Ontology-Based Modelling of Web Content: Example Leipzig Health Atlas

9

Alexandr Uciteli, Christoph Beger, Katja Rillich, Frank A. Meineke, Markus Loeffler, and Heinrich Herre

Key Statements

1. The realisation of a complex web portal, including the modelling of content, is a challenging process. The contents describe different interconnected entities that form a complex structure.
2. The entities and their relations have to be systematically analysed, the content has to be specified and integrated into a content management system (CMS).
3. Ontologies provide a suitable solution for modelling and specifying complex entities and their relations. However, the functionality for automated import of ontologies is not available in current content management systems.
4. In order to describe the content of a web portal, we developed an ontology. Based on this ontology, we implemented a pipeline that allows the specification of the portal's content and its import into the CMS Drupal.
5. Our method is generic. It enables the development of web portals with the focus on a suitable representation of structured knowledge (entities, their properties and relations). Furthermore, it makes it possible to represent existing ontologies in such a way that their content can be understood by users without knowledge of ontologies and their semantics.

A. Uciteli (✉) · C. Beger · K. Rillich · F. A. Meineke · M. Loeffler · H. Herre
University of Leipzig, Leipzig, Germany
e-mail: auciteli@imise.uni-leipzig.de; christoph.beger@imise.uni-leipzig.de; katja.rillich@imise.uni-leipzig.de; frank.meineke@imise.uni-leipzig.de; markus.loeffler@imise.uni-leipzig.de; heinrich.herre@imise.uni-leipzig.de

T. Hoppe et al. (eds.), *Semantic Applications*,
https://doi.org/10.1007/978-3-662-55433-3_9

9.1 Introduction

The field of systems medicine[1] deepens the understanding of physiological and pathological processes in order to derive new diagnostic and therapeutic approaches. In addition to clinical data, extensive genomic data are processed. Data from various studies are also collected, analysed and combined. The methods for analysing and modelling are very closely linked to the data. The scientific gain in knowledge cannot be passed on alone through publications since the publication of methods and data is equally important. The preparation of the data provided by a local research group for a broad user community requires comprehensive research and data management, but also a carefully thought-through data-sharing concept.

The Leipzig Health Atlas (LHA)[2], launched in 2016, delivers a multifunctional, quality-assured and web-based repository of health-relevant data and methods (models and applications) for a broad research population. Partner teams in Leipzig contribute extensive data, methods and experience from clinical and epidemiological studies, research collaborations in systems medicine, bioinformatics and ontological research projects. The LHA brings together ontologists, modellers, clinical and epidemiological study groups, bioinformaticians and medical informaticians.

The LHA manages extensive content and representation metadata on the publications, data and methods of the participating research projects. The web portal of the LHA serves as a shop window and marketplace for data and innovative methods (models and applications). Depending on the legal regulations, clinical and genomic microdata can be downloaded directly or via appropriate access controls. Where applicable, applications and models can be run interactively in the portal and evaluations can be carried out on an ad hoc basis.

The creation of a complex web portal, including modelling the content, is a challenging process. The contents describe different interconnected entities and have a complex structure. The entities and their relations have to be systematically analysed, and the content has to be specified and integrated into a content management system (CMS). The ontology provides a suitable solution for modelling and specifying complex data and their dependencies. However, since automated import of ontologies in web portals is lacking, we have focused on this problem.

In order to describe the metadata on the projects, publications, methods and datasets to be represented in the LHA portal, we developed an ontology. Based on this ontology, we implemented an ETL (extract/transform/load) pipeline (Fig. 9.1) that allows the specification of the portal's content and its import into the CMS Drupal (Version 8).

---

[1] "*Systems Medicine is the implementation of Systems Biology approaches in medical concepts, research and practice. […]*" (https://www.casym.eu/what-is-systems-medicine/).

[2] Funded by the German Ministry of Education and Research (reference number: 031L0026, program: i:DSem – Integrative Datensemantik in der Systemmedizin).

