationsabhängigen Faktoren, die Einfluss darauf haben, ob für den Benutzer eine Information in einem bestimmten Kontext relevant ist.
Beispiel: Ein Anwender benutzt sein neues Mobiltelefon das erste Mal. In diesem Kontext ist wahrscheinlicher, dass der Nutzer das Informationsanliegen *Überblick verschaffen* hat als das Informationsanliegen *Erklären lassen*.

Diese vier Kategorien sind nicht vollständig unabhängig voneinander. So hängt die Verfügbarkeit verschiedener Rollen im Softwarebereich oft von der Produktkonfiguration ab. Auch zwischen Kontext und Benutzereigenschaften kann es zu Beeinflussungen kommen (vgl. KLEMT/KUTTER/MÄSER 2013). Siehe hierzu auch die Ausführungen in Abschnitt 2.5.4.

2.5.3.2 *Faktoren*

Relevanzfaktoren für die Softwaredokumentation

Aufbauend auf dieser Kategorisierung lassen sich die einzelnen Relevanzfaktoren einordnen, die für die Softwaredokumentation gelten und realistisch gepflegt werden können. Daraus entsteht als Hypothese ein vollständiges benutzerorientiertes Metadatenmodell (siehe Abb. 5 auf Seite 24).

17 Beispiel: Bezieht ein Handyhersteller baugleiche Displays von verschiedenen Herstellern führt dies zur Variantenbildung, allerdings ohne direkte Auswirkung auf die Benutzerinteraktion.

Bei den Relevanzfaktoren handelt es sich um Klassifikationen, die einer Information zugeteilt werden können. Da es sich um ein benutzerorientiertes Konzept handelt, und somit immer der Bezug zum (externen) Benutzer hergestellt werden soll, können alle Faktoren als extrinsisch interpretiert werden (siehe Tabelle 1 auf Seite 27).

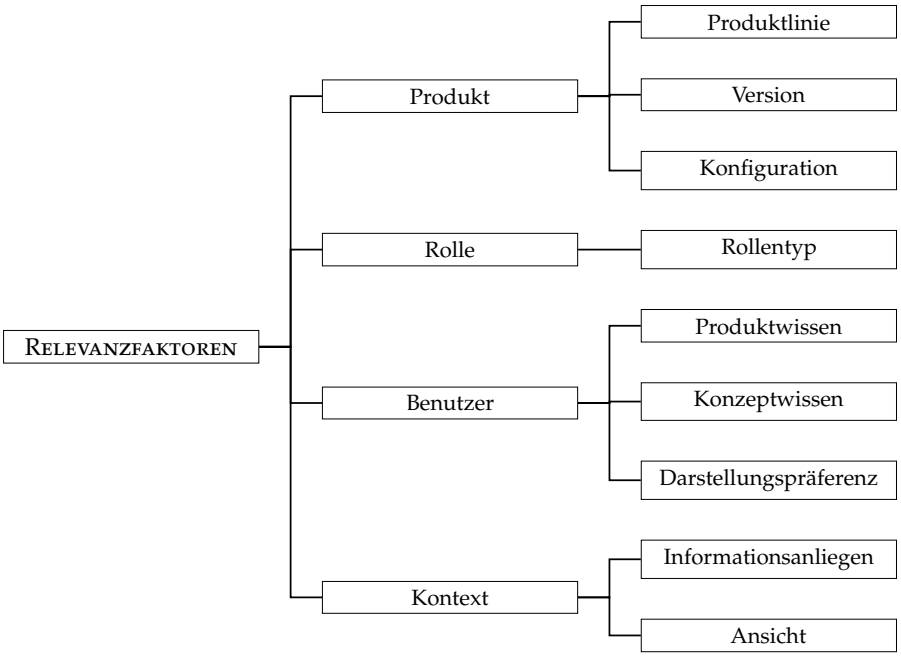


Abb. 5: Relevanzfaktoren für Softwaredokumentation

Im Folgenden werden die einzelnen Relevanzfaktoren kurz beschrieben, eingegrenzt sowie Beispiele und – falls eingrenzbar – mögliche Werte genannt:

PRODUKTLINIE

Als *Produktlinie* oder auch als *Variante* bzw. *Edition* werden im Softwarebereich in der Regel zielgruppenspezifische Ausprägungen einer Software bezeichnet. Varianten einer Software basieren oft auf dem gleichen Programmkern und werden meistens parallel vertrieben und entwickelt. Hauptunterschiede bestehen in Funktionsumfang und Oberfläche. Dies sind auch die Berührungspunkte mit dem Endanwender, die sich je nach Produktlinie unterschieden können. Bekanntes Beispiel sind die beiden Produktlinien *Home Edition* und *Professional Edition* des Betriebssystems *Windows XP* des Herstellers Microsoft.

VERSION

Eine *Version* ist ein bestimmtes Entwicklungsstadium einer Software. Oft wird in der Zählung zwischen *Major- und Minorversionen* unterschieden, wobei erstere größere Änderungen wie Funktionsneuerungen mit sich bringen, während letztere oft nur Fehlerbehebungen und kleinere Anpassungen unterschei-

den. Bei der verbreitetsten Zählweise werden die unterschiedlichen Versionsprünge durch einen Punkt getrennt. Beispielsweise erscheint Projektron BCS zur Zeit in Version 7.6. Gerade beim Sprung auf eine neue Major-Version können für den Benutzer Änderungen in der Funktionsweise oder der Oberfläche relevant werden.

KONFIGURATION

Als Konfiguration wird die Gesamtheit der Grundeinstellungen, Module und Parameter bezeichnet, mit der eine Software betrieben wird. Programme unterscheiden sich teilweise sehr stark in Umfang und Möglichkeiten, an individuelle Anforderungen angepasst zu werden. Diese können vom Zu- oder Abschalten von ganzen Modulen oder Bausteinen bis hin zum Konfigurieren von einzelnen Betriebsparametern reichen. Entsprechend unterschiedlich können auch die direkten Auswirkungen auf den Benutzer sein.

ROLLENTYP

Analog zu 2.5.3.1: Die Rolle, Funktion oder Lizenz, die einem Nutzer von eine höheren Institution (z.B. dem Arbeitgeber) zugewiesen wird und ihm den Rahmen für sein Handeln vorgibt.

PRODUKTWISSEN

Das Vorwissen, das ein Benutzer zur Software besitzen muss, so dass eine Information relevant für ihn wird. Dieses korreliert stark mit Nutzungsdauer und -häufigkeit (vgl. NIELSEN 1994:44f). Es umfasst Bereiche wie die Anordnung der Oberflächenelemente, die Navigation innerhalb des Programms oder auch das Wissen, dass bestimmte Funktionen oder Ansichten existieren (vgl. MEIER 2012:69). Mögliche Werte entsprechen dem OH3-Konzept: Anfänger, Fortgeschrittener und Experte.

KONZEPTWISSEN

Das Vorwissen, das ein Benutzer zum Konzeptthema, das die Software behandelt, haben muss, um die Information zu benötigen (vgl. OEVERMANN/MEIER 2012a:9). Gerade Einsteiger in Konzeptthemen benötigen oft Zusatzinformationen. Bei Microsoft Word könnte das zu *Layout* oder *Schriftsatz* sein, bei Projektron BCS zu *Projektmanagement*. Bei NIELSEN 1994:44 wird dies als „Knowledgeable about domain" bezeichnet. Mögliche Werte entsprechen dem OH3-Konzept: Anfänger, Fortgeschrittener und Experte.

DARSTELLUNGSPRÄFERENZ

Benutzer unterscheiden sich in der favorisierten Art der Informationsaufnahme voneinander (vgl. MANGOLD 2007:312). Unterschieden werden die Informationsarten nach ihrer Kodalität. Typische Beispiele sind Texte, Bilder, Videos, Sprachaufnahmen oder interaktive Darstellungen. Hierbei lassen sich Informatio-

nen am besten zwischen verbalisiert und stark visuell unterscheiden und einteilen. Schlussfolgernd aus den Ergebnissen des OH3-Konzepts und Klassifizierungen von Dokumentationsdienstleistern (vgl. ARVATO SERVICES TI GMBH 2013) lassen sich drei mögliche Werte festlegen: *eher verbal*, *neutral* und *eher visuell*.

INFORMATIONSANLIEGEN

Das Anliegen, mit dem der Benutzer die Hilfe konsultiert, ist ein entscheidender Faktor, ob eine Information relevant für ihn ist (vgl. MEIER 2012:65). Informationen müssen demnach entsprechend ihrer kommunikativen Funktion klassifiziert werden. In vielen standardisierten Informationsmodellen geschieht das bereits (siehe z.B. DITA: *task* vs. *concept*). Welche Informationsart der Nutzer gerade benötigt (z.B. *anleitend* vs. *erklärend* vs. *Übersicht schaffend*) entscheidet der Kontext, in dem die Hilfe aufgerufen wird.[18] Um eine Abfrage an den Benutzer zu realisieren, die den Nutzer nicht auf dem Weg zur richtigen Information aufhält, erscheint hier die Reduzierung auf zwei mögliche Werte (im Gegensatz zu drei wie bei MEIER) sinnvoll, die durch einen Zweiwege-Schalter mit sinnvollem Standardwert abgefragt werden (vgl. GOLDMANN/NITSCHE 2013 und KLEMT/KUTTER/MÄSER 2013). Die möglichen Werte, die sich daraus ergeben sind *Anleitung* vs. *Anleitung + Erklärung*.

ANSICHT

Ein kontextabhängiger Faktor für die Informationsrelevanz ist auch die Ansicht, in der sich ein Benutzer gerade innerhalb der Software befindet. Man kann davon ausgehen, dass ein Anwender, der die Dokumentation vom Begrüßungsbildschirm aus aufruft, eine andere Information erwartet als jemand, der innerhalb einer tiefen Menühierarchie die Online-Hilfe konsultiert. Diese kontextsensitive Softwareanbindung gilt schon lange als eine der großen Vorteile von Online-Hilfen gegenüber der klassischen Printdokumentation (vgl. CLOSS 2012a:39).

2.5.3.3 *Relation und Objektbezug*

Die Relevanzfaktoren können, wie in Abschnitt 2.1.2.2 erläutert, nach Relation und Objektbezug kategorisiert werden.

Extrinsisches Metadatenmodell

Da es sich beim OH3-Konzept um ein benutzerorientiertes Modell handelt, ergeben sich alle Relevanzfaktoren durch die Beziehung, die sie zum Benutzer haben. Damit wird ein rein extrinsisches Modell aufgebaut, auch wenn manche Faktoren weiterhin intrinsisch interpretiert werden können (siehe auch Abschnitt 2.5.3.2).

18 Es bestehen auch Abhängigkeiten zu Benutzereigenschaften, siehe Kapitel 2.5.4

RELEVANZFAKTOR	RELATION	OBJEKT
Produktlinie	extrinsisch	Produkt
Version	extrinsisch	Produkt
Konfiguration	extrinsisch	Produkt
Rollentyp	extrinsisch	Information
Produktwissen	extrinsisch	Information
Konzeptwissen	extrinsisch	Information
Darstellungspräferenz	extrinsisch	Information
alternativ	intrisisch	Information
Informationsanliegen	extrinsisch	Information
alternativ	intrisisch	Information
Ansicht	extrinsisch	Produkt

Tabelle 1: Kategorisierung von Relevanzfaktoren nach Abb. 1

Beispiel: Die Information, dass eine Information *anleitend* statt *beschreibend* ist, wird zum extrinsischen Metadatum Informationsanliegen, da der Bezug zum Benutzer lautet „Welchem Informationsanliegen des Nutzers bin ich zugeordnet?“. Es kann aber auch als intrinsisches Metadatum Informationsart interpretiert werden, also „Welche Informationsart bin ich?“.

2.5.3.4 *Zuweisung und Auswertung*

Kategorisierung nach Zuweisung und Auswertung

Die Methodiken, wie die Relevanzfaktoren im Erstellungsprozess zugewiesen bzw. beim Ausgabeprozess ausgewertet werden sollen, unterscheiden sich. Dadurch ergeben sich Möglichkeiten zur Automatisierung oder Systemunterstützung.

Die Relevanzfaktoren müssen bei der Erstellung der entsprechenden Information zugewiesen werden. Bei der Art der Zuweisung kann zwischen *explizit* und *implizit* unterschieden werden:

EXPLIZIT
: Der Wert muss vom Redakteur vergeben werden. Hierbei kann er vom System durch vorbelegte Masken oder eine integrierte Vergabelogik[19] unterstützt werden. Die Auswahl des Wertes muss er aber selbst vornehmen.

IMPLIZIT
: Der Wert kann aus bestehenden Auszeichnungen ermittelt werden. Dazu gehören z.B. semantische Elemente in standardisierten Informationsmodellen. Auch auf Grundlage des Informati-

19 Die Logik kann z.B. durch eine Taxonomie gegeben sein.

onstyps (Text, Bild, Video, etc.) kann eine Zuweisung zu einem Faktor erfolgen. Damit ist z.B. eine (teil-)automatisierte Bewertung von Beständen möglich.

Bei Ausgabe und Filterung der Informationen kann unterschieden werden, ob das *System* die Auswertung der Relevanzfaktoren übernimmt oder der *Benutzer* diese selbst beeinflussen kann:

SYSTEM
: Das System, das die Hilfe aufruft, belegt Relevanzfaktoren selbstständig und ohne Einfluss durch den Benutzer mit Werten. Ein Beispiel wäre die Übergabe der Software-Version.

BENUTZER
: Der Benutzer kann die Werte von Relevanzfaktoren selbst beeinflussen. Dies kann nach der Kategorisierung aus Abb. 4 nur bei selbstbestimmten Faktoren der Fall sein. Nach dem OH3-Konzept werden Werte durch Schieberegler in der Oberfläche der Online-Hilfe eingestellt (vgl. OEVERMANN/MEIER 2012a:8). Grundsätzlich sind aber auch andere Eingabemöglichkeiten und Schnittstellen denkbar.

Die Einordnung der Relevanzfaktoren in diese Kategorien ist in Tabelle 2 auf Seite 28 dargestellt.

RELEVANZFAKTOR	ZUWEISUNG	AUSWERTUNG
Produktlinie	explizit	System
Version	explizit	System
Konfiguration	explizit	System
Rollentyp	explizit	System
Produktwissen	explizit	Benutzer
Konzeptwissen	explizit	Benutzer
Darstellungspräferenz	implizit	Benutzer
Informationsanliegen	implizit	Benutzer
Ansicht	explizit	System

Tabelle 2: Zuweisung und Auswertung von Relevanzfaktoren

2.5.4 *Abhängigkeiten*

Nach der Einteilung in Produkt- und Informationsbezug der Relevanzfaktoren (siehe Tabelle 1) können diese nun eingeteilt werden in systemische Abhängigkeiten (z.B. „die Konfiguration ist von der Produktlinie abhängig“) und Abhängigkeiten, die die Informations-

übermittlung an den Nutzer betreffen (z.B. „Anfänger mögen lieber Bilder als Text“).

Nur Benutzerfaktoren betrachtet

Da die Abhängigkeiten, die in der Software (also dem System) bestehen, sich von Hersteller zu Hersteller stark unterscheiden können, geht die Analyse auf diese produktbezogenen Faktoren nicht weiter ein. Diese müssen je nach Software betrachtet werden.

Genauer betrachtet werden die Faktoren: Rollentyp, Produktwissen, Konzeptwissen, Darstellungspräferenz [Darst.-Präf.] und Informationsanliegen [Info-Anliegen]. Abhängigkeiten sind entweder belegt (durchgezogene Linie) oder vermutet (gestrichelte Linie). Siehe Abb. 6 auf Seite 29.

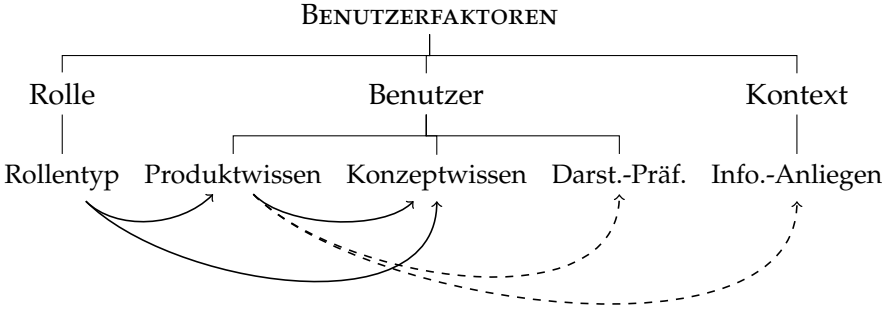


Abb. 6: Abhängigkeiten bei Benutzerfaktoren

Wie stark sich diese Faktoren gegenseitig beeinflussen, unterscheidet sich je nach beschriebenem System und Heterogenität der Zielgruppen. Je komplexer die Software und je heterogener die Zielgruppen, desto stärker sind die Abhängigkeiten ausgeprägt (vgl. Goldmann/ Nitsche 2013). Bei einer Umsetzung müssen diese Abhängigkeiten beachtet und Verknüpfungen entsprechend implementiert werden.

ROLLENTYP -> PRODUKTWISSEN

Je nach Rolle, die einem Benutzer zugeteilt wird, kommt er unterschiedlich oft mit der Software in Kontakt. So haben Benutzer mit der Rolle *Administrator* in der Regel ein sehr ausgeprägtes Produktwissen. Auch bei einer granulareren Rollenaufteilung (z.B. *Projektleiter* vs. *Controller*) können diese Abhängigkeiten beobachtet werden. Beispiel: „Controller verwenden die Software laut den Trainer-Aussagen eher seltener als Projektleiter“ (Meier 2012:50).

ROLLENTYP -> KONZEPTWISSEN

Gerade bei einer Unternehmenssoftware werden viele Rollen entsprechend der fachlichen Kompetenz vergeben. Dies ist darauf zurückzuführen, dass in der Regel mit umfassenderen Rollen auch erweiterte Rechte innerhalb der Software zur Verfügung stehen. Diese Rechte sollten wiederum nur Mitarbeiter

mit entpsrechender Kompetenz ausführen dürfen. Daraus ergibt sich eine Abhängigkeit zum Konzeptwissen eines Benutzers. Beispiel: Ein Projektleiter benötigt mehr Konzeptwissen zum Thema Projektmanagement als ein Projektmitarbeiter (vgl. MEIER 2012:48f).

PRODUKTWISSEN -> KONZEPTWISSEN

Mit der häufigen Nutzung einer Software und somit steigendem Produktwissen verändert sich auch das Wissen über das übergeordnete Konzept bzw. wie es in der Software umgesetzt ist (Konzeptwissen). Daraus wird deutlich, „[...] dass Produktwissen und Konzeptwissen [...] untrennbar miteinander verbunden sind“ (MEIER 2012:61).

PRODUKTWISSEN -> DARSTELLUNGSPRÄFERENZ

Nach Erfahrungen von TD-Dienstleistern und Redakteuren kann in vielen Fällen davon ausgegangen werden, dass Einsteiger visuelle Informationen bevorzugen (z.B. in Form eines Videotutorials), während Experten öfter textuelle Informationen (z.B. in Form einer Troubleshooting-Tabelle) benötigen (vgl. ARVATO SERVICES TI GMBH 2013).

PRODUKTWISSEN -> INFORMATIONSANLIEGEN

Auch der Kenntnisstand, den der Benutzer beim Aufrufen der Hilfe hat, kann entscheidend dafür sein, welche Informationen für ihn relevant sind. Experten benötigen oft anleitende Inhalte in möglichst kompakter Form, z.B. als „Guided Fault finding [sic]“ oder „Guided Function“ (vgl. ARVATO SERVICES TI GMBH 2013). Im Gegensatz dazu haben Einsteiger oft das Informationsbedürfnis nach Überblick und Erklärung. Dieses Bedürfnis kann z.B. durch Tutorials oder ein Computer-Based-Training (CBT) befriedigt werden (vgl. GOLDMANN/NITSCHE 2013 und ARVATO SERVICES TI GMBH 2013).

3

CONTENT-MANAGEMENT-SYSTEME ALS BASIS

In diesem Kapitel werden Content-Management-Systeme im Allgemeinen und COSIMA *go! im Speziellen charakterisiert. Darauf aufbauend werden die Ergebnisse aus Kapitel 2 umgesetzt und implementiert.*

3.1 SYSTEMCHARAKTERISTIK

3.1.1 *Einführung*

Content-Management-Systeme sind für das „Erfassen, Verwalten und Publizieren von unternehmensinternen oder -externen technischen Informationen [...]“ (DREWER/ZIEGLER 2011:291) zuständig. Sie dienen als zentrale Datenquelle im Redaktionsprozess.

Content und Informationsobjekte

Als Content werden Informationen textueller, visueller und in manchen Fällen auch multimedialer Art bezeichnet, die in einem CMS als Informationsobjekte verwaltet werden (siehe auch Abschnitt 2.1.3). Ein Informationsobjekt (IO) ist die Kombination aus Content und zugehörigen Metadaten (siehe Abschnitt 2.1.2).

CMS werden in der Regel von mehreren Redakteuren gleichzeitig benutzt (*Mehrbenutzersystem*).

3.1.2 *Methoden*

In CMS werden fast immer die Methoden SSP und CMP umgesetzt:

SINGLE-SOURCE-PUBLISHING
: Unter Single-Source-Publishing (SSP) versteht man die kontrollierte Wiederverwendung („Re-Use“) von Informationseinheiten, den sog. Modulen (vgl. DREWER/ZIEGLER 2011:297).

CROSS-MEDIA-PUBLISHING
: Mit Cross-Media-Publishing (CMP) wird die auf medienneutraler Datenhaltung basierende Publikation der gleichen Informationen in verschiedenen Ausgabemedien bezeichnet.

Auch das Content-Variant-Management (CVM) ist immer öfter zentraler Bestandteil eines CMS (vgl. ZIEGLER 2005). Das CVM ermöglicht durch systemeigene Mechanismen die Variantenbildung bei der Publikation von Inhalten aus einem CMS. Oft handelt es sich dabei um

individualisierte Anpassungen an die Produktkonfiguration oder Lokalisierungsvarianten (vgl. DREWER/ZIEGLER 2011:296).

3.1.3 *Datenhaltung und Schnittstellen*

Im Umfeld der TD setzen die meisten CMS als Speicherformat für ihre Inhalte XML ein (vgl. STRAUB/ZIEGLER 2008:203/224). Die XML-Daten werden dann in einer relationalen Datenbank verwaltet und durch Ein- und Auscheckmechanismen den Redakteuren zur Bearbeitung bereitgestellt (vgl. STRAUB/ZIEGLER 2008:224).

Schnittstellen zu anderen Systemen

Neben Schnittstellen zu weiteren Datenquellen (z.B. dem Enterprise-Ressource-Planning (ERP)) oder Sprachkontrollsystemen (z.B. einem Controlled-Language-Checker (CLC)) bieten CMS eine große Anzahl an angebundenen oder integrierten Programmen zur Publikation des Contents (vgl. STRAUB 2011:215ff).

Die Publikation erfolgt in der Regel über eine XSL-Transformation, die die XML-Quelldaten in ein verwandtes (z.B. HTML) oder vorgeschaltetes Format (z.B. XSL-FO) transformiert (vgl. DREWER/ZIEGLER 2011:434). In manchen CMS werden proprietäre Publikations-Engines zur Umwandlung eingesetzt (z.B. Ovidius TopLeaf).

3.1.4 *Customizing*

Anpassungen fast immer notwendig

Zur optimalen Integration eines CMS in den Redaktionsprozess eines Unternehmens sind fast immer Anpassungen am System notwendig. Diese können von kleineren Anpassungen (z.B. des Freigabeprozesses) bis zu weitreichenden Veränderungen (z.B. Anpassung des Informationsmodells) reichen.

In dieser Arbeit wird die Bezeichnung *Customizing* für alle Änderungen an einem System verwendet, die der Anpassung an individuelle Kundenanforderungen dienen – unabhängig davon, ob sie durch Einstellung, Konfiguration oder Programmierung vorgenommen werden.

Konfiguration vs. Programmierung

Die meisten CMS sind konfigurierbar und geben dem Kunden entsprechend weitreichende Rechte, Anpassungen selbst vorzunehmen (vgl. DREWER/ZIEGLER 2011:390). Tiefgreifende Systemänderungen, die z.B. die Geschäftslogik[20] betreffen, können vom Hersteller hingegen nur durch Programmierung umgesetzt werden und werden als Dienstleistung angeboten (vgl. STRAUB/ZIEGLER 2008:227ff).

Zunehmend bieten Hersteller CMS-Produktlinien an, die auf mittelständische Unternehmen angepasst und vorkonfiguriert sind, in ih-

20 Bei Softwaresystemen, die auf einer Client-Server-Architektur beruhen, bezeichnet *Geschäftslogik* oder auch *Business Logic* die von der technischen Implementierung getrennte Schicht, die die Problemstellung des Systems behandelt.

rem Kern aber auf komplexeren CMS-Plattformen basieren (vgl. DREWER/ZIEGLER 2011:390).

3.2 DOCUFY COSIMA

3.2.1 *Einführung*

COSIMA go! (CGO) ist ein CMS des Herstellers Docufy[21] und seit 2004 am Markt vertreten (vgl. STRAUB/ZIEGLER 2008:163). Seitdem hat es sich zu einem der führenden Redaktionssysteme[22] im Bereich der TD entwickelt (vgl. KOTHES! GMBH 2010:1). Das System ist konzipiert für „[...] die multilinguale Erstellung modularisierter Dokumentation und vollautomatische Produktion von Dokumentvarianten für beliebige Zielmengen [...]“ (vgl. STRAUB/ZIEGLER 2008:174).

COSIMA als Basis-Plattform

Entstanden ist COSIMA go! aus der CMS-Plattform COSIMA, die für große Redaktionen als stark individualisierbares System konzipiert wurde (vgl. KOTHES! GMBH 2010:4). Da die Anfangsinvestitionen in DTD-Entwicklung oder Stylesheet-Programmierung für mittelständische Unternehmen nicht tragbar waren, wurde in Zusammenarbeit mit dem TD-Dienstleister Kothes[23] eine Vorkonfiguration des Systems entwickelt, die auf die Bedürfnisse von mittelständischen Unternehmen angepasst ist (vgl. KOTHES! GMBH 2008:1).

3.2.2 *Architektur*

Client-Server-Modell als Basis

COSIMA go! basiert auf einem Client-Server-Modell, bei dem der Systemkern die Verwaltung des Contents in einer relationalen Datenbank und die Auslieferung der Inhalte an Clients oder in eine Publikation steuert (siehe Abb. 7 auf Seite 34). Eine Extension API bindet externe Systeme an, die interne API stellt die Verbindung zu den Clients her (vgl. KOTHES! GMBH 2008:3). Docufy bietet neben einem RichClient zur Installation auf Windows-Systemen auch einen browserbasierten WebClient an, dessen Funktionsumfang allerdings stark eingeschränkt ist und dessen Fokus im Einsatz im Review-Prozess liegt (vgl. KOTHES! GMBH 2008:3).

