ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-TUBPL01

AUTOMATIC WEB APPLICATION GENERATION FROM AN IRRADIATION EXPERIMENT DATA MANAGEMENT ONTOLOGY (IEDM)*

B. Gkotse†,1,2, P. Jouvelot1, F. Ravotti2
1CERN, Geneva, Switzerland
2MINES ParisTech, PSL University, Paris, France

Abstract

Detectors and electronic components in High-Energy Physics experiments are nowadays often exposed to harsh radiation environments. Thus, to insure reliable operation over time, their radiation tolerance must be assessed beforehand through dedicated testing experiments in irradiation facilities. To prevent data loss and perform accurate experiments, these facilities need to rely upon a proper data management system.

In prior work, we provided a formal description of the key concepts involved in the data management of irradiation experiments using an ontology (IEDM). In this work, we show how this formalisation effort has a practical by-product via the introduction of an ontology-based methodology for the automatic generation of web applications, using IEDM as a use case. Moreover, we also compare this IEDM-generated web application to the IRRAD Data Manager (IDM), the manually developed web application used for the data handling of the CERN Proton Irradiation facility (IRRAD). Our approach should allow irradiation facility teams to gain access to state-of-the-art data management tools without incurring significant software development effort.

INTRODUCTION

Ontologies have been used in Artificial Intelligence (AI) for years for various purposes such as knowledge formalisation, interoperability, complex querying and inference [1]. Nowadays, some companies choose to base their information systems on ontologies [2] and knowledge graphs [3], the descendants of ontologies, in order to allow for better data integration and communication. Ontologies have been shown to be suitable for domain-specific knowledge formalisation, and are broadly used in various domains such as biomedicine [4], bioinformatics [5] and law [6]. In addition to formalisation, ontologies can have many practical applications, which were not explored deeply so far. We focus here on one particular application, namely the management of ontology-related data.

Data management is an important issue in several scientific domains [7], and in the High-Energy Physics (HEP) as well. This is true when running physics experiments at the CERN Large Hadron Collider (LHC) or similar accelerator infrastructures, but also at earlier stages, during their development. Detectors and electronic components used in HEP experiments are often exposed to harsh radiation environments. Thus, to insure reliable operations over time, their radiation hardness must be assessed beforehand through dedicated testing experiments in irradiation facilities. To prevent data loss and perform accurate experiments, these facilities need also to rely upon a proper data management system.

In our previous work [8], we introduced the domain ontology for Irradiation Experiment Data Management (IEDM), which formalises the concepts involved in irradiation testing experiments. In this paper, we introduce both a new methodology for generating automatically web applications from domain ontologies and our ontology-driven, data management web application generator (GenAppi). In this way, we enable non-computer scientists to easily build and deploy such applications. We use IEDM as a representative example to illustrate our approach.

An interesting by-product of our proposal is that GenAppi-generated web applications end up being actually enriched beyond their initial intended goal by the presence of a new underlying ontology, the Ontology-based Web Application Ontology (OWAO), and other web semantic technologies. More specifically, these generated web applications can rely upon well-established open standards, while user data are not only stored in private databases but also as interconnected knowledge graphs. This opens opportunities for both developers and domain experts, for instance in terms of inference or coherence checking.

The structure of the paper is as follows. In the second section, we provide some background information about ontologies, describe relevant user-interface ontologies and discuss related work about ontology-based user interface generation. In the third section, our new methodology is described, focusing on the OWAO ontology designed to enable the description of the concepts and axioms integrated in our web application generator (GenAppi). In the fourth section, we present the IEDM ontology, used as an example ontology to demonstrate the functionalities and User Interface (UI) features of automatically generated web applications. In the fifth section, we compare the IEDM-derived web application to the IRRAD Data Manager (IDM), a custom-made web application currently used in the CERN proton irradiation facility (IRRAD)[1] [9]. Finally, in the sixth section, we conclude our work and present our ideas regarding possible

* This work has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement no. 654168 (AIDA-2020).

† Blerina.Gkotse@cern.ch

[1] http://cern.ch/ps-irrad

extensions of the GenAppi-derived functionalities, including UI customisation, as future work.

STATE OF THE ART

This section describes briefly what ontologies are and how they are defined, present examples of UI ontologies and discuss related work about ontology-based UI generation.