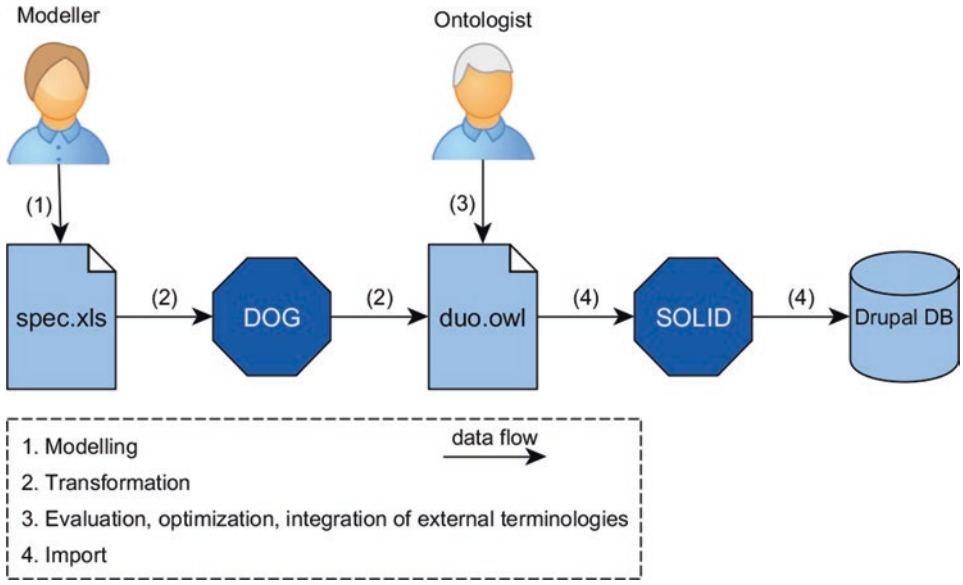

Fig. 9.1 Pipeline for the import of content in Drupal

The pipeline consists of the following four steps:

1. Modelling of the content using a spreadsheet template (Fig. 9.3).
2. Transformation of the domain-specific entities from the spreadsheet template to the ontology using the Drupal Ontology Generator (DOG).
3. Optional optimization of the ontology using an ontology editor including the import of external ontologies/terminologies.
4. Importing the ontology into Drupal's own database using the Simple Ontology Loader in Drupal (SOLID).

The approach and the individual components will be discussed in detail in the following sections.

9.2 Content Specification of the LHA Portal

For the metadata specification of the projects, publications, data and methods, a metadata model of the LHA was developed (Fig. 9.2). The metadata model consists of three interconnected levels (entity types).

Publications can be assigned to several projects. The datasets (OMICS-datasets [1], clinical trial data, and other specific datasets) and associated methods are mostly assigned to publications and form the lowest level for capturing the accompanying metadata. It is

possible to refer comprehensive datasets to more than one publication. References between entities are realized using IDs.

The collection and processing of metadata is based on a spreadsheet software. This allows for a flexible approach in the development phase.

The metadata model (Fig. 9.2) is implemented in spreadsheets (Fig. 9.3) and it forms the basis for the collection of metadata. The spreadsheet queries specific information in the individual worksheets for the respective entity types (project, publication, OMICS-dataset, clinical dataset, and method).

