M. Koch, A. Butz & J. Schlichter (Hrsg.): Mensch und Computer 2014 Tagungsband,
München: Oldenbourg Wissenschaftsverlag, 2014, S. 225-234.

Entwicklung semantischer Produktdatenmodelle durch Domänenexperten: Fehleranalyse und Werkzeugunterstützung

Werner Gaulke, Jürgen Ziegler

Universität Duisburg-Essen

Zusammenfassung

Semantisch modellierte Daten in Form von Ontologien bilden eine Grundlage für unterschiedlichste intelligente Funktionen, insbesondere ermöglichen sie eine verbesserte Unterstützung bei der Suche und Filterung von Information. Dies gilt auch im Bereich des Electronic Commerce, wo semantische Produktdatenmodelle eine zunehmende Rolle spielen. Typischerweise werden diese Modelle aber von Domänenexperten erstellt, die mit semantischen Modellierungstechniken nicht oder nur wenig vertraut sind. Dieser Beitrag stellt einen auf die E-Commerce-Domäne zugeschnittenen Editor vor, der nichttechnische Benutzer dabei unterstützt, formal und fachlich korrekte Modelle zu erstellen. Auf Grundlage einer erstellten Fehlerklassifizierung wird dieser Editor empirisch mit einer generischen Modellierungsmethode verglichen. Die Ergebnisse zeigen einerseits einen Vorteil des domänenspezifischen Werkzeugs, andererseits werden aus einer Fehleranalyse Gestaltungsvorschläge zur weiteren Optimierung domänenspezifischer Modellierungswerkzeuge abgeleitet.

1 Einleitung

Die Verfügbarkeit von Datenmodellen, welche die Anwendungsdomäne in fachlich angemessener und korrekter Weise reflektiert, bildet eine der grundlegenden Voraussetzungen nicht nur für die Entwicklung der Funktionalität eines Systems sondern auch für die Gestaltung gebrauchstauglicher Nutzerschnittstellen (Akscyn, Yoder, & McCracken, 1988). Dies gilt in besonderem Maß auch für den Bereich des Electronic Commerce, wo sinnvoll strukturierte Produktdaten leistungsfähige Mechanismen für das Durchsuchen, Filtern, Empfehlen und Präsentieren von Produkten z. B. im Online-Shop ermöglichen (s. z. B. Hearst, 2008; Jannach et. al. 2011). In den letzten Jahren werden im E-Commerce zunehmend formale Datenmodelle und –repräsentationen auf Basis semantischer Techniken

herangezogen, um eindeutige, anbieterübergreifende und vernetzbare Produktbeschreibungen zu erhalten (s. z.B. Barbau et al. 2012). Meta-Modelle wie Good Relations (Hepp, 2008) sollen die einheitliche Modellierung semantischer Produktdaten unterstützen.

Ein zentrales Problem für einen breiteren Einsatz semantischer Techniken ist in den Kenntnissen und Fähigkeiten der Zielgruppe zu sehen, die in der Praxis üblicherweise für das Erstellen von Produktdaten zuständig ist. Hierbei handelt es sich meist um Produkt-verantwortliche bei Herstellern, Shop-Betreibern oder Vergleichsportalen. Diese Personen besitzen zwar fundiertes Domänenwissen, haben aber in der Regel keinen (informations-) technischen Hintergrund oder Vorbildung in semantischen Techniken. Bisherige Werkzeuge zur Modellierung semantischer Daten wie z.B. Protégé[1] richten sich jedoch primär an Experten mit ausgeprägtem Wissen über verwendete Modellierungssprachen, wie das Resource Description Framework (RDF) oder die Web Ontology Language (OWL)[2]. Eine höhere Verbreitung semantischer Technologien erfordert deshalb zielgruppengerechte Werkzeuge.

Ein wesentlicher Ansatzpunkt, um ein handhabbares Werkzeug zur Produktdatenmodellierung für Fachexperten zu erreichen, kann in einer domänenspezifischen Reduktion der Komplexität gesehen werden. Hierbei sind mehrere kritische Abwägungen zu treffen. Zunächst stellt sich die Frage, welche Modellierungsaspekte überhaupt unterstützt werden. Im allgemeinen Ontology Engineering wird stark auf das Erstellen korrekter Klassifikationen und das Erzeugen von Relationen zwischen Ressourcen abgehoben. Aus durchgeführten informellen Analysen mit Anwendern im E-Commerce-Bereich zeigte sich jedoch, dass Produktmanager meist nur für einen engen Ausschnitt eines Sortiments zuständig sind und sich häufig vorgegebener Klassifikationsstrukturen oder –Standards wie eCl@ass, UNSPC[3] oder händlereigenen Sortierung bedienen (müssen). Hingegen bildet die korrekte Modellierung der typischerweise umfangreichen Produkteigenschaften (‚Properties' in der Diktion onto-logischer Modellierung) oft den Kern der Aufgabenstellung und bildet gleichzeitig - aufgrund vieler möglicher Fehlerquellen - ein zentrales Problem. Auch stellt sich die Frage, welche Vorgaben von Properties, Typen oder Wertebereichen die Aufgabe unterstützen können.

