Towards bridging ontological and closed-world modelling with synchronised EMF views of RDF models

Owen James Reynolds
ASE research group,
University of York
York, UK
owen.reynolds@york.ac.uk
0000-0002-5639-0533

Antonio García-Domínguez
ASE research group,
University of York
York, UK
a.garcia-dominguez@york.ac.uk
0000-0002-4744-9150

Dimitris Kolovos
ASE research group,
University of York
York, UK
dimitris.kolovos@york.ac.uk
0000-0002-1724-6563

Gianmaria Bullegas
Leonardo UK,
Edinburgh, Scotland
Gianmaria.Bullegas@leonardo.com

Campbell Mccausland
Leonardo UK,
Edinburgh, Scotland
campbell.mccausland@leonardo.com

***Abstract*—Closed-world modelling is a common approach for modelling high-integrity systems where robust verification and validation is required. However, closed-world modelling can be rigid, and at times this rigidity can hinder system development. Ontological modelling has several advantages over closed-world modelling, most notably its flexibility, which enables unanticipated information to be freely attached to a model. Some organisations have made significant investments in closed-world modelling techniques for designing high-integrity systems, including developing domain-specific model editors on top of mature frameworks, such as Xtext and Sirius. Similarly, some model management programs that focus on controlled models will still be needed. However, they may benefit from having the option to access some additional information outside that controlled subset.**

Prior works have explored ways to bridge the worlds of Resource Description Framework (RDF) and Eclipse Modeling Framework (EMF) modelling. These approaches are often based on model transformations and fail to take advantage of Semantic Web technologies such as reasoners. They can also result in a loss of data when reducing the ontological model to fit a closed-world modelling language.

In this paper, we propose an approach to bridge EMF closed-world and RDF open-world ontological modelling, by exposing editable closed-world views of an ontological model for EMF-based modelling tools. The exposed closed-world view supports reading and writing to the model: this is achieved through synchronisation of the EMF and RDF representations of the model. We demonstrate the approach with an open-source implementation built using EMF and Apache Jena. A series of case studies illustrate how we handle some of the differences between ontological models and closed-world models.

***Index Terms*—Model-Driven Engineering, closed-world modelling, open-world modelling, ontological modelling.**

I. Introduction

Developing high-integrity systems demands robust verification and validation techniques to meet various regulatory and safety requirements for the system's domain. The tools and practices provided by Model-Driven Engineering (MDE) can help develop a high-integrity system when combined with a methodology [1]. In MDE, models are treated as first-class citizens in the development process that creates a system or a product. The models in the development process can be managed with automated processes. Some automated model management tasks can be embedded into a DevOps pipeline, enabling a more agile approach to the MDE development process [2], [3]. Thus, MDE-based projects can benefit from the use of automation and version control, like other software development practices.

While MDE has provided solutions to many problems, a survey by Alfraihi and Lano [3] identified some issues. In their findings, Alfraihi and Lano, comment on MDE tools and notations having a strong 'lock-in' to specific ways of working. This rigidity becomes a challenge to the desire to work incrementally or in an agile way. It also restricts the MDE development processes to using MDE tools, which can have issues with poor efficiency on large-scale models.

An open-world approach to modelling is considered to be more agile than closed-world MDE modelling, and as such could provide solutions for some of MDE's challenges, such as scalability. There is a long history of different approaches to trying to combine these two techniques for different purposes [4], [5]. OpenCAESAR [6] is a recent attempt to combine Model-Based System Engineering (MBSE) with the open-world modelling techniques seen in the Semantic Web (SW) [7] communities.

Developing a system with MDE requires a significant investment in time, effort and resources. It is also hard to establish teams of developers that are skilled in MDE, as noted by Alfraihi and Lano's survey [3] on MDE practices.

Thus, an organisation that has invested in a set of MDE tools and training for developers may not want to migrate to a newer MDE tool like OpenCAESAR, which may offer some advantages from open-world models.