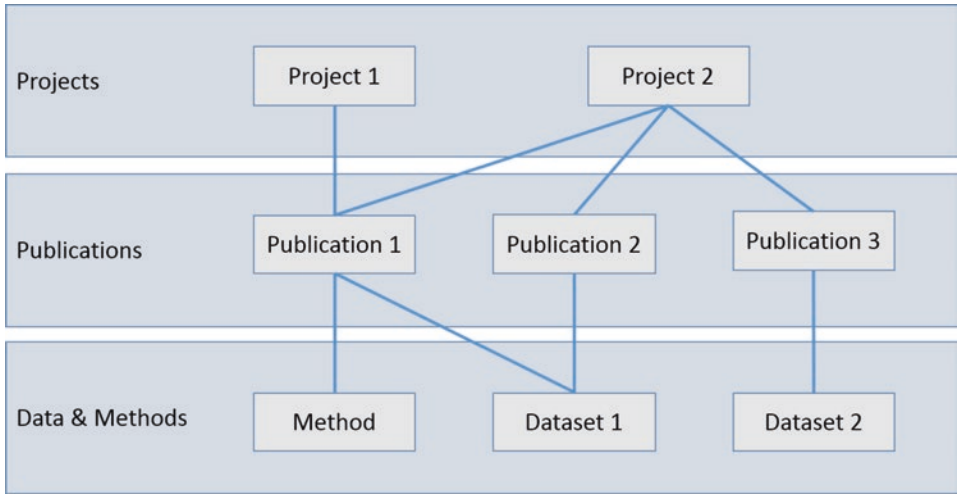

Fig. 9.2 Metadata model of the LHA

Merkmal	Format	Frage	Antwort
title	text	Bitte geben Sie den Titel des Projektes/der Studie an:	Leipzig Research Center for Civilization Diseases. Head and Neck Group
field_study_shortcut	id	Gibt es ein Projekt-/Studienkürzel? Falls ja, wie heißt das Kürzel?	LIFE-HNG
field_study_grouplink	text	Link zur Webseite der Studiengruppe:	http://life.uni-leipzig.de/
field_founding_year	integer	In welchem Jahr wurde das Projekt/die Studie begonnen?	2012
field_publication_date	date	Publikationsdatum:	24.06.2015
field_sponsor	text list	Durch welche(n) Förderer wird das Projekt/die Studie finanziell unterstützt?	Leipzig Research Center for Civilization Diseases (LIFE) University Leipzig European Union, the European Fund for Regional Development (EFRE) Free State of Saxony
field_disease	taxonomy_reference list	Bitte nennen Sie einige Stichwörter zu diesem Projekt/der Studie (bitte nennen Sie dabei auch die erforschten Krankheiten):	head and neck squamous cell carcinoma| human papillomavirus
field_author	node list	Autoren (werden von uns ergänzt):	Wichmann G| Rosolowski M| Krohn K| Kreuz M| Boehm A| Reiche A| Scharrer U| Halama D| Bertolini J| Bauer U| Holzinger D| Pawlita M| Hess J| Engel C| Hasenclever D| Scholz M| Ahnert P| Kirsten H| Hemprich A| Wittekind C| Herbarth O| Horn F| Dietz A| Loeffler M| Leipzig Head and Neck Group (LHNG).

Project | Publication | OMICS-Dataset | Clinical Dataset | Method

Fig. 9.3 Example of a spreadsheet for metadata collection

The definition of which meta information must be queried is based on different sources and standards. The higher-level archive functionality of the LHA is based on the OAIS (Open Archival Information System) ISO standard [2]. The OAIS metadata model used for the LHA was supplemented in further steps and compared with broadly generic publication-related standards (for example, schemas of the MEDLINE/PubMed database [3]). If a publication is listed in MEDLINE, it is sufficient to enter the MEDLINE ID, and the corresponding bibliographical data can be completed automatically. For the definition of domain-specific properties of genetic and clinical data, schemes of existing data portals, e.g., GEO [4], TCGA [5], cBioPortal [6] and CGHUB [7], were taken into account. Properties perceived as missing were added. The resulting metadata list was reviewed and revised based on application to existing projects with a wide range of entity types (e.g., publication, dataset, method) and data types (e.g., text, date, number, reference) jointly with the responsible scientists. In doing so, irrelevant requirements were eliminated, the metadata queries were linguistically defined and new aspects were included. In order to check the data acquisition and display of the metadata, the data tables from each entity type and from each data type were filled with several examples and loaded into the content management system using our pipeline (Fig. 9.1).

In addition to the bibliographic data, the spreadsheet template gather further information on the contents of the projects, publications and data records in such a way that the context of a project or the publications and associated datasets is known before the data is downloaded or a request for access to the data is made. At the project level, the content of this website includes, e.g., links to existing project websites, information on the objectives of the projects, the funding and sponsors, information regarding specific questions on data management and biometrics, as well as annotations with concepts of external terminologies. On the publication level, the abstract, the link to the original publication, the data on sponsors, relevant keywords and authors are recorded. At the dataset level, the content of the datasets including case numbers and design is briefly described. Additionally, information on the responsible scientists (name, address, email address, perspective ORCID (an identifier for scientists)) are collected at all levels.

Depending on the context, the metadata itself is entered as a link, text, text enumeration separated by a concatenation character (vertical stroke), numeric entries or as a date.

9.3 Ontological Architecture

We developed the Drupal Upper Ontology (DUO), which models the standard components of Drupal (field, node, file and vocabulary). According to the 3-Ontologies-Method [8], DUO is a task ontology, i.e., an ontology for the problem to be solved by the software. Furthermore, we implemented a domain ontology, the Portal Ontology of LHA (POL), which is embedded in DUO and used to model the content of the portal. For the integration and formal foundation of both the task and domain ontologies, we used the General Formal Ontology (GFO) [9, 10] as a top-level ontology (Fig. 9.4).