Im vorliegenden Beitrag stehen zur Beantwortung dieser Fragen zwei Beiträge im Fokus. Zum einen wird ein von den Autoren entwickeltes domänenangepasstes Werkzeug zur semantischen Produktmodellierung vorgestellt. Zum zweiten wird auf Grundlage verwandter Arbeiten sowie den Erfahrungen aus einem Forschungsprojekt zu diesem Thema eine Klassifikation von Fehlern bei der Erstellung ontologischer Datenmodelle aufgestellt. Diese Fehlerklassifikation wird herangezogen, um in einer empirischen Studie die Erstellung von Produktontologien in zwei unterschiedlichen Konditionen (Modellierung mittels Ontologie-beschreibungssprache vs. Modellierung mit domänenspezifischen Werkzeug) zu bewerten. Hieraus werden Hinweise für die Gestaltung fachspezifischer Ontologiewerkzeuge abgeleitet.

[1] http://protege.stanford.edu/

[2] http://www.w3.org/RDF/ bzw. http://www.w3.org/TR/owl2-overview/

[3] www.eclass.de bzw. http://www.unspsc.org/ (United Nations standard for products and services code)

2 Verwandte Arbeiten

Modellierung von Ontologien ist ein aktives Forschungsfeld, welches auf Grundlage semantischer Modellierungssprachen zahlreiche Werkzeuge und Entwicklungsprozesse hervorgebracht hat. Grundsätzlich kann die Modellierung semantischer Modelle unter Zuhilfenahme eines einfachen Texteditors direkt in der Notation der entsprechenden Sprache wie OWL oder RDF erfolgen. Demgegenüber stehen dedizierte Ontologieeditoren, die durch eine Abstraktion der zugrunde liegenden formalen Methode dem Anwender einen alternativen Zugang ermöglichen sollen. Vergleichsweise niedrige Abstrahierung erfolgt durch Editoren wie Protégé (Knublauch, Fergerson, Noy, & Musen, 2004) oder dem NeOn Toolkit (Haase et al., 2008), welche durch hierarchische Auflistungen die enthaltenen Konzepte verdeutlichen. Ein höherer Abstraktionsgrad wird in Wiki-basierten Ansätzen verfolgt. In (Bry, Schaffert, Vrandečić, & Weiand, 2012) wird ein Überblick über kollaborative Editoren auf Basis eines Wiki-Systems gegeben. Diese Editoren bilden Verknüpfungen zwischen Einträgen semantisch ab und erlauben die Angabe formaler Konzepte für beschriebene Inhalte.

Ungeachtet des eingesetzten Werkzeugs gestalten sich die Prozesse zur Modellierung einer Domäne inkrementell. Der Begriff des Ontology Maturing (Braun, Kunzmann, & Schmidt, 2012) bezeichnet den Entwicklungsprozess der schrittweisen Formalisierung abzubildender Konzepte. Während der Entwicklung kann dabei auf bewährte Entwurfsmuster zurückgegriffen werden, die sich aus Erfahrungen von Modellierungsexperten ableiten (Roussey, Corcho, & Vilches-Blázquez, 2009). Ferner ist eine Unterstützung durch das eingesetzte Werkzeug denkbar. Die Wahl sinnvoller Standardwerte und Hilfemechanismen (Rector et al., 2004) sowie die Integration automatisierter Prüfmechanismen zur Identifikation von Inkonsistenzen und Widersprüchen (Kalyanpur, Parsia, Sirin, & Cuenca-grau, 2006) sind denkbar.