In this paper, we explore an approach to combining MDE closed-world models based on the Eclipse Modeling Framework (EMF) [8] with SW open-world models based on the W3C Resource Descriptor Framework (RDF) [9], [10]. Specifically, the development of an EMF resource, which is an implementation of a Java interface that supports models expressed using different persistence formats, for example, file formats like XML or JSON. An EMF resource hides the implementation details of the model persistence from the modelling tools; thus, different EMF resources can be used transparently from a tooling perspective. This new EMF resource can be used with the Eclipse EMF IDE [8], Eclipse Epsilon [11], or other EMF-compatible tools. This resource is envisaged as a means to enable existing EMF models and tools to have access to SW technologies. This interoperability could then enable new ways of working that build upon current MDE practices. The EMF-RDF resource presented in the paper is open-source and under development at its Epsilon Labs GitHub repository[1].

The paper is structured as follows: some background is provided in Section II. In Section III, there is a discussion of the problems and objectives this research is exploring. The design and development of the software artefacts are presented in Section IV, followed by a review of related works in Section V and the paper offers conclusions in Section VI.

II. Background

Before proceeding, it is necessary to define some of the key concepts being discussed and provide some background on the technologies presented in this paper.

A. Definition of open- and closed-world for this paper

In this paper, we will use the terms open-world and closed-world to describe modelling or models. The use of these terms comes from Open-World Assumption (OWA) and Closed-World Assumptions (CWA). Statements about knowledge made under OWA conditions never assume things are false if there is no proof of it being true [12]; a state of 'unknown' can exist. Under CWA conditions, statements are either true or false; the 'unknown' state is not permitted, and anything not known is assumed to be false.

These approaches to assumptions can be applied to modelling; a modeller can create or handle a model under OWA or CWA conditions [6]. For example, under OWA conditions, it would be possible to create model elements without defining their types; an unknown type is permitted. This enables a very flexible approach to developing a model. However, OWA conditions are not appropriate for all modelling tasks. For example, a model verification process requires more rigid CWA conditions; CWA conditions require all model elements to have a type, unlike the model under OWA conditions.

[1]https://github.com/epsilonlabs/emf-rdf

B. Eclipse Modelling Framework - closed-world modelling

The Eclipse Modeling Framework (EMF) [8] is an open-source modelling ecosystem that many other modelling tools are built on or support. In EMF, models conform to closed-world metamodels (modelling languages) that define the concepts a model can comprise, including their permitted relations and properties. Using the EMF IDE, a modelling language can be created for domain experts, and a new EMF IDE application can be automatically generated (using code generation) for the domain experts to install and use to create models.

Eclipse Epsilon [11] is a model management tool that natively supports EMF models with its family of scripting languages for automating common model-based software engineering tasks. However, Epsilon also provides a model connectivity layer (EMC) that enables many different model formats to be accessed through different drivers. The drivers shield developers from the details of the technology used to implement a model; their programs query/modify models through a uniform presentation independent of the model format.

C. Semantic Web - open-world modelling

W3C has a vision of a technology stack to support the creation of a web of data, similar to the web of documents on the internet [7]. This vision includes the idea that people would create and store data on the web using vocabularies. Vocabularies would help with writing rules for handling data, by providing some guidance or structures to help interpret stored data. Ontologies can be created using technologies such as OWL [13], which are examples of standard vocabularies used in the Semantic Web.

Apache Jena, from the Apache Software Foundation [14], is an open-source Java framework for building web and linked data applications. The Jena framework provides some APIs for working with RDF data, ontologies (OWL) and inference for reasoning about data. Jena can work with triple stores such as TDB[2] and Fuseki[3] for persisting semantic open-world models.

III. Research problems and objectives

EMF models require a modelling language to be defined before creating a model. Changing a modelling language after creating a model can be difficult; models based on the language being changed can become invalid and require a transformation process to correct them. It would be desirable to have the ability to add new concepts to a model without this expense. Modellers could benefit from a low-cost method to handle new information that does not require extending a modelling language.