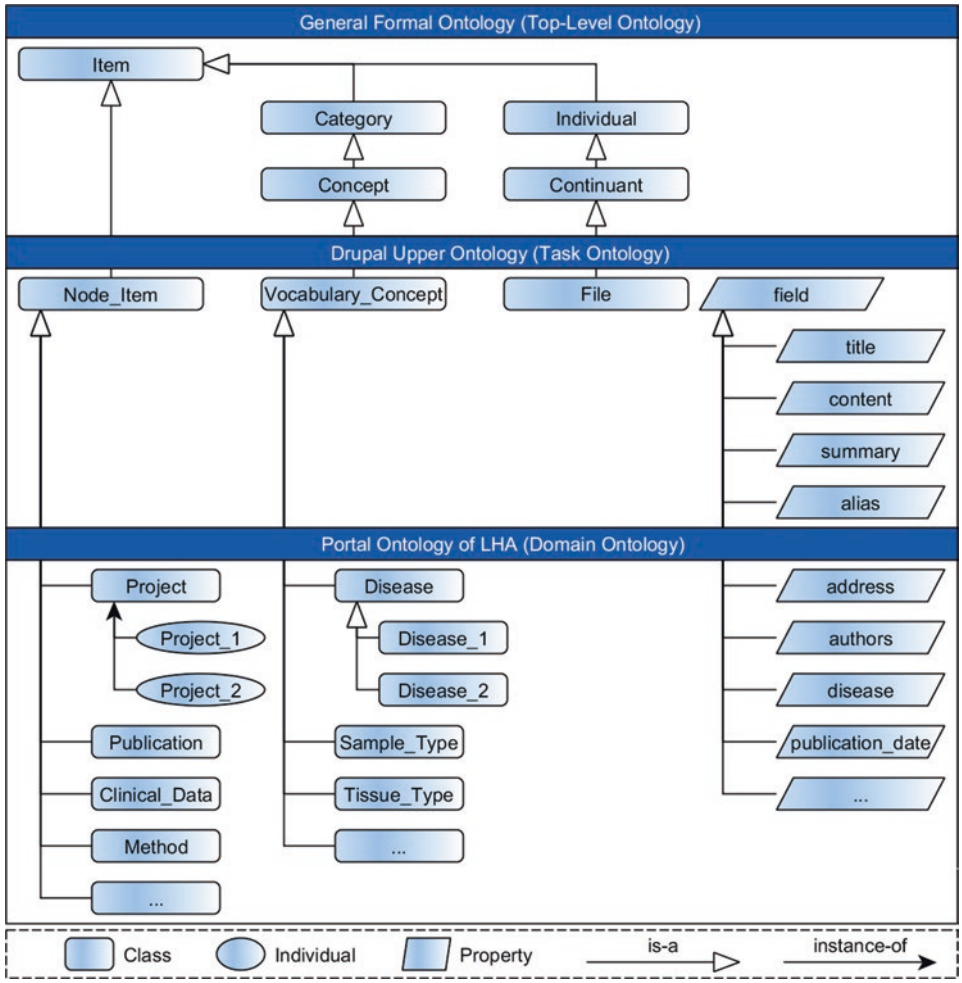

Fig. 9.4 Ontological architecture

According to GFO, we distinguish between symbolic structures (e.g., the content of web pages, such as text and images) and entities (categories or individuals, such as persons or projects) that are represented by the symbolic structures. For the sake of simplicity, we only model the entities to be represented in the web portal, while their representations on the web pages are generated. To reference entities of a particular ontology the notation <ontology_name>:<entity_name> is used in this chapter. For example, the class Node_Item from DUO is designated as duo:Node_Item.

Since both individuals and categories can be represented in a portal, we derive the class duo:Node_Item, to model entities to be represented, from the class gfo:Item, which has the classes gfo:Individual and gfo:Category as subclasses. The class duo:Vocabulary_Concept is used for integrating the concepts of external ontologies/terminologies (e.g., disease or

phenotype ontologies) and is derived from gfo:Concept. We consider the files (duo:File) as continuants (gfo:Continuant) in the GFO sense, since they are concrete individuals with a certain lifetime.

The categories of DUO are specialized and instantiated in POL. Various entity types, e.g., pol:Project, pol:Method and pol:Clinical_Data are defined as subclasses of duo:Node_Item, and concrete instances of these classes are created and linked. In addition, external terminologies (such as classifications of diseases) are referenced in POL, so that the POL entities can be annotated with their concepts.