Ontologies

The term "Ontology" derives from the ancient Greek words ὄν (on), the present participle of the verb "to be", and λόγος (logos), which stands for (one) who speaks (in a certain way), or (one) who treats of (a certain subject). Initially, the term was used in philosophy when referring to the subject of existence [2], the science of being, and, more simply, to "what exists" in the world [10]. As in philosophy, also in computer science ontologies are used for the formal description and classification of "what exists". However, in computer science, "what exists" is what can be explicitly represented as knowledge. Therefore, an ontology is an explicit specification of a conceptualisation [11], a model that defines concepts and relations among them for representing an area of knowledge or domain. In detail, an ontology is a set of domain-specific definitions and descriptions specified using the following concepts:

- **class**, i.e., an entity of the domain;
- **property**, i.e., an attribute, of specific type (e.g., string, integer, etc.), that helps to describe a class – such an attribute is also called a **data property**;
- **relation**, i.e., a link between two or more classes in order to describe a semantic relation among them – a relation is often called an **object property**, as well.

The primary purpose of defining and using an ontology is to reach a common understanding of the structure of a specific information among people or software agents. In practice, ontologies are thus used for sharing domain information and to foster interoperability. Furthermore, ontologies contribute in analysing and reusing domain knowledge [12].

An ontology can be represented by a graph-like or only tree-like structure where the nodes are the classes and the edges are the relations[2]. Once an ontology is defined, instances of these classes can be created and linked together with the purpose of representing or annotating datasets and resources. Nowadays, an ontology is considered to be the sole schema describing the interrelations and restrictions of resources whereas the whole set of schema and instances is referred to as a knowledge base (KB) or knowledge graph (KG) [3], term coined by Google [13].

Ontologies and KGs can be specified by languages dedicated to ontology description; each has its own structure and syntax. The languages that have prevailed nowadays because of their simplicity and expressiveness are the Resource Description Framework (RDF) [14], the Resource Description Framework Schema (RDFS) [15] and the Web Ontology Language (OWL) [16]. These standards are used in this work.

[2] In the case of a tree, the ontology is actually a taxonomy, since classes are only organised in a hierarchical manner, representing a specific classification and inheritance.

UI Ontologies

The formalisation and standardisation of knowledge that ontologies provide can be used for the description of the UI components that form the fabrics of the vast majority of computer software. In the literature, there have been several attempts to describe at least part of this type of knowledge [17–19]; UI ontologies have been developed to fulfil specific requirements and describe certain parts of a UI or web application. For example, the Semantic UI ontology includes concepts related to the interface elements of its own UI framework, Semantic UI [17]. Even though this ontology seems suitable for our work, it does not include concepts related to the visualisation of the elements (e.g., font size), since Semantic UI has specific predefined themes that are customizable only at compile time.

Concepts such as font size are, however, included in the UI ontology of [18] supported by the community behind Linked Open Vocabularies (LOV) [20]. This ontology describes concepts of UI elements, relates these entities to properties of style (e.g., colour or background colour) and is suitable for describing forms and sequences in widgets.

Yet, both ontologies describe only the graphical front-end part of a user interface, and they do not address concepts related to actual data operations. Such an issue is treated by the RDFa[3] User Interface Language (RaUL) [19], a user interface ontology used for the description and structure of web forms as RDF statements that introduces concepts such as *CRUDOperation* about specific operations (Create, Read, Update and Delete, thus CRUD).

Ontology-based UI Generation

The idea of automatically generating UIs based on ontologies has been envisioned before. One example is the work about ontology-based UI development where a User Interface Ontology (UIO) is used to describe UI properties and their semantic relationships [21]. By the use of UIO and a VCards[4] domain ontology, mappings between these two ontologies are created and used to instantiate a user interface. Even though this idea is similar to our work, it does not get up to the point of automatically generating user interfaces, as we do. Moreover, UIO is neither properly documented nor publicly available, and therefore this work is not readily reusable.

Another example, where an ontology of user interfaces is used, is of Hierarchical User Interface Component Architecture (HUICA) [22]. This ontology describes the whole process of user-interface development from the wireframes to a Model-View-Controller (MVC) architecture for UI interactive components. This ontology is divided in three parts:

[3] RDFa provides ways to add metadata annotations to Web documents.
[4] VCards is a file format for electronic business cards.