Transformations in MDE are typically one-to-one mappings (Fig. 1), requiring a new transformation program to be written to exchange models between formats or tools. Integrating the different MDE tools at scale using model transformations becomes difficult and costly to maintain because of the number

[2]https://jena.apache.org/documentation/tdb/

[3]https://jena.apache.org/documentation/fuseki2/

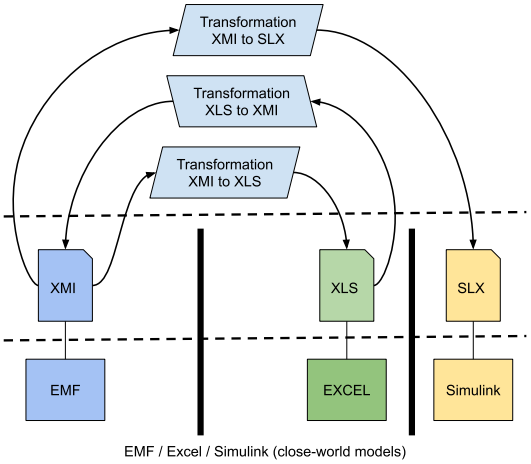

Fig. 1: MDE Model transformations for integrating tools

of transformation programs that can be involved. It would be desirable to explore an open-world RDF approach to performing these transformations. For example, whether a reasoner and schema could be used to create a transformation, or whether an RDF presentation of a model could be used to synthesise many different model notations (formats) through a generic transformation process that is defined per MDE tool (one-to-many arrangement).

The knowledge required to answer some questions about a system may be difficult to collect: for example, the knowledge might be spread over more than one model. These questions require queries to be written to extract the data from multiple models. A further complication is added when the models are managed with different MDE tools, each with their own query language and/or modelling languages. It would be more convenient if the models could be combined in a single space, as a combined view of the models. A combined view of the models could be more easily queried for knowledge; it may also be possible to infer new knowledge from connections that can be established between model elements.

Within the context of an MDE modelling tool, there can be limitations imposed on the kind of knowledge that can be stored with a model. This knowledge is not part of the model itself, but could be related to the development of the model or a relationship with another artefact within the project. For example, some configuration values in a model could be derived from values given to the modeller in a specification document, and a record of this relationship could be useful for traceability for regulatory compliance. There may be advantages to being able to store additional knowledge about a model that enhances how it is managed.

Organisations may have a significant investment in MDE tools through customisations or automations. Therefore, an approach to open-world modelling that can integrate with an organisation's existing MDE investments could help to bridge the gap between open- and closed-worlds and aid in the adoption of more ontological modelling approaches. The vision of how this transition could be achieved is explained below.

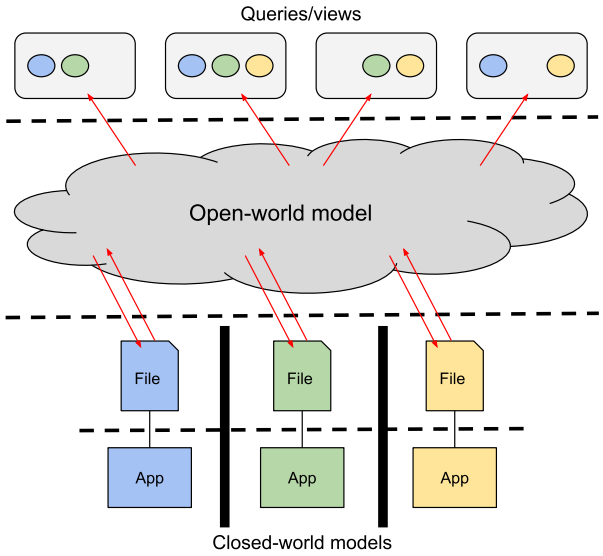

Fig. 2: Exchanging closed-world models via an open-world model

Our long-term objective is to enable the creation of a virtual open-world model, composed of several closed-world models that can be accessed with either ontological or MDE tools. It seems feasible that several closed-world models could exist in the same open-world model, assuming the open-world model has a federated namespace that all the closed-world models conform to (Fig. 2). Bringing closed-world models into an open-world modelling space, and maintaining interoperability with the original closed-world modelling tools while avoiding unintentional data losses from transformations could help address the problems discussed above, through a fusion of open- and closed-world modelling technologies.