Both the Drupal fields (such as "title" or "content") and the user-defined domain-specific fields (such as "address", "author" and "disease") are modelled as annotation properties and used for description and linking of the instances in POL.

9.4 Drupal Ontology Generator (DOG)

We developed the Java application Drupal Ontology Generator (DOG) to transformation the domain ontology from the spreadsheet template to Web Ontology Language (OWL).

When reading a completed spreadsheet template (Fig. 9.3), the DOG interprets each data sheet as a representation of one or more instances of a particular type/class. If a datasheet, for example, is called "Project", then specific individual projects are represented/described on this datasheet. For each datasheet name, the DOG generates a subclass of the class duo:Node_Item (if the class does not already exist) and creates the appropriate instances of that class based on their properties. The DOG passes through all specified properties line by line and varies its procedure depending on the defined format.

If "id" is selected for the format field of a property in the spreadsheet, the value of the property is used to generate the Internationalized Resource Identifier (IRI) of the instance and allows referencing of the instance in the same or other files.

When specifying one of the default data types ("text", "integer", "double", "date") in the format column, an annotation is created. The annotation property is used with the name specified in the column "Merkmal" (the German for "feature"), the data type defined in the column "Format", and the value entered in the column "Antwort" (the German for "answer"). If the annotation property with the desired name does not already exist, it is generated as a subproperty of duo:field.

If "taxonomy_reference" or "taxonomy_reference list" is selected in the format column, a subclass of the class duo:Vocabulary_Concept is generated, which is named using the property (without the "field" prefix, e.g., "Disease" from "field_disease") and represents the root node of the corresponding vocabulary. Subclasses of the root class are created for all values of the property (for example, various diseases). Next, an annotation is created that links the corresponding instance to the vocabulary class representing the desired disease (e.g., a link joining a project instance with the vocabulary concept of the disease it deals with). In this way, it is modelled that the current instance is tagged/annotated by certain concepts of defined vocabularies.

The “node”, “nodelist”, “node_reference”, and “node_reference list” formats are used to create links between individual instances. For this purpose, we also use annotation properties. In addition to the defined relations, all instances specified in a spreadsheet file are linked together. The names of the required annotation properties are formed from the class names of the two entities to be linked. For example, the annotation property for linking the instances of the classes “Project” and “Publication” has the name “field_project_publication”, and the inverse property is named “field_publication_project”.

All list formats (the format name ends with “list”, e.g. “node list”) allow the specification of multiple values. The order of the values may be important, for example for authors of a publication. The order is represented in the ontology by “annotation of annotation”, i.e., annotating the corresponding annotation (for example “field_author”) using the property “ref_num” and the specification of the sequence number.

Another important function of the DOG is the generation of the directory structure for storing files (for example, datasets, images, applications, etc.) to be imported into the LHA. The DOG proceeds as follows. For each project, a directory is generated that contains a subdirectory for each of all related instances (i.e., all publications, records, methods, etc.). The subdirectories, for their part, are divided into “public” and “private”. All directories are generated only if they were not already present. The DOG also links the ontology with the directory structure (under duo:File). If a file exists in one of the directories when the directory structure is generated, the DOG creates an instance of the corresponding directory class in the ontology and annotates it with the file path.

9.5 Simple Ontology Loader in Drupal (SOLID)

The content management system (CMS) Drupal facilitates the creation of web content (nodes) by providing simple web forms. Additionally, it allows the annotation of content with terms of self-defined vocabularies. Different node types can be defined and may be provided with fields. Fields serve as containers for the information of a concrete node. The fields support simple data types such as character strings or numbers, as well as complex types such as files and references to other nodes or vocabulary terms. However, large amounts of content to be managed can yield a complex interconnection of nodes and terms. Therefore, ontologies would be suitable for the modelling of content.

In order to enable users to import ontologies into Drupal, we have developed the Drupal module Simple Ontology Loader in Drupal (SOLID) [11]. SOLID supports both, ontologies generated by the Drupal Ontology Generator (DOG) as well as any other standard ontology (e.g., downloaded from BioPortal). Ontologies merely have to be integrated into the Drupal Upper Ontology (DUO). The module is PHP-based and interacts directly with the Drupal API. Hence, the created content does not lead to collisions or inconsistencies in Drupal’s Database Management System.