Technisch ist das System auf Basis des J2EE-Standards[24] implementiert (vgl. KOTHES! GMBH 2008:4). Für die Datenhaltung können verschiedene relationale Datenbanksysteme eingesetzt werden (z.B. MS

21 Docufy GmbH, Bamberg (www.docufy.de)

22 Im Bereich der TD werden die Bezeichnungen Redaktionssystem und Content-Management-System oft synonym verwendet.

23 Kothes! Technische Kommunikation GmbH & Co. KG, Kempen (www.kothes.de)

24 *Java Plattform, Enterprise Edition*, eine Softwarearchitektur für die Ausführung von in Java programmierten Anwendungen.

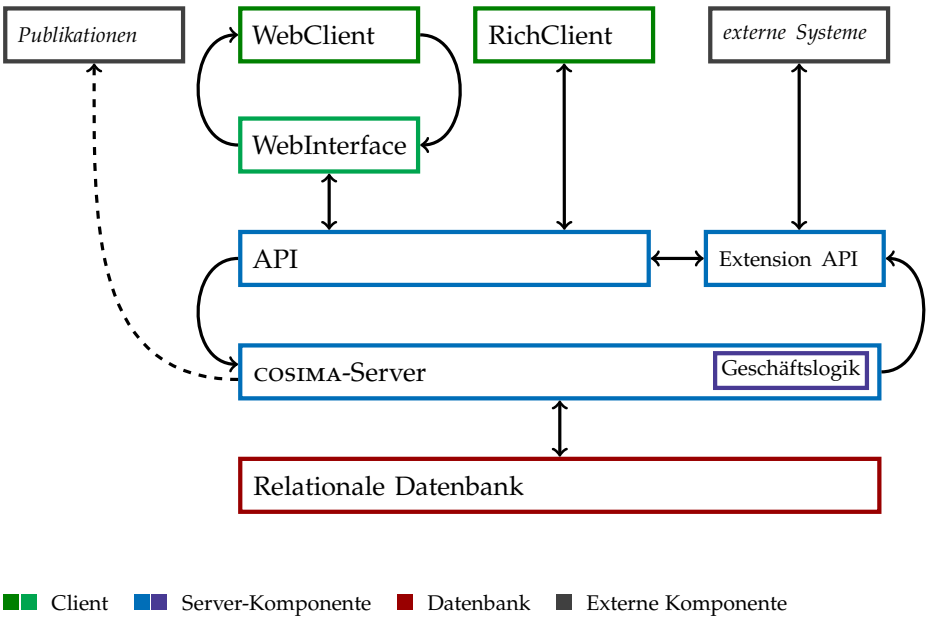


Abb. 7: COSIMA go! Systemarchitektur
(abgeleitet aus KOTHES! GMBH 2008 / KOTHES! GMBH 2010)

SQL, Oracle, IBM DB2). Die Server-Komponente ist weitgehend plattformunabhängig, der RichClient basiert auf der Eclipse-Plattform und ist für Windows verfügbar (vgl. DOCUFY GMBH 2012a:14).

3.2.3 *Variantenmanagement*

Cosima-CVM Gültigkeiten

Das in CGO integrierte CVM wird vom Hersteller über sog. *Gültigkeiten* abgebildet (vgl. DOCUFY GMBH 2012a:67f). Dabei können Variantenstrukturen mit Hilfe von Taxonomien im System hinterlegt werden. Aus methodischer Sicht handelt es sich um eine Variantensammlung bzw. die Erstellung von Maximalvarianten (vgl. DREWER/ZIEGLER 2011:348). Das System verwaltet die Logik und Verwendung der Gültigkeiten. Gespeichert werden die Gültigkeiten als interne Metadaten in Form von XML-Attributen an den entsprechenden Elementen.

Der Redakteur kann im System Publikationsprofile anlegen, die die Varianten über zugewiesene Gültigkeiten charakterisieren. Während des Publikationsvorgangs kann eines der hinterlegten Publikationsprofile zugeordnet werden und darauf basierend eine Variante erstellt werden (vgl. DOCUFY GMBH 2012b:194). Dabei werden die Attribute von den Elementen entfernt.

3.2.4 *Informationsmodell*

Das Informationsmodell, das CGO im Standard verwendet, wird mit dem System ausgeliefert und ist für die Branchen Maschinen- und Anlagenbau, Medizin, Software und Fabrikation passend (vgl. KOTHES! GMBH 2008:3). Abgebildet ist das Modell als DTD, die sich aus einer Stammdatei und mehreren referenzierten Moduldateien zusammensetzt (aggregierte DTD).

Generische Makrostruktur

Das CGO-Informationsmodell setzt sich aus einer generischen Makrostruktur und semantischen Elementen unterhalb der Absatzebene zusammen (vgl. SCHULZE 2013). Dieser Ansatz folgt der Vorgehensweise vieler Hersteller, die übergeordneten Strukturelemente allgemein zu halten (siehe auch Abschnitt 2.1.1.3) und auf Absatz- und Wortebene eine Reihe von semantischen Elementen anzubieten, die in branchenspezifischen Anwendungsfällen zum Einsatz kommen können, z.B. Schaltflächenbezeichnungen in der Softwaredokumentation oder Werkzeuglisten im Anlagenbau.

In Abb. 8 werden alle Elemente erster Ebene entsprechend ihrer Funktion eingeordnet. Bei diesem Aufbau handelt es sich um die klassische Buchstruktur, die auch bei anderen Informationsmodellen (z.B. DocBook) verwendet wird.

Inhaltsmodule werden zum größten Teil in `chapter`-Elementen erfasst. Ein `chapter` setzt sich wiederum aus einem `title` und `block`-Elementen sowie Metadaten (`topicmeta`), Modulreferenzen (`block-ref`, `chapter-ref`) und automatisch generierten Listen (z.B. `tec-data-comp-list`) zusammen. `block`-Elemente gliedern innerhalb eines `chapters` textuelle Inhalte und Bilder.

Semantische Elemente auf Block-Ebene

Innerhalb von block-Elementen kommen vermehrt semantische Elemente zum Einsatz. Beispiele sind: `procedure` (Handlungsanweisung), `example` (Beispiel) oder `environment` (Entsorgungshinweis).

In Abb. 9 sind augewählte Elemente in ihrer Hierarchie dargestellt, semantische Elemente wurden ***`kursiv`*** gekennzeichnet.

3.2.4.1 *Aufbau der DTD*

Die Gesamt-DTD setzt sich aus mehreren Dateien zusammen, die aus der *manual.dtd* heraus referenziert werden (siehe Abb. 10 auf Seite 38). Die DTD-Fragmente werden mit Parameterentitäten eingebunden, die auf Formal Public Identifier (FPI) und lokale Datei der entsprechenden DTD verweisen. Mit der Datei *cgo-customization.dtd* steht eine spezielle Datei für Benutzeranpassungen an der CGO-Standardkonfiguration zur Verfügung (vgl. SCHULZE 2013). Für das Customizing der COSIMA-Plattform wird die Datei *dcms-customization.dtd* verwendet. Die anderen referenzierten Dateien entsprechen Auslagerungen von Standardelementen, -strukturen und -attributen. Innerhalb der einzelnen

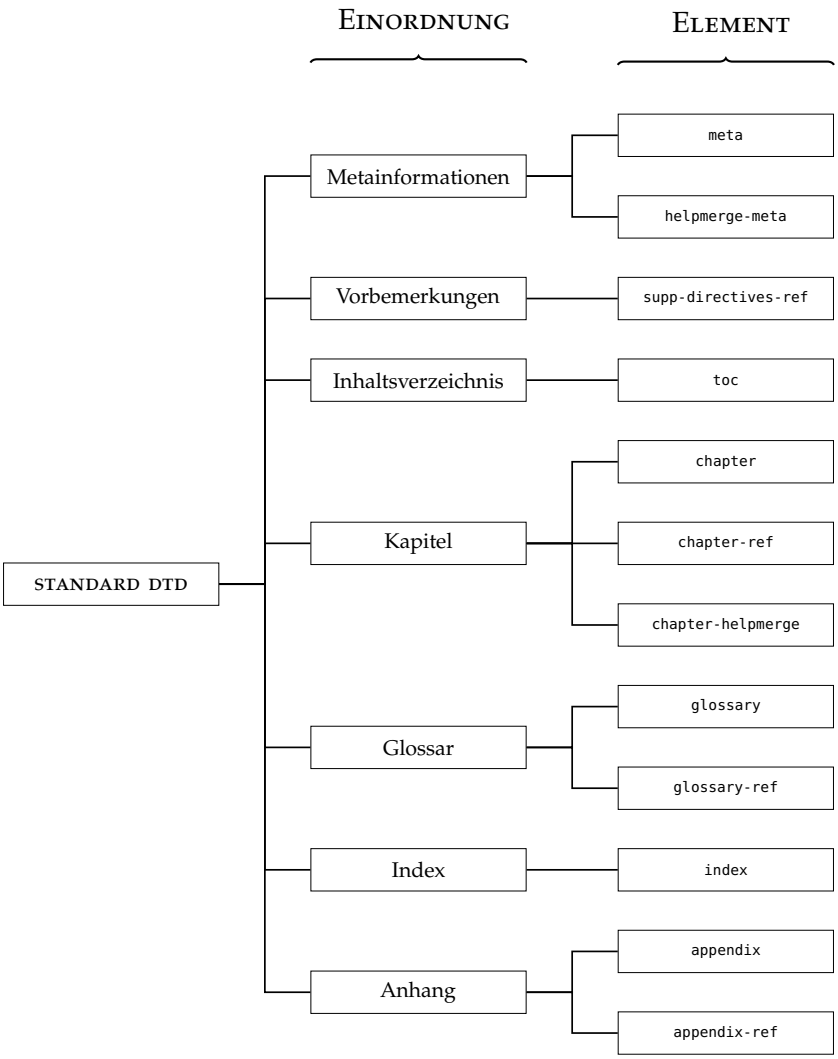


Abb. 8: Schematische Makrostruktur der Docufy Standard DTD

Dateien stehen die zu verarbeitenden Strukturanweisungen (in Abbildung: *PCDATA* für *parsed character data*).

3.2.5 *Customizing-Eigenschaften*

Durch Architektur und Konzeption des Systems können COSIMA go! gute Customizing-Eigenschaften zugesprochen werden. Interessante Anpassungsmöglichkeiten bestehen in den folgenden Punkten:

INFORMATIONSMODELL
: Durch die Modularisierung der DTD in Standardbestandteile und Benutzeranpassungen ergeben sich sehr gute Customizing-Eigenschaften (siehe Abschnitt 3.2.4.1). Es kann auf bestehende Element- und Attributmengen über Entitäten zugegriffen werden und die DTD ist in den meisten Teilen kommentiert.

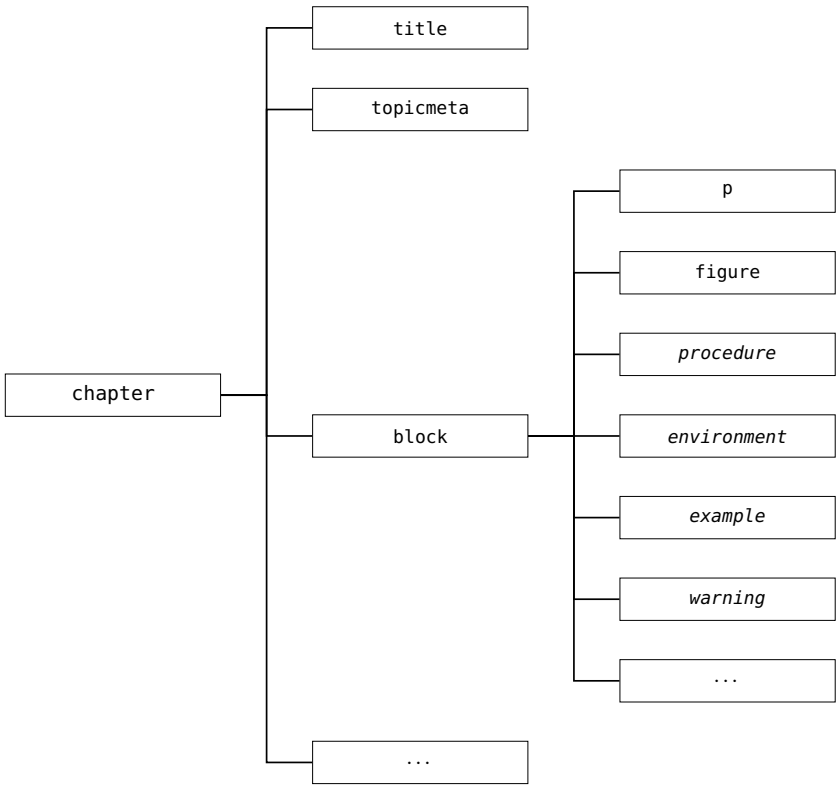


Abb. 9: Semantische Elemente der Docufy Standard DTD

Name	Docufy Standard DTD
FPI	`-//docufy//Docufy Standard DTD 20080125//EN`
Stammdatei	`manual.dtd`

Tabelle 3: Cosima DTD

METADATEN

Im System können Metadaten, Metadatendomänen und -namensräume sowie Metadatenabhängigkeiten frei konfiguriert werden (vgl. KOTHES! GMBH 2008:4).

GÜLTIGKEITEN (CVM)

Besonders komfortabel ist die Erstellung und Verwaltung für das Variantenmanagement umgesetzt. Hier können über einen Gültigkeiten-Hierarchie-Editor neue Domänen und Werte erstellt oder bearbeitet werde, die anschließend allen Benutzern zur Verfügung stehen (vgl. DOCUFY GMBH 2012a:67).

OBJEKTE UND FORMATE

Die Verwaltungseigenschaften des Systems beruhen auf Objekttypen von Informationsobjekten und deren abgeleiteten Formaten. Für jedes IO können gültige Formate festgelegt werden und die Übergänge zwischen diesen als Transformationen definiert werden. Daraus ergeben sich die Publikationsmöglichkeiten für

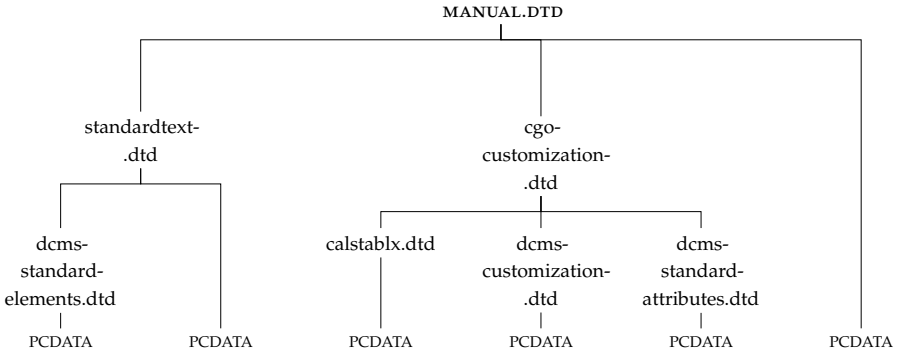


Abb. 10: Dateistruktur der Docufy-Standard-DTD

ein IO (vgl. DOCUFY GMBH 2012a:44). Für diese Anpassungen wird von CGO eine webbasierte Administrationsoberfläche, der *WebClient*, zur Verfügung gestellt (vgl. DOCUFY GMBH 2012a:21).

EDITOR

In CGO kommt als XML-Editor *XMetaL® XMAX™* zum Einsatz. Dieser Editor lässt sich über Konfigurationsdateien gut an verschiedene Benutzeranforderungen anpassen (vgl. JUSTSYSTEMS 2007:3).

Transformationen können nicht offiziell bearbeitet werden. Anpassungen werden von Docufy und seinen Partnern als kostenpflichtige Dienstleistung angeboten:

> „Eigene Modifikationen sind nicht möglich. Sollen Änderungen an den zur Verfügung stehenden Transformationen erfolgen, ist die Mitarbeit der Entwickler erforderlich." (DOCUFY GMBH 2012a:62)

Transformationen schwer anpassbar

Die entsprechenden Dateien für eigene Anpassungen liegen zwar frei zugänglich im System bereit, die darin enthaltenen Parameter und Verarbeitungshinweise sind allerdings nicht öffentlich dokumentiert (vgl. SCHULZE 2013). Dieser Umstand erschwert das Customizing, macht es aber nicht unmöglich.

3.3 IMPLEMENTIERUNG

3.3.1 *Vorgehensweise*

Die Ergebnisse aus Kapitel 2 und der Analyse der Customizing-Eigenschaften von CGO ergeben folgende Implementierungsabsichten:

1. Abbildung der Relevanzfaktoren über das CGO-eigene Variantenmanagement *Gültigkeiten*

2. Ergänzung des Informationsmodells um ein neutrales Inline-Element, um Teilsätze und -wörter mit Bewertungen auszeichnen zu können
3. Erstellen einer eigenen Transformation auf Basis des vorhandenen HTML-Stylesheets, die die intern von CGO verwendeten Gültigkeiten-Attribute mit ausgibt
4. Implementieren eines integrierten Leitfadens für Redakteure (Anwenderunterstützung)

3.3.2 *Relevanzfaktoren*

3.3.2.1 *Auswahl*

Die Zuweisung der OH3-Relevanzfaktoren zum entsprechenden Inhalt kann über verschiedene Wege erfolgen. Die Möglichkeiten können vom direkten Bearbeiten der XML-Quelle bis zur intelligenten Autorenunterstützung reichen. Generell empfiehlt es sich, eine Methode zu wählen, die den gängigen Vergabeprinzipien von Metadaten entspricht (siehe Abschnitt 2.1.2.3).

Systemunterstützung bei der Vergabe

Die manuelle Vergabe der Metadaten durch den Redakteur sollte grundsätzlich durch das System und seine Mechanismen unterstützt werden (vgl. DREWER/ZIEGLER 2011:392). Insbesondere die Logik der Metadaten (mögliche Werte, Hierarchie, etc.) muss dabei überwacht werden. Dies lässt sich innerhalb von CMS am Besten durch ein Variantenmanagement abbilden. COSIMA go! stellt dafür das CVM *Gültigkeiten* bereit.

3.3.2.2 *Implementierung*

Das CVM *Gültigkeiten* in CGO stellt zum Bearbeiten und Erstellen von hierarchischen Klassifikatoren eine GUI zur Verfügung, den *Gültigkeiten-Hierarchie-Editor*. Über diesen können innerhalb CMS sehr einfach neue Klassifikatoren hinzugefügt und eingeordnet werden.

Zuordnung über eigenen Editor

Innerhalb des Gültigkeiten-Hierachie-Editors gibt es verschieden Einordnungs- und Feldtypen. Neue Klassifikatorengruppen werden in Gültigkeiten-Domänen zusammengefasst. Diese erscheinen dem Redakteur als Tabs. Innerhalb der Domänen können Klassifikatoren in unterschiedlicher Hierarchietiefe erstellt werden. Jeder Klassifikator hat eine Bezeichnung, die an der Oberfläche erscheint und einen Bezeichner, der intern verarbeitet wird.[25] Den Klassifikatoren können zudem kurze Beschreibungstexte zugewiesen werden. Eine Beispielstruktur ist in Abb. 11 dargestellt.

25 Dies ist auch der Wert, der an einem XML-Element als Attribut festgehalten wird.

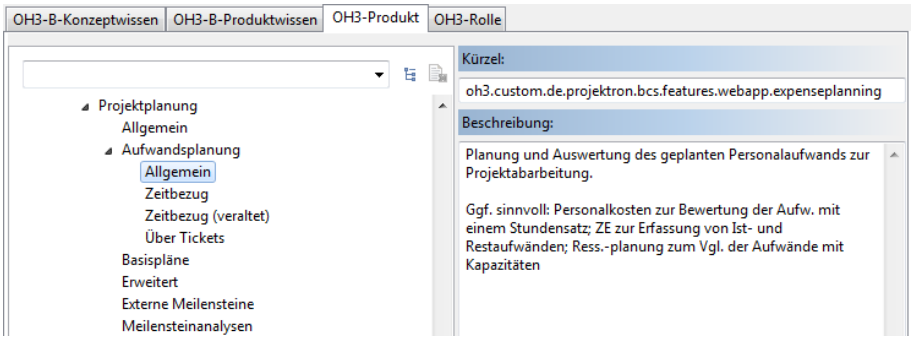


Abb. 11: Gültigkeiten-Hierachie-Editor

Die Zuweisung einer Klassifikation zu einer Information durch den Redakteur erfolgt im Moduleditor über einen Rechtsklick innerhalb eines XML-Elements und der Auswahl *Element-Gültigkeiten bearbeiten* (siehe Abb. 12 und (Docufy GmbH 2012b:188)). Dadurch öffnet sich die Übersicht aller Gültigkeiten, die durch Auswahl von Kontrollkästchen zugewiesen werden können (siehe Abb. 13).

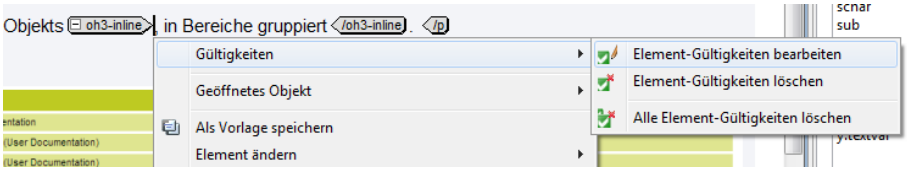


Abb. 12: Gültigkeiten-Zuweisung durch Redakteur

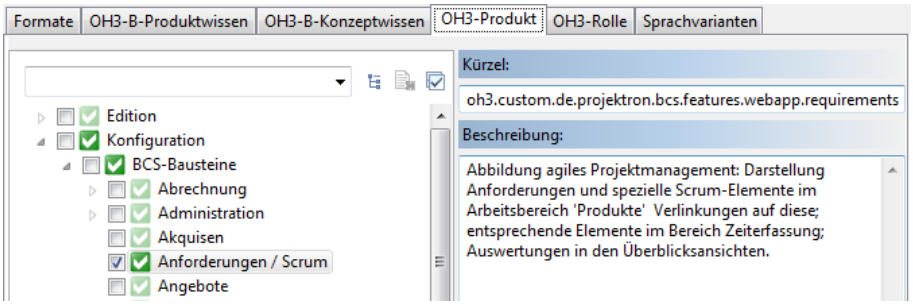


Abb. 13: Gültigkeiten-Auswahl durch Redakteur

Aufteilung in Domänen

Die zu implementierenden Relevanzfaktoren aus Abschnitt 2.5.3.2 wurden in die Domänen *Produkt*, *Rolle*, *B-Produktwissen* und *B-Konzeptwissen* unterteilt. Damit lassen sich (mit Ausnahme von *Ansicht*) alle Relevanzfaktoren, die explizit zugewiesen werden müssen, einordnen (vgl. Tabelle 2).

Die beiden Relevanzfaktoren *Produktwissen* und *Konzeptwissen* aus der Kategorie *Benutzer* werden in zwei Domänen aufgeteilt, da das CVM es sonst nicht zulässt, beide Klassifikationen gleichzeitig zuzuweisen.

Der Relevanzfaktor *Ansicht* wird aus zwei Gründen nicht als Gültigkeit sondern als klassisches Metadatum gepflegt:

- Der Relevanzfaktor wird in Projektron BCS über den *PageKey* abgebildet (vgl. Abschnitt 5.1.3). Dieser kann ca. 1500 verschiedene Werte annehmen (vgl. SAUTER 2013). Diese müssten alle als Klassifikatoren gepflegt werden.
- Der *PageKey* wird in der TD von Projektron schon seit einiger Zeit als Metadatum gepflegt. Eine Altdatenmigration wird dadurch vereinfacht.

Die Implementierung des neuen Metadatums wird in Abschnitt 3.3.4 beschrieben.

Innerhalb der Domänen wurden die Klassifikatoren entsprechend der Ergebnisse aus Abschnitt 2.5.3.2 und Abschnitt 5.1.3 erstellt. Daraus ergibt sich die Struktur, die in Abb. 14 dargestellt ist.

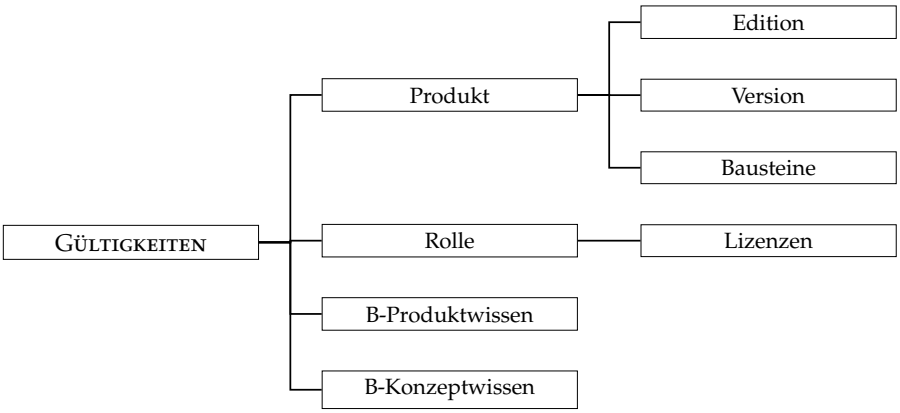


Abb. 14: Struktur der Gültigkeiten in CGO

Genaue Auszeichnung der Inhalte möglich

Damit ist es nun möglich , Inhalte so auszuzeichnen, dass diese nur Benutzern angezeigt werden, für die sie tatsächlich relevant sind. Wird beispielsweise der Arbeitsbereich *Administration* nur bei zugewiesener Lizenz *Administrator* und aktivierten Modul *admin_general* angezeigt, so können dem Inhalt diese Klassifikatoren zugewiesen werden (siehe Abb. 15).

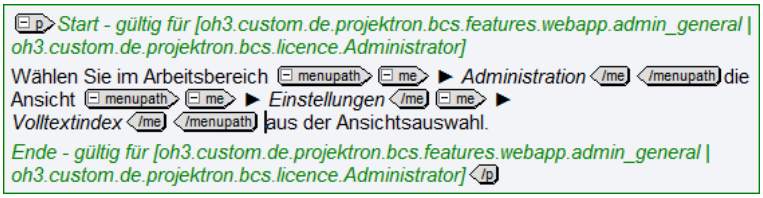


Abb. 15: Beispiel für Inhaltsklassifikation

3.3.2.3 *Namenskonventionen*

Die internen Bezeichner für Gültigkeiten, die vom CMS als XML-Attribut am jeweiligen Element gespeichert werden, dienen als Grundlage

für die spätere Filterung durch den Benutzer. Darum muss deren Benennung nach Konventionen erfolgen, um eine eindeutige Differenzierung durch das OH3-Framework zu gewährleisteten.