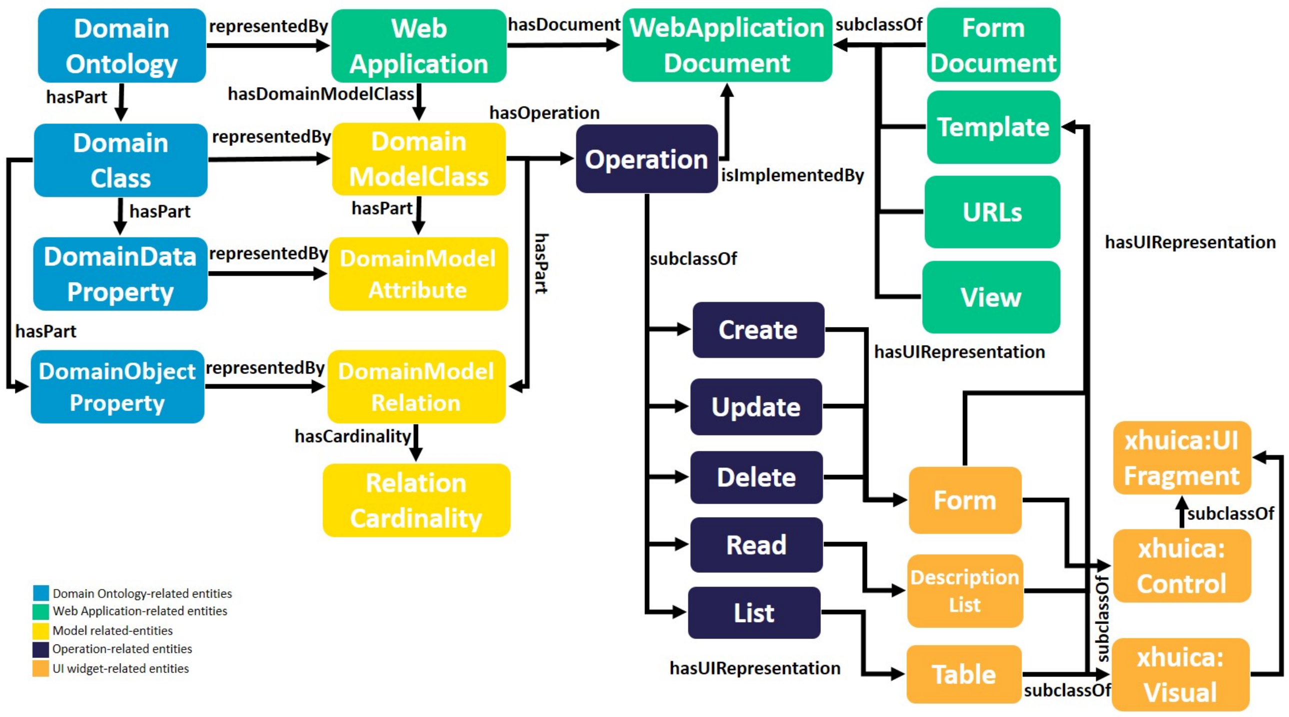


Figure 1: OWAO excerpt.

the design part, where concepts such as wireframes are described; the UI part, where UI elements such as widgets or composites are included; and the last part, used for an MVC architecture. Since the main focus of this work is the HUICA architecture, the author provides only a visual schema of this ontology without providing the ontology in a language format (e.g., RDF). Moreover, the MVC architecture is used only in the front end for UI interactive components, which is a domain where several software tools exist and is not the focus of our work.

In the work of Mahmudi *et al.* [23], ontologies are transformed into relational databases based on specific rules that can then be used for the development of a web application. However, a final operational web application is not presented. Another related work about ontology-based UI generation is the work of Hitz *et al.* [24]. In their paper, users must provide input to an Application Ontology that is transformed into a Target Ontology used for UI generation. Nevertheless, some proprietary annotations are required, adding some limitations to its universality.

METHODOLOGY

Inspired by the existing UI ontologies and the related work on ontology-based UI generation, in our work we first introduce OWAO, the new Ontology-based Web Application Ontology, which describes concepts, operations and axioms related to the generation of web applications via data extraction from a domain ontology. These concepts are integrated in our web application generator GenAppi, which can be used to create a Django web application specific to any user-provided domain ontology [25]. General additional web semantic services are also presented.

Ontology-based Web Application Ontology (OWAO)

OWAO can be seen as a meta-ontology that builds upon a set of (meta-)concepts to describe all entities present in any domain ontology, together with those needed to automatically manage the user interface of ontology instances. Specifically, as shown in Fig. 1, a Domain Ontology (*DomainOntology*) is composed of the classes *DomainClass*, *DomainDataProperty* and *DomainObjectProperty* concepts. These domain concepts are then mapped to the OWAO concepts that describe any Django web application, thus opening the door to the ultimate management of all the data belonging to the domain ontology.