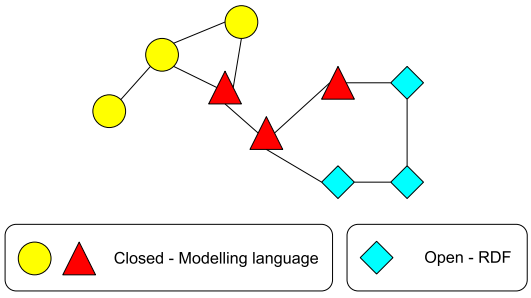

Fig. 3: Model composed of elements conforming to a modelling language and generic RDF triples

A closed-world subset of an open-world model could be modified by changing the RDF statements that directly relate to it. Additionally, it would also be possible to add other RDF statements to a model, and these would not have to conform to the modelling language of the closed-world subset. However, under closed-world conditions, the additional RDF statements that do not relate to a closed-world model's modelling language should be hidden to maintain compatibility with the original MDE tooling.

Figure 3 shows an RDF model containing circles, triangles and diamonds. The circles and triangles represent information

that conforms to a closed-modelling language. An MDE tool that is confined to the closed-modelling language should be able to access the information in these RDF nodes as native MDE model resources. However, the MDE tool could also have access to a second *generic RDF* modelling language, which enables access to the information in the diamond RDF-only nodes. Accessing information in these nodes would not be as convenient as the closed-modelling language; the MDE tool would need to more directly query the RDF without being able to make closed-world assumptions.

Typically, in a transformation approach, the circles and triangles are extracted from the RDF model into an MDE model. The diamond nodes that are not part of the closed-modelling language would not be copied into the MDE model. As such, transforming the MDE model back into RDF to save changes would lose the diamonds if the original RDF were overwritten. MDE tools should not lose the additional (diamond) RDF statements when saving the MDE model back to this RDF model. The open-world model containing the closed-world model should be preserved in its entirety, even when accessed by a closed-world MDE tool.

Having the ability to make additional statements about a closed-world model when it is part of the open-world model presents some interesting opportunities. These additional statements could capture a wide array of knowledge about a model like: provenance data for changes made to the closed-world model, keeping versioning information, or linking parts of a model with sources of documentation outside an MDE tool. This is where the flexibility of the open-world model leads to potentially innovative ways of working that traditionally would have been difficult or costly using only MDE tools.

A closed-world model represented in RDF can be combined with other RDF models, like a schema, that contains additional knowledge or explanations about a model. Information might be added to a model before communicating/sending it to someone else. For example, this additional information might be instructions on the conditions under which to use a model or the model's limitations, possibly mappings between or explanations of domain-specific terminology. Thus, RDF with a collection of suitable ontologies could become the lingua franca for communicating models between teams that use different MDE- or ontology-based tools.

The open-world model of closed-world models would be virtual; it does not need to be persisted as a single large database or file. Graph unions make it possible to divide a large open-world model into sub-graphs, which can be combined to create ad-hoc models for a specific purpose on demand. A natural fragmentation of the virtual open-world model occurs with each instance of a closed-world model. Each model created using an MDE tool could be considered as an individual subgraph. The unified namespace of the virtual open-world model would separate the models (or join them on common parts that should overlap), enabling any combination of closed-world models to be loaded into the same space as a graph union.

With all the closed-world models existing in a single open-world model, it would be possible to query or view any combination of the models. Graph reasoners might then enable new knowledge to be added to the open-world model; for example, a reasoner could identify a common element in two different domain-specific models, where each model uses different domain-specific names for the same part. Or a query could be written to explore the dependencies between various models to estimate the effect of a change in a design.