Die Bewertung der Qualität erstellter Modelle erfolgt in der Regel durch Experten der formalisierten Domäne mit entsprechenden technischen Kenntnissen. Zur Unterstützung dieser Reviewprozesse stellen (Poveda-Villalón, Suárez-Figueroa, & Gómez-Pérez, 2010) eine Sammlung und Klassifizierung von typischen Fehlern in der Ontologieentwicklung vor. Die Autoren unterscheiden insgesamt 24 mögliche Fehler, die sie in zwei mögliche Kategorisierungen einsortieren. Zum einen sind die Fehler in Struktur, Funktion und Usability unterscheidbar und zum anderen wird eine Aufteilung in Konsistenz, Vollständigkeit und Prägnanz vorgeschlagen. Eine Benutzerstudie ermittelt die Verteilung der Fehler. Es häufen sich unkritische Fehler wie vergessene Annotationen. Semantische Konzepte werden meist korrekt angewendet, was der technischen Vorbildung der Teilnehmer zuzuschreiben ist.

Anhand der betrachteten Literatur lässt sich erkennen, dass sowohl Entwicklungsprozesse als auch Strategien zur Fehlervermeidung und Erkennung bis dato primär auf den technisch versierten Anwender abzielen. Vorgestellte Sammlungen von Entwurfsmustern und Fehlerarten erfüllen im jeweils angewendeten Kontext ihre Funktion, sind aber teilweise zu unspezifisch bzw. zu feingranular um allgemeiner angewendet werden zu können. Die Autoren des vorliegenden Artikels stellen im Folgenden eine generische Fehlerklassifizierung vor, die in der nachfolgenden Studie Anwendung findet.

3 Fehlerklassifizierung

An dieser Stelle wird eine Klassifizierung möglicher Fehler vorgestellt, die bei der Modellierung semantischer Daten auftreten können. Die Klassifizierung soll generell eine systematischere Untersuchung von Modellierungsfehlern ermöglichen, speziell wird sie im Weiteren für einen empirischen Vergleich zweier Modellierungsansätze herangezogen. Definierte Fehlerarten leiten sich teilweise aus den vorgestellten Arbeiten ab und wurden für eine vollständigere Abdeckung um zusätzliche Typen ergänzt. Um eine Balance zwischen Handhabung und Detailgrad zu erreichen, werden einige Fehlertypen zusammengefasst.

Auf der obersten Ebene wird zwischen formalen, semantischen und Usability-bezogenen Fehlern unterschieden. Letztere treten bei Verwendung eines Modellierungswerkzeugs auf.

Formale Fehler

Formale Fehler entstehen bei der fehlerhaften Anwendung der Konstrukte der verwendeten Modellierungssprache. Beispiele hierfür sind Syntaxfehler oder falsch eingesetzte Schlüsselwörter. Ein fehlendes oder falsches Verständnis semantischer Grundkonzepte fällt ebenfalls in diese Kategorie. Damit sind formale Fehler abhängig von der verwendeten Modellierungssprache sowie der Kenntnis des Anwenders über deren Aufbau und Konzepte.

Zur weiteren Unterteilung werden folgende Fehlertypen unterschieden:

- *[F1] Syntaxfehler*: Ausdrücke die, durch falsch eingesetzte/fehlende Symbole, nicht dem schematischen Aufbau der verwendeten Modellierungssprache entsprechen.
- *[F2] Formalsemantische Fehler*: Semantische Grundkonzepte der verwendeten Modellierungssprache sind nicht bekannt oder werden falsch eingesetzt.

Semantikfehler

In der Kategorie der Semantikfehler werden Fehler eingeordnet, die einen Sachverhalt zwar formal korrekt modellieren, aber in Hinblick auf die modellierte Domäne oder den Verwendungszweck des Modells unvollständig, falsch oder widersprüchlich sind. Darunter fallen zum Beispiel unpassende Datentypen oder zyklische Definitionen. Semantikfehler sind in der Regel nur mit fachlicher Kenntnis der jeweiligen Domäne zu identifizieren.

Es werden folgende Fehlertypen unterschieden:

- *[S1] Fachterminologische Fehler*: Es werden fachlich nicht korrekte Begriffe als Bezeichner für Klassen / Ressourcen bzw. angegebene Synonyme verwendet.
- *[S2] Eigenschaftsfehler*: Definition von nicht klassenzugehörigen Eigenschaften.
- *[S3] Typfehler*: Falsche Verwendung von Datentypen, insbesondere zu breite oder unspezifische Typen / Wertebereiche für modellierte Eigenschaften.
- *[S4] Klasse-Instanz-Verwechslung*: Entitäten werden fälschlicherweise als Klasse statt als Instanz (oder umgekehrt) definiert.
- *[S5] Kategorisierungsfehler*: Falsche Einordnung von Klassen in Hierarchien.
- *[S6] Relationsfehler:* Modellierte Beziehungen zwischen Ressourcen sind widersprüchlich, zyklisch oder führen zu falschen Beschränkungen des Wertebereichs.