Django web applications follow a Model-View-Template (MVT) architecture. The Model layer defines the domain data structure describing the tables and fields of the underlying database storing the application data. The View layer defines the processes used for retrieving, formatting or saving the data to the database defined in the Model layer. The Templates are used to render the information to the client. In order to generate a domain-specific web application, the concepts of the domain ontology have to be mapped to the corresponding concepts of the MVT architecture, which will enable the automatic generation of the MVT-compliant documents (instances of *WebApplicationDocument*) for the web application.

The domain concepts must be mapped to the web application related concepts (*DomainModelClass*, *DomainModelAttribute* and *DomainModelRelation*) of the Django web application model. These mappings are used for the generation of the Model of the web application (see the GenAppi subsection for more details). Then, for each *DomainModelClass*, the UI *Operation*s such as *Create*, *Read*, *Update*, *Delete* or *List* that are to be supported must be specified. In the generated web application, this means that appropriate *WebApplicationDocument*s will be available to the user, for example *View* documents implementing the functionalities of each *Operation*, and this for each *DomainModelClass*. At the front-end level, every *Operation* is represented by a specific UI fragment *UIFragment*, e.g., *Form* or *Table*.

Web Application Generator (GenAppi)

Exploiting state-of-the-art technologies for ontologies and web applications such as Django [25], Jinja2 [26] or Owlready2 [27], the GenAppi software tool follows a specific workflow for the generation of OWAO-based web applications. The main steps are displayed in Fig. 2 and described here.

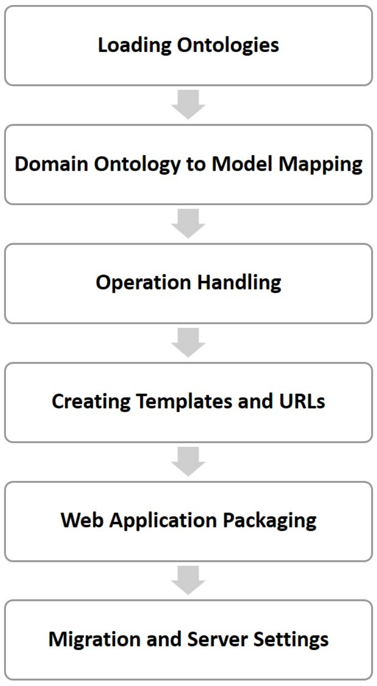

Loading Ontologies
Domain Ontology to Model Mapping
Operation Handling
Creating Templates and URLs
Web Application Packaging
Migration and Server Settings

Figure 2: GenAppi generator workflow.

Loading Ontologies Owlready2 [27] is the module that GenAppi uses for handling ontologies within Python; it enables existing ontologies to be represented as Python data structures, opening the way to their dynamic management via Python at run time. Since the Django framework is based on Python, the Owlready2 module is simply imported in GenAppi. When GenAppi is started, the domain and OWAO ontologies are loaded into GenAppi via Owlready2 in order to extract the necessary information.

Domain Ontology to Model Mapping Following the user-provided OWAO mappings, the domain ontology classes (*DomainClass*) are transformed to Python classes (*DomainModelClass*) used to create the Model of the Django web application. Data properties (*DomainDataProperty*) are transformed to attributes (*DomainModelAttribute*) and their datatypes are transformed to the equivalent datatype in the model, e.g., *xsd:string* to *TextField*. Object properties (*DomainObjectProperty*) are mapped to relations (*DomainModelRelation*) of the Model. Their cardinality (*RelationCardinality*) describes the Django relationship (e.g., ForeignKey or ManyToMany). For example, the OWL triple *IrradiationExperiment createdBy exactly 1 User* is transformed to the field *createdBy = ForeignKey("User")* in the Python class corresponding to the *IrradiationExperiment* domain class.

Operation Handling As displayed in Fig. 1, certain UI operations such as *Create* or *Update* can be associated to Model classes (*DomainModelClass*). For each of these operations, GenAppi provides a corresponding Jinja2 [26] template that includes the code implementation of the operation, independently of the Model. GenAppi uses these templates to generate *View* Python documents, used for the execution of the operations for each class.

Creating Templates and URLs Jinja2 templates are also used to generate Django templates. These Django templates are needed to provide the UI presentation logic of the generated web application, rendering and presenting data to the users. A unique URL is associated of each template and acts as a link to the corresponding *View* document.