However, exploring the potential for using RDF as a lingua franca for EMF models requires a tool that enables MDE models to be persisted in RDF, without causing a loss of information. As such, our initial objective is to implement an EMF resource (compatible with several existing MDE tools) that directly uses RDF to persist the model, instead of XMI or a model repository. This *EMF-RDF* resource should enable different models and modelling languages to be used like a normal EMF resource, but it should also enable direct access to the RDF model to access information outside of the MDE tools' typical closed-world model.

IV. Design and development

In designing a method to persist MDE models in RDF, it was clear from reviewing past attempts that MDE model-to-model transformations should be avoided. Instead, a more direct approach that simultaneously maintains both closed-world and open-world representations could be used, removing the need for a model-to-model transformation. Ideally, this method for persisting EMF models with RDF would also maintain compatibility with existing MDE tools; for example, an RDF representation of an EMF model should not require that a tool or program be changed to handle a new notation introduced by the method to store the model in RDF. The MDE model in its RDF form can be freely extended/changed by editing RDF statements. Additional RDF statements outside of the model's modelling language can be accessed by an MDE tool through a *generic RDF* modelling language. This generic RDF modelling language exposes the RDF statements using EMF EObjects that an MDE tool can interpret.

The architecture of the Eclipse Modeling Framework (EMF) is amenable to being extended for working with different types of structured models that can be persisted in many different ways. For this reason, EMF has been selected as the MDE technology for this research, which is the foundation of many existing modelling tools. The open-source Apache Jena library has been selected to handle open-world RDF models. Jena can read and write RDF from files using several notations, or a triple store like TDB or Fuseki.

Our approach creates an EMF resource which synchronises changes between EMF and RDF representations of an in-memory EMF model: we call this an *EMF-RDF resource*. Only the RDF representation is persisted on disk. When an RDF representation is loaded, a *deserialiser* recreates the in-memory model containing EMF and RDF representations. RDF statements relating to the EMF model's modelling language become EMF EObject instances in the EMF representation. Thus, the EMF-RDF resource is expected to be

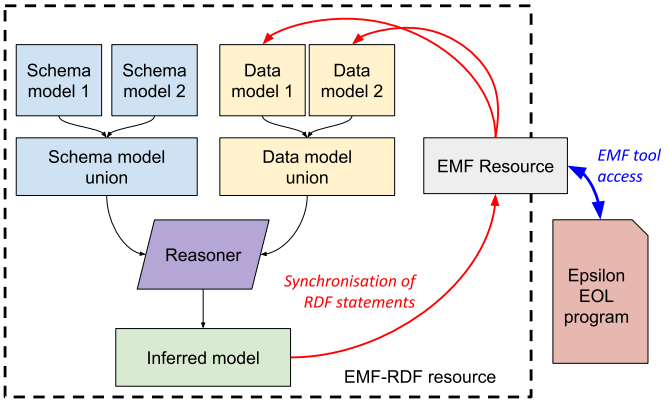

Fig. 4: EMF-RDF resource architecture

```
@base <http://my.org/>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix orgc: <http://york.ac.uk/emf-rdf/examples/orgchart#>.

<#organisation>
  a orgc:Organisation;
  orgc:name "My Organisation";
  orgc:teams ( <teams#dev> <teams#marketing> ).

<teams#dev>
  a orgc:Team;
  orgc:name "Development Team";
  orgc:employees <employees#1234>;

<employees#1234>
  a orgc:Employee;
  orgc:name "John Doe".
```

Listing 1: Organisational chart's shortened turtle file

```
@namespace(uri="http://york.ac.uk/emf-rdf/examples/orgchart")
package orgchart;

class Organisation {
    attr String name;
    val Team[*] teams;
}

class Team {
    attr String name;
    val Employee[*] employees;
}

class Employee {
    attr String name;
}
```

Listing 2: The organisation chart's EMF Ecore metamodel (modelling language)

fully compatible with existing EMF-based tools, as it should behave like a native EMF resource. Additionally, the resource includes an API to access the original RDF resource backing an EObject directly.

The facilities within Jena enable the EMF-RDF resource to load several RDF data models and schema models into a single RDF model, before deserialising them as an