Usability-bezogene Fehler

Unter Usability-bezogenen Fehlern verstehen wir Probleme, die auf eine Fehlbedienung des verwendeten Editors zurückzuführen sind, ohne dass das Modellieren selbst als problematisch einzustufen wäre. Usability-Fehler können in ihrer Konsequenz entweder zu formalen oder semantischen Fehlern führen. Die Einordnung von Problemen als Usability-Fehler kann in der Regel nur auf Basis des beobachteten Nutzerverhaltens erfolgen. Der somit erfasste Einfluss der Gebrauchstauglichkeit auf das erstellte Modell verhindert eine Irrtümliche Anlastung der Fehler an die Modellierungssprache oder das zugrundeliegende Meta-Modell.

Es werden folgende Fehlertypen unterschieden:

- *[U1] Aufwandserhöhende Fehler*: Die Zielerreichung erfordert durch ineffiziente Pfade / inkonsistente Gestaltung zusätzlichen Korrekturaufwand durch den Anwender.
- *[U2] Ergebnisverfälschende Fehler*: Die Zielerreichung wird z.B. durch falsche Bezeichner und nicht eindeutige Wahlmöglichkeiten durch das Werkzeug verhindert.

4 Ein Editor für Produktontologien

Im Rahmen eines Projektes zu ontologie-basiertem Produktdatenmanagement[4] wurde ein speziell auf den E-Commerce-Bereich zugeschnittener Produkt-Ontologieeditor (PONTE Editor) entwickelt. Mit der Domänenfokussierung geht eine Spezialisierung auf zielgruppenabgestimmte Funktionalitäten des Editors einher. Im Gegensatz zu den in Abschnitt 2 vorgestellten Ansätzen stehen hier Anwender mit ausgeprägtem Domänenwissen ohne spezielle technische Schulung im Vordergrund. Im Mittelpunkt der Tätigkeiten befindet sich die Erfassung von Produkteigenschaften für die Weiterverarbeitung bzw. Einsatz in E-Commerce Systemen. Die exakte Definition der Eigenschaften, bestehend aus Datentyp, Wertebereich, Einheit und ggf. Aufzählungen, ist für diesen Anwendungsfall entscheidend.

Den Anforderungen der Zielgruppe entsprechend verbirgt der Editor zugrunde liegende semantische Formate und nutzt die Metapher und Terminologien eines Datenblattes zur Visualisierung der modellierten Produkte. Der Editor ermöglicht die Einordnung von Produktklassen in standardisierte Kategorisierungen, welche im E-Commerce üblich sind, sowie die Definition von Produkteigenschaften. Technische Basis bildet dabei das auf E-Commerce zugeschnittene Good-Relations Meta-Modell (Hepp, 2008). Die mit dem PONTE-Editor erstellte Modelle entsprechen diesem Modell und sind im- und exportierbar.

Abbildung 1 zeigt im linken Bereich die Hauptansicht des Editors welche einen Ausschnitt einer Produktontologie für Fahrräder darstellt. Das Datenblatt unterteilt sich in grundlegende Attribute (Name, Synonyme und die verknüpfte Kategorie) sowie Produkteigenschaften. Diese unterteilen sich ferner in Standardeigenschaften (*standard product features*) wie z.B. Gewicht und Größe sowie produktspezifische Eigenschaften (*custom product features*).