Web Application Packaging After the generation of all the previously mentioned files, GenAppi assembles them in specific directories. Moreover, the OWAO and domain ontologies are copied in the web application, allowing for easier accessibility and portability, while making it possible to create future instances of the domain ontology through the web application.

Migration and Server Settings The last step of GenAppi is to handle the model migration to a dedicated database. The default database managed by Django is SQLite3, but this can be changed easily in the settings. After the migration, a web server is also started and the web application is finally ready for use. From that point, instances of the user-specific domain classes can be generated and managed via the UI client automatically generated by GenAppi, in compliance with OWAO-specified requirements.

Additional Services

In addition to the generation of a proper UI for the handling of ontology-linked data, useful additional features have been introduced within our framework.

Authentication and Authorisation The Django framework provides an integrated authentication and authorisation system. In order to use these functionalities, specific Jinja2 templates were created to generate the user interfaces that allow users to register and sign into the web application.

Ontology Visualisation WebVOWL is a software tool that allows for the easy visualisation and editing of ontologies [28]. WebVOWL is integrated in our framework and customised to fit the configurations of a Django application. This allows for an easy graph-like visualisation of the domain ontology used in the generated application.

UI Preferences In the generated web application, users can adapt the display view of the user interfaces. For example, they can change the font size or background colour. This is implemented for better user experience.

THE IEDM ONTOLOGY USE CASE

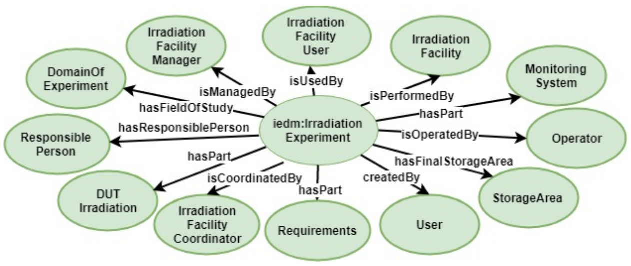


Figure 3: IEDM core classes.

For the purpose of testing our automatic web application generation methodology, the Irradiation Experiment Data Management ontology (IEDM) was used as a domain ontology. As shown in Fig. 3, the IEDM ontology formalises concepts of irradiation experiments focusing on the management of the data involved in them. IEDM was developed by investigating and analysing the common elements and practices used in typical irradiation experiments and by the help of domain experts. IEDM contains also instances describing the specific use case of an actual irradiation experiment performed at the CERN proton irradiation facility (IRRAD) [8].

When IEDM is used as an input to GenAppi, its classes and object and data properties are transformed into the Django model of an IEDM-specific web application. This model is mapped by default to a SQLite3 database. Views, templates and URLs of the MVT architecture are generated and implement the operations for each class of the ontology. In Fig. 4 and Fig. 5 at the bottom, some screenshots of the generated user interfaces are presented. Note that instances of IEDM classes can be also saved directly in the ontology (or in a triple store, in order to make the storage and querying of large amount of data more efficient).

As described in the previous paragraphs, the key UI operations are defined in OWAO, and the relevant functions for the model classes are generated by GenAppi. In Fig. 4, one specific example of a *Create* user instance operation form is demonstrated, where OWL superclasses in the form of triples corresponding to object or data properties are transformed into the input fields of a form adequate for creating the corresponding class instances. If this data property were instead an object property, the GenAppi-generated application would add a foreign key to the instance. In addition, if there are no instances of the class referred to, the user is prompted by the application to first add an instance of the class to which the foreign key is associated.