SOLID is based on Drupal’s module architecture and must be installed in Drupal (Version 8) to be functional. The module is accessible from the administration section of

Drupal and provides a small form for data upload and configuration to simplify the import process. Attention should be paid to the fact that nodes can only be imported if a respective node type was created prior to the import. For each property in an ontology, there must exist a corresponding field in Drupal. Automated creation of fields is not supported by SOLID because the required configuration parameters for each field are too extensive to add them into an ontology. It is much more appropriate to use the user interface provided by Drupal to create the fields. Regarding the LHA instance, we had to create the node types "project", "publication", "clinical dataset" and so forth with their respective fields (as described in Sect. 9.2). Additionally, Drupal provides the functionality to manage files (e.g., data sets or applications). Prior to the importing the ontology however, these files must be placed on the server according to the properties of the respective duo:File instance.

Hereafter, the structure and functionality of SOLID are described briefly. The module contains two types of components: parsers and importers (Fig. 9.5). Parsers are responsible for the processing of uploaded input files. OWL and JSON are supported, but this section will focus on the import of the OWL ontologies. Importers (node- respective vocabulary-importer) interact with the Drupal API, to check for existing entities and to create new ones.

In the LHA pipeline, SOLID receives the Portal Ontology of LHA (POL) from the DOG as an OWL file. The file is processed by the OWL parser (based on EasyRDF [12]). The parser extracts each subclass of the duo:Vocabulary_Concept and transmits them to the vocabulary importer. The importer creates a vocabulary in Drupal for each direct subclass of duo:Vocabulary_Concept and it adds all descending classes as terms into the vocabulary. In this step subclass/superclass relations are preserved and saved as hierarchies. Depending on the configuration, the OWL parser searches for instances of respective subclasses of duo:Node_Item in the ontology. Besides standard properties of nodes such as title, node type and alias, the parser also collects data, object and annotation properties and delivers all found properties to the node importer, where all nodes are inserted

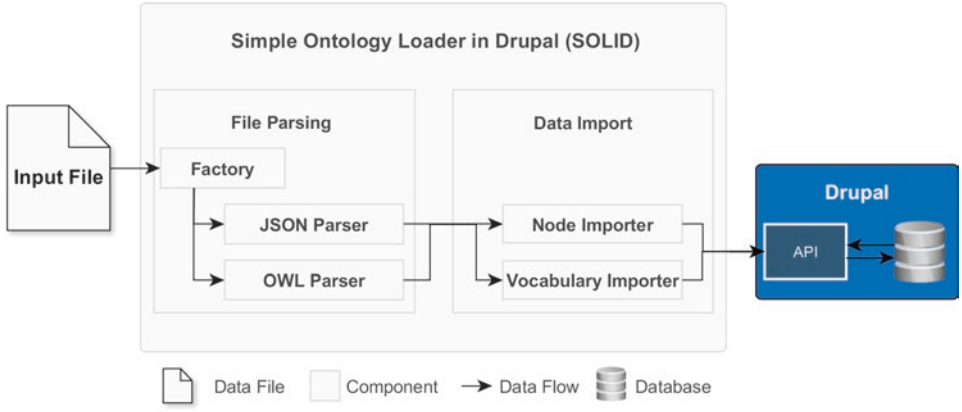

Fig. 9.5 Architecture of the Simple Ontology Loader in Drupal

into the CMS (Fig. 9.6). In case a property references another entity in the ontology, the respective field cannot be inserted into the database immediately, because referenced nodes are potentially not yet processed and created. Therefore, referencing properties are processed after all nodes are created.

Drupal uses a Universally Unique Identifier (UUID) for a bijective identification of content which is stored in the database. To guarantee a connection between nodes after import and their source entities in the ontology, we use the entities Internationalized Resource Identifiers (IRI) as UUIDs in Drupal. By this means, the module can determine if a class or individual which is extracted by the parser already exists in the database. In