[4] http://opdm-project.org

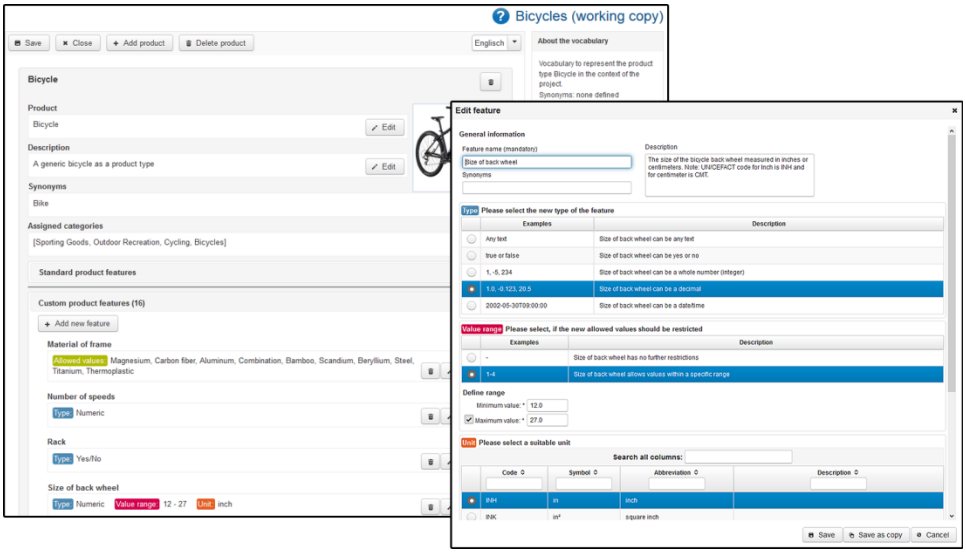

Abbildung 1 (Links) Hauptansicht des PONTE Editors. Produktklassen werden als Datenblatt dargestellt mit Auflistung aller Eigenschaften. (Rechts) Dialog zur Erstellung/Bearbeitung von Produkteigenschaften.

Produktspezifische Eigenschaften werden in einem Dialog (Abbildung 1, rechte Hälfte) bearbeitet. Die Konfiguration besteht aus mehreren Schritten. Zunächst wird der Datentyp („type“) gewählt, der den Wert String, Boolean, Ganzzahl, Fließzahl sowie Datumswert annehmen kann. Obwohl diese Auswahl einschränkend erscheint, hat die Analyse existierender Produktblätter gezeigt, dass diese Typen gängige Produktbeschreibungen lückenlos erfassen können, wenn sie zusätzlich durch Wertebereiche und die Angabe von Einheiten ergänzt wird. Diese Angabe erfolgt in Schritt zwei und drei. Für numerische Typen können hier obere und untere Grenzen angegeben, sowie passende Einheiten aus dem UN/CEFACT[5] Standard gewählt werden. Ferner besteht die Möglichkeit, Aufzählungen von konkret vorgegebenen Werten für String-Datentypen anzugeben. Im Editor vorgenommene Änderungen werden sofort übernommen und sorgen so für direktes Feedback.

5 Studie zur Onlogieerstellung

In einer empirischen Studie wurde untersucht, wie sich der Einsatz eines maßgeschneiderten Editors gegenüber einem generischen, textuellen Modellierungsansatz auf die Fehlerquote auswirkt. Im Rahmen der Studie lösen die Probanden Modellierungsaufgaben. Die Lösung erfolgt entweder mit Hilfe des in Abschnitt 4 vorgestellten PONTE Editors oder durch Formulierung von N3 Ausdrücken in einem Texteditor. Die N3 Notation[6] wurde gewählt, da diese gegenüber RDF/XML Notation einfacher nachzuvollziehen und zu formulieren ist.

[5] United Nations Centre for Trade Facilitation and E-business (http://www.unece.org/cefact.html)
[6] http://www.w3.org/TeamSubmission/n3/

5.1 Studienaufbau und Durchführung

Insgesamt 18 Probanden nahmen an der Studie teil, davon 12 weibliche und 6 männliche Teilnehmer, mit einem durchschnittlichen Alter von 22,17 (SD=3,365) Jahren. Die Teilnehmer waren Studierende eines medieninformatischen Studiengangs ohne Vorerfahrung in semantischen Techniken. Teilnahme wurde durch Vergabe von Versuchspersonenstunden motiviert. In einem Within-Subjects-Design lösten die Probanden drei Modellierungsaufgaben von ansteigender Komplexität jeweils mit beiden Editoren.

In Aufgabe 1 wurde eine Ontologie (TabletPC) bearbeitet. Teilaufgaben waren die Ergänzung von Synonymen, Entfernung/Bearbeitung von Eigenschaften sowie Einfügen eines Attributes. Inhalt von Aufgabe 2 war die Umformung einer existierenden Ontologie (Bücher) in eine verwandte Domäne (Comics). Dies umfasste Änderungen an der Produktklasse, Anpassen der Klassifizierung sowie Ergänzung von drei vorgegebenen Eigenschaften. Aufgabe 3 forderte die selbständige Definition einer neuen Ontologie (Smartphones). Anhand exemplarischer Produktbeschreibungen sollten folgende Ziele erfüllt werden: Definition der Produktklasse, Auswahl der Kategorie sowie Erstellung von mindestens drei Eigenschaften.