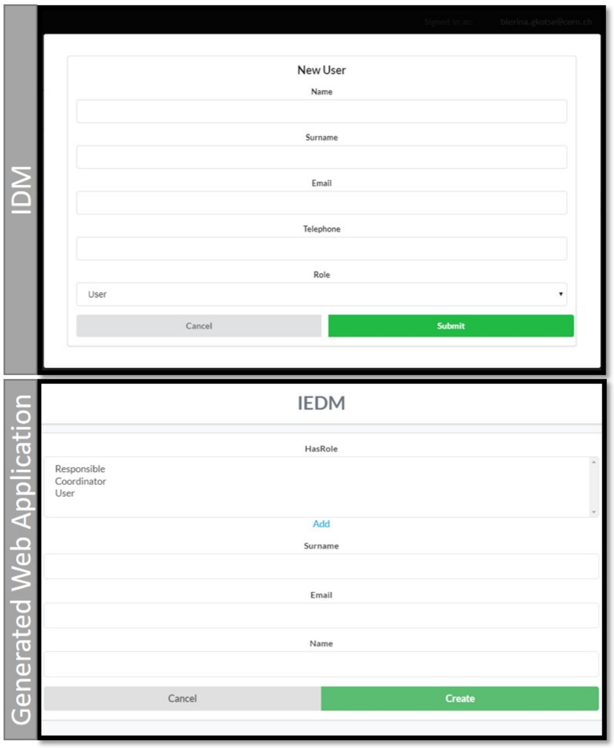
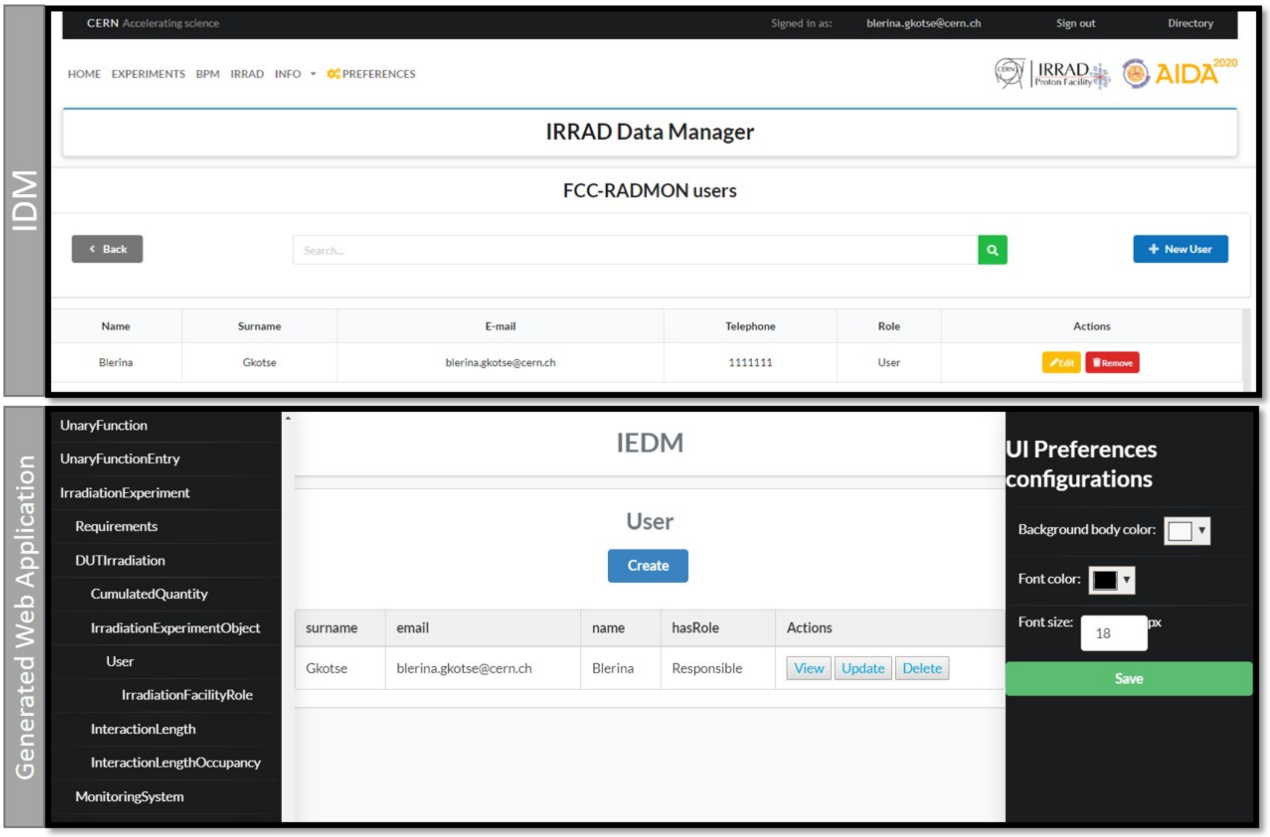


Figure 4: "Create user instance" form of IDM, at the top; generated web application version, at the bottom.

COMPARISON WITH THE IRRAD DATA MANAGER (IDM)

Currently, in the IRRAD facility, a custom-made Django web application is used for the data management of irradiation experiments, the IRRAD Data Manager (IDM). IDM follows the concepts of the IEDM ontology, but it has been manually developed and fulfils the specific requirements for the IRRAD experiments. IDM is used by both the IRRAD coordinator, operators and users for the registration of data. It provides specific functionalities dedicated to the planning of the irradiation experiments and their follow up. The results of the experiments are also available via this tool, which communicates with external systems. Finally, it keeps a history of the performed experiments and of the components that were irradiated [29].