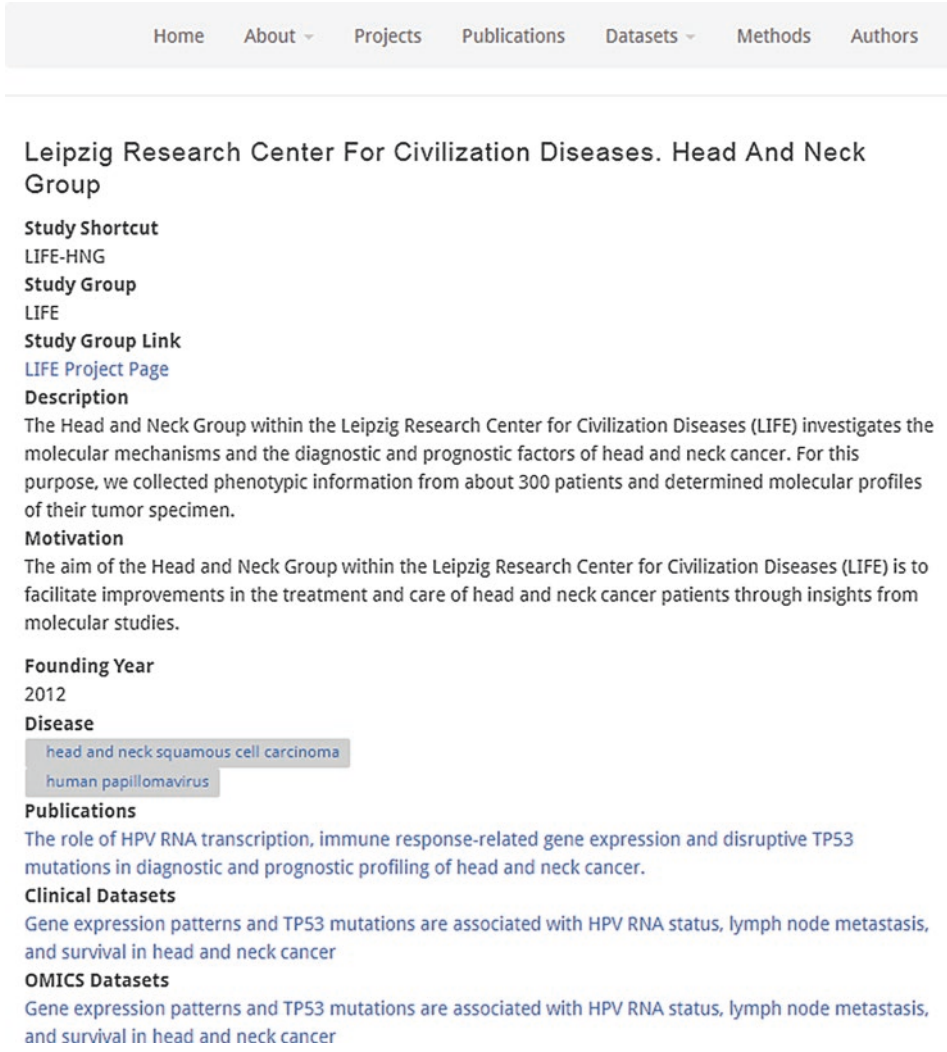

Fig. 9.6 Generated page example in the Leipzig Health Atlas web portal

case the ontological entity was imported earlier, the former node is expanded by a new revision, which contains the new fields.

The described utilization of SOLID requires the use of the web form for upload and configuration to simplify the import process. But it is also possible to direct the module via command line, to e.g., create a periodic import. New spreadsheet files may be placed in a directory of the server's file system so that DOG can create an OWL file which SOLID can subsequently import.

9.6 Recommendations

Our generic approach offers a solution to two kinds of problems:

1. *Development of web portals*

Our method is usable for the development of web portals with the focus on a suitable representation of structured knowledge. The following criteria should be satisfied for the development of a web portal based on our approach:

- Different types of entities having certain properties should be represented.
- There are different relationships between the particular entities.
- The entities to be represented should be annotated with concepts of terminologies/ontologies to simplify the search.
- The content to be represented is dynamic.

In this case, the entities, their properties and relations are modelled using a spreadsheet, transformed into OWL by the DOG, and loaded in Drupal by the SOLID.

Our approach is not suitable for representing static or one-dimensional content (such as blogs), or for creating portals that require complex program logic or interaction with the user (such as forms).

2. *Representation of existing ontologies*

The number of ontologies which are available for widespread domains is growing steadily. BioPortal alone embraces over 500 published ontologies with nearly eight million classes. In contrast, the vast informative content of these ontologies is only directly intelligible by experts. To overcome this deficiency, it could be possible to represent ontologies as web portals, which do not require knowledge of ontologies and their semantics, but still, carry as much information as possible to the end-user [11]. Using our approach, ontological entities are presented to the user as discrete pages with all appropriate properties and links (to internal or external pages and files).

9.7 Conclusion

In this chapter, we presented an approach for specifying and automatically loading the contents of web portals into the CMS Drupal. Our approach has successfully been applied in building the LHA portal ([13], the layout of the portal is still under development), which makes available metadata, data, publications and methods from various research projects at the University of Leipzig. Ontologies have proven to be a suitable tool for modelling complex contents of web portals. Our pipeline facilitates the specification of the content by domain experts and replaces the manual input of the data in Drupal by an automated import.

Our method is generic. On the one hand, it enables the development of web portals with the focus on suitable representation of structured knowledge. On the other hand, it makes it possible to represent existing ontologies in such a way that their content is intelligible for users without background knowledge about underlying ontological entities and structures (e.g., the distinction between concepts, individuals, relations, etc.). The representation of ontological entities as traditional web pages and links facilitates access to the semantic information and improves the usability of the ontologies by domain experts.

To import an existing domain ontology into Drupal using SOLID, only a few relatively simple modifications are required. To avoid errors during the import process, some restrictions and requirements for the ontology design needed to be defined. The ontology has to be embedded in DUO, i.e., their classes and properties have to be derived from those of the DUO. Only classes and properties that are defined in DUO and are specialized or instantiated in the domain ontology are processed by SOLID. The classes whose instances are to be represented as web pages (nodes) have to be defined as subclasses of duo:Node_Item, while the root nodes of the external terminologies have to be placed under duo:Vocabulary_Concept. All annotation properties have to be subproperties of the duo:field, and their names have to match the names of the fields created in Drupal.

Our approach is a promising solution for the development of complex web portals. Additionally, it can be applied to make existing ontologies available. Future applications should be established and evaluated in further projects.

References

1. Horgan RP, Kenny LC (2011) “Omic” technologies: genomics, transcriptomics, proteomics and metabolomics. Obstet Gynaecol 13(3):189–195
2. ISO 14721:2012. Space data and information transfer systems – Open archival information system (OAIS) – Reference model. https://www.iso.org/standard/57284.html
3. MEDLINE/PubMed XML data elements. https://www.nlm.nih.gov/bsd/licensee/data_elements_doc.html
4. Gene Expression Omnibus (GEO). https://www.ncbi.nlm.nih.gov/geo/
5. Hanauer DA, Rhodes DR, Sinha-Kumar C, Chinnaiyan AM (2007) Bioinformatics approaches in the study of cancer. Curr Mol Med 7(1):133–141(9)

6. Cerami E, Gao J, Dogrusoz U, Gross BE, Sumer SO, Aksoy BA, Jacobsen A, Byrne CJ, Heuer ML, Larsson E, Antipin Y, Reva B, Goldberg AP, Sander C, Schultz N (2012) The cBio cancer genomics portal: an open platform for exploring multidimensional cancer genomics data. Am Assoc Cancer Res. https://doi.org/10.1158/2159-8290.CD-12-0095
7. Grossman RL, Heath AP, Ferretti V, Varmus HE, Lowy DR, Kibbe WA, Staudt LM (2016) Toward a shared vision for cancer genomic data. N Engl J Med 375:1109–1112. https://doi.org/10.1056/NEJMp1607591
8. Hoehndorf R, Ngomo A-CN, Herre H (2009) Developing consistent and modular software models with ontologies. In: Fujita H, Marik V (eds) New trends in software methodologies, tools and techniques: proceedings of the Eighth SoMeT_09. Volume 199. IOS Press, pp 399–412. [Frontiers in Artificial Intelligence and Applications]
9. Herre H, Heller B, Burek P, Hoehndorf R, Loebe F, Michalek H (2006) General formal ontology (GFO): a foundational ontology integrating objects and processes. Part I: basic principles (Version 1.0). Onto-Med report. Research Group Ontologies in Medicine (Onto-Med), University of Leipzig
10. Herre H (2010) General formal ontology (GFO): a foundational ontology for conceptual modelling. In: Poli R, Healy M, Kameas A (eds) Theory and applications of ontology: computer applications. Springer, Dordrecht, pp 297–345
11. Beger C, Uciteli A, Herre H (2017) Light-weighted automatic import of standardized ontologies into the content management system Drupal. Stud Health Technol Inform 243:170–174
12. Humfrey N. RDF library for PHP. http://www.easyrdf.org/
13. Leipzig Health Atlas (LHA). https://www.health-atlas.de/