Testumgebung und Instruktionen

Zu Beginn wurde den Probanden Sinn und Funktion strukturierter Produktbeschreibungen erläutert. Aufgaben wurden in schriftlicher Form ausgehändigt. Vor Beginn der Arbeiten am Texteditor wurden Grundlagen des N3 Formates erläutert und ein Referenzblatt ausgehändigt. Die Reihenfolge der Editoren wurde mit jedem Teilnehmer gewechselt. Die Studie wurde an einem PC mit Windows 7 und Firefox 22 für den PONTE Editor, sowie Eclipse 4.3 als Texteditor durchgeführt. Interaktionen wurden mit der Software Morae 3.3 aufgezeichnet.

5.2 Versuchsergebnisse

Von den Probanden erstellte Modelle wurden in einem Expertenreview bewertet. Die Ergebnisse wurden in die Kategorien „gelöst“ für fehlerfreie Lösungen, „mit Fehlern“ für Modelle die einen oder mehrere Fehler enthielten, welche die Lösung nicht invalidierten, sowie „nicht gelöst“ für abgebrochene oder komplett falsche Lösungen.

Abbildung 2 zeigt die prozentuale Verteilung des Lösungsgrades der einzelnen Aufgaben nach Editor. Der zusammengefasste Lösungsgrad für den Editor beträgt 41%, für PONTE 84%. Werden mit Fehlern gelöste Teilaufgaben hinzugefügt erhöht sich der Lösungsgrad auf 65% für den Editor und 99% für PONTE.

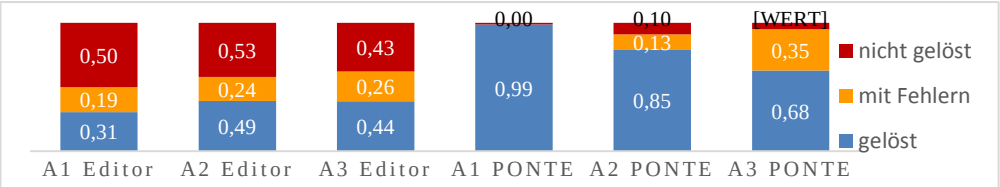

Abbildung 2: Prozentuale Verteilung des Aufgabenlösungsgrades

5.2.1 Fehlerzahlen

In den Lösungen wurden insgesamt 199 Fehler aus der in Abschnitt 3 vorgestellten Kategorisierung identifiziert. Zu beachten ist, dass der Aufgabenlösungsgrad beider Editoren unterschiedlich ausfällt. Aus diesem Grund wurde eine gewichtete Fehlerzahl errechnet, welche die rund 35% nicht erfolgreich gelösten Aufgaben mit dem Texteditor berücksichtigt. Dazu wurde die Anzahl absoluter Fehler pro Fehlertyp durch den durchschnittlichen Lösungsgrad geteilt. Abbildung 3 stellt die absoluten und gewichteten Fehlerzahlen dar.

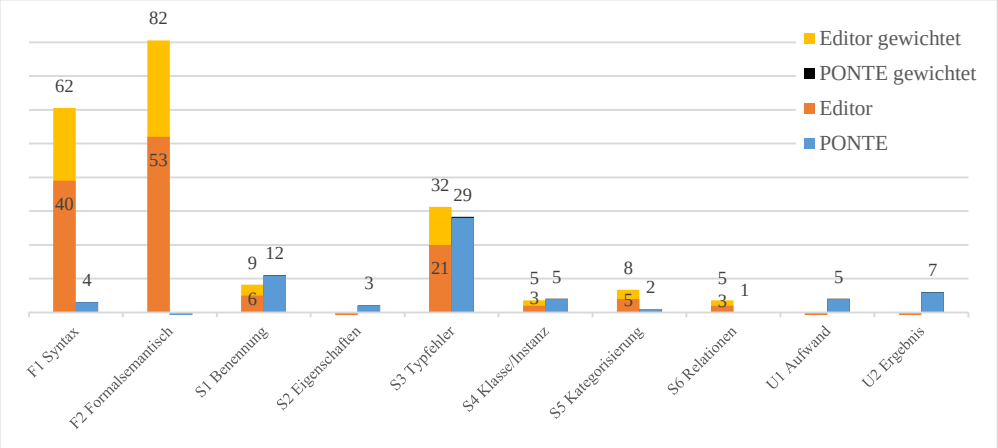

Abbildung 3: Häufigkeit der absoluten und gewichteten Fehlerarten nach Typ. Formale Fehler häufen sich im Texteditor. Semantikfehler sind in beiden Werkzeugen zu finden und konzentrieren sich auf die Benennungs- (S1) und Typfehler (S3). Für eine bessere Übersicht wird beim PONTE Editor nur der gewichtete Wert abgebildet.

Die Verteilung zeigt, dass sich im Texteditor formale Fehler häufen. Dies umfasst zum einen Syntaxfehler (F1), z.B. durch nicht abgeschlossene Statements, zum anderen aber ebenfalls die falsche Anwendung vorhandener Sprachmittel (F2). In PONTE beobachtete Syntaxfehler beziehen sich auf die Eingabe von Synonymen mit nicht korrektem Trennzeichen. Semantikfehler häufen sich im Bereich Typfehler (S3). Dieser entsteht bei falscher oder fehlender Angabe von Datentypen für Produkteigenschaften. Usability-Fehler wurden im PONTE-Editor bei der Wahl der Kategorisierung für die Produktklasse identifiziert. Einigen Anwendern gelang nicht auf Anhieb die Auswahl der gewünschten Kategorie. Als Resultat wurden entweder zusätzliche Schritte benötigt (U1) oder die Auswahl blieb fehlerhaft (U2).

6 Diskussion und abgeleitete Gestaltungsempfehlungen

Die im vorherigen Abschnitt vorgestellte Studie verdeutlicht, dass eine Beschränkung der Ausdrucksmächtigkeit auf die Erfordernisse des Anwendungsbereichs sowie das Verbergen der Formalismen den Nutzern hilft, formale Fehler bei der Ontologie-Erstellung zu reduzieren. Zwar könnten Syntaxfehler (F1) durch eine Validierung bei der freien Eingabe vermieden werden, ausgeprägtes Fachwissen semantischer Grundkonzepte ist aber weiterhin

erforderlich. Die Abstraktionen eines domänenspezifischen Editors helfen hingegen Fehler, insbesondere formale Fehler, zu verhindern. Unter Verwendung eines hinreichend spezifizierten Meta-Modells, können zudem Eingaben bis zu einem gewissen Grad überprüft werden.

Die Verteilung semantischer Fehler zeigt bei beiden Editoren gewisse Ähnlichkeiten. Insbesondere die Modellierung von Produktattributen erwies sich oft aufgrund fehlender Datentypen (Texteditor) und Einheiten (PONTE) als nicht detailreich genug. Im Falle des PONTE Editors wird zwar meist der richtige Datentyp gewählt, die Angabe einer Maßeinheit wurde aber oft vergessen oder war nicht korrekt. Andere aufgetretene Fehlertypen lassen sich hauptsächlich auf ein falsches Verständnis der modellierten Domäne zurückführen. Diese Aspekte verdeutlichen, dass eine gebrauchstaugliche Oberfläche zwar einen wichtigen Schritt zur Fehlerreduzierung darstellt, aber darüber hinaus zusätzliche Unterstützung für den Anwender nötig ist. Zum Beispiel könnte der Editor durch Beispiele motivieren Produkteigenschaften möglichst vollständig zu definieren. Auch die Integration einer Kooperationsunterstützung oder von Peer-Review-Verfahren wären zur Fehlervermeidung hilfreich.

Aus der vorliegenden Untersuchung und den gewonnenen Erfahrungen lassen sich einige Leitlinien für domänenspezifische Editoren ableiten. (1) Die Wahl einer strukturell geeigneten und *zielgruppengerechten Metapher* (im vorliegenden Fall ein Datenblatt) für die entsprechende Domäne unterstützt das Verstehen des Modellierungsziels und des erforderlichen Vorgehens. (2) Neben einer geeigneten Visualisierung ist die Wahl einer *passenden Terminologie* von hoher Relevanz. (3) Die *Verwendung eines geeigneten Meta-Modells* ist wesentlich sowohl für den Entwurf eines domänenspezifischen Editors als auch für die Vermeidung von Fehlern bei der Nutzung. Auch wird der Austausch mit externen Systemen erleichtert. (4) Zusätzlich hilft eine *Restriktion der Sprachmittel* auf eine sinnvolle Untermenge die Komplexität des Editors zu reduzieren. Die Teilmenge semantischer Konzepte ist so zu wählen, dass für die jeweilige Domäne die fachlichen Anforderungen in (fast) allen Fällen erfüllt werden können. (5) Der Editor selbst sollte gängigen Empfehlungen benutzerfreundlicher Gestaltung folgen und durch *Verdeutlichung von Aktionen* dem Benutzer die Auswirkungen von Aktionen unmittelbar darstellen. (6) Durch die *Veranschaulichung gängiger Modellierungsmuster und -beispiele* aus der entsprechenden Domäne können Anwender in ihrer Modellierungsaktivität unterstützt werden. Eine Empfehlungsfunktion könnte dem Benutzer Beispiele für den Modellierungsfall präsentieren. Im Fall von PONTE könnte z.B. die Darstellung möglicher Instanzdaten die Wahl korrekter Einheiten erleichtern.

Abschließend kann festgestellt werden, dass die Einstiegshürde der Modellierung semantischer Daten für Domänenexperten mit Hilfe eines maßgeschneiderten Werkzeuges gesenkt werden konnte. Die mit Hilfe der aufgestellten Fehlerklassifizierung durchgeführte Analyse zeigt, dass neben einer Abbildung der technischen Konzepte auf eine verständliche Metapher zusätzliche, ggf. aktive Unterstützungsfunktionen geboten werden sollten. Die vorgestellten Empfehlungen sollen helfen, schon in der Entwurfsphase maßgeschneiderter Werkzeuge relevante Fragestellungen zu beachten. Eine weitere Vertiefung dieser Gestaltungsempfehlungen wird im Rahmen von Folgearbeiten angestrebt.

Danksagung

Die vorliegende Entwicklung wurde gemeinsam mit Partnern des von der EU geförderten Projekts OPDM durchgeführt, denen an dieser Stelle herzlich gedankt sei.

Literaturverzeichnis

Akscyn, R., Yoder, E., & McCracken, D. (1988). The data model is the heart of interface design. In *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '88* (pp. 115–120). New York, New York, USA: ACM Press.

Barbau, R., Krima, S., Rachuri, S., Narayanan, A., Fiorentini, X., Foufou, S., & Sriram, R. D. (2012). OntoSTEP: Enriching product model data using ontologies. *Computer-Aided Design*, *44*(6), 575–590.

Braun, S., Kunzmann, C., & Schmidt, A. P. (2012). Semantic people tagging and ontology maturing: an enterprise social media approach to competence management. *International Journal of Knowledge and Learning*, *8*(1/2), 86.

Bry, F., Schaffert, S., Vrandečić, D., & Weiand, K. (2012). Semantic wikis: Approaches, applications, and perspectives. *Reasoning Web. Semantic Technologies for Advanced Query Answering*, *7487*, 329–369.

Gangemi, A. (2005). Ontology design patterns for semantic web content. *The Semantic Web–ISWC 2005*, *3729*, 262–276.

Haase, P., Lewen, H., Studer, R., Tran, T. D., Erdmann, M., D'Aquin, M., & Motta, E. (2008). The NeOn Ontology Engineering Toolkit. In *WWW 2008 Developers Track*.

Hearst, M. A. (2008). UIs for Faceted Navigation Recent Advances and Remaining Open Problems. In *Workshop on Computer Interaction and Information Retrieval, HCIR 2008*.

Hepp, M. (2008). Goodrelations: An ontology for describing products and services offers on the web. *Knowledge Engineering: Practice and Patterns*, 329–346.

Jannach, D., Zanker, M., Felfernig, A., & Friedrich, G. (2011). *Recommender Systems: An Introduction*. Cambridge University Press.

Kalyanpur, A., Parsia, B., Sirin, E., & Cuenca-grau, B. (2006). Repairing Unsatisfiable Concepts in OWL. *Lecture Notes in Computer Science*, *4011*, 170–184.

Knublauch, H., Fergerson, R. W., Noy, N. F., & Musen, M. A. (2004). The Protégé OWL Plugin: An Open Development Environment for Semantic Web Applications. In S. McIlraith, D. Plexousakis, & F. van Harmelen (Eds.), *The Semantic Web – ISWC 2004* (pp. 229–243). Hiroshima: Springer Berlin Heidelberg.

Poveda-Villalón, M., Suárez-Figueroa, M. C., & Gómez-Pérez, A. (2010). A double classification of common pitfalls in ontologies. In *Proceedings of the Workshop on Ontology Quality - OntoQual 2010* (pp. 1–12).

Rector, A. L., Drummond, N., Horridge, M., Rogers, J., Knublauch, H., Stevens, R., … Wroe, C. (2004). Designing user interfaces to minimise common errors in ontology development: the CO-ODE and HyOntUse projects. In *Proceedings of the UK e-Science All Hands Meeting*.

Roussey, C., Corcho, O., & Vilches-Blázquez, L. M. (2009). A catalogue of OWL ontology antipatterns. In *Proceedings of the fifth international conference on Knowledge capture - K-CAP '09* (p. 205). New York, New York, USA: ACM Press.