IDM was developed for the same purpose as the GenAppi-generated web application presented above, namely the management of irradiation experiments data. However, IDM may not be suitable for other facilities, because it strictly fol-



IRRAD Data Manager

FCC-RADMON users

Name	Surname	E-mail	Telephone	Role	Actions
Blerina	Gkotse	blerina.gkotse@cern.ch	1111111	User	Edit Remove

User

surname	email	name	hasRole	Actions
Gkotse	blerina.gkotse@cern.ch	Blerina	Responsible	View Update Delete

UI Preferences configurations

Background body color: ⬜

Font color: ⬛

Font size: 18 px

Figure 5: List user interface of IDM at the top, generated web application at the bottom.

lows the operational requirements as specified by the IRRAD management.

But, more importantly, the technologies IDM is based on are strongly linked to the CERN software infrastructure. First, the Django model of IDM is based on the Oracle relational database, which may not be available in other facilities. In GenAppi, the user can specify the database that the web application should use. Moreover, it is not built based on semantic technologies as GenAppi, which allow for easier knowledge sharing and interoperability. These web semantic technologies facilitate complex querying, reasoning and inference. In addition to that, saving instances directly in the ontology or in a triple store is also possible.

As shown in Fig. 4 and Fig. 5, there is a resemblance among the UIs of the IDM and the generated web application. Regarding the additional services provided by both systems, note that the IDM authentication service is the CERN Single Sign On (SSO) system, while the generated application relies upon the default Django authenticating system. Also, IDM implements additional and complex functionalities that need to be defined by the developer. For instance, it integrates formulas to compute physics-related values that cannot be easily defined in an ontology. Nevertheless, the generated web application can be considered as a backbone for further implementation and functionality customisation according to the needs of the domain as done with other commercial software tools (e.g., SIMATIC WinCC [30])

CONCLUSION AND FUTURE WORK

In this paper, we introduce a methodology for the automatic generation of Django web applications from domain ontologies. For this purpose, the new Ontology-based Web Application Ontology (OWAO) is defined for formalising the concepts of the methodology; its instances describe the expected UI properties of the web application to be generated from a domain ontology. An OWAO-based web application generator, GenAppi, is described, and an example based on the Irradiation Experiment Data Management (IEDM) domain ontology is provided to illustrate the web application interfaces that are automatically generated. Finally, we compare this generated web application with the IRRAD Data Manager (IDM), the custom-made web application used in the IRRAD proton irradiation facility at CERN.

Our work illustrates that our ontology-based generated web application implements core functionalities similar to IDM's, but that it can be more easily adapted to other irradiation facility requirements, while relying on web semantic technologies that enhance knowledge sharing, data interoperability and inference.

The generated web application is simpler than IDM and does not yet implement its most complex functionalities. However, it is more flexible and adaptable to other use cases. Moreover, even though GenAppi-generated UIs allow for some level of customisation, they still need to provide more degrees of freedom for the users. Therefore, more focus will be given in the future to UI customisation, so that the generated applications are more adaptable and offer different “look and feel”, thus better fitting the user preferences.

REFERENCES

[1] N. Noy, “Semantic Integration: A Survey Of Ontology-Based Approaches”, *SIGMOD Record*, vol. 33, pp. 65-70, Dec. 2004. https://doi.org/10.1145/1041410.1041421

[2] B. Smith, “Ontology”, *Blackwell Guide to the Philosophy of Computing and Information*, Blackwell, Oxford, 2004, pp. 155–166, https://doi.org/10.1002/9780470757017.ch11

[3] H. Paulheim, “Knowledge graph refinement: A survey of approaches and evaluation methods”, *Semantic Web*, vol.8, no. 3, pp. 489–508, 2017.