Bezeichnung folgt Hierarchie

Jeder Bezeichner setzt sich aus einem Klassenpfad zusammen, der – durch Punkte getrennt – die hierarchische Einordnung eines Wertes angibt. Die Formulierung des Klassenpfades erfolgt auf Englisch. Alle Klassifikatoren, die vom OH3-Framework ausgewertet werden sollen, beginnen mit der Wurzel `oh3`. Für Benutzerwerte folgt der Pfad `user`, bei System- und Rollenwerten werden nach dem Pfad `custom` die internen Pfad-Bezeichnungen aus Projektron BCS verwendet, da diese sich je nach eingesetzter Software unterscheiden.

Eindeutige Klassifikatoren

Daraus ergeben sich für alle Klassifikatoren, die als Gültigkeit abgebildet (siehe Abb. 14) und explizit vergeben werden (siehe Tabelle 2), eindeutige Bezeichner. So erhält ein Textteil, der für Anfänger im Bereich Produktwissen relevant ist, die Gültigkeit:

```
oh3.user.product.beginner
```

Ein Absatz, der nur für bei aktiviertem Spesen-Modul relevant ist, erhält die Gültigkeit:

```
oh3.custom.de.projektron.bcs.features.webapp.allowances
```

So bleibt nach dem Export für jede Gültigkeit ihre hierarchische Zugehörigkeit erhalten. Eine Übersicht über die möglichen Zusammensetzungen eines Bezeichners ist in Abb. 16 auf Seite 43 dargestellt. Implizit vom OH3-Framework vergebene[26] Gültigkeiten sind in *Kursivschrift* gekennzeichnet.

3.3.3 *Informationsmodell*

3.3.3.1 *Notwendigkeit*

Das OH3-Konzept sieht die Auszeichnung von Informationsfragmenten auf Satz-, Teilsatz- und Wortebene vor. Das CVM von COSIMA go! kann Klassifikatoren nur Elementen zuweisen. Da es im bestehenden Informationsmodell aber bisher keine Inline-Elemente gibt, die weder semantisch, noch formatierend sind, muss ein neues neutrales Inline-Element eingeführt werden, das die Zuweisung von Gültigkeiten zu diesen Fragmenten zulässt.

3.3.3.2 *Implementierung*

Wie in Abschnitt 3.2.5 beschrieben, sieht CGO Anpassungen des Informationsmodells durch den Benutzer vor und stellt dafür ein separates DTD-Modul zur Verfügung. Um die OH3-Erweiterungen getrennt von

26 Siehe dazu die Erläuterungen in Kapitel 4.

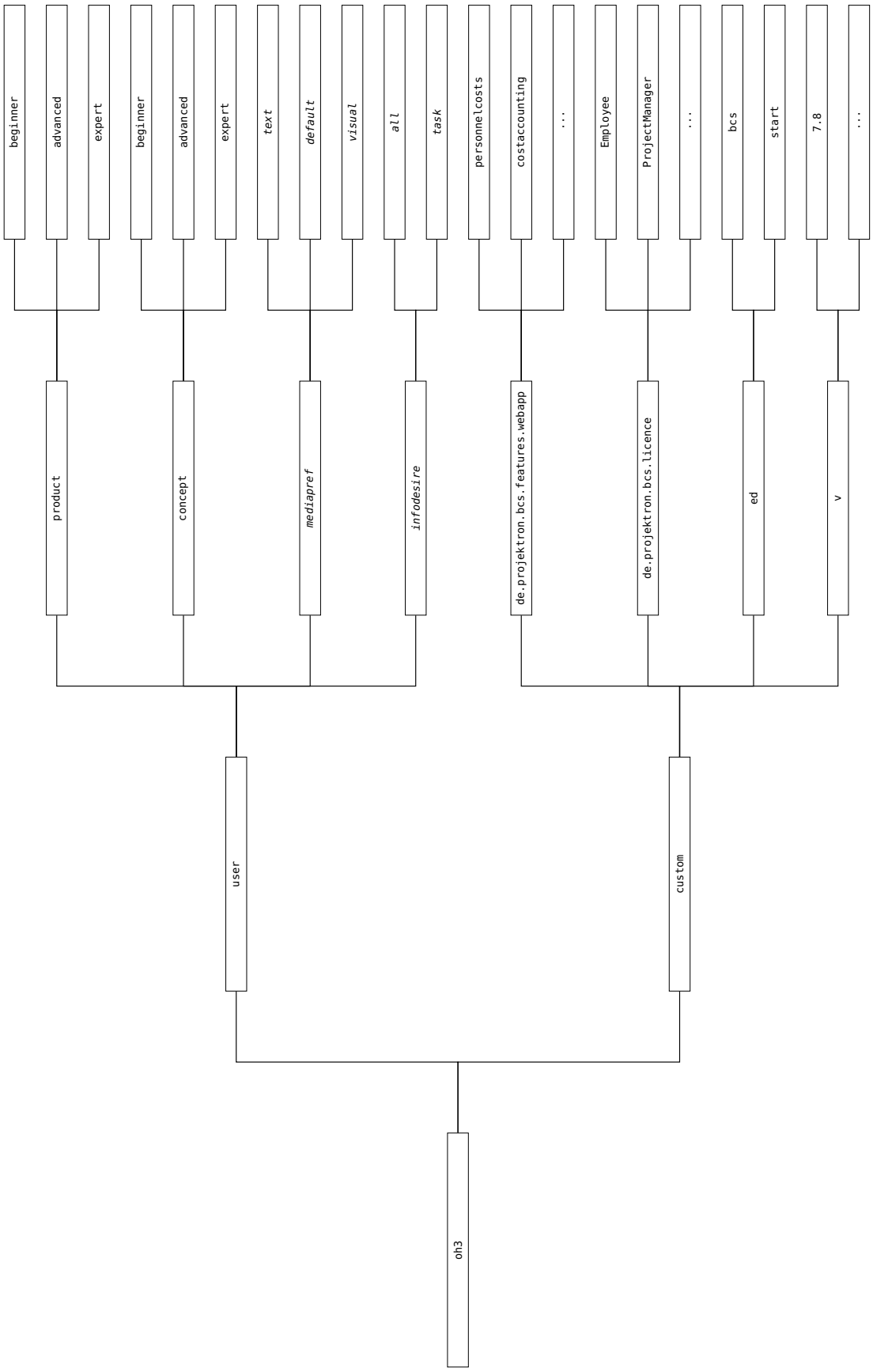


Abb. 16: Bezeichner für Gültigkeiten im CMS

weiteren Anpassungen behandeln zu können, empfiehlt es sich, diese in ein eigenes DTD-Modul auszulagern (vgl. SCHULZE 2013). Dem Modul werden Name und FPI zugeteilt, sowie eine DTD-Datei erstellt.

Name	Online-Hilfe 3.0 in Docufy COSIMA go!
FPI	`-//oh3//OH3-Customizing COSIMA go 1.0//DE`
Stammdatei	`oh3-customization.dtd`

Tabelle 4: OH3-Customization DTD

Referenzierung über Entity

Um das OH3-Modul einzubinden, muss es in der Hauptdatei *manual.dtd* referenziert werden (siehe Code 2 auf Seite 44). Die Referenzierung muss an erster Stelle erfolgen, da bei DTD Entities, die zuerst geparst werden, Priorität haben. Sie können nicht an späterer Stelle überschrieben werden (vgl. WEINHEIMER 2010:31).

```
1 <!--Einbinden der Anpassungen fuer Online-Hilfe 3.0-->
2 <!ENTITY % oh3.customization PUBLIC "-//oh3//OH3-Customizing
      Cosima go! 1.0//DE" "oh3-customization.dtd">
3 %oh3.customization;
```

Code 2: Referenzierung in manual.dtd

Zusätzlich wird in den beiden internen XML-Katalogen `catalog.cat` und `catalog-filesystem.cat` ein neuer Eintrag erstellt, der den Formal Public Identifier (FPI) des neuen Moduls der lokalen DTD-Datei zuordnet (Beispiel für `catalog.cat` in Code 3).

```
1 <catalog xmlns="urn:oasis:names:tc:entity:xmlns:xml:catalog">
2   ...
3   <public publicId="-//oh3//OH3-Customizing COSIMA go 1.0//DE"
        uri="cms:///system/cosimago/dtd/oh3-customization.dtd"/>
4 </catalog>
```

Code 3: Eintragung in XML-Katalog catalog.cat

Das Modul mit dem Dateinamen *oh3-customization.dtd* enthält die notwendigen Schritte zur Implementierung:

1. Erstellen eines neuen Elements
2. Zuweisen der CGO-Gültigkeiten-Attribute
3. Aufnehmen des Elements in bestehende Elementmengen

Um auf Standardattribute zugreifen zu können, wird das Modul *dcms-standard-attributes.dtd* eingebunden. Einige benötigte Element-Entity-Definitionen aus dem Modul *cgo-customization.dtd* werden in das OH3-Modul verschoben.

Neues Element: oh3-inline

Das neue Inline-Element mit dem Namen `oh3-inline` wird über die Entity `oh3.elements` in die vorhandenen Elemente-Mengen referenziert. Sollen zu einem späteren Zeitpunkt weitere Inline-Elemente eingeführt werden, kann das über Hinzufügen der Elementnamen in die Entität `oh3.elements` getan werden.

```
<!--Definieren der OH3-Elemente als Entity-->
<!ENTITY % oh3.elements "oh3-inline" >

<!--Einbinden von Standard-Entities-->
<!ENTITY % dy.std.atts PUBLIC "-//docufy//ENTITY DCMS standard
    attributes 20040723//EN" "kernel/dcms-standard-attributes.dtd
    ">
%dy.std.atts;

<!--Einbinden von Entities aus cgo-customization.dtd-->
<!ENTITY % technical-data "tec-data">
<!ENTITY % notes "danger | danger-ref  | warning | warning-ref |
    caution | caution-ref | notice | notice-ref | note | note-ref
     | environment | environment-ref">
<!ENTITY % textvar "y.textvar | v-ref">
<!ENTITY % pcdata "#PCDATA | schar | %textvar;">

<!--Hinzufuegen zu Inline-Elementen-->
<!ENTITY % inline "%pcdata; | b | symbol | sup | sub | key | link
     | nb.text | code | url | menupath | gui-element | i | u | %
    oh3.elements;">

<!--Hinzufuegen zu Block-Elementen-->
<!ENTITY % normal.block "p | ul | dl | figure | icon | table |
    procedure | verbatim | maintenance-table | troubleshooting |
    %technical-data; | example | gear-list | skill-list | tool-
    list | mat-list | safety-comp-list | safety-category-list |
    safety-phrase-list |  pdf | %notes; | xe | %oh3.elements;">
<!ENTITY % simple.block "p | ul | dl | table | xe | %oh3.elements
    ;">

<!--Definieren des inline-Elements-->
<!ELEMENT oh3-inline (%inline;)*>
<!ATTLIST oh3-inline
    %dy.id;
    %dy.validity; >
```

Code 4: OH3-DTD-Modul

Die Änderungen, die in der Modulstruktur der Docufy-DTD vorgenommen wurden, sind in Abb. 18 auf Seite 47 dargestellt. Umrandete Elemente sind neu hinzugefügt, kursive Elemente neu referenziert worden.

Aus diesen Änderungen ergibt sich die Möglichkeit, auch Satzteilen oder Wörtern Gültigkeiten bzw. Klassifikationen zuzuweisen (siehe Abb. 17).

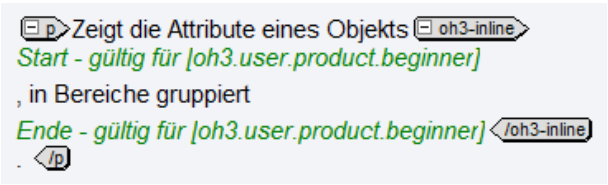


Abb. 17: oh3-inline-Element mit Gültigkeiten

3.3.4 *Metadaten*

3.3.4.1 *Notwendigkeit*

Durch die hohe Anzahl an möglichen Werten für den Relevanzfaktor *Ansicht* in Projektron BCS und für eine leichtere Altdatenmigration wird dieser Relevanzfaktor als Metadatum gepflegt (siehe Abschnitt 3.3.2). Dazu muss ein neues Metadatum eingerichtet und innerhalb von CGO entsprechenden IO-Typen zugeordnet werden.

3.3.4.2 *Namensräume*

Verwaltung in Namensräumen

Metadaten werden in COSIMA go! als *Eigenschaften* von Informationsobjekten innerhalb sog. Metadaten-Namensräume verwaltet (vgl. DOCUFY GMBH 2012a:36).

Im Auslieferungszustand des Systems sind zwei Namensräume standardmäßig eingerichtet:

DOCUFY DYNAMIC METADATA
: `http://metadata.docufy.de/cms/docufy/meta`
Für vom System intern verwendete Metadaten.

CUSTOMER DYNAMIC METADATA
: `http://metadata.docufy.de/cms/dynamic/meta`
Für vom Benutzer definierte Metadaten.

Wie von Docufy empfohlen (vgl. DOCUFY GMBH 2012a:36) wird das neue Metadatum innerhalb von *Customer dynamic metadata* eingerichtet.

3.3.4.3 *Definition*

Bei der Definition neuer Metadaten muss der u.a. der Datentyp (z.B. *String* oder *Integer*) und die Zuordnung zu einer IO-Ebene (z.B. *Version*

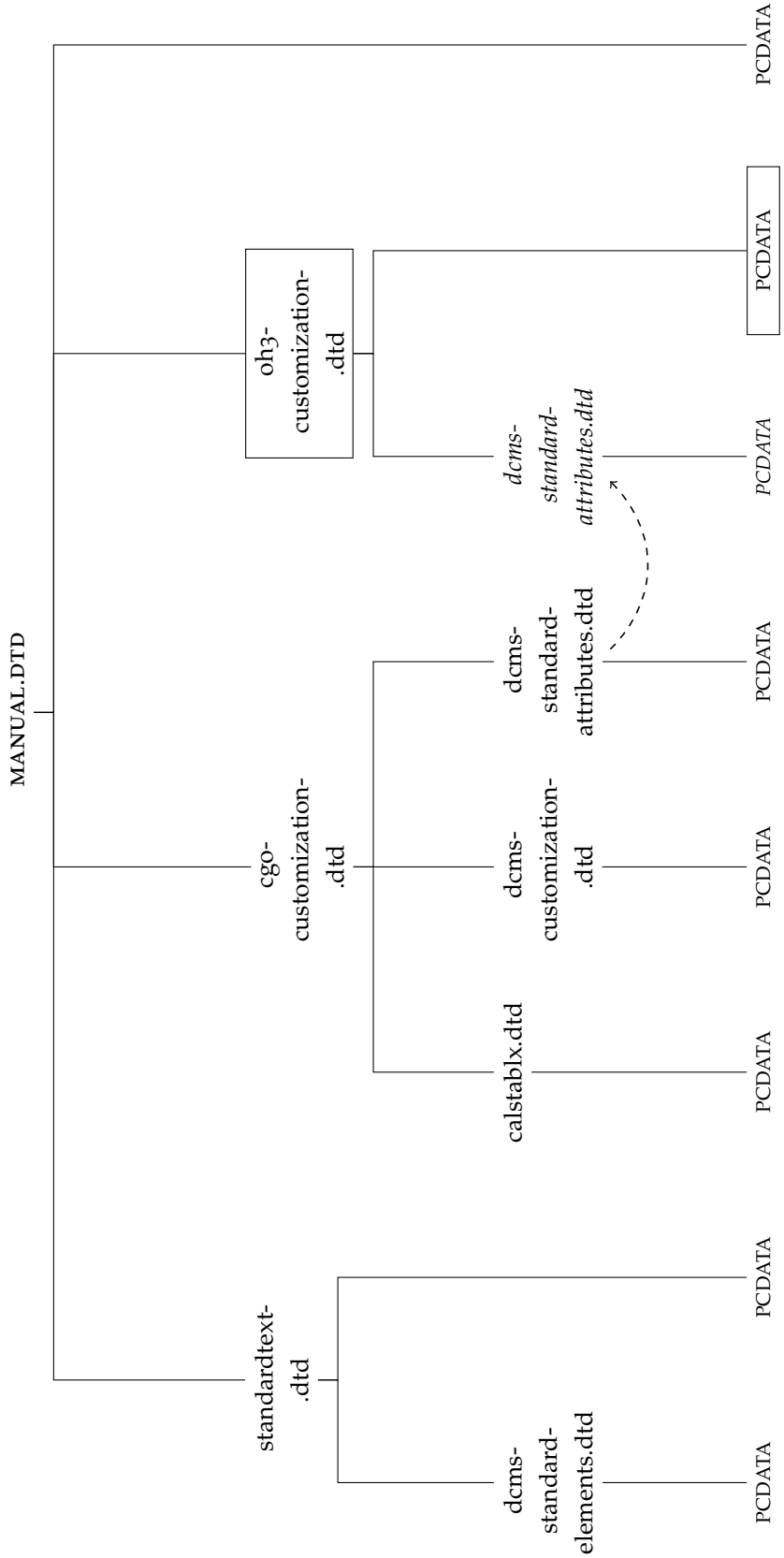


Abb. 18: Änderungen der Docufy-Standard-DTD

oder *Sprachvariante*) festgelegt werden. Zusätzlich können Einschränkungen[27] definiert werden.

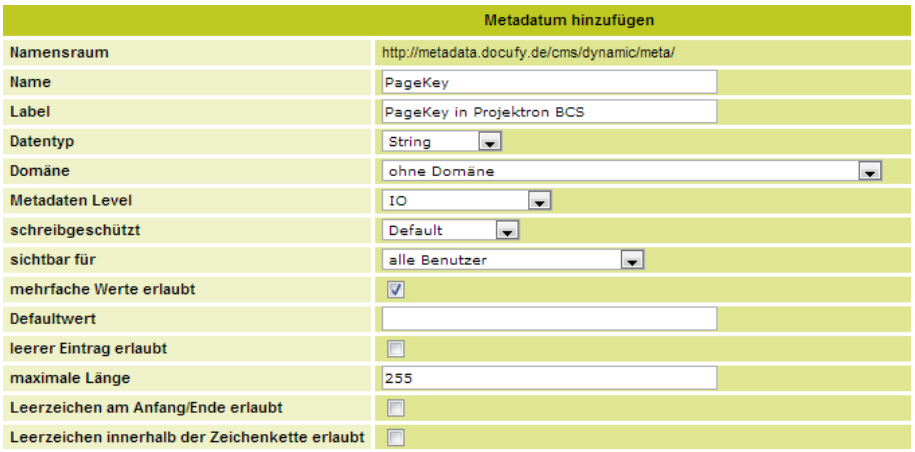


Abb. 19: Metadatum PageKey hinzufügen

3.3.4.4 *Zuordnung*

Um das neu definierte Metadatum anwenden zu können, muss es den entsprechenden IO-Typen in der Verwaltungsansicht des *WebClients* zugeordnet werden (vgl. DOCUFY GMBH 2012a:37). Anschließend kann das neue Metadatum z.B. beim Anlegen eines neuen Moduls als Pflichtfeld abgefragt werden (siehe Abb. 20).

3.3.5 *Transformation*

Wie schon in Abschnitt 3.2.5 beschrieben, ist die Anpassung von Transformationen durch den Benutzer in COSIMA go! nicht vorgesehen. Die Steuerdateien für die Transformationen (*transformation configs*) und die Stylesheets sind aber grundsätzlich zugänglich und veränderbar (vgl. SCHULZE 2013).

Keine öffentliche Dokumentation

Aufgrund der fehlenden öffentlichen Dokumentation musste zunächst durch Analyse der vorhandenen Transformationen und Anwenden des *Trial-&-Error*-Prinzips ein korrektes Vorgehen für das Customizing ermittelt werden. Dazu zählt neben der zentralen Stylesheet-Erstellung auch das Registrieren des neuen Ausgabeformats im System über den CGO-WebClient (siehe Abschnitt 3.2.5).

Die konkreten Schritte umfassen:

1. Registrieren eines neuen Formattyps in CGO (WebClient)

27 Einschränkungen legen z.B. fest, ob Leerzeichen am Anfang oder innerhalb der Zeichenkette erlaubt sind oder Mehrfacheinträge möglich sein sollen.

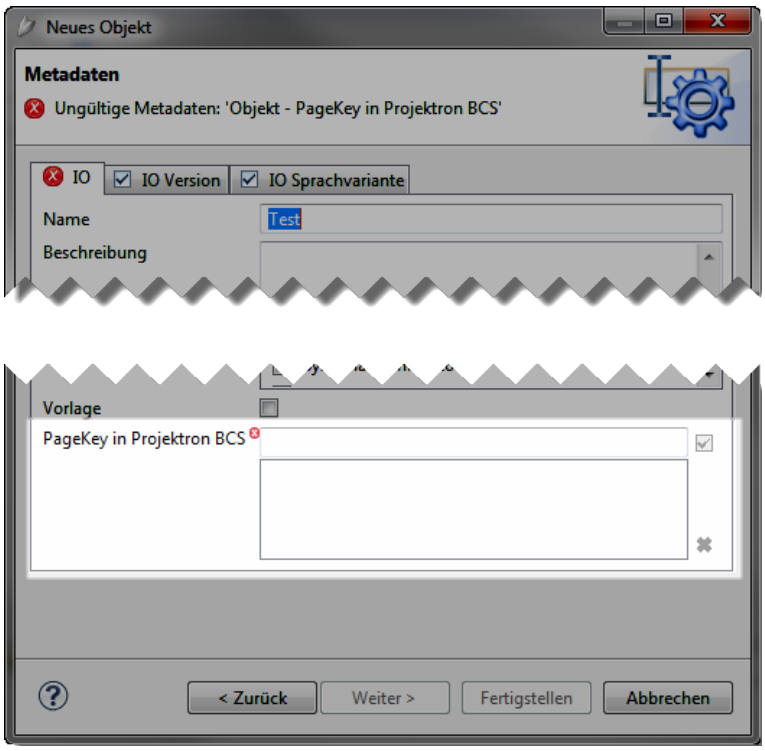


Abb. 20: Metadatum PageKey abfragen

2. Hinzufügen des neuen Formattyp als abgeleitetes Format des Standard-Dokumentationstyps (IO) (WebClient)
3. Definieren einer Transformation für den Übergang zwischen den Formaten (WebClient)
4. Erstellen einer neuen Steuerdatei (XML) für die neue Transformation (*transformation config*)
5. Erstellen neuer XSL-Stylesheets für die Transformation
6. Erstellen eines Dokumentschemas (DTD) für das Ausgabeformat
7. Generieren von XMAX-Anpassungen für die Ausgabe in der CGO-internen Vorschau (CSS & CTM)

3.3.5.1 *Anpassungen im WebClient*

Im WebClient müssen die Schritte 1-3 aus der voranstehenden Liste (siehe Liste 1) durchgeführt werden:

FORMATTYP

Für jede Publikationsmöglichkeit muss ein eigener sog. „Format-Typ“ *[sic]* in CGO definiert sein. Dieser ist zunächst einmal unabhängig vom eigentlichen *Dateiformat* der Ausgabe. Format-Typen können sich in Dateiformat, Struktur, Ausgabemedium, Verwendungszweck oder auch weiteren Merkmalen unterscheiden (vgl. SCHULZE 2013). Das Modifizieren und Anlegen von

Format-Typen kann laut Hersteller nicht vom Benutzer vorgenommen werden (vgl. DOCUFY GMBH 2012a:43). Der WebClient stellt aber die benötigte Verwaltungsoberfläche dafür bereit. Darin kann über einen Dialog ein neuer Format-Typ angelegt und seine Eigenschaften können bearbeitet werden (siehe Abb. 21).

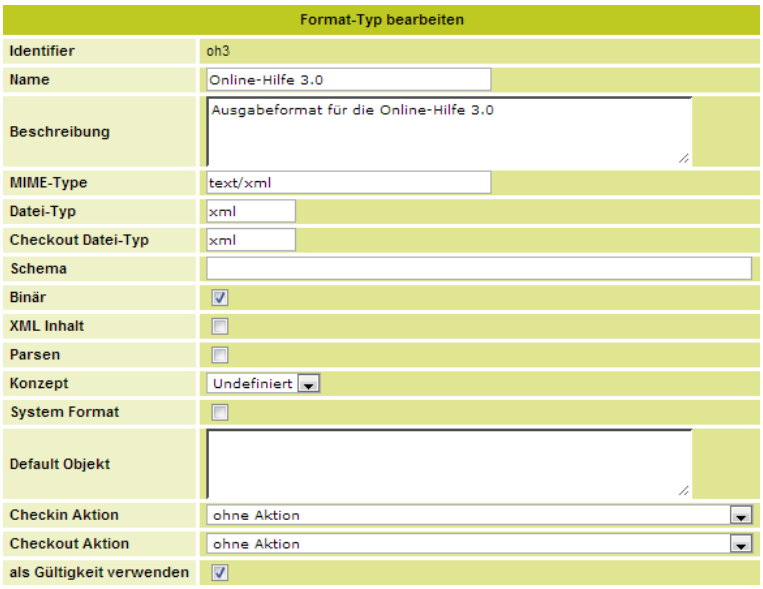


Abb. 21: Format-Typ hinzufügen

An dieser Stelle wurde der Format-Typ *Online-Hilfe 3.0* definiert. Er soll das Ausgabeformat repräsentieren, das später an das Framework weitergegeben wird. Der neue Format-Typ hat innerhalb des Systems zunächst keine Auswirkung, da er noch keinem IO-Typ zugeordnet ist (vgl. SCHULZE 2013).

OBJEKTTYP

Der Objekttyp oder auch IO-Typ[28] definiert die Art eines Informationsobjekts. Informationsobjekte sind innerhalb von CGO alle Inhalts- oder Systemdateien, die vom System verwaltet werden können. Dazu gehören neben dem eigentlichen Content z.B. auch Übersetzungsaufträge oder Stylesheets. Jedem IO-Typ wird ein führender Format-Typ und ggf. abgeleitete Format-Typen zugeordnet. Darüber wird entschieden, welche Publikationsmöglichkeiten sich für ein Informationsobjekt ergeben.

Soll es nun möglich sein, die Standard-Benutzerdokumentation in den *Online-Hilfe 3.0*-Format-Typ zu publizieren, muss ihrem IO-Typ *Documentation (User Documentation)* der neue Format-Typ zugeordnet werden. Diese Änderung kann innerhalb des Web-Clients im Bereich IO-Typen im Menü *Format-Typen bearbeiten* vorgenommen werden (vgl. DOCUFY GMBH 2012a:44).

28 Die Docufy-Handbücher selbst verwenden keine eindeutige Terminologie (vgl. DOCUFY GMBH 2012a:44).

TRANSFORMATION

In CGO werden die Übergänge zwischen Formaten eines IO-Typs durch Transformationen erzeugt. Diese können nach Herstellerangaben nicht vom Benutzer bearbeitet werden (vgl. DOCUFY GMBH 2012a:62), eine Verwaltungsoberfläche steht mit dem Web-Client aber bereit. Einer Transformation werden Ausgangs- und Zielformat zugeordnet und auf eine Steuerdatei im System (siehe Abschnitt 3.3.5.2) verwiesen. Für jeden möglichen Übergang zwischen Format-Typen muss eine Transformation im System hinterlegt werden (vgl. SCHULZE 2013).

Um den Übergang des IO-Typs *Documentation (User Documentation)* in den neuen abgeleiteten Format-Typ *Online-Hilfe 3.0* zu definieren, werden der neuen Transformation zunächst Ausgangs- und Ziel-Format-Typ zugeordnet und anschließend wird auf eine zu erstellende Transformation-Steuerdatei verwiesen.

Transformation bearbeiten	
Quell Format-Type	Documentation (User Documentation)
Ziel Format-Type	Online-Hilfe 3.0
Name	OH3 (User Documentation)
Beschreibung	Online-Hilfe 3.0 (User Documentation)
Konfigurationpfad	/system/cosimago/transformation_config/oh3-export.xml

Abb. 22: Transformation bearbeiten

3.3.5.2 *Transformation Config*

Transformationen werden in CGO über sog. *transformation configs* gesteuert (siehe auch Abschnitt 3.2.5). Diese sind als XML-Dateien im CMS hinterlegt und enthalten neben Pfadangaben und Reihenfolge der zu verwendenden Stylesheets diverse Parameter mit Verarbeitungsangaben, Pfad- und Dateiverweise, Metadaten-Zuordnungen etc. Da die Parameter nicht dokumentiert sind, können nicht für alle Angaben Rückschlüsse gezogen werden.

Customizing nicht vorgesehen

Da eine Bearbeitung der Steuerdateien durch den Benutzer nicht vorgesehen ist (vgl. DOCUFY GMBH 2012a:62), empfiehlt es sich zunächst eine bestehende ähnliche Steuerdatei zu kopieren und anschließend anzupassen (vgl. SCHULZE 2013). Dazu wurde die Steuerdatei `html_preview.xml` gewählt – die im Standard ausgegebene Online-Hilfe von COSIMA go!.

Innerhalb der Original-Steuerdatei werden drei Stylesheets referenziert:

REMOVE_FILTERED.XSL

Vorverarbeitung der Daten und Entfernen der nicht gültigen Varianten (Filterung)

CHAPTERREF2CHAPTER.XSL
: Auflösen von `chapter`-Referenzierungen

MAIN.XSL
: Hauptverarbeitungsdatei für Ausgabe Online-Hilfe, die wiederum auf zahlreiche Untermodule verweist

Angepasst wurden die Angaben des ersten und dritten Stylesheets in der Verarbeitungsreihenfolge Code 5. Der Rest der Steuerdatei blieb unverändert.

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE config PUBLIC '-//Docufy//DTD Transformation Config//EN
    ' '/system/cosima/dtd/transformation/config.dtd'>
<config>
  <step processId="de.docufy.cms.service.transformation.jobs.
      XSLTransformation" server="" duration="0">
    <property name="xsl.cmspath.1" value="/system/cosimago/
        transformation/oh3/oh3-preprocessing.xsl"/>
    <property name="xsl.cmspath.2" value="/system/cosimago/
        transformation/common/chapterref2chapter.xsl"/>
    <property name="xsl.cmspath.3" value="/system/cosimago/
        transformation/oh3/oh3-main.xsl"/>
      ...
  </step>
</config>
```

Code 5: Transformation Config OH3-Export

3.3.5.3 *XSL-Stylesheets*

Für die Ausgabe des OH3-Ausgabeformats musste ein Stylesheet der Standard-Online-Hilfe angepasst und eines neu erstellt werden.

OH3-PREPROCESSING.XSL
: Am Stylesheet `remove_filtered.xsl` wurde das Template, das für das Entfernen der gefilterten Elemente zuständig ist, entfernt, da nach dem OH3-Konzept die Filterung auf Seite des Ausgabe-Frameworks stattfindet.

OH3-MAIN.XSL
: Das Stylesheet `oh3-main.xsl` wurde komplett neu erstellt, da Anpassungen am bestehenden, stark fragmentierten Stylesheet `main.xsl` zu aufwändig gewesen wären.

  Zentrale Änderung, die für das Umsetzen des OH3-Konzepts notwendig war, ist das Ausspielen der Gültigkeiten-Attribute als Klassenelemente (Attribut `class`) an HTML-Elemente. Auch die semantischen Eigenschaften für eine spätere gestalterische Auszeichnung der Elemente werden als Attributwert der Klasse zugeordnet.

Zentrales Template in Stylesheet

Für eine möglichst effiziente Umsetzung der Elementerzeugung innerhalb des Stylesheets wurde ein allgemeingültiges Template entworfen, das die Parameter `node` (Kontextknoten), `name` (Neuer Elementname) und `add-class` (Zusätzliche Klasse) entgegennimmt.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/
       Transform" version="2.0">
3      <xsl:output encoding="UTF-8" method="xml" doctype-public
           ="-//oh3//OH3-Output COSIMA go 1.0//DE"
           doctype-system="/system/cosimago/dtd/oh3-output.dtd"
           />
4     <xsl:param name="base">https://demo02.docufy.de</
          xsl:param>
5     <!--Elemente generieren und Klasse zuweisen-->
6      <xsl:template name="generate-element">
7          <xsl:param name="node" select="."/>
8          <xsl:param name="validities" select="$node/@y.
               validity.text"/>
9          <xsl:param name="name"/>
10         <xsl:param name="add-class"/>
11         <xsl:element name="{$name}">
12             <xsl:choose>
13                 <xsl:when test="$validities != '' and $
                       add-class != ''">
14                  <xsl:attribute name="class">
15                      <xsl:value-of select="$add-class"/>
16                      <xsl:text> </xsl:text>
17                      <xsl:value-of select="replace($
                            validities,' \| ',' ')"/>
18                  </xsl:attribute>
19              </xsl:when>
20              <xsl:when test="$validities != '' and $
                    add-class = ''">
21                  <xsl:attribute name="class">
22                      <xsl:value-of select="replace($
                            validities,' \| ',' ')"/>
23                  </xsl:attribute>
24              </xsl:when>
25              <xsl:when test="$add-class != ''">
26                  <xsl:attribute name="class">
27                      <xsl:value-of select="$add-class"/>
28                  </xsl:attribute>
29              </xsl:when>
30              <xsl:otherwise/>
31          </xsl:choose>
32          <xsl:apply-templates/>
33      </xsl:element>
34   </xsl:template>
```

Code 6: generate-element Template in oh3-main.xsl

Dieses Template kann nun gruppiert mit entsprechenden Parametern aufgerufen werden, so dass z.B. alle Inline-Elemente in ein span-Element mit entsprechender Klasse transformiert werden (siehe Code 7).

```
1 <!--Inline-Elemente, die ein <span> erzeugen sollen-->
2 <xsl:template match="code | gui-element | key | link | nb.
      text | oh3-inline">
3   <xsl:call-template name="generate-element">
4     <xsl:with-param name="name">span</xsl:with-param>
5     <xsl:with-param name="add-class"><xsl:value-of select="
          local-name(.)"/></xsl:with-param>
6   </xsl:call-template>
7 </xsl:template>
```

Code 7: Aufruf von generate-element in oh3-main.xsl

Das generierte XML-Dokument enthält eine Untermenge von HTML-Elementen, die in den bestehenden Anzeigecontainer des OH3-Frameworks eingebunden wird.

3.3.5.4 *Vorschau in CGO*

Vorschau des Exports im internen Editor

Um eine Vorschau des Ausgabe-Formats in COSIMA go! zu ermöglichen, benötigt der integrierte Editor XMAX ein hinterlegtes Dokumentschema und dazugehörige Style-Informationen in Form von CSS- und CTM-Dateien (vgl. JUSTSYSTEMS 2007:5).

Um die korrekte Vorschau in CGO und ein valides Dokumentschema des Exports zu gewährleisten, wurde für das Ausgabeformat der Online-Hilfe 3.0 ein Dokumentschema entworfen, das die verwendete HTML-Untermenge abbildet.

Name	Online-Hilfe 3.0 Ausgabe aus CGO
FPI	`-//oh3//OH3-Output COSIMA go 1.0//DE`
Stammdatei	`oh3-output.dtd`

Tabelle 5: OH3-Output DTD

Dabei wurde auf eine gute Anpassbarkeit durch den Einsatz von Entities für einzelne Elementgruppen und Attribute geachtet (Code 8).

```
1 <!--Definieren der Elementmengen als Entity-->
2 <!ENTITY % block.elements "div | h1 | h2 | h3 | p" >
3 <!ENTITY % list.elements "dl | ul | ol | li" >
4 <!ENTITY % inline.elements "span | b | i | u | sub | sup | img" >
5 <!ENTITY % pcdata "#PCDATA" >
```

```
6 <!ENTITY % all.elements "(%pcdata; | %block.elements; | %list.
      elements; | %inline.elements;)*" >
7 <!ENTITY % most.elements "(%pcdata; | %list.elements; | %inline.
      elements;)*" >
8 <!ENTITY % text.elements "(%pcdata; |%inline.elements;)*" >
```

Code 8: Entities in OH3-Output DTD

3.3.5.5 *Publikation aus COSIMA go!*

Durch die vorherigen Schritte und die daraus resultierende Registrierung des Online-Hilfe-3.0-Formats als mögliches Ausgabe-Format einer Standard-Dokumentation kann nun innerhalb von CGO zusätzlich in das OH3-Format publiziert werden.

Der Redakteur kann die Publikation durch Rechtsklick auf eine Dokumentation (IO-Typ: *Documentation (User Documentation)*) und anschließende Auswahl von *Format anzeigen* → *Online-Hilfe 3.0* anstoßen.

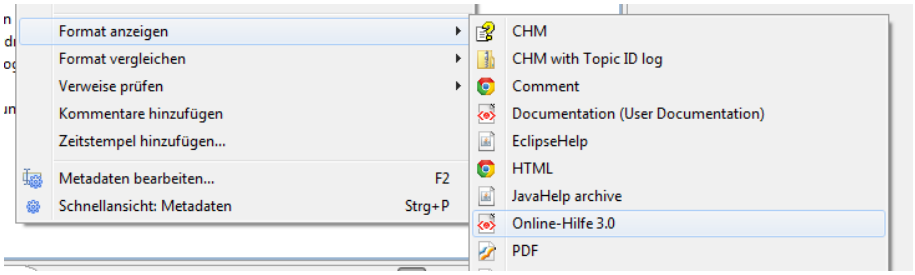


Abb. 23: Online-Hilfe 3.0 publizieren

Wird *Online-Hilfe 3.0* als Ausgabeformat ausgewählt, wird vom System die Transformation angestoßen und nach erfolgreicher Fertigstellung eine Vorschau des Formats im Editor von CGO angezeigt. Dadurch kann eine korrekte Transformation kontrolliert werden.

3.3.6 *Anwenderunterstützung*

Hilfestellung für Redakteure

Um den Redakteur bei der Vergabe der Relevanzfaktoren zu unterstützen, soll eine integrierte Anwenderunterstützung in das System implementiert werden. Durch die Customizing-Möglichkeiten von COSIMA go! und XMAX (siehe Abschnitt 3.2.5) kann dies in Form von Beschreibungstexten oder zusätzlichen Einblendungen erfolgen.

3.3.6.1 *Beschreibungstexte Relevanzfaktoren*

Die Beschreibungstexte für einzelne Gültigkeiten können im *Gültigkeiten-Hierarchie-Editor* über ein Freitextfeld vergeben werden. Allerdings sind Texte in ihrer Länge auf 256 Zeichen beschränkt. An dieser

Stelle wurden für die einzelnen Relevanzfaktoren und ihre möglichen Werte Beschreibungstext hinzugefügt, die den Redakteur bei der Auswahl unterstützen sollen.

3.3.6.2 *Beschreibungstexte XML-Elemente*

In XMetal und XMAX können für beliebige XML-Elemente Beschreibungen vergeben werden, die dem Redakteur beim Editieren von Dokumenten im Editor eingeblendet werden, wenn er das Element auswählt (vgl. JUSTSYSTEMS 2007:29).

Gepflegt werden die deutschen Beschreibungstexte als Key-Value-Paare in der Datei `manual_de.properties` (siehe Code 9). Hier wurde für das neu hinzugefügte Element `oh3-inline` ein kurzer Beschreibungstext hinterlegt.

```
1 ...
2 element.notice-ref=Sicherheitshinweis Typ "Hinweis" (Referenz)
3 element.oh3-inline=Online-Hilfe 3.0: Auszeichnung f\u00FCr inline
      -Elemente, denen Relevanzfaktoren \u00FCber G\u00FCltigkeiten
       zugewiesen werden sollen.
4 element.original=Hinweis, ob es sich um das Original oder eine \
      u00DCbersetzung des Dokumentes handelt
5 ...
```

Code 9: Beschreibungstexte in XMAX

3.3.7 *Bereitstellung*

Bereitstellung als Import-Pakete

Alle Anpassungen, die am CMS vorgenommen wurden, können als Systemkonfiguration zur Verfügung gestellt werden (vgl. SCHULZE 2013). Eine Gesamtkonfiguration besteht aus einer XML-Datei, die alle Konfigurationsparameter beinhaltet, sowie einem ZIP-Archiv, das alle veränderten bzw. neuen Dateien in der internen CGO-Ordnerstruktur enthält.

Die Ergebnisse aus Kapitel 3 können somit als Konfigurationspaket zusammengefasst werden und auf andere CGO-Systeme durch einen Import übertragen werden. Das Konfigurationspaket für das OH3-Customizing befindet sich im Anhang auf beiliegender CD. Eine Anleitung zur Einrichtung ist in Anhang A.2 zu finden.

Um die implementierten Funktionen zu testen, kann vorgefertigter Beispielcontent verwendet werden. Eine Paket zum automatischen Export befindet sich im Anhang auf beiliegender CD. Eine Anleitung zum Import ist in Anhang A.2 zu finden.

4

DYNAMISCHE FILTERUNG DES CONTENTS

In diesem Kapitel wird, aufbauend auf den Ergebnissen aus Kapitel 3, ein Framework zur dynamischen Filterung konzipiert, angepasst und mit Web-Technologien umgesetzt.

4.1 EINFÜHRUNG

4.1.1 *Anforderungen*

Rahmenstruktur wird benötigt

Mit den Erkenntnissen aus den vorherigen Kapiteln ist es möglich, mit dem CMS COSIMA go! benutzerorientierten Content zu erstellen und auszuspielen. Um diese Content-Art auch dynamisch filtern zu können, bedarf es einer Rahmenstruktur, die diese Funktionalität unabhängig vom Inhalt zur Verfügung stellt. Diese Rahmenstruktur, die wiederum aus mehreren Konstrukten besteht, wird im Folgenden als *Framework* bezeichnet. Das Framework hat in diesem Fall die Aufgabe, den Content darzustellen, Benutzereingaben entgegenzunehmen und weiterzuverarbeiten. Es soll dabei selbst aber nur einen Rahmen zur Verfügung stellen, dessen spezielle Konfiguration durch separate Dateien gesteuert wird (vgl. OEVERMANN 2012:4).

Das OH3-Framework soll alle konzeptionellen Eigenschaften der Online-Hilfe 3.0 umsetzen können, ohne dabei jedoch von den jeweiligen speziellen Implementierungen für die Stammsoftware abhängig zu sein. Dabei sind die zentralen Funktionalitäten: das Bereitstellen einer Filterfunktion und die flexible Verarbeitung und Darstellung der Daten aus dem CMS. Darüber hinaus sollen Abhängigkeiten und Regeln für die Content-Darstellung formuliert werden und eine automatische Verwaltung von lokalen Benutzerprofilen erfolgen.

Konkrete Anforderungen durch Konzept

In Tabelle 2 auf Seite 28 sind konkrete Anforderungen an das Framework aufgelistet. So müssen Relevanzfaktoren, die nicht explizit vom Redakteur zugewiesen werden (*Darstellungspräferenz*, *Informationsanliegen*) durch das System automatisiert und regelbasiert vergeben werden. Ebenso muss die Auswertung der entsprechend kategorisierten Relevanzfaktoren (*Produkt*, *Rolle*, *Ansicht*) im Hintergrund durch das Framework erfolgen. Die in Abb. 6 auf Seite 29 dargestellten Abhängigkeiten zwischen Benutzerfaktoren sollen implementierungsabhängig ausgewertet und verarbeitet werden können.

4.1.2 *Vorleistungen*

Bei Darstellung und Filterung des benutzerorientierten Contents kann auf ein bestehendes Framework aufgebaut werden, das bereits auf grundlegende Anforderungen des OH3-Konzepts eingeht (siehe ausführlich in Abschnitt 2.4.3.2 und OEVERMANN 2012).

Anpassungen notwendig

Durch die Ergebnisse aus Kapitel 2 und Kapitel 3, die sich teilweise stark von den Grundlagen unterscheiden, auf denen das initiale OH3-Konzept aufbaut, musste das Framework in vielen Bereichen stark ergänzt und umstrukturiert werden.

Übernommen wurden die grundsätzlichen Prinzipien:

- Clientseitige Architektur
- Technische Umsetzung auf Basis moderner Webtechnologien (HTML5, CoffeeScript etc.)
- Umfangreiche Konfigurierbarkeit und Content-Verknüpfungen über *Maps* in ausgelagerten XML-Dateien
- Dynamisches Anzeigeverhalten und Deep-Linking[29]

Des weiteren wurden der Aufbau der Benutzeroberfläche und die Funktionen zu DOM-Manipulation sowie URL-Verarbeitung weiterverwendet.

In vielen zentralen Bereichen, z.B. dem Filter-Prinzip des Frameworks, mussten allerdings umfangreiche Anpassungen gemacht werden, so dass nur ein kleiner Teil der ursprünglichen Vorlage unverändert blieb.

4.1.3 *Technische Umsetzung*

4.1.3.1 *Markup*

Zum Aufbau der Grundstruktur, Auszeichnung des Contents (*Markup*) und Gestaltung des Frameworks wird HTML5 eingesetzt. Bei HTML5 handelt es sich *de facto* um die Kombination mehrerer neuartiger Webtechnologien (vgl. SCHOBER 2012:36). Im Kern sind das die Dokumentenbeschreibungssprache HTML5, die verarbeitende Skriptsprache JavaScript (JS) und die deklarative Stylesheetsprache CSS3. HTML5 ist noch kein verabschiedeter Standard,[30] wird aber von fast allen modernen Browsern schon unterstützt (vgl. SCHOBER 2012:36).

Einzelne Bereiche, die für die Umsetzung des Frameworks von besonderer Bedeutung sind, werden im Folgenden aufgeführt.

29 Unter *Deep-Linking* versteht man die Möglichkeit, über die URL direkt zu einem *Application State* der Anwendung zu gelangen.

30 Derzeit ist das noch HTML 4.01 (ISO/IEC 15445 2000) bzw. das XML-konforme XHTML 1.0 (W3C Recommendation)

SEMANTISCHE ELEMENTE
: Die mit HTML5 eingeführten semantischen Auszeichnungselemente werden für den Aufbau der Grundstruktur des Frameworks benutzt. Durch den Einsatz semantischer Elemente kann die Kompatibilität zu mobilen Endgeräten verbessert werden und den Inhalten eine Funktion (Navigation, Zusatz, etc.) zugewiesen werden (vgl. MÜNZ/GULL 2010:89ff).

FORMULARELEMENTE
: Mit HTML5 wurden eine Reihe neuer Formularelemente eingeführt, darunter das `input-range`-Element. Durch dieses Element wird ein Schieberegler erzeugt – das zentrale Bedienelement des OH3-Konzepts.

LOKALER SPEICHER
: Eine weitere Neuerung, die mit HTML5 eingeführt wurde, ist die Möglichkeit, Daten lokal (innerhalb des Browsers) im sog. *LocalStorage* abzulegen. Bisher war dies nur über Cookies[31] möglich. Gespeichert werden Daten im LocalStorage in Attribut-Wert-Paaren. Diese Daten verbleiben auch nach Beenden des Browser im Speicher erhalten. Diese Eigenschaften erlauben es, Benutzereinstellungen im lokalen Speicher zu hinterlegen und sie beim nächsten Aufruf der Seite wiederherzustellen.

CSS3
: Für die Gestaltung der Benutzeroberfläche kommen viele der neuen Möglichkeiten zum Einsatz, die Cascading Style Sheets (CSS)3 bietet.

4.1.3.2 *Scripting*

Als Scripting-Sprache für die Verarbeitung und Filterung des Contents wird *CoffeeScript* verwendet. CoffeeScript (CS) transkompiliert in JavaScript, während es dessen Funktionsumfang erweitert und die Syntax vereinfacht (vgl. BURNHAM 2011:3f). Dadurch wird der zu schreibende Code kürzer und besser lesbar, was zu einem schnelleren und agileren Entwicklungsprozess führt:

> „Shorter code is easier to read, easier to write, and, perhaps most critically, easier to change."
> (BURNHAM 2011:xvii)

CoffeeScript (CS) wird vor der Auslieferung (*Deployment*) an den Nutzer in JavaScript (JS) kompiliert. In nicht-produktiven Umgebungen kann eine Kompilierung auch clientseitig im Browser erfolgen (vgl. OEVERMANN 2012:6). CS bleibt für den Endnutzer somit verborgen, es wird nur zur Entwicklung eingesetzt.

31 Kleine Textdateien, die eine Website im Cache des Browsers anlegen und wieder auslesen kann.

4.1.3.3 *Libraries*

Für Grundfunktionalitäten zur DOM-Navigation und -Selektion sowie für URL- und Integer-Verarbeitung wird auf frei verfügbare JavaScript-Libraries zurückgegriffen. Durch ihren Einsatz wird gleichzeitig die Browser-Kompatibilität erhöht. Die Libraries werden beim Aufruf als JS-Dateien eingebunden. Sie werden im Folgenden kurz dargestellt:

JQUERY
: Bei *jQuery* handelt es sich um eine der bekanntesten und am weitesten verbreiteten JS-Libraries (vgl. BONGERS/VOLLENDORF 2011:23). Oft auch selbst als „Framework" bezeichnet (vgl. BONGERS/VOLLENDORF 2011:24) erleichtert jQuery die Selektion, Manipulation und Ergänzung von Document Object Model (DOM)-Elementen und stellt darüber hinaus Funktionen zu Event-Handling, Animation und Datenverarbeitung zur Verfügung.

  jQuery wird unter der MIT-Lizenz als freie Software vertrieben (`http://jquery.com`).

JQUERY ADDRESS
: Zum Uniform Resource Locator (URL)-Handling wird das jQuery Plugin *jQuery Address* der bulgarischen Firma *asual* eingesetzt. Die Library bietet einfachen Zugriff auf URL-Parameter und Adressstämme sowie eine Überwachung von Adressänderungen.

  jQuery Address wird unter den Lizenzen GPL und MIT als freie Software vertrieben (`http://www.asual.com/jquery/address`).

JQUERY TOOLS
: Um die bisher fehlende Unterstützung von *Mozilla Firefox* auszugleichen und das Aussehen des mit HTML5 eingeführten `range`-Element Browser-übergreifend zu vereinheitlichen, wird auf die jQuery-Tools-Bibliothek von `jquerytools.org` zurückgegriffen, die das Element mit CSS-Eigenschaften und JavaScript simuliert.

  jQuery Tools wird lizenzfrei als freie Software vertrieben (`http://jquerytools.org`).

BIG INTEGER LIBRARY
: Innerhalb von JavaScript ist die Verarbeitung von Ganzzahlen (*Integers*) größer als 2^{53} (9007199254740992) nicht möglich (vgl. IEEE 754 2008). Da es sich bei den in Abschnitt 5.2.2.3 beschriebenen Konfigurationsschlüsseln aber um Zahlen in der Größenordnung von ca. 2^{80} handelt, muss eine externe Library eingebunden werden, die eine Verarbeitung ermöglicht. Die *Big Integer Library* von Leemon Baird kann Zahlen dieser Dimension verarbeiten, indem sie sie als Array aufsplittet und darin weiterverarbeitet.

  Big Integer Library ist unter `Public Domain` als freie Sofware verfügbar (`http://www.leemon.com/crypto/BigInt.html`).

4.1.3.4 *Konfiguration: XML*

Das OH3-Framework wurde so konzipiert, dass es in großem Umfang konfigurierbar ist. Die dafür benötigten Konfigurationsdateien werden im XML-Format gespeichert. Durch den Einsatz von XML kann auf bestehende Funktionalitäten wie die Datenübertragung via AJAX (Asynchronous JavaScript and XML) und DOM-Selektion mit jQuery zurückgegriffen werden.

4.1.4 *Vorgehen*

Aus der Analyse der bestehenden Funktionalitäten und neuen Anforderungen heraus ergeben sich folgende umfassende Änderungen:

1. Anpassen des Filtermechanismus. Dieser darf nicht mehr progressiv[32] vorgehen, sondern muss es zulassen, dass Elemente explizit für verschiedene Zielgruppen angezeigt werden.
2. Erweitern der Konfigurationsfähigkeit, Ergänzen eines Customizing-Moduls und Integrieren einer Lokalisierungsmöglichkeit.
3. Hinzufügen einer definierbaren *Business Logic* zum Festlegen von Bewertungs-Abhängigkeiten und zum automatischen Bewerten von Content.
4. Hinzufügen eines weiteren Bedienelementes zum Umschalten in einen Handlungsanweisungen-Modus (vgl. Abschnitt 2.5.3.2).

Darüber hinaus müssen weitere Anpassungen im Bereich Profilverwaltung, Datenaustausch und Fehler-Handling vorgenommen werden. Diese wurden in der Konzeption berücksichtigt.

4.2 KONZEPTION

4.2.1 *Konzeptionelle Beschreibung*

Grundkonzept des Frameworks ist die Kombination von Informationen aus drei verschiedenen Quellen (Benutzereinstellungen, Informationen aus der aufrufenden Software und bewerteter Content) und eine darauf aufbauende Darstellung des Contents, die für den Benutzer die größtmögliche Benutzerorientierung hat (siehe Abb. 24).

Als zentrale Daten, die von den drei Informationsquellen an das Framework geliefert werden, dienen die sog. Relevanzfaktoren bzw. ihre Repräsentation als eindeutiger Schlüssel (vgl. Abb. 16 auf Seite 43). Innerhalb des Frameworks werden die Schlüssel der Content-Elemente

32 Der im Prototyp verwendete Filtermechanismus reduzierte die Inhaltsmenge mit steigendem Wissen in einem Bereich. Damit war es z.B. nicht möglich, Inhalt nur für Experten anzuzeigen.

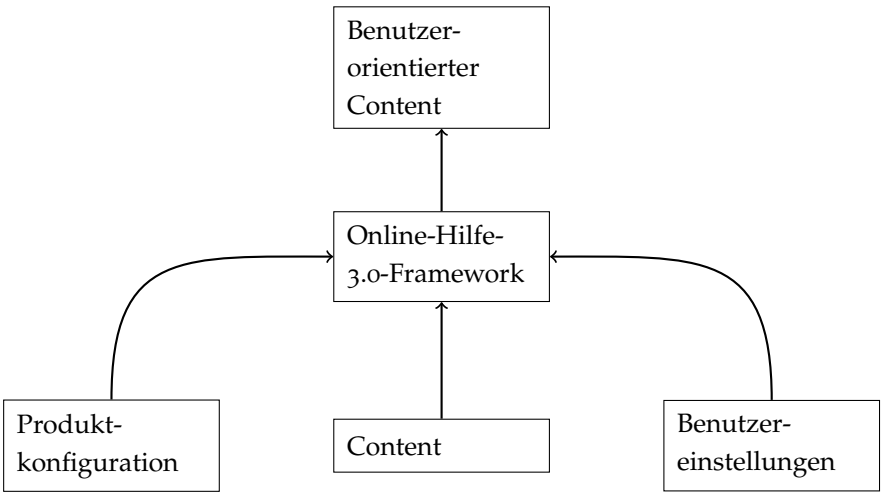


Abb. 24: OH3-Framework-Konzept

mit denen aus Produktkonfiguration und Benutzereinstellungen verglichen und je nach Übereinstimmung das Element entsprechend angezeigt oder ausgeblendet. Dieser Vorgang entspricht der Filterung aus den Vorgänger-Implementierungen.

Transformation in CMS ausgelagert

Im Gegensatz zum ursprünglichen Framework erfolgt der Hauptteil der Transformation von CMS-Daten in das OH3-Format nicht mehr *on the fly* im Browser sondern im CMS selbst. Dadurch können Kompatibilitätsprobleme mit verschiedenen Browsern ausgeschlossen werden. Eine clientseitige Transformation im Browser findet nur noch zur Vorverarbeitung der CMS-Daten zum Integrieren in das DOM der HTML-Struktur statt.

Konfiguration, Business Logic, Content-Maps und sprachspezifische Strings sind getrennt vom Code in XML-Dateien gespeichert und können angepasst und erweitert werden. Für das Customizing des Frameworks ist eine getrennte CS-Datei vorgesehen, die über die API mit dem Kernteil des Frameworks kommuniziert.

Informationen aus Stammsoftware

Das Framework empfängt Konfigurations- und Rolleninformationen von der Stammsoftware über URL-Parameter. Die aktuelle Ansicht (*PageKey*) der Software dient als eindeutiger Schlüssel (im Framework: *ContentKey*) für die anzuzeigende Content-Datei. Die Verknüpfung zwischen ContentKey und Content-Datei ist in sogenannten Maps festgehalten.

Fehlerhafte Angaben in der URL werden erkannt und ggf. durch Defaultwerte ersetzt. Bei nicht lösbaren Fehlern wird der Nutzer auf eine Fehlerseite weitergeleitet. Ebenfalls wird durch eine Prüfsumme[33] geprüft, ob die URL seit der Generierung durch die Stammsoftware verändert oder fehlerhaft übertragen wurde.

33 Ein durch eine Einweg-Hash-Funktion generierter eindeutiger Schlüssel für einen gegebenen String.

4.2.2 *Architektur*

Das Framework setzt sich aus unterschiedlichen Funktionsbereichen zusammen, die sich in Ihrer technischen Umsetzung unterscheiden (siehe Abb. 25).

Framework-Kern besteht aus Funktionsgruppen

Der Kernbereich des Frameworks besteht aus einer CS-Datei, die in wiederum in verschiedene Funktionsgruppen aufgeteilt ist. Die Funktionsgruppe die für das Customizing notwendige Funktionen enthält ist in einer separaten CS-Datei ausgelagert.

Alle weiteren Anpassungsdateien (Konfigurationen, Lokalisierungen, Logiken etc.) liegen als XML-Dateien im System. Profile werden auf dem Benutzerrechner im LocalStorage des Browser gespeichert. Die GUI besteht aus einer HTML5-Datei, deren Darstellung durch die Controller-Funktionsgruppe (Teil des Framework-Kerns) gesteuert wird.

Inhalte des CMS werden über den in Abschnitt 3.3.5.5 beschriebenen Export als XML-Dateien an das Framework übergeben werden. Daten der Stamm-Software werden beim Aufruf als URL-Parameter and das Framework übergeben (siehe Abschnitt 5.2.1).

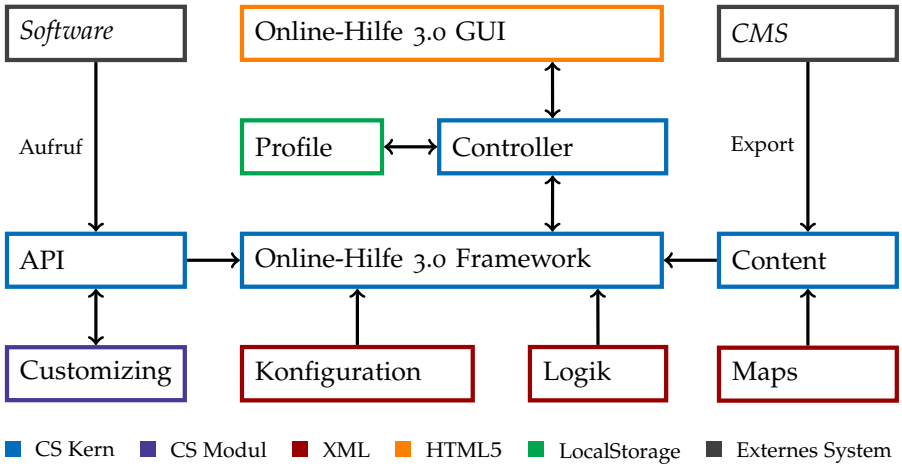


Abb. 25: OH3-Framework-Systemarchitektur

4.3 UMSETZUNG

4.3.1 *Funktionsgruppen*

4.3.1.1 *API*

Das Application Programming Interface (API) wird durch globale Objekte abgebildet, die Informationen bereitstellen oder entgegennehmen können. Variablen, Objekte und Arrays werden in CoffeeScript automatisch im jeweiligen Kontext gekapselt (vgl. BURNHAM 2011:19).

API über globale Objekte

Sollen andere Module oder externe Funktionen darauf zugreifen können, müssen sie als globales Objekt definiert werden (vgl. BURNHAM 2011:61). Im OH3-Framework wird das globale Objekt `oh3` definiert. Es entspricht dem `window`-Objekt des Browsers. Ihm werden Eigenschaften zugewiesen, die dann global verfügbar sind. Die aktuellen Benutzereinstellungen werden so z.B. im Objekt `oh3.user` verwaltet. Diese globalen Objekte können auch über die Web-Entwickler-Konsole verschiedener Browser (z.B. *Mozilla Firefox* oder *Google Chrome*) betrachtet werden (siehe Abb. 26).

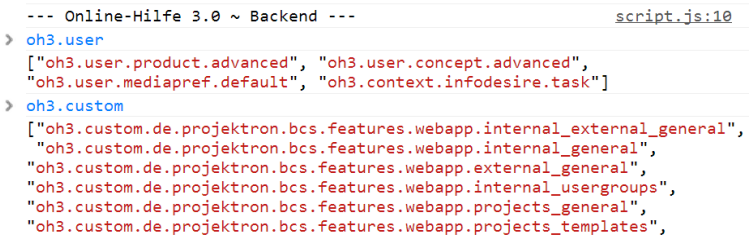


Abb. 26: Globale Objekte im OH3-Framework

Die aufrufende Software übermittelt Informationen über URL-Parameter (siehe Kapitel 5), die entsprechend in das globale Objekt `oh3._url` übernommen werden, wenn sie in der Datei `oh3.conf.params.xml` konfiguriert sind. Von dort aus können sie vom Framework oder externen Modulen weiterverarbeitet werden.

Das Customizing-Modul tauscht seine Daten über das globale Objekt `oh3.custom` mit dem Framework aus. Darin werden alle produktspezifischen Eigenschaften aus den Kategorien Produkt und Rolle verwaltet (siehe Abb. 4). Auch der Schlüssel `oh3.contentKey` wird hier definiert. Er wird vom Framework für das Auffinden des passenden Contents in der Maps-Datei benötigt.

4.3.1.2 *Customizing*

Separate Datei für Anpassungen

Das Modul Customizing stellt ein vom eigentlichen Framework gekapseltes Funktionsmodul dar, das alle systemspezifischen Besonderheiten abhandelt. In der Implementierung für Projektron BCS ist das z.B. die Dekodierung von Konfigurationsinformationen (siehe Abschnitt 5.2.2.3). Das Customizing-Modul wird in der Datei `customizing.coffee` (bzw. `customizing.js`) gepflegt und muss mindestens die Funktion `customizing` enthalten, die vom Framework zum entsprechenden Zeitpunkt initialisiert wird. Diese Funktion kann dann innerhalb des Customizing-Moduls beliebige andere Funktionen aufrufen. Der Datenaustausch mit dem Framework erfolgt über die API. Das Customizing-Modul hat Zugriff auf eine begrenzte Zahl an Systemfunktionen des Frameworks (z.B. Warnmeldungen ausgeben, Fehler triggern, Variablen abfragen).

```
makeUserProductSetting = () ->
    oh3.custom = new Array
    oh3.custom = oh3._bcsconf.concat oh3._bcslic, oh3._bcsedition
        , oh3._bcsversion
    oh3.contentKey = oh3._url.pkey
```

Code 10: Auszug aus customizing.coffee

4.3.1.3 *Konfiguration*

Die Gesamtkonfiguration des Systems ist in mehreren modularisierten XML-Dateien definiert (siehe Tabelle 6).

DATEINAME	FUNKTION
`oh3.setup.xml`	Referenzierung aller Konfigurationsmodule und Objektzuordnungen
`oh3.conf.controls.xml`	Steuerelemente und Referenzierung von Reglerwerten zu Klassifikatoren
`oh3.conf.params.xml`	Registrierung von URL-Parametern
`oh3.conf.vars.xml`	Einstellungen und Default-Werte

Tabelle 6: Standard-Konfigurationsdateien OH3-Framework

Zentrale Setup-Datei

Durch einen Eintrag in der Datei `oh3.setup.xml` werden Konfigurationsmodule im Framework registriert (siehe Code 11). Auch externe Gültigkeitslisten, Logik- und Map-Dateien werden über diese Datei referenziert. Jedes Konfigurationsmodul wird vom Framework zunächst eingelesen und in ein globales Objekt geschrieben. Die darin enthaltenen Informationen sind dadurch über die API zugänglich.

```
...
<Config>
  <File>conf/oh3.maps.xml</File>
  <Type>array</Type>
  <Object>_maps</Object>
  <Key>Map</Key>
</Config>
<Config>
  <File>conf/oh3.conf.params.xml</File>
  <Type>list</Type>
  <Object>_params</Object>
  <Key>Param</Key>
</Config>
...
```

Code 11: Referenzierung in oh3.setup.xml

```
makeConf = (file, type, obj, key) ->
  _xml = requestFile file
  oh3[obj] = $(_xml)
  switch type
    when "list" then buildList obj, key
    when "array" then buildArray obj, key
```

Code 12: Verarbeitung von Konfigurationen

Die Konfigurationsdateien unterscheiden sich in zwei Typen, die unterschiedlich verarbeitet werden: Listen (Typ: `list`) und Datensätze (Typ: `array`). Bei Listen handelt es sich um einfache Aufzählungen, bei Datensätzen um die Gruppierung mehrerer Angaben in zusammengehörige Datensätze.

Lokalisierung konfigurierbar

Ein weiterer Bestandteil der Systemkonfiguration ist die Lokalisierung der grafischen Benutzeroberfläche (GUI). Innerhalb des Frameworks werden Oberflächenelemente nur über sog. *Keys* (eindeutige Bezeichner) beschriftet, denen bei der Ausgabe ein sprachspezifisches *Label* zugeordnet wird. Die Referenzierung von Labels zu Keys wird über Lokalisierungsdateien vorgenommen (siehe Code 13).

```
...
<Tuple>
  <Key>oh3.msg.content.none</Key>
  <Label>Es wurde kein passender Inhalt gefunden.\n\nSie werden
      stattdessen auf die Startseite umgeleitet.</Label>
</Tuple>
```

Code 13: Lokalisierung in oh3.l12n.de.xml

4.3.1.4 *Logik*

Im Logik-Teil des Frameworks werden Abhängigkeiten zwischen Benutzereinstellungen, Produktkonfigurationen und Content-Arten geprüft und umgesetzt. Als Basis dafür dienen Konfigurationsdateien, in denen die Abhängigkeiten in einem *Wenn-Dann*-Muster definiert sind. Diese sind in drei Module aufgeteilt (siehe Tabelle 7).

Es können beliebig viele Abhängigkeiten definiert werden. Diese werden nacheinander von den jeweiligen Funktionen verarbeitet.

```
<Rule>
  <ifHasProperty>oh3.custom.de.projektron.bcs.licence.
      ProgramManager</ifHasProperty>
  <setControlTo>oh3.user.concept.expert</setControlTo>
</Rule>
```

Code 14: Regeldefinition in Logik-Datei

DATEINAME	FUNKTION
`oh3.logic.content.xml`	Zuordnungen von automatischen Bewertungen für best. Elementmengen
`oh3.logic.startup.xml`	Abhängigkeiten, die beim Aufruf getestet werden, z.B. Voreinstellungen für bestimmte Rollentypen
`oh3.logic.dynamic.xml`	Abhängigkeiten, die dynamisch geprüft werden, z.B. sich beeinflussende Reglerstellungen

Tabelle 7: Logik-Konfigurationsdateien OH3-Framework

4.3.1.5 *Maps*

In Maps werden die Beziehungen von Content-Dateien (die aus dem CMS stammen) untereinander festgehalten und die Metadaten *Titel*, *ContentKey* und *Sprache* zugeordnet.

Die Maps-Datei `oh3.maps.xml` kann manuell erstellt, ggf. aber auch vom CMS automatisch generiert werden[34]. In der Implementierung für Projektron BCS entspricht der *ContentKey* in den meisten Fällen dem PageKey der Software. Kann ein Hilfeinhalt nicht eindeutig einer Ansicht zugeordnet werden, kann dem Content in der Map ein generischer Schlüssel zugeordnet werden.

Der verknüpfte Content wird an der Oberfläche im Menü *Verwandte Themen* dargestellt und wird innerhalb der Maps-Datei mit dem Element `related` ausgezeichnet.

```
1  ...
2  <Map>
3     <Key>projectbrowser.display.preparation</Key>
4     <Lang>de</Lang>
5     <Source>content/Projektvorbereitung, 1, de_DE.xml</Source>
6     <Title>Projektvorbereitung</Title>
7     <Related>bcs.docs.common.introduction.views</Related>
8     <Related>projectdetail.edit.teamplanning</Related>
9  </Map>
10 ...
```

Code 15: Beispiel für einen Maps-Eintrag

34 Diese Möglichkeit wurde im Rahmen der Arbeit nicht verfolgt, ist aber für eine spätere Anwendung denkbar

4.3.1.6 *Content*

In der Funktionsgruppe *Content* sind alle Funktionen und Methoden zusammengefasst, die mit der Verarbeitung der Inhalte vom Einlesen der Quelldatei bis zum Filtern der Ausgabe in Zusammenhang stehen. Dabei durchläuft der Content mehrere Phasen:

MAPPING
: Innerhalb der Funktion `buildContent` wird der übergebene *ContentKey* durch eine Referenzierung in der `oh3.maps.xml` ausgewertet, die zugehörige Content-XML-Datei eingelesen und Verknüpfungen zu verwandten Themen werden hergestellt.

ANREICHERUNG
: Der eingelesene Content wird entsprechend der in `oh3.logic.content.xml` hinterlegten Regeln von der Funktion `enrichContent` mit zusätzlichen Relevanzfaktoren angereichert. Die Regelformulierung orientiert sich an der CSS-Selektorlogik.

ANALYSE
: Die Funktion `analyzeContent` sucht nach zugeteilten OH3-Relevanzfaktoren in `class`-Attributen der Content-Elemente. Sind einem Element Relevanzfaktoren zugeordnet, werden diese in Kategorie aufgeteilt als jQuery-data-Objekt am jeweiligen DOM-Element hinterlegt.[35] Intern wird – entsprechend den Ergebnissen aus Kapitel 2 – zwischen den Kategorien *user* (Benutzereigenschaften), *custom* (Produkt- und Rolleneingenschaften) und *context* (Kontexteigenschaften) unterschieden.

FILTERUNG
: Die eigentliche Filterung des Contents erfolgt durch die Funktion `renderContent` und nach Kategorien getrennt. Diese Trennung ist nötig, um verschiedene *Filterlogiken* und *Konfliktprioritäten* anwenden zu können (s.u.). Die Filterfunktion ermittelt die Gesamtgültigkeit eines Elements und blendet es entsprechend ein oder aus. Ob ein Relevanzfaktor gültig ist, wird durch den Abgleich der Relevanzfaktoren, die einem Element zugeteilt sind, mit den aktuellen Benutzereinstellungen bzw. der übergebenen Produktkonfiguration ermittelt.

```
1 $("*[class*='oh3.#{part}']").each ->
2   validities = $(this).data("oh3." + part)
3   elementValid = false
4   for i in [0...validities.length]
5     if validities[i] in oh3.user then elementValid = true
6     if elementValid then show this else hide this
```

Code 16: Ausschnitt aus der Filterfunktion

35 jQuery nutzt hierbei ein eigenes Objekt, das als interner Cache dient und auch manuell über `$.cache` angesteuert werden kann.

Ob ein Content-Element angezeigt wird, entscheidet sich während des Filtervorgangs und ist von zwei Faktoren abhängig:

FILTERLOGIK
: Die Filterung folgt grundsätzlich einer einfachen zweiwertigen Logik (*gültig* oder *nicht gültig*) für einzelne Relevanzfaktoren. Ist einem Element mehr als ein Relevanzfaktor zugeordnet, muss eine Gesamtgültigkeit für das Element ermittelt werden. Die Ermittlung der Gesamtgültigkeit unterscheidet sich für die internen Kategorien *user/context* und *custom*. Bei Produkteigenschaften (*custom*) müssen *alle* Eigenschaften der Kategorie für ein Element zutreffen, damit es als gültig eingestuft wird, bei Benutzereinstellungen (*user/custom*) muss nur *eine* der Eigenschaften zutreffen.

KONFLIKTPRIORITÄT
: Sind einem Element mehrere Relevanzfaktoren zugeordnet, so kann es Konflikte in der Gültigkeitsauswertung zwischen Kategorien geben. Innerhalb des Frameworks entscheidet die Reihenfolge des Aufrufs über die priorisierte Kategorie: Die Filterung nach Produkteigenschaften wird einmalig beim Aufruf des Frameworks vorgenommen; die Filterung nach Benutzereigenschaften beim Verändern der Schieberegler; die Filterung nach Kontext beim Betätigen des Zwei-Wege-Schalters.

4.3.1.7 *Profile*

Lokale Profilspeicherung

Um Einstellungen eines Benutzers (Schieberegler, Zwei-Wege-Schalter) beim nächsten Aufruf der Online-Hilfe 3.0 wiederherstellen zu können, werden im lokalen Speicher des Browsers Profile abgelegt. Als Schlüssel für den Zugriff auf Profildaten wird die übertragene User-ID aus der Stammsoftware verwendet. Ruft ein Benutzer die Hilfe zum ersten Mal auf, wird ein neues Benutzerprofil angelegt, das mit jeder Einstellungsänderung aktualisiert wird. Ist beim Aufruf der Hilfe ein vorhandenes Profil für die jeweilige User-ID gespeichert, wird dieses wiederhergestellt.

Key	Value
123456	oh3.user.product.expert,oh3.user.concept.beginner,oh3.user.med…
333333	oh3.user.product.expert,oh3.user.concept.advanced,oh3.user.me…
654321	oh3.user.product.beginner,oh3.user.concept.advanced,oh3.user.…

Abb. 27: OH3-Benutzerprofile im LocalStorage

Profile werden nur lokal auf dem Rechner des Anwenders angelegt und nicht an den Server übertragen. Der Benutzer hat über das Optionen-Menü die Möglichkeit, sein Profil zu löschen.

4.3.1.8 *Controller*

Schnittstelle zum Benutzer

Die Controller-Funktionsgruppe dient als Schnittstelle zwischen GUI und Framework-Kern. Sie wertet Benutzeraktionen aus, sammelt und verwaltet Einstellungswerte und steuert Kontrollelemente. Zentrale Schnittstellen zwischen Benutzer und System stellt hierbei die Funktion `handleUserInput` dar, die bei jeder Veränderung eines Kontrollelements ausgeführt wird und entsprechend der Eingabeart weitere Aktionen veranlasst.

4.3.2 *Prozesse*

4.3.2.1 *Contentverarbeitung*

Der vom Redakteur ursprünglich im CMS erfasste Content durchläuft bis zur endgültigen Darstellung innerhalb der OH3-Oberfläche diverse Zwischenschritte, die schematisch im Abb. 28 aufgeführt sind.

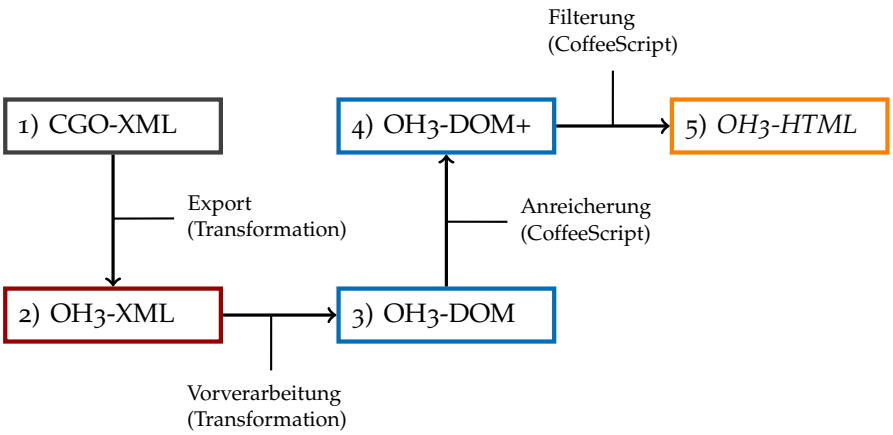


Abb. 28: OH3-Contentverarbeitung

1. Der Content wird vom Redakteur im CMS-Standardformat erfasst. Bei CGO ist dies der IO-Typ *Documentation (User Documentation)* bzw. XML nach Schema der `manual.dtd`.

   Beim OH3-Export aus CGO (siehe Abschnitt 3.3.5.5) wird das im CMS vorliegende XML in das OH3-Format transformiert.

2. Das aus dem CMS ausgespielte OH3-Format bzw. XML nach Schema der `oh3-export.dtd` wird dem Framework über einen Map-Eintrag bekannt gemacht (siehe Abschnitt 4.3.1.5).

   Beim Aufruf des entsprechenden Contents wird die OH3-XML-Datei durch eine Transformation vorverarbeitet und in das DOM der Framework-Oberfläche integriert.

3. Der Content liegt im DOM vor und kann durch das Framework und dessen Funktionsgruppen selektiert und manipuliert werden.

   Basierend auf ausgelagerten Regeln wird der Content mit weiteren Relevanzfaktoren angereichert. Dabei werden einzelnen Elementen zusätzliche Klassen-Werte zugewiesen.

4. Der angereicherte Content kann nun durch das Framework analysiert und gefiltert werden. Dabei werden basierend auf den Benutzereinstellungen durch CSS-Elemente ein- oder ausgeblendet. Ausgeblendete Elemente verbleiben weiterhin im DOM.

4.3.2.2 *Voreinstellungskonfiguration*

Um dem Benutzer beim Aufrufen der Hilfe bereits eine optimale Voreinstellung anbieten zu können, werden verschiedene Informationen ausgewertet.

Dreistufiger Prozess

Besucht ein Anwender die Online-Hilfe, werden in einem dreistufigen Prozess, basierend auf konfigurierbaren Regeln, die Information, die über Ihn vorliegen, ausgewertet. Dabei wird zunächst geprüft, ob für Eigenschaften, die dem Nutzer zugeteilt sind, Regeln vorliegen.[36] Trifft eine Regel zu, wird die hinterlegte Einstellung vorgenommen.

Im zweiten Schritt werden die Abhängigkeiten der Benutzereinstellungen untereinander geprüft. Wurde durch die erste Voreinstellungs-Stufe eine Einstellung verändert, von der eine andere Einstellung abhängig ist (siehe Abb. 6), wird diese Abhängigkeit ausgeführt.

Im letzten Schritt wird der Profilstatus des Benutzers geprüft. Hat der Nutzer die Online-Hilfe bereits schon einmal benutzt, werden die Einstellungen seiner letzten Sitzung wiederhergestellt.

Abb. 29: OH3-Voreinstellungskonfiguration

4.3.3 *Oberfläche*

4.3.3.1 *Aufbau*

Die grafische Benutzeroberfläche des Frameworks orientiert sich an seinem Vorgänger (siehe Abb. 31). Zentrale Bedienelemente sind drei

36 Das können z.B: Voreinstellungen für bestimmte Rollen (anhand der Lizenz) oder auch für spezifische Nutzer (anhand der User ID) sein.

Schieberegler sowie ein Zwei-Wege-Schalter (b), um die vom Benutzer selbst bestimmten Profileigenschaften (siehe Tabelle 2) einzustellen. Durch eine Schaltfläche mit dem Produktlogo der aufrufenden Software hat der Anwender die Möglichkeit, schnell zur Software zurückzukehren; eine zweite Schaltfläche öffnet das Optionen-Menü des Frameworks (a). Im Optionen-Menü des Frameworks hat der Anwender die Möglichkeit, über Schaltflächen eine Übersicht aufzurufen oder seine Profilinformationen zu löschen (c). Mit Hilfe eines Auswahlmenüs kann der Anwender direkt zu verwandten Themen springen (d).

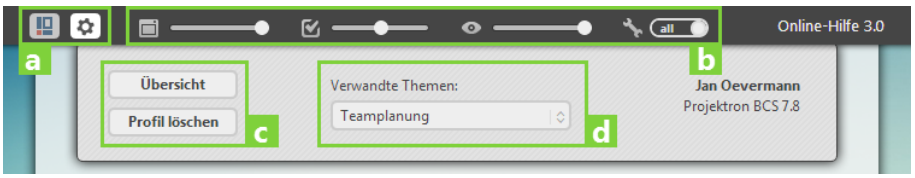


Abb. 30: Optionen-Menü des OH3-Frameworks

Der Hauptbereich des Frameworks dient zur Inhaltsanzeige und ist in einer lesefreundlichen Breite in der horizontalen Mitte des Bildschirms platziert (siehe Abb. 31). Die Gestaltung des Inhaltsbereichs und seiner Elemente kann getrennt von der Gestaltung des Frameworks im Stylesheet `content.css` vorgenommen werden.

4.3.3.2 *Lokalisierung*

Die Oberfläche den Frameworks kann lokalisiert werden. Alle sprachspezifischen Bezeichnungen und Meldungen sind in XML-Lokalisierungsdateien ausgelagert und werden innerhalb der Framework-Programmgruppen über eindeutige Keys und die globale Funktion `oh3.getLabel` referenziert. Lokalisierungsdateien müssen nach dem Muster `oh3.l12n.`*Sprachkürzel*`.xml` benannt werden.

4.3.3.3 *Personalisierung*

Zur firmen- oder benutzerspezifischen Anpassung der Oberfläche können zusätzlich Stylesheets hinterlegt werden, die beim Aufruf durch die Software über die Schnittstelle bestimmt werden (siehe Abschnitt 5.2.2). Durch diese kann z.B. das Hintergrundbild oder die Farbgebung der Kontrollleiste der Software-Oberfläche angepasst werden. Die Stylesheets sind unter `/style/user` hinterlegt.

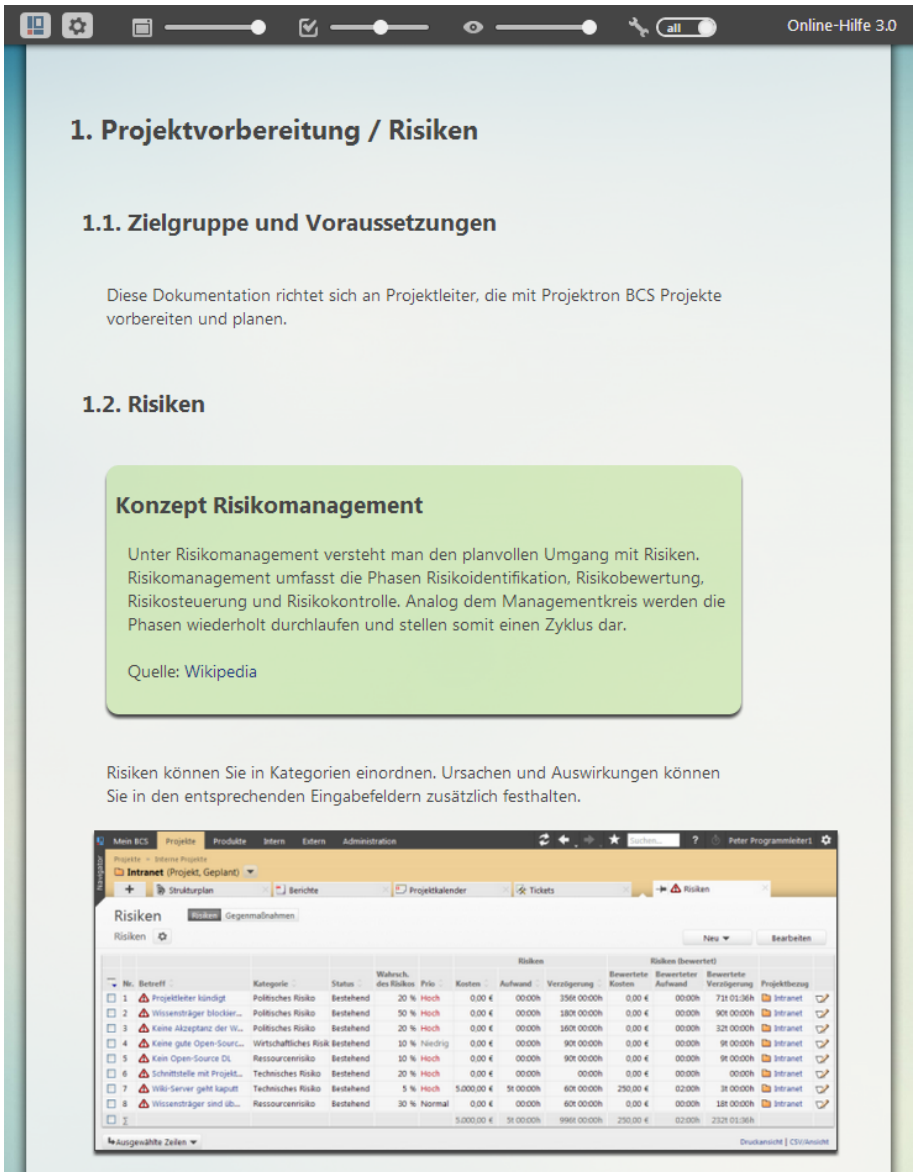


Abb. 31: Benutzeroberfläche des OH3-Frameworks

5

ANBINDUNG ALS HILFESYSTEM

In diesem Kapitel wird die Online-Hilfe aus Kapitel 4 an die Software Projektron BCS angebunden. Dafür werden systemspezifische Besonderheiten herausgearbeitet, eine Schnittstelle wird definiert und eine prototypische Implementierung vorgenommen.

5.1 PROJEKTRON BCS

5.1.1 *Einführung*

Webbasierte Projektmanagementsoftware

Projektron BCS ist eine webbasierte Projektmanagement-Software, die vom Berliner Software-Hersteller Projektron[37] entwickelt und vertrieben wird. BCS steht für Business Coordination Software, und soll den Ansatz der Software beschreiben, neben der Kernkompetenz Projektmanagement den gesamten Unternehmensprozess abzudecken.

Projektron beschäftigt über 70 Mitarbeiter an 6 Standorten und hat europaweit ca. 450 Kunden. Firmen, die die Software einsetzen, sind oft im mittelständischen IT-Bereich angesiedelt, aber auch Firmen aus anderen Branchen und Großkonzerne zählen zum Benutzerkreis (vgl. Projektron GmbH 2013c). Seit ca. einem Jahr gibt es mit Projektron BCS.start eine eigene Produktlinie für Kleinunternehmen oder einzelne Abteilungen (vgl. Projektron GmbH 2012b).

5.1.2 *Einsatzgebiet und Benutzergruppen*

Die Software kann für einzelne Organisationsbereiche, wie etwas das Projekt- oder Vertragsmanagement eingesetzt werden, aber auch tiefgreifend in die Geschäftsprozesse eingebunden werden. Der Funktionsumfang enthält neben den Kernfunktionen (Multi-)Projektmanagement, -controlling und Ressourcenverwaltung auch angrenzende Bereiche wie Customer Relationship Management (CRM), Angebots- und Rechnungsstellung sowie Qualitätsmanagement.

Durch die große Anzahl an Einsatzgebieten, die sich auch je nach Branche weiter unterscheiden können, kommt eine große Anzahl unterschiedlicher Benutzergruppen mit Projektron BCS in Berührung. Diese heterogene Zielgruppenstruktur führt auch zu einer komplexe-

37 Projektron GmbH, Berlin (www.projektron.de)

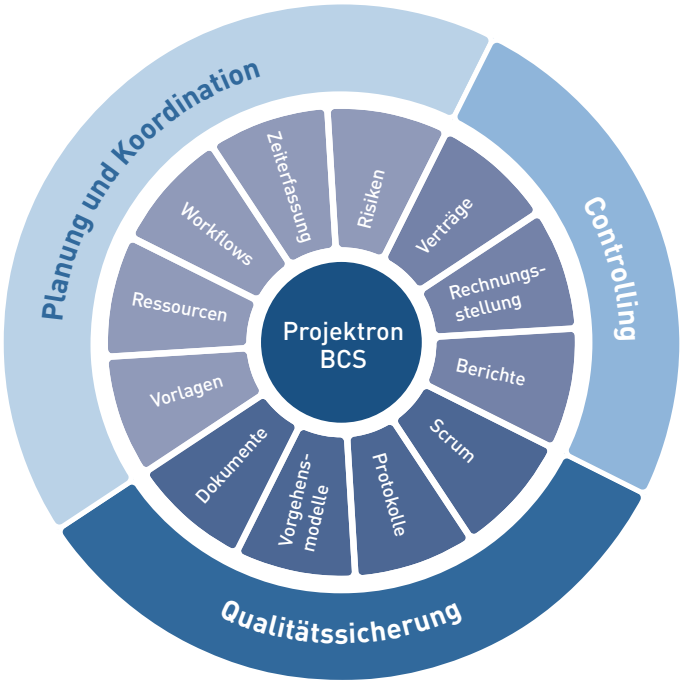


Abb. 32: Funktionsbereiche von Projektron BCS (Quelle: Projektron GmbH)

ren Dokumentation, da im Idealfall alle Kombinationen aus Benutzer und System abgedeckt sind.

5.1.3 *Besonderheiten*

Teilt man die systemspezifischen Eigenschaften von Projektron BCS den Relevanzfaktoren aus Abschnitt 2.5.3.2 zu, kommt man zu folgendem Ergebnis:

PRODUKTLINIE
: Die Software wird in zwei verschiedenen Produktlinien angeboten: Projektron BCS und Projektron BCS.start. Der Hersteller nennt die parallel angebotenen Produktlinien *Editionen* (vgl. PROJEKTRON GMBH 2013b:74). BCS.start entspricht einer Sonderkonfiguration des Systems, bei der bestimmte Module (*Bausteine*) der Software nicht aktivierbar sind. Damit wird der Funktionsumfang der Software eingeschränkt, das Basissystem bleibt hingegen das gleiche (vgl. SAUTER 2013).

VERSION
: Die Versionsnummer von Kundenreleases gliedert sich in Major-Version (derzeit 7) und gerade Minor-Version (derzeit 6) durch Punkt getrennt. Eine weitere Unterteilung erfolgt für Zwischen-

releases bzw. Revisionen, die z.B. Fehlerbeseitigungen enthalten. Die derzeit aktuelle Version von Projektron BCS ist:

$$\underbrace{7}_{\text{Major}} . \underbrace{6}_{\text{Minor}} . \underbrace{8}_{\text{Revision}}$$

Umfassendere Funktionsänderungen fließen nur in Major- oder Minor-Versionen ein, weshalb die Revision nicht als relevant für den Benutzer eingestuft wird.

KONFIGURATION

Die Gesamtkonfiguration der Software wird durch viele Faktoren bestimmt; eine direkte Auswirkung auf den Nutzer hat aber vorwiegend die Kombination aus aktivierten Modulen (vgl. HEINZE 2013). Diese Module werden bei Projektron BCS *Bausteine* genannt und können sich vom Einblenden verschiedener Ansichten[38] (z.B. alternative Buchungsmaske) bis hin zum Zuschalten von Grundfunktionalitäten (z.B. Mehrwährungsfähigkeit) auswirken.

ROLLENTYP

Der Rollentyp wird bei Projektron BCS zu großen Teilen[39] durch die Benutzerlizenz bestimmt (vgl. HEINZE 2013). Diese Lizenzen können Benutzern zugeteilt werden, um ihnen weitreichendere Recht im System zu geben. Sie entsprechen dem Anwendertyp (z.B. Zeiterfasser, Projektmanager, Controller, etc.), der dem Mitarbeiter von der Firma, die die Software einsetzt, zugeteilt wird. Einige Sonderlizenzen wirken sich aber auf aktivierbare Schnittstellen (z.B. *ExchangeSync-Adapter*) oder einstellbare Oberflächensprachen aus (z.B. Lizenz *Italian* für Italienisch).

ANSICHT

Die Stelle, an der man die Hilfe innerhalb der Software aufruft (*kontextsensitive Online-Hilfe*), kann in Projektron BCS über den sog. `PageKey` abgefragt werden. Dabei handelt es sich um einen eindeutigen Bezeichner-String der gerade geöffneten Ansicht. Anhand des `PageKeys` kann auch der Modus der Ansicht (Ansehen/Bearbeiten) abgelesen werden.

```
1 <meta content="projectdetail.edit.teamplanning" name="
      PageKey">
```

Code 17: Beispiel: PageKey in Projektron BCS

38 Ansichten sind in Projektron BCS

39 Weitere Einflussfaktoren, wie einzeln vergebene Aktions- oder Sichtrechte oder unternehmensspezifisch definierte Rollen haben eine zu hohe Divergenz für den praktischen Einsatz.

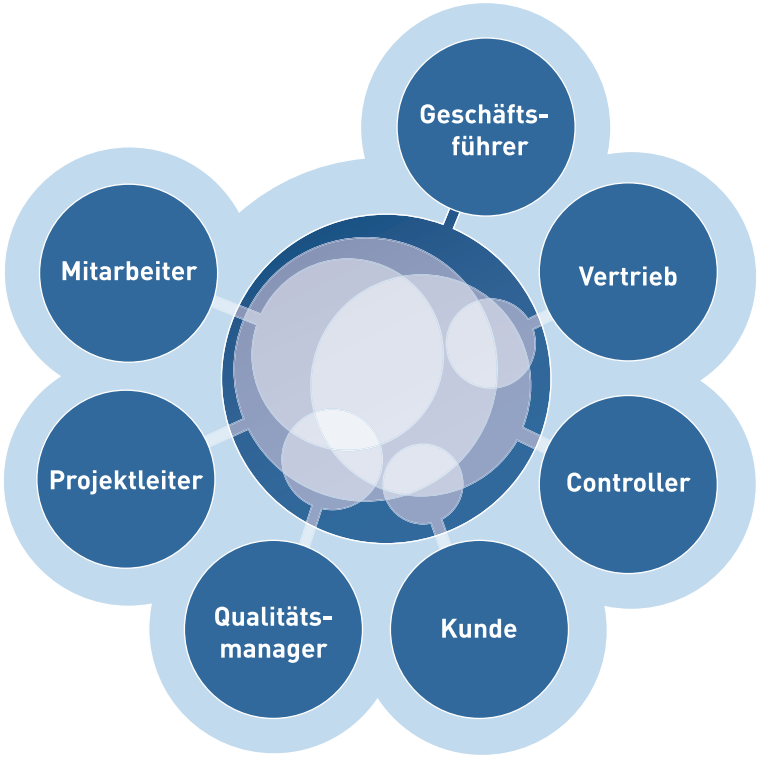


Abb. 33: Rechteverteilung in Projektron BCS (Quelle: Projektron GmbH)

5.2 SCHNITTSTELLE

5.2.1 *Technische Umsetzung*

URL als Schnittstelle

Das in Kapitel 4 vorbereitete Framework zur Darstellung und Filterung des benutzerorientierten Contents kann Daten zu Benutzereigenschaften und Systemkonfiguration in Key-Value-Paaren[40] über den *query string* einer URL entgegennehmen und verarbeiten.

Der query string wird einer URL nach der Pfad- bzw. Dateiangabe durch Fragezeichen (?) getrennt angehängt (vgl. IETF RFC 1738 1994):

$$\underbrace{\text{http}}_{\text{Schema}} \text{://} \underbrace{\text{localhost}}_{\text{Host}} \text{/} \underbrace{\text{oh3/help.html}}_{\text{Pfad}} \text{?} \underbrace{\text{key1=value1\&key2=value2}}_{\text{Query}}$$

Wird das Framework mit einem angehängten query string aufgerufen, werden die darin enthaltenen Key-Value-Paare ausgewertet, falls sie dem Framework bekannt sind. Um Daten zu übertragen, muss die aufrufende Software (in diesem Fall: Projektron BCS) der URL, die auf die Hilfe verlinkt, entsprechende Key-Value-Paare anhängen.

40 Key-Value-Paare gliedern sich in *Schlüssel* und *Wert*, durch Gleichheitszeichen (=) getrennt. Mehrere Paare werden durch *Kaufmännisches Und* (&) voneinander abgegrenzt.

5.2.1.1 *Einschränkungen*

Obwohl die Spezifikation des Hypertext Transfer Protocols (HTTP) keine Begrenzung für die Länge von URLs nennt (vgl. IETF RFC 2616 1999), gibt es *de facto* Einschränkungen durch den verwendeten Browsertyp (vgl. MICROSOFT KNOWLEDGEBASE 2007):

> „Microsoft Internet Explorer has a maximum Uniform Resource Locator (URL) length of 2,083 characters."

Durch diese Begrenzung muss darauf geachtet werden, dass umfangreiche Informationen, wie die Systemkonfiguration so kompakt wie möglich zusammengefasst werden, und dass die maximale Länge der URL von ca. 2.000 Zeichen nicht überschritten wird.

5.2.2 *Spezifikation*

Um einen standardisierten Datenaustausch zwischen COSIMA go!, OH3-Framework und Projektron BCS zu gewährleisten, muss eine Schnittstelle spezifiziert werden, die die Relevanzfaktoren des OH3-Konzepts unter Berücksichtigung der Besonderheiten der eingesetzten Systeme abdeckt.

Informationen als Key-Value-Paare

Hierzu müssen innerhalb der Schnittstelle eindeutige Schlüssel und mögliche Werte für jedes Key-Value-Paar festgelegt werden. Durch die Einschränkungen der Schnittstellenart (URL-Länge) sollten alle Festlegungen so kompakt wie möglich gestaltet werden, ohne jedoch einen ausreichend großen Spielraum für langfristige Änderungen einzuschränken.

5.2.2.1 *Produktlinie*

Schlüssel: `ed` (für *edition*)

Die Produktlinie lässt sich über die gängigen Kurzbezeichnungen der Editionen übertragen (siehe Tabelle 8).

PRODUKTLINIE	KÜRZEL	QUERY
Projektron BCS	bcs	`?ed=bcs`
Projektron BCS.start	start	`?ed=start`

Tabelle 8: Schnittstelle Produktlinie

5.2.2.2 *Version*

Schlüssel: `v` (für *version*)

Die relevanten Versionsnummern-Bestandteile, Major- und Minor-Version, lassen sich in ihrer natürlichen Form übertragen (siehe Tabelle 9).

VERSION	KÜRZEL	QUERY
Projektron BCS 7.4	7.4	`?v=7.4`
Projektron BCS 7.6	7.6	`?v=7.6`

Tabelle 9: Schnittstelle Version

5.2.2.3 *Konfiguration*

Schlüssel: `conf` (für *configuration*)

Wie in Abschnitt 5.1.3 beschrieben, wird der für den Benutzer relevante Teil der Systemkonfiguration über die Kombination aktivierter Bausteine abgebildet. Momentan gibt es 79 verschiedene Bausteine.

Innerhalb von Projektron BCS werden die einzelnen Bausteine mit *Keys*[41] referenziert. Konfigurationen werden in XML-Dateien nach dem folgenden Muster gespeichert:

```
1 <Features>
2   ...
3   <Feature Name="de.projektron.bcs.features.webapp.
        projects_templates" State="on" />
4   <Feature Name="de.projektron.bcs.features.webapp.
        projects_topdown" State="off" />
5   ...
6 </Features>
```

Code 18: Baustein-Konfiguration in Projektron BCS

Den Baustein-Key in ähnlicher Weise über die URL-Schnittstelle zu übertragen ist nicht möglich, da sonst die Längenbegrenzung für URLs zu schnell überschritten wird. Selbst beim Entfernen des Stammanteils[42] jedes Keys, verbleibt eine Länge von durchschnittlich 15 Zeichen pro Baustein, was multipliziert mit der momentanen Anzahl von 79 Bausteinen schon eine query-Länge von 1.264 Zeichen ergibt, wenn jeweils ein Trennzeichen verwendet wird.

Besonders kompakte Übertragung

Um eine solche Auflistung so kompakt wie möglich zu übertragen, kann man sich den booleschen Charakter der Konfiguration zu Nutze machen: Bausteine können immer nur aktiviert oder deaktiviert sein. Dadurch lässt sich der Aktivierungsstatus eines Bausteins durch eine Ziffer im Binärsystem darstellen (`1` = aktiviert, `0` = deaktiviert).

Betrachtet man die Konfiguration als Liste von Bausteinen, denen je nach Status eine `1` oder `0` zugeordnet wird, so kann jede Gesamtkonfiguration eindeutig durch eine Kette von `1`en und `0`en, also einer Bi-

41 Keys (=Schlüssel) werden in Projektron BCS in englischer Sprache nach einem intern gepflegten Schema festgelegt.

42 Der Teil, der sich innerhalb der Baustein-Keys nicht unterscheidet (`de.projektron.bcs.features.webapp.`).

närzahl dargestellt werden. Bei einer Kombination aus 79 Bausteinen wäre diese Zahl also 79 Zeichen lang.

Um in der Übertragung weitere Zeichen zu sparen, kann diese Binärzahl (Basis: 2) in ein Zahlensystem mit der Basis 36 umgewandelt werden. In einem solchen System werden Ziffern sowohl durch die Zahlen 0-9 (insg. 10) und Buchstaben (insg. 26) dargestellt. Schon bei einer Konfiguration von 10 Bausteinen lassen sich so 80% Zeichen einsparen:

$$1011110001_{(2)} = kx_{(36)}$$

Eine Konfiguration von 79 Bausteinen lässt sich in durchschnittlich 16 Zeichen übertragen:

$$\text{help.html?conf} = \underbrace{28ey0a1nchj4ow04}_{1101000100100100 1\ldots}$$

Vorteile dieser Methode sind die gute Komprimierung und die problemlose Erweiterung auf weitere Bausteine. Nachteile ergeben sich durch die Fehleranfälligkeit bei Nichteinhaltung der vorgegebenen Reihenfolge innerhalb der Binärzahl. Hier muss bei der Anwendung eine zentral gepflegte Liste verwaltet werden, auf die beide Systeme zugreifen können.

Im CMS wird vom Redakteur einer Information die Abhängigkeit von einem Baustein über eine Maske zugewiesen (siehe Abschnitt 3.3.2). Im Hintergrund wird diese Abhängigkeit als Key am Element-Attribut `y.validity.text` hinterlegt.

```
1 <oh3-inline y.id="ID_5fc4d7323f5536ce0a0a1e1000da1889" y.validity
      .ids="56065779de3d94d40a0a1e10010fd2df" y.validity.allowed="
      true" y.validity.mode="positive" y.validity.text="oh3.custom.
      de.projektron.bcs.features.webapp.bookmarks">
2   Dieser Teil wird nur angezeigt, wenn der Baustein Lesezeichen
        aktiviert ist.
3 </oh3-inline>
```

Code 19: Gültigkeiten als XML-Attribute

Rückreferenzierung in Framework

Deshalb muss auf Seite des OH3-Frameworks eine Rückreferenzierung erfolgen. Zunächst muss die Zahl mit der Basis 36 wieder in eine Binärzahl umgewandelt werden, anschließend die jeweils n-te Stelle der Binärzahl der n-ten Stelle der gemeinsamen Bausteinliste zugeordnet werden. Diese Liste kann zum Beispiel über das Internet in Form einer XML-Datei zur Verfügung gestellt werden.

Zwar gibt es kompaktere Speicherformen[43] für eine solche Dateiart, aber da sowohl Projektron BCS als auch das OH3-Framework Schnitt-

43 Alternative Möglichkeiten sind die Übertragung als JSON-Objekt über einen AJAX-Request oder eine Key-Value-Textdatei auf dem Server

stellen zum Verarbeiten von XML-Dateien haben, bietet sich dieses Format an.

Um mit der Verarbeitungslogik des Frameworks kompatibel zu sein (vgl. OEVERMANN 2012:9), bietet sich eine einfache XML-Listenform an (siehe Code 20). Diese Liste lässt sich durch eine einfache XSL-Transformation aus den im System hinterlegten Konfigurationsdateien erstellen.

```
1 <Features>
2 ....
3   <Feature>de.projektron.bcs.features.webapp.todos</Feature>
4 ...
5   <Feature>de.projektron.bcs.features.webapp.faqs</Feature>
6 ....
7   <Feature>de.projektron.bcs.features.webapp.bookmarks</Feature>
8 ....
9 </Features>
```

Code 20: Referenz-Bausteinliste für Framework

5.2.2.4 *Rollentyp*

Schlüssel: `lic` (für *licence*)

Rechtemanagement durch Lizenzen

Lizenzen werden innerhalb von Projektron BCS mit englischen Bezeichnungen verwaltet. Derzeit gibt es 29 unterschiedliche Lizenzen, die sich in Rechte-, Schnittstellen- und Sprachlizenzen untergliedern. Um eine vollständige Benutzerorientierung zu gewährleisten, müssen demnach alle Lizenzen übertragen werden, die einem Benutzer zugeteilt sind (vgl. HEINZE 2013).

Die Zusammenstellung der Lizenzen wird auf Grund der ähnlichen Problemstellung in der gleichen Form wie die Baustein-Konfiguration übertragen. Die Referenzierung muss dementsprechend analog zur Konfiguration über eine zentral gepflegte Datei erfolgen.

5.2.2.5 *Ansicht*

Schlüssel: `pkey` (für *page key*)

Die Ansicht, von der der Benutzer die Hilfe aus aufruft, wird über den PageKey übermittelt (siehe Abschnitt 5.1.3). Aufgrund der hohen Anzahl an unterschiedlichen Ansichten[44] und der hohen Änderungswahrscheinlichkeit (z.B. durch eine neue Ansicht in einem Minor-Release), muss dieser komplett übertragen werden und kann nicht

44 Da der PageKey bei Projektron BCS für jede Ansicht dynamisch erstellt wird, ist die Gesamtzahl nicht festgelegt, liegt aber in der Dimension von ca. 1.500 möglichen Werten (vgl. SAUTER 2013).

wie die Systemkonfiguration durch binäre Abbildungen von Listen abgebildet werden.

5.2.2.6 *Zusätzliche Parameter*

PROFIL

Schlüssel: `uid` (für *user id*), `u` (für *user name*)

Das OH3-Framework speichert Benutzerprofile lokal im Browser (vgl. Framework). Um zwischen Benutzern unterscheiden zu können, die von unterschiedlichen Projektron-BCS-Accounts, aber dem gleichen Betriebssystem-Account auf die Online-Hilfe aus zugreifen, muss die Benutzer-ID von Projektron BCS mit übertragen werden. Der Benutzername aus der Stammsoftware kann für Personalisierungszwecke (z.B. eine individuelle Begrüßung) verwendet werden.

SPRACHE

`lang` (für *language*)

Der Hilfeinhalt und die Oberfläche der Online-Hilfe sollen automatisch in der Oberflächensprache des Benutzers angezeigt werden. Dazu muss die eingestellte Sprache übermittelt werden. Als Werte werden die zweistelligen internationalen Sprachkürzel aus ISO 639-1 verwendet. Wird die Sprache nicht erkannt, wird auf eine Default-Sprache zurückgegriffen.

PERSONALISIERUNG

`style` (für *style*)

Die Hilfe kann mit verschiedenen Personalisierungsthemen z.B. an das Corporate Design des Unternehmens oder die Gestaltung der Software angepasst werden. Durch den Aufruf des Parameters wird ein alternatives Stylesheet geladen, das die nötigen Anpassungen enthält.

5.2.2.7 *Integritätsprüfung*

`check` (für *checksum*)

Um sicherzustellen, dass die URL nicht fehlerhaft übertragen wurde oder eine Manipulation der Parameter stattgefunden hat, wird als zusätzlicher Parameter eine Prüfsumme angehängt, die sich aus dem vorhergegangenen query-String der URL berechnet. Vor der Verarbeitung der URL durch das OH3-Framework wird die Prüfsumme erneut berechnet und mit der übergebenen Prüfsumme verglichen (siehe Code 21). Erst bei einer Übereinstimmung erfolgt eine Weiterverarbeitung durch das Framework.

```
oh3.hash = (string) ->
  a = string.split("").reduce ((a, b) ->
    a = ((a << 5) - a) + b.charCodeAt(0)
    a & a
  ), 0
  b = a.toString(36)
```

Code 21: Prüfsummenberechnung des Frameworks

5.3 BEISPIELHAFTE IMPLEMENTIERUNG

Implementierung durch Link-Generator

Um die Schnittstelle mit einer Vielzahl an möglichen Wertkombinationen testen zu können, wurde ein Link-Generator erstellt. Mit dem Generator können alle vom System an das Framework übertragenen Benutzereigenschaften und Systemkonfigurationen simuliert werden (siehe Abb. 34). Die generierte URL kann anschließend vom OH3-Framework verarbeitet und aufgelöst werden.

Die technische Umsetzung des Link-Generators entspricht der des Frameworks. Das Scripting erfolgte in CoffeeScript, das Markup in HTML5 und für den Datenimport (Lizenz- und Bausteinlisten) wird XML verwendet.

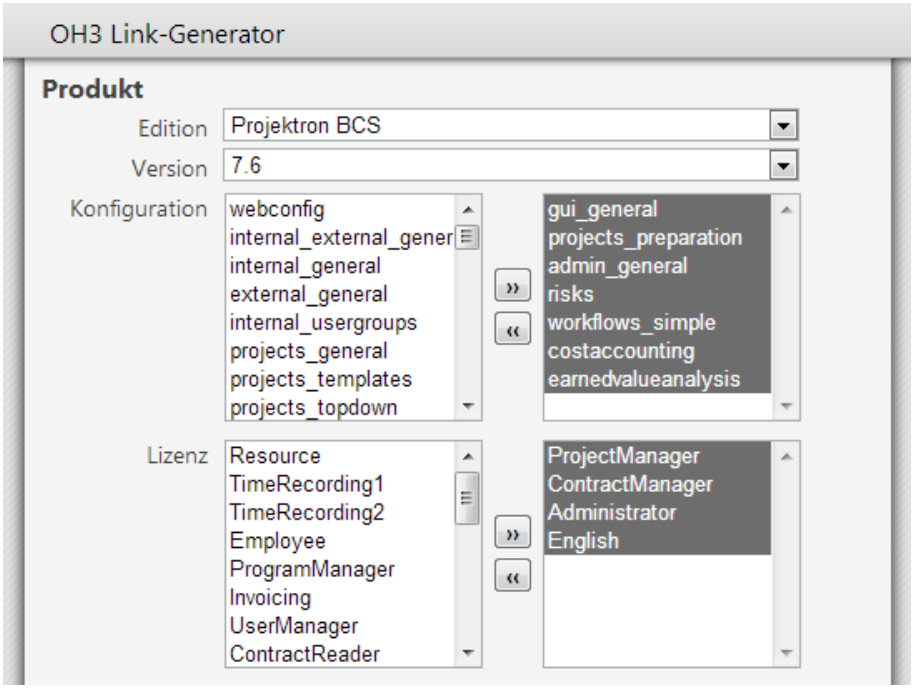


Abb. 34: Implementierung durch Link-Generator

Die komprimierte Übertragung von Konfiguration und Lizenzen auf Basis von gemeinsam geteilten Listen ist vollständig implementiert

und kann als funktionale Grundlage für eine Umsetzung in Projektron BCS dienen.

Die Parameterwerte, die beim Link-Generator vom Nutzer eingegeben werden, können von Projektron BCS im Hintergrund bereitgestellt und zu einem Link zusammengesetzt werden (vgl. SAUTER 2013 und HEINZE 2013). Dargestellt in Abb. 35

Einhaltung der Anforderungen

Durch den Link-Generator konnte mit echten Basisdaten eine durchschnittliche URL-Länge zwischen 180-190 Zeichen ermittelt werden. Das entspricht ca. 9% der Maximallänge von 2.000 Zeichen (siehe Abschnitt 5.2.1.1). Damit kann gezeigt werden, dass die spezifizierte Schnittstelle alle Anforderungen bei Einhaltung der Einschränkungen erfüllen kann. Auch eine Erweiterbarkeit durch zusätzliche Parameter wird dadurch möglich.

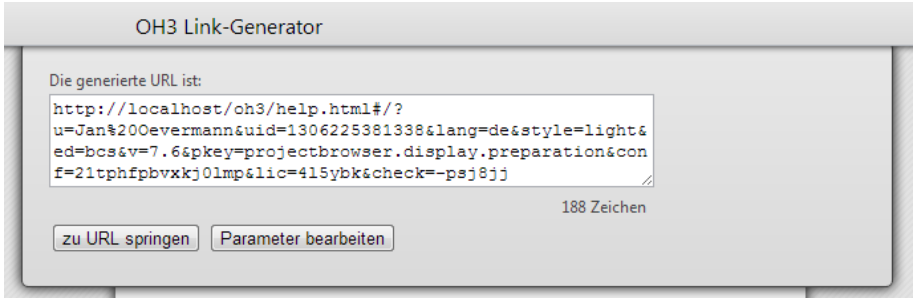


Abb. 35: URL-Generierung durch Link-Generator

Zur besseren Veranschaulichung möglichen Wertkombinationen können im Linkgenerator auch vier vorgefertigte Anwendungsfälle (*Use-Cases*) ausgewählt werden. Sie stellen jeweils eine typische Konfiguration für eine Zielgruppe dar (siehe Tabelle 10).

NR.	ANSICHT	ZIELGRUPPE
1	Projektvorbereitung	Projektmanager mit benötigten Bausteinen
2	Projektvorbereitung	Zeiterfasser 2 ohne benötigte Bausteine
3	Projektvorbereitung	Administrator mit benötigten Bausteinen
4	Ansichten in Projektron BCS	Zeiterfasser 1 mit benötigten Bausteinen

Tabelle 10: UseCases Framework

6

FAZIT UND AUSBLICK

6.1 ZUSAMMENFASSUNG

Als Ergebnis der Arbeit konnte das Online-Hilfe 3.0-Konzept erfolgreich prototypisch implementiert werden. Nach eingehender Analyse bestehender Konzepte zur Benutzerorientierung wurde in Kapitel 2 ein für die Software-Dokumentation gültiges Modell für Relevanzfaktoren entworfen. Die konzeptionellen Ergebnisse wurden in Kapitel 3 in das CMS COSIMA go! implementiert, so dass es nun möglich ist, im Rahmen des normalen Redaktionsprozesses, benutzerorientierte Inhalte zu erstellen und zu publizieren. Um diese Inhalte dynamisch darzustellen und filterbar zu machen, wurde in Kapitel 4 auf Basis von Web-Technologien ein Framework entwickelt. Die Anbindung dieses Frameworks an die Software Projektron BCS wurde abschließend in Kapitel 5 vorgenommen.

Rückblickend konnten aussagekräftige Antworten auf die eingangs formulierten Fragen gefunden werde:

- Die Implementierung des OH3-Konzepts in ein Content-Management-System ist stark vom verwendeten System und dessen Customizing-Eigenschaften abhängig. Das untersuchte CMS COSIMA go! lässt sich weitgehend sehr schnell und einfach anpassen. In Teilbereichen (z.B. der Transformation) ist eine Anpassung durch den Anwender hingegen nicht vorgesehen, was eine Implementierung erschwert.
- Durch den Einsatz des CMS-internen CVM wurde eine für den Redakteur sehr komfortable Möglichkeit zur Vergabe der Relevanzfaktoren gewählt. Durch die Implementierung einer Exportstrecke und einer Vorschau des OH3-Formats integriert sich das Konzept sehr gut in den bestehenden Redaktionsprozess.
- Durch die Konzeption des Frameworks konnte eine Trennung zwischen Kernfunktionen und implementierungsabhängigen Programmteilen erreicht werden. Diese Architektur erlaubt die einfache Übertragung des Frameworks auf andere Softwareprodukte und erleichtert die Implementierung in andere Systemumgebungen.

6.2 FAZIT

Im Rahmen der Arbeit konnten erfolgreich verschiedenste Technologien und Systeme zu einem schlüssigen Gesamtkonzept vereint werden. Durch die Kombination bewährter Standards und neuer Web-Technologien wie HTML5 und CoffeeScript kann eine zukunftssichere und mittelfristige Kompabilität auf einer großen Breite an Endgeräten sichergestellt werden.[45]

Die ursprüngliche Idee, einen OH3-Leitfaden als zusätzliche Ansicht in das CMS zu integrieren, um Redakteuren Richtlinien für Vergabe, Abhängigkeiten und Struktur von Bewertungen zu geben, wurde im Laufe der Arbeit verworfen, da durch den Einsatz der CVM-Logik innerhalb des CMS und den möglichen Logik-Konfigurationen im Framework diese Aufgaben für den Redakteur obsolet geworden sind.

Als besonders komplex in der Umsetzung haben sich die konfigurierbare Regellogik und die priorisierte Filterung des Frameworks erwiesen. Die Spezifikation und Implementierung der Schnittstelle zur Übertragung von System- und Lizenzkonfigurationen war ebenfalls eine Herausforderung. Unproblematisch war hingegen das Customizing von COSIMA go! über den WebClient, sowie die Anpassung der mitgelieferten DTD.

Bemerkenswert ist auch die große Anzahl an Gesprächen, die im Rahmen der Arbeit durchgeführt werden konnte. Dadurch konnten Erfahrungen und Meinungen vieler verschiedener Sichtweisen in die Problemlösung mit einfließen. Aus technologischer Perspektive interessant waren hierbei die Gespräche mit Entwicklern der unterschiedlichen Systeme, aus wirtschaftlicher Position die Diskussionen mit Produktmanagern sowie Geschäftsführern und aus praktikabler Sicht die Befragung von Technischen Redakteuren.

Abschließend können die Ergebnisse der Arbeit als positiv eingestuft werden. Es konnte gezeigt werden, dass es möglich ist, ein benutzerorientiertes Konzept für Software-Dokumentation in ein bestehendes CMS zu implementieren und eine flexible Benutzerschnittstelle zur Filterung der Inhalte bereitzustellen.

45 Getestet wurden die derzeit aktuellen Versionen von Google Chrome (26), Mozilla Firefox (20), Microsoft Internet Explorer (10) und Apple Safari (5).

6.3 AUSBLICK

Auf Basis der Implementierung muss das Konzept nun seine Praxistauglichkeit beweisen. Dazu müssen zunächst Teile des Dokumentationsbestandes für Projektron BCS in das neue Format überführt werden. Weiterhin muss auf Basis des Link-Generators und dessen Methoden eine entsprechende Funktionalität in Projektron BCS implementiert werden.

Ein erster produktiven Einsatz kann zunächst firmenintern oder mit einer begrenzten Nutzergruppe erfolgen. Anschließend sollten Usability-Tests zum Überprüfen der Anwenderakzeptanz durchgeführt werden.

Bereiche, Aufgabengebiete und Systeme, die in weiterführenden Arbeiten betrachtet werden müssen, sind:

- Die Implementierung einer Suchfunktion, die Ergebnisse nach benutzerorientierten Faktoren (dynamisch) filtern kann.
- Die Lokalisierung benutzerorientierter Dokumentationen in andere Sprach- oder Kulturkreise.
- Eine automatisierte Erstellung von Maps-Dateien aus dem Content-Management-System.

Weiterhin kann der Bedarf für eine Umkonzeptionierung des Frameworks in eine Client-Server-Architektur geprüft werden. Hierbei kann bestehender CS-Code auf Serverseite unter *Node.js* ausgeführt werden, einer Plattform zum Betreiben von JavaScript-basierten Serveranwendungen. Dadurch kann z.B. bei sensiblen Inhalten eine Filterung der Inhalte schon vor Auslieferung an den Benutzer erfolgen oder Inhaltsbewertungen können durch Verhaltensanalysen dynamisch verbessert werden.

A ANHANG

A.1 BEISPIELE

In diesem Teil des Anhangs werden einige ausgewählte Beispiele vorgestellt, die die Funktionsweise des Gesamtkonzepts verdeutlichen.

Auszug aus Beispielmodul im CMS

Der folgende Auszug *Code 22* zeigt den Content in der Form, in der er im CMS vorliegt (siehe dazu Abb. 28-1). Alle nicht relevanten Attribute wurden zur besseren Lesbarkeit entfernt und durch `[*]` ersetzt.

```
1   <chapter [*]>
2     <title [*]>Risiken</title>
3     <block [*] y.validity.text="oh3.custom.de.projektron.bcs.
          features.webapp.risks">
4       <example [*] y.validity.text="oh3.user.concept.beginner |
            oh3.user.concept.advanced">
5         <title [*]>Risikomanagement</title>
6         <p [*]>
7           Unter Risikomanagement versteht man den planvollen
              Umgang mit Risiken.
8           Risikomanagement umfasst die Phasen
              Risikoidentifikation, Risikobewertung,
              Risikosteuerung und Risikokontrolle. Analog dem
              Managementkreis werden die Phasen wiederholt
              durchlaufen und stellen somit einen Zyklus dar.
9         </p>
10        <p [*]>
11          Quelle:
12          <url address="http://de.wikipedia.org/wiki/Risiko#
              Risiko_in_der_Wirtschaftswissenschaft">Wikipedia<
              /url>
13        </p>
14      </example>
15    </block>
16    <block [*]>
17      <notice [*] y.validity.text="oh3.user.product.beginner |
            oh3.user.product.advanced">
18        <p [*]>
19          Um in Projektron BCS Risiken verwalten zu k&amp;#246;nnen
              , ben&amp;#246;tigen Sie den aktivierten Baustein <i>
              Risiken</i>.
```

```
20          </p>
21          <ul [*]>
22            <li [*] y.validity.text="oh3.custom.de.projektron.bcs
                  .features.webapp.risks">
23              <p [*]>
24                <i>Der ben&#246;tigte Baustein   Risiken wurde
                      bei Ihnen gefunden</i>
25              </p>
26            </li>
27          </ul>
28        </notice>
29      </block>
30      <block [*] y.validity.text="oh3.custom.de.projektron.bcs.
            features.webapp.risks">
31        <p [*] y.validity.text="oh3.user.concept.beginner">
32          Mit einem Risiko, das eventuell eintreten wird, sind
              unter Umst&#228;nden zus&#228;tzliche Kosten, zus
              &#228;tzlicher Aufwand sowie eine zeitliche Verz
              &#246;gerung des Projekts verbunden. Solche Kosten
              und Aufw&#228;nde k&#246;nnen Sie f&#252;r jedes
              Risiko eintragen.
33        </p>
34        <p [*]>
35          Risiken k&#246;nnen Sie in Kategorien einordnen
36          <oh3-inline [*] y.validity.text="oh3.user.product.
                beginner | oh3.user.concept.beginner | oh3.user.
                product.advanced">
37            und als Politisches Risiko, Terminliches
38            Risiko, Wirtschaftliches Risiko, Ressourcenrisiko
                oder Technisches Risiko
39          </oh3-inline>
40          . Ursachen und Auswirkungen k&#246;nnen Sie in den
              entsprechenden Eingabefeldern zus&#228;tzlich
              festhalten.
41        </p>
42        <figure [*]>
43          <img [*]></img>
44          <subtitle [*]>Ansicht Risiken</subtitle>
45        </figure>
46        <p [*]>
47          Gleichzeitig k&#246;nnen Sie die Risiken mit Hilfe der
              Status unterschiedlich einordnen.
48          <oh3-inline [*] y.validity.text="oh3.user.concept.
                beginner">
49            Risiken, die noch eintreten k&#246;nnen, erhalten den
                Status Bestehend, eingetretene Risiken werden
                mit dem Status Eingetreten und Risiken, die nicht
                mehr eintreten k&#246;nnen, mit dem Status Nicht
                eingetreten versehen.
50          </oh3-inline>
51        </p>
52        <p [*] y.validity.text="oh3.user.concept.beginner | oh3.
              user.concept.advanced">
```

```
53          Durch pr&#228;ventive Gegenma&#223;nahmen kann die
                Eintrittswahrscheinlichkeit von Risiken reduziert
                werden. Kurative Gegenma&#223;nahmen dienen dazu,
                den Schaden durch bereits eingetretene Risiken zu
                mindern.
54        </p>
55      </block>
56    </chapter>
```

Code 22: Beispielmodul im CMS

Auszug aus Beispielmodul im OH3-Framework

Der folgende Auszug *Code 23* zeigt den Content in der Form, in der er im DOM des OH3-Frameworks nach der Anreicherung (siehe dazu Abb. 28-4) vorliegt.

```
1     <div class="chapter">
2       <h2>1.2. Risiken</h2>
3       <div class="block oh3.custom.de.projektron.bcs.features.
            webapp.risks">
4         <div class="example oh3.user.concept.beginner oh3.user.
              concept.advanced oh3.context.infodesire.all">
5           <h2>Risikomanagement</h2>
6           <p class="oh3.context.infodesire.all">
7             Unter Risikomanagement versteht man den planvollen
                  Umgang mit Risiken. Risikomanagement umfasst die
                  Phasen Risikoidentifikation, Risikobewertung,
                  Risikosteuerung und Risikokontrolle. Analog dem
                  Managementkreis werden die Phasen wiederholt
                  durchlaufen und stellen somit einen Zyklus dar.
8           </p>
9           <p class="oh3.context.infodesire.all">
10            Quelle:
11            <a href="http://de.wikipedia.org/wiki/Risiko#
                  Risiko_in_der_Wirtschaftswissenschaft">Wikipedia<
                  /a>
12          </p>
13        </div>
14      </div>
15      <div class="block">
16        <div class="notice oh3.user.product.beginner oh3.user.
              product.advanced oh3.context.infodesire.all" style="
              display: block;">
17          <p class="oh3.context.infodesire.all">
18            Um in Projektron BCS Risiken verwalten zu können, ben
                  ötigen Sie den aktivierten Baustein
19            <i>Risiken</i>
20            .
21          </p>
```

```
22          <ul>
23            <li class="oh3.custom.de.projektron.bcs.features.
                  webapp.risks">
24              <p class="oh3.context.infodesire.all">
25                <i>Der benötigte Baustein   Risiken wurde bei
                      Ihnen gefunden</i>
26              </p>
27            </li>
28          </ul>
29        </div>
30      </div>
31      <div class="block oh3.custom.de.projektron.bcs.features.
            webapp.risks">
32        <p class="oh3.user.concept.beginner oh3.context.
              infodesire.all" style="display: none;">
33          Mit einem Risiko, das eventuell eintreten wird, sind
                unter Umständen zusätzliche Kosten, zusätzlicher
                Aufwand sowie eine zeitliche Verzögerung des
                Projekts verbunden. Solche Kosten und Aufwände kö
                nnen Sie für jedes Risiko eintragen.
34        </p>
35        <p class="oh3.context.infodesire.all">
36          Risiken können Sie in Kategorien einordnen
37          <span class="oh3-inline oh3.user.product.beginner oh3.
                user.concept.beginner oh3.user.product.advanced"
                style="display: inline;">
38            und als Politisches Risiko, Terminliches Risiko,
                  Wirtschaftliches Risiko, Ressourcenrisiko oder
                  Technisches Risiko
39          </span>
40          . Ursachen und Auswirkungen können Sie in den
                entsprechenden Eingabefeldern zusätzlich festhalten
                .
41        </p>
42        <img class="figure oh3.user.mediapref.visual" src="https:
              //demo02.docufy.de/cmsweb/view?io=
              cd5124f61319bd8b05093a250af8ee31&amp;version=1&amp;
              language=--&amp;variant=--&amp;formattype=PNG72&amp;
              preview=false" style="display: inline;">
43        <p class="subtitle oh3.user.mediapref.visual oh3.context.
              infodesire.all" style="display: block;">Ansicht
              Risiken</p>
44        <p class="oh3.context.infodesire.all">
45          Gleichzeitig können Sie die Risiken mit Hilfe der
                Status unterschiedlich einordnen.
46          <span class="oh3-inline oh3.user.concept.beginner"
                style="display: none;">
47            Risiken, die noch eintreten können, erhalten den
                  Status Bestehend, eingetretene Risiken werden mit
                   dem Status Eingetreten und Risiken, die nicht
                  mehr eintreten können, mit dem Status Nicht
                  eingetreten versehen.
48          </span>
```

```
49          </p>
50          <p class="oh3.user.concept.beginner oh3.user.concept.
                advanced oh3.context.infodesire.all">
51            Durch präventive Gegenma ß nahmen kann die
                Eintrittswahrscheinlichkeit von Risiken reduziert
                werden. Kurative Gegenma ß nahmen dienen dazu, den
                Schaden durch bereits eingetretene Risiken zu
                mindern.
52          </p>
53        </div>
54      </div>
```

Code 23: Beispielmodul im OH3-Framework

Beispielmodul als PDF

Der folgende Screenshot Abb. 36 zeigt das Beispielmodul als PDF, das mit dem CMS erzeugt wurde. Die vollständige PDF ist auf beiliegender CD enthalten.

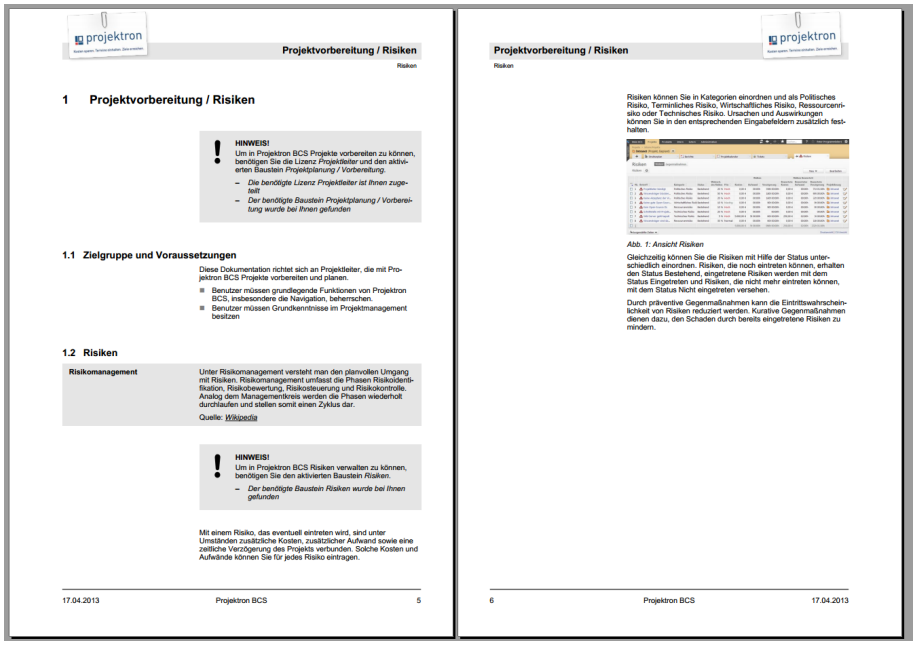


Abb. 36: Beispielmodul als PDF

Beispielmodul als Online-Hilfe 3.0

Der folgende Screenshot Abb. 37 zeigt das Beispielmodul als Online-Hilfe 3.0 im Browser.

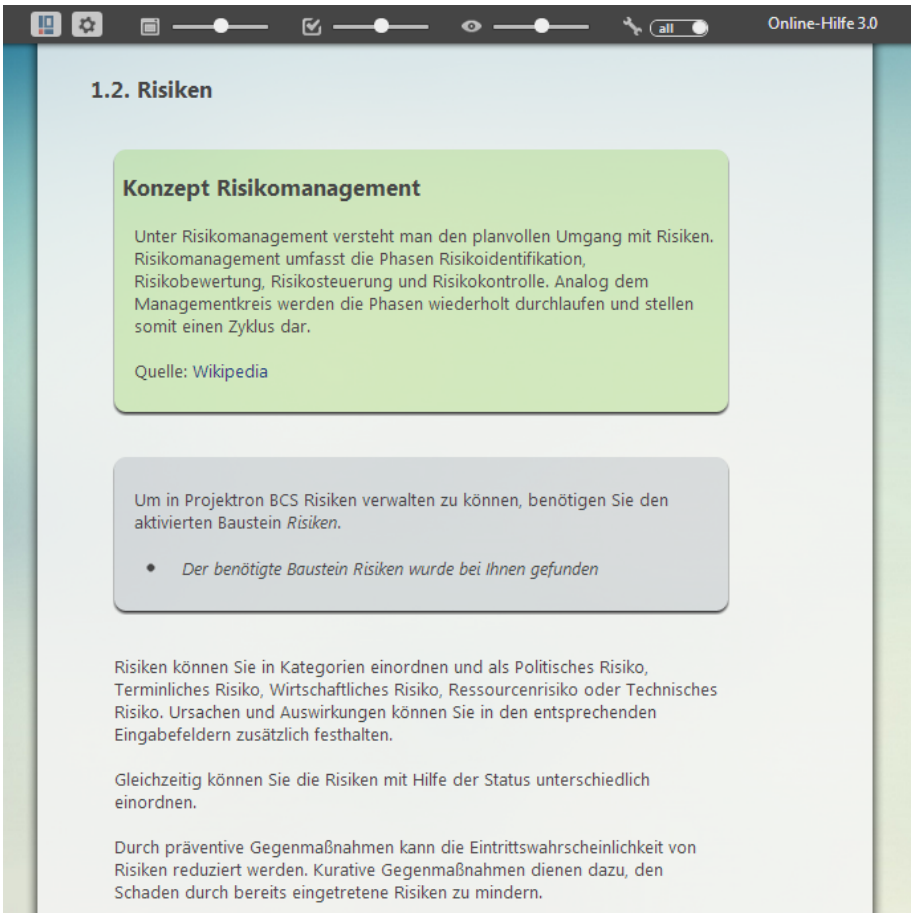


Abb. 37: Beispielmodul als Online-Hilfe 3.0

A.2 ANLEITUNGEN

In diesem Teil des Anhangs werden wichtige Vorgänge zur Inbetriebnahme der Ergebnisse erläutert.

OH3-Konfiguration in CGO importieren

In den folgenden Schritten wird erläutert, wie die Ergebnisse aus Kapitel 3 als Konfigurationspaket in ein COSIMA go!-System der Version 4.2.50 importiert werden können.

1. Starten Sie den COSIMA go! RichClient und melden Sie sich am System an.
2. Wählen Sie im Menü *CMS* auf *System-ZIP* importieren.
3. Wählen Sie die Datei `OH3_CGO_systemzip.zip` im Ordner `CGO-Systemkonfiguration` auf der beiliegenden CD und klicken Sie auf *Öffnen*.
4. Wählen Sie im Dialogfenster *System-ZIP importieren* die Option *Nein*.

   Der Import des System-ZIP ist abgeschlossen.
5. Wählen Sie im Menü *CMS* auf *System-Konfiguration importieren*.
6. Wählen Sie die Datei `OH3_CGO_systemconfig.xml` im Ordner `CGO-Systemkonfiguration` auf der beiliegenden CD und klicken Sie auf *Öffnen*.

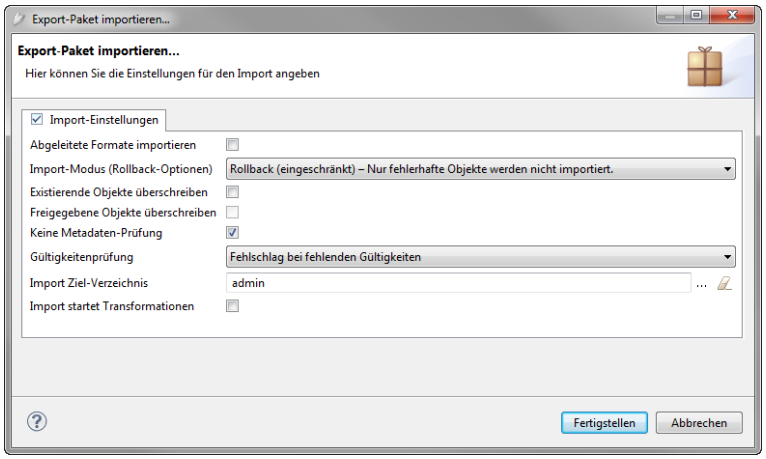
   Der Import der Systemkonfiguration ist abgeschlossen.

OH3-Beispielcontent in CGO importieren

In den folgenden Schritten wird erläutert, wie OH3-Beispielcontent in ein COSIMA go!-System der Version 4.2.50 importiert werden kann.

1. Starten Sie den COSIMA go! RichClient und melden Sie sich am System an.
2. Stellen Sie sicher, dass die OH3-Systemkonfiguration auf Ihrem System eingerichtet wurde (siehe Kapitel A.2)
3. Klicken Sie mit gedrückter STRG-Taste und Rechtsklick auf den Ordner, in den der Beispielcontent importiert werden soll und wählen Sie *Export-Paket importieren...*.
4. Wählen Sie die Datei `OH3_Content-Export-Paket.zip` im Ordner `OH3-Beispielcontent` auf der beiliegenden CD und klicken Sie auf *Öffnen*.

5. Wählen Sie im Dialogfenster *Export-Paket importieren...* folgende Optionen:



   - Import-Modus: *Rollback (eingeschränkt)*
   - Keine Metadatenprüfung: *aktiviert*
   - Import-Zielverzeichnis: *Ihren Benutzerordner*
   - Import startet Transformation: *deaktiviert*

   Alle anderen Optionen des Dialogfensters können unverändert übernommen werden.

   Der OH3-Beispielcontent wurde importiert. Die importierten Dateien befinden sich in Ihrem Benutzerordner.

OH3-Framework in Betrieb nehmen

In den folgenden Schritten wird erläutert, wie das Framework aus Kapitel 4 in Betrieb genommen werden kann.

1. Stellen Sie sicher, dass ein Webserver auf Ihrem System installiert ist (z.B. *xampp*).
2. Kopieren Sie den Ordner `oh3` aus dem Ordner `OH3-Framework` auf beiliegender CD in den `htdocs`-Ordner des Webservers.
3. Starten Sie den Webserver
4. Rufen Sie in einem Browser[46] die folgende Adresse auf:

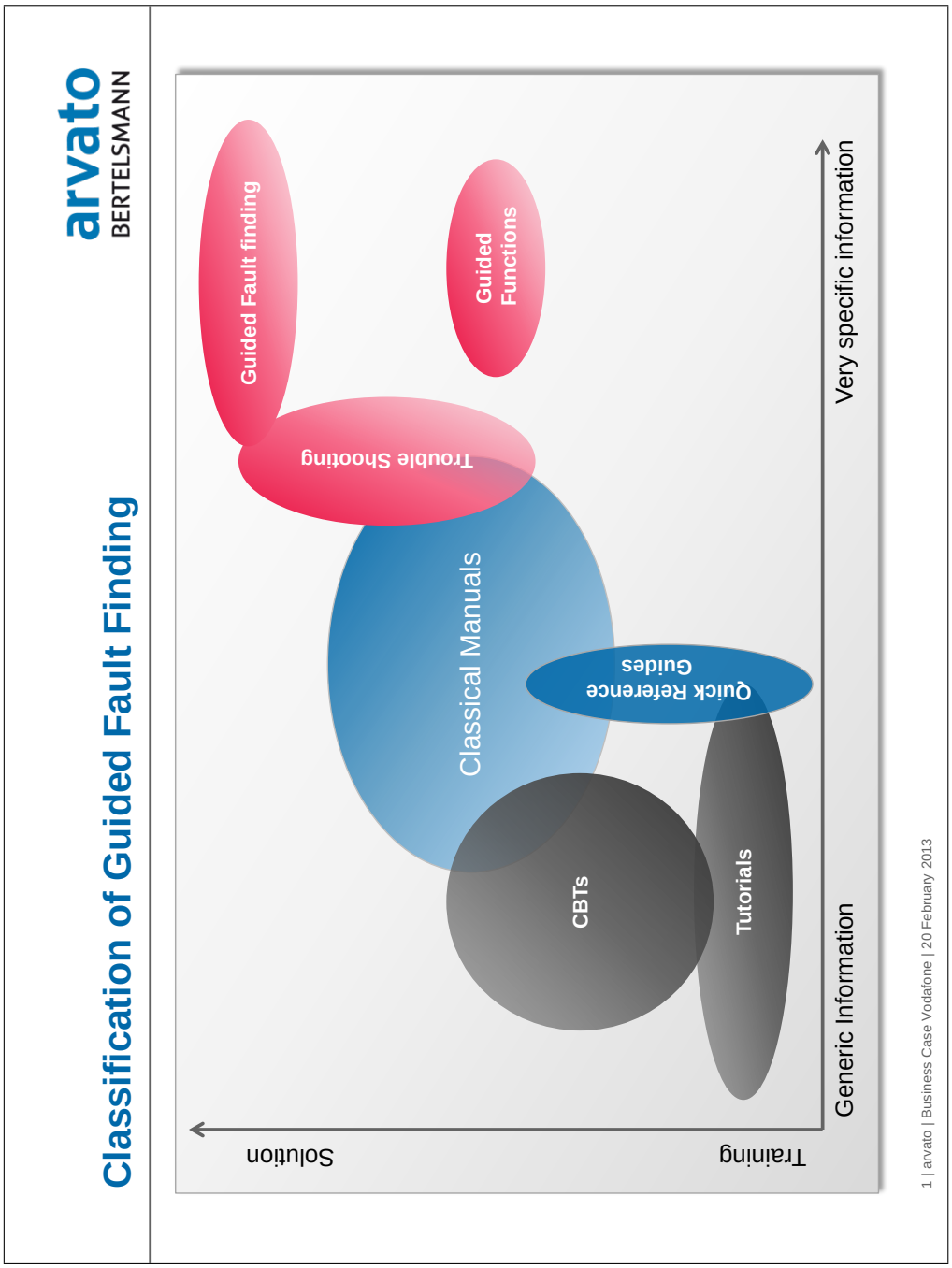
   ```
   http://localhost/oh3/
   ```

5. Wählen Sie einen UseCase oder generieren Sie eine URL um zum Framework zu gelangen.

46 Folgende Browser sind ab der in Klammern genannten Version kompatibel und getestet: Google Chrome (26), Mozilla Firefox (20), Microsoft Internet Explorer (10) und Apple Safari (5)

A.3 ABBILDUNGEN

Classification of Guided Fault Finding (arvato services TI)



LITERATURVERZEICHNIS

Arvato Services TI GmbH (2013): Classification of Guided Fault Finding. Grafik von Holger Nitsche aus dem Business Case Vodafone vom 20.02.2013. Siehe Anhang/Abbildungen.

Belikan, Oliver (2007): Mit Personalisierung gegen die Informationsüberflutung. <http://blog.doubleslash.de/mit-personalisierung-gegen-die-informationsuberflutung/> [Stand: Dezember 2007. Letzter Zugriff: 2013-02-12]

Benda, Ruth von (2009): Die sprachliche Gestaltung von Links in eingebetteter Hilfe. Master-Thesis. Studiengang Technische Kommunikation. Donau-Universität Krems

Bertram, Miriam H. (2012): „Junge Doku am Start - Generation 3.0: in multimedialen, virtuellen Räumen zu Hause." In: C-Blatt, Nr. 23, 14–15

Bongers, Frank / Vollendorf, Maximilian (2011): jQuery - das Praxisbuch. Bonn : Galileo

Burnham, Trevor (2011): Coffeescript - Accelerated JavaScript Development. Dallas, TX : Pragmatic Bookshelf

Chase, Nicholas (2003): XML primer plus. Indianapolis, IN : Sams

Closs, Sissi (2006): „Agil und eXtrem - die Zukunft der Technischen Dokumentation." In: C-Blatt, Nr. 14, 5–7

Closs, Sissi (2009): Technische Dokumentation im Wandel. <http://www.documanager.de/magazin/technische_dokumentation_im_wandel.html> [Stand: April 2009. Letzter Zugriff: 2012-02-12]

Closs, Sissi (2011): Single Source Publishing - Modularer Content für EPUB & Co. Frankfurt/Main : Entwickler.press

Closs, Sissi (2012a): „Hilfe mit Zukunft - Schwachstellen aktueller Hilfen und Anforderungen an neue Konzepte." In: technische kommunikation 34. Jg., Nr. 1, 38–42

Closs, Sissi (2012b): „Informationen aus der Dose?" In: Hennig, Jörg / Tjarks-Sobhani, Marita (Hrsg.): Technische Kommunikation im Jahr 2041: 20 Zukunftsszenarien. Lübeck : Schmidt-Römhild

Dambeck, Holger (2013): Facebook-Studie: Likes enthüllen Persönlichkeit. Spiegel Online. <http://www.spiegel.de/netzwelt/web/facebook-studie-likes-enthuellen-persoenlichkeit-a-888151.html> [Stand: März 2013. Letzter Zugriff: 2013-03-12]

DIN EN ISO 13407 (2000): Benutzer-orientierte Gestaltung interaktiver Systeme (abgelöst durch DIN EN ISO 9241-210).

DIN EN ISO 9241-210 (2011): Ergonomie der Mensch-System-Interaktion – Teil 210: Prozess zur Gestaltung gebrauchstauglicher interaktiver Systeme.

Docufy GmbH (2010): Broschüre COSIMA go! <http://www.docufy.de/uploads/media/Broschuere_COSIMAgo__web.pdf>
[Stand: September 2010. Letzter Zugriff: 2013-02-15]

Docufy GmbH (2012a): COSIMA 4.2 - Administrationshandbuch.

Docufy GmbH (2012b): COSIMA 4.2 - Anwenderhandbuch.

Docufy GmbH (2012c): COSIMA go! 4.2 - Schulungsunterlagen.

Dreusicke, Michael (2011): Authoring with Linked Data: Social Media, E-Learning and SEO (Folien zur tekom Jahrestagung 2011). <http://www.tekom.de/upload/3383/TA7_Dreusicke.pdf>
[Stand: 2011. Letzter Zugriff: 2013-02-12]

Dreusicke, Michael (2013): Geschäftsführer der PAUX Technologies GmbH. Gespräch zu Content-Enhancement-Technologien und Benutzerorientierung. Bei Projektron, Berlin (27.03.2013 16:00 Uhr MEZ).

Dreusicke, Michael / Lieske, Christian / Sasaki, Felix (2012): „Evolution der Topic-Idee - Neue Möglichkeiten durch Microcontent.“ In: technische kommunikation 34. Jg., Nr. 3, 52–55

Drewer, Petra / Ziegler, Wolfgang (2011): Technische Dokumentation. Würzburg : Vogel

DuCharme, Bob (2001): XSLT quickly. Greenwich, CT : Manning

Ebenhoch, Peter (2012): „Sorgenkind Redaktionssystem - Fehlerursachen und Lösungen bei Einführung und Betrieb.“ In: technische kommunikation 34. Jg., Nr. 1, 43–48

Eckl, Roland / Arafat, Oliver / Wendel, Heinrich (2012): „Skripte vom Band: JavaScript für Industrie- und Geschäftsanwendungen.“ In: ix: Magazin für professionelle Informationstechnik, Nr. 10, 46–54

EU-Maschinenrichtlinie (2006): Richtlinie 2006/42/EG des europäischen Parlaments und des Rates vom 17. Mai 2006 über Maschinen und zur Änderung der Richtlinie 95/16/EG (Neufassung). <http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2006:157:0024:0086:DE:PDF>
[Stand: Mai 2006. Letzter Zugriff: 2013-02-20]

Goldmann, Stephan / Nitsche, Holger (2013): Geschäftsführung der arvato services technical information GmbH. Diskussion zu dynamischen Hilfesystemen. Bei arvato, Neukirchen-Vluyn (05.03.2013 13:30 Uhr MEZ).

GRÜNWIED, Gertrud (2009): „Softwaredokumentation gewinnt an Bedeutung.“ In: technische kommunikation 31. Jg., Nr. 4, 48–52

HAROLD, Elliotte Rusty / MEANS, W. Scott (2003): XML in a Nutshell. Deutsche Ausgabe. 3. Auflage. Köln : O'Reilly

HASEBRINK, Uwe (2011): „Veränderungen der Mediennutzung.“ In: HENNIG, Jörg / TJARKS-SOBHANI, Marita (Hrsg.): Veränderte Mediengewohnheiten - andere Technische Dokumentation? Lübeck : Schmidt-Römhild

HAUSER, Tobias (2010): XML Standards. 2. Auflage. Frankfurt/Main : Entwickler.Press

HEINZE, Bertram (2013): Entwickler bei der Projektron GmbH. Gespräch zur Anbindung einer benutzerorientierten Online-Hilfe an Projektron BCS. Bei Projektron, Berlin (14.03.2013 10:00 Uhr MEZ).

HOCHSCHULE KARLSRUHE (2012): Studierende der HsKA mit Innovationspreis ausgezeichnet. <http://www.hs-karlsruhe.de/hochschule/aktuelles/presse/tekom-preis.html>
[Stand: Mai 2012. Letzter Zugriff: 2013-02-12]

IBM DEVELOPERWORKS (2002): Customize a DTD with parameter entities. <http://www.ibm.com/developerworks/xml/library/x-tiparam/index.html>
[Stand: Juli 2002. Letzter Zugriff: 2013-02-21]

IEEE 754 (2008): Standard for Floating-Point Arithmetic.

IETF RFC 1738 (1994): Uniform Resource Locators (URL). <http://tools.ietf.org/html/rfc1738>
[Stand: Dezember 1994. Letzter Zugriff: 2013-03-20]

IETF RFC 2616 (1999): Hypertext Transfer Protocol – HTTP/1.1. <http://tools.ietf.org/html/rfc2616>
[Stand: Juni 1999. Letzter Zugriff: 2013-03-20]

ISO 8879 (1986): Information processing; Text and office systems; Standard Generalized Markup Language (SGML).

ISO/IEC 15445 (2000): Information technology; Document description and processing languages; HyperText Markup Language (HTML).

ISO/IEC 9070 (1991): Information technology; SGML support facilities; registration procedures for public text owner identifiers.

JUSTSYSTEMS: Creating Customizations for XMetaL. <http://xmetal.com/tutorials/customizing_xmetal_manually.html>
[Stand: k.A.. Letzter Zugriff: 2013-02-22]

JUSTSYSTEMS (2007): XMetaL Author 5.0 Customization Guide. <http://na.justsystems.com/guides/CG_50.pdf>
[Stand: Januar 2007. Letzter Zugriff: 2013-02-22]

KLEMT, Claudia / KUTTER, Claudia / MÄSER, Frank (2013): Technische Redakteure bei der Projektron GmbH. Diskussion zu Metadatenmodellen in der Softwaredokumentation. Bei Projektron, Berlin (06.03.2013 10:00 Uhr MEZ).

KNOPP, Sandra (2000): Aufbau, Gestaltung und Struktur bei Online-Hilfesystemen - im Kontext der Mensch-Computer-Interaktion. Lübeck : Schmidt-Römhild

KOTHES! GMBH (2008): Infoservice COSIMA go! - Das High-End-Redaktionssystem mit Standard-Applikation für den Mittelstand.

KOTHES! GMBH (2010): Infoservice COSIMA go! - Produktivitätssteigerung in der Technischen Dokumentation.

KRÖMKER, Heidi / NORBEY, Marcel (2012): „Utility, Usability und User Experience 2041: Ein Traum wird wahr." In: HENNIG, Jörg / TJARKS-SOBHANI, Marita (Hrsg.): Technische Kommunikation im Jahr 2041: 20 Zukunftsszenarien. Lübeck : Schmidt-Römhild

KRÖNER, Peter (2010): HTML5 - Webseiten innovativ und zukunftssicher. München : Open Source Press

KRÜGER, Manfred (2012): „Alles XML?" In: HENNIG, Jörg / TJARKS-SOBHANI, Marita (Hrsg.): Technische Kommunikation im Jahr 2041: 20 Zukunftsszenarien. Lübeck : Schmidt-Römhild

KRÜGER, Manfred / ZIEGLER, Wolfgang (2008): „Standards für strukturierte technische Informationen - ein Überblick." In: MUTHIG, Jürgen (Hrsg.): Standardisierungsmethoden für die technische Dokumentation. Lübeck : Schmidt-Römhild, tekom Hochschulschriften 16, 11–40

MANGOLD, Roland (2007): Informationspsychologie: Wahrnehmen und Gestalten in der Medienwelt. München : Elsevier

MEIER, Eva-Maria (2012): Entwicklung eines Konzepts für nutzerzentrierte Kommunikation in Online-Hilfen anhand von Kritereien zur Ermittlung des Informationsbedarfs spezifischer Nutzergruppen. Bachelor-Thesis. Studiengang Technische Redaktion. Hochschule Karlsruhe - Technik und Wirtschaft

MICROSOFT KNOWLEDGEBASE (2007): Maximum URL length is 2,083 characters in Internet Explorer. <http://support.microsoft.com/kb/208427/EN-US>
[Stand: Oktober 2007. Letzter Zugriff: 2013-03-20]

MURTHY, Ramana (2010): „Technical communication: Content is the key." In: tcworld e-magazine <http://www.tcworld.info/tcworld/technical-communication/article/technical-communication-content-is-the-key/>

MUTHIG, Jürgen (2012): „Karlsruhe Studierende gewinnen Innovationspreis.“ In: magazin der Hochschule Karlsruhe 33. Jg., Nr. 66, 13

MUTHIG, Jürgen / SCHÄFLEIN-ARMBRUSTER, Robert (2012): „Weil Redakteure verstehen müssen, was sich Nutzer wünschen.“ In: technische kommunikation 34. Jg., Nr. 4, 18–25

MÜNZ, Stefan / GULL, Clemens (2010): HTML5 Handbuch - die Webgrammatik für das Internet der Zukunft. Poing : Franzis

NIELSEN, Jakob (1994): Usability Engineering. Amsterdam : Morgan Kaufmann

OEHMIG, Peter (2011): „Zur Wirtschaftlichkeit verschiedener Medien.“ In: HENNIG, Jörg / TJARKS-SOBHANI, Marita (Hrsg.): Veränderte Mediengewohnheiten - andere Technische Dokumentation? Lübeck : Schmidt-Römhild

OEVERMANN, Jan (2012): Webbasiertes Framework zur Anzeige und Filterung von XML-Inhalten. Projektarbeit im Rahmen der Vorlesung Multimedia Programmierung (T2B732) bei Prof. Martin Schober. Hochschule Karlsruhe - Technik und Wirtschaft. Siehe Anhang auf CD

OEVERMANN, Jan / MEIER, Eva-Maria (2012a): Online-Hilfe 3.0 - Innovativ und individuell. Wettbewerbsbeitrag zum Studentischen Innovationspreis der tekom- intro 2012. Siehe Anhang auf CD

OEVERMANN, Jan / MEIER, Eva-Maria (2012b): tekom intro - DEMO. <http://www.projektron.de/intro/>
[Stand: Januar 2012. Letzter Zugriff: 2013-02-25]

PAUX TECHNOLOGIES GMBH (2013a): Anwendungen. <http://paux.de/w/paux-technologies-gmbh/s.36-anwendungen>
[Stand: k.A.. Letzter Zugriff: 2013-02-12]

PAUX TECHNOLOGIES GMBH (2013b): Editor. <http://www.paux.de/w/paux-technologies-gmbh/s.19-editor>
[Stand: k.A.. Letzter Zugriff: 2013-02-12]

PELSTER, Ulrich (2011): „XML für den passenden Zweck - Offener und standardisierter Umgang mit XML.“ In: technische kommunikation 33. Jg., Nr. 1, 54–57

PROJEKTRON GMBH (2012a): Dokumentation Navigation für Projektron BCS 7.0.

PROJEKTRON GMBH (2012b): Ganz groß für Kleinunternehmen: Projektron BCS.start - Projektmanagement-Software Projektron BCS. <http://www.projektron.de/aktuelles/projektron-bcs-aktuell/ganz-gross-fuer-kleinunternehmen-projektron-bcsstart-1801>
[Stand: Mai 2012. Letzter Zugriff: 2013-03-19]

PROJEKTRON GMBH (2013a): Dokumentation Administration für Projektron BCS 7.6.

PROJEKTRON GMBH (2013b): Produktbeschreibung Projektron BCS 7.6, Version 1.1 (25.02.2013).

PROJEKTRON GMBH (2013c): Unternehmensprofil. <http://www.projektron.de/ueber-uns/unternehmensprofil> [Stand: 2013. Letzter Zugriff: 2013-02-12]

RATH, Holger (2005): Varianten in den Griff bekommen - Wie deklarative Metadaten die automatisierte dynamische Publikation ermöglichen. <http://www.transline.de/de/infocenter/wissenswertes/varianten-in-den-griff-bekommen-wie-deklarative-metadaten-die-automatisierte-dynamische-publikation-ermoeglichen.html> [Stand: Mai 2005. Letzter Zugriff: 2013-02-12]

ROCKLEY, Ann / KOSTUR, Pamela / MANNING, Steve (2003): Managing Enterprise Content: A Unified Content Strategy. Berkley, CA : New Riders

SAUTER, Florian (2013): Produktmanager Projektron BCS. Gespräch zu Systemeigenschaften von Projektron BCS. Bei Projekron, Berlin (22.03.2013 12:00 Uhr MEZ).

SCHOBER, Martin (2012): „Mobil, mehrsprachig und multimedial - Grundlagen von HTML5." In: technische kommunikation 34. Jg., Nr. 6, 36–42

SCHULZE, Philipp (2013): Entwickler bei der Docufy GmbH. Telefongespräch zur Systemarchitekturvon COSIMA und Customizing-Konzepten (12.02.2013 15:00 Uhr MEZ).

SCHÄFER, Gregor / OEVERMANN, Jan / MEIER, Eva-Maria (2012): „Der Nutzer sagt, was er möchte (Interview)." In: technische kommunikation 34. Jg., Nr. 4, 45–47

SCHÄFLEIN-ARMBRUSTER, Robert (2008): „Ohne Leit kein Freud." In: technische kommunikation 30. Jg., Nr. 4

SIEGEL, Siegfried (2013): „Technische Consumer-Produkte durch zielgruppenspezifische Anleitungen leichter bedienbar machen." In: HENNIG, Jörg / TJARKS-SOBHANI, Marita (Hrsg.): Zielgruppen für Technische Kommunikation. Lübeck : Schmidt-Römhild, 147–156

SKULSCHUS, Marco / WIEDERSTEIN, Marcus (2005): XSLT und XPath für HTML, Text und XML. Bonn : Mitp

STRAUB, Daniela (2011): Branchenkennzahlen für die Technische Dokumentation 2011 - tekom Frühjahrsumfrage (Studie). <http://www.tekom.de/upload/alg/Branchenkennzahlen_Ergebnisuebersicht%20tekom_2011_Sept.pdf> [Stand: September 2011. Letzter Zugriff: 2013-02-20]

STRAUB, Daniela / ZIEGLER, Wolfgang (2008): Effizientes Informationsmanagement durch spezielle Content-Management-Systeme - Praxishilfe und Leitfaden zu Grundlagen - Auswahl und Einführung - Systemen am Markt. 2. Auflage. Stuttgart : TC and more

TAUBER, James K. (1996): Formal Public Identifiers (FPIs). <http://xml.coverpages.org/tauber-fpi.html>
[Stand: April 1996. Letzter Zugriff: 2013-02-16]

TEKOM (2012): „Die tekom verleiht erstmals Preis an Studierende." In: technische kommunikation 34. Jg., Nr. 3, 11–12

TEKOM (2013): Der intro - eine tekom-Initiative für Studierende. <http://www.tekom-intro.de>
[Stand: 2013. Letzter Zugriff: 2013-02-25]

THIEMANN, Petra (2008): Benutzerfreundliche Online-Hilfen - Grundlagen und Umsetzung mit Madcap Flare. Berlin : Springer

ULLY, Frank (2009): Implementierung des TIM-InfoManager - Entwicklung eines Tutorials für die Hochschulnutzung anhand typischer Use-Cases. Diplomarbeit. Studiengang Technische Redaktion. Hochschule Karlsruhe - Technik und Wirtschaft

W3C (2008): Extensible Markup Language (XML) 1.0 (Fifth Edition). <http://www.w3.org/TR/xml/#dt-doctype>
[Stand: November 2008. Letzter Zugriff: 2013-02-06]

W3C (2013): HTML 5.1 Nightly - Editor's Draft. <dev.w3.org/html5/spec/>
[Stand: Februar 2013. Letzter Zugriff: 2013-02-25]

WEINHEIMER, Benjamin (2010): Customizing des XML-Informationsmodells PI- Mod - Konzept und Dokumentation. Master-Thesis. Studiengang Technische Redaktion. Hochschule Karlsruhe - Technik und Wirtschaft

ZIEGLER, Wolfgang (2005): „Variantenverwaltung in CMS - Fünf Methoden für die Feinarbeit." In: technische kommunikation 27. Jg., Nr. 3, 40–44

ZIEGLER, Wolfgang (2009): „Content Management 2.0 - Stand und Perspektiven des Systemeinsatzes." In: technische kommunikation 31. Jg., Nr. 4, 16–18

ZIEGLER, Wolfgang / STEURER, Stephan (2010): „Mit PI-Mod dokumentieren - Standardisiertes Informationsmodell für den Anlagen- und Maschinenbau." In: technische kommunikation 32. Jg., Nr. 6, 51–55

EIDESSTATTLICHE ERKLÄRUNG

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit selbstständig angefertigt, keine anderen als die angegebenen Mittel benutzt und alle wörtlichen oder sinngemäßen Entlehnungen als solche gekennzeichnet habe.

Berlin, 19. April 2013

Jan Oevermann