[4] B. Smith,*et al.*, “The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration”, *Nat Biotechnol.*, vol.25, pp. 1251, 2007, Nov., Nature Publishing Group. doi:10.1038/nbt1346

[5] The Gene Ontology Consortium and M. Acencio, “The Gene Ontology Resource: 20 years and still GOing strong”, *Nucleic Acids Research*, vol. 47, no. D1, Nov. 2018, pp. D380–D388. doi:10.1093/nar/gky1055

[6] H. J. Pandit, F. Kaniz, D., O’Sullivan, and D. Lewis, “GDPRtEXT - GDPR as a Linked Data Resource”, in *The Semantic Web*, Jun. 2018, pp. 481–4955, doi:0.1007/978-3-319-93417-4_31

[7] R. Hernández de Diego,*et al.*, “STATegra EMS: an Experiment Management System for complex next-generation omics experiments”, *BMC Systems Biology*, vol.8 (Suppl 2), p. S9, 2014, http://www.biomedcentral.com/1752-0509/8/S2/S9

[8] B. Gkotse, P. Jouvelot, and F. Ravotti, “IEDM: An Ontology for Irradiation Experiments Data Management”, presented at the 16th Extended Semantic Web Conf. (ESWC2019), Portoroz, Slovenia, Jun. 2019, to be published.

[9] F. Ravotti, B. Gkotse, M. Moll, and M. Glaser, “IRRAD: The New 24GeV/c Proton Irradiation Facility at CERN”, in *Proc. AccApp’15*, Washington, DC, USA, Nov. 2015, pp. 182–187. http://accapp15.org/wp-content/data/index.html

[10] Oxford English Dictionary https://www.oed.com/view/Entry/131551?redirectedFrom=ontology

[11] T. R. Gruber, “A translation approach to portable ontologies”, *Knowledge Acquisition*, vol. 5, no. 2, 1993, pp. 199-220. doi:10.1006/knac.1993.1008

[12] N. F. Noy and D. L. McGuinness, “Ontology Development 101: A Guide to Creating Your First Ontology”, Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880. http://www-ksl.stanford.edu/people/dlm/papers/ontology-tutorial-noy-mcguinness-abstract.html

[13] Google Knowledge Graph, developers.google.com/knowledge-graph/

[14] Resource Description Framework (RDF), https://www.w3.org/RDF/

[15] Resource Description Framework Schema (RDFS), https://www.w3.org/2001/sw/wiki/RDFS

[16] Web Ontology Language (OWL), https://www.w3.org/OWL/

[17] Semantic User Interface ontology, https://old.datahub.io/dataset/ui

[18] LOV User Interface ontology, https://lov.linkeddata.es/dataset/lov/vocabs/ui

[19] A. Haller,J. Umbrich, and M. Hausenblas,“RaUL: RDFa User Interface Language – A Data Processing Model for Web Applications”, in *Proc. WISE 2010*, Hong Kong, China, Dec. 12-14, 2010, pp. 12–14. doi:10.1007/978-3-642-17616-6_36

[20] Linked Open Vocabularies (LOV), https://lov.linkeddata.es/dataset/lov

[21] S. K. Shahzad, “Ontology-based User Interface Development:User Experience Elements Pattern”, *Journal of Universal Computer Science*, vol.17 , no. 7, pp. 1078–1088, 2011.

[22] R. S. Engelschall, "Hierarchical User Interface Component Architecture", Ph.D. thesis, Inf. Univ., Augsburg University, Augsburg, Germany, 2018.

[23] K. Mahmudi,*et al.*, “Ontology to relational database transformation for web application development and maintenance”, *J. Phys.: Conf. Ser.*, vol. 971, p. 012031, 2018.

[24] M. Hitz, T. Kessel and D. Pfisterer, "Towards Sharable Application Ontologies for the Automatic Generation of UIs for Dialog based Linked Data Applications" in *Proc. 5th Int. Conf. on Model-Driven Engineering and Software Development (MODELSWARD 2017)*, Porto, Portugal , Feb. 2017. pp. 567–569. doi:10.5220/0006137600650077

[25] Django framework, https://www.djangoproject.com

[26] Jinja2, https://jinja.palletsprojects.com

[27] JB Lamy, “ Owlready: Ontology-oriented programming in Python with automatic classification and high level constructs for biomedical ontologies”, *Artificial Intelligence In Medicine*, vol. 80C, pp. 11-28, 2017.

[28] V. Wiens, S. Lohmann, and S. Auer, “WebVOWL Editor: Device-Independent Visual Ontology Modeling”, in *Proc. 17th Int. Semnatic Web Conference (ISWC 2018)*, CEUR Workshop Proceedings, 2180, CEUR-WS.org, 2018.

[29] B. Gkotse, P. Jouvelot, G. Pezullo, and F. Ravotti, “The IRRAD Data Manager (IDM)”, presented at ICALEPCS’2019, New York, USA, Oct. 2019, paper MOPHA048, this conference.

[30] SIMATIC WinCC, https://new.siemens.com/global/en/products/automation/industry-software/automation-software/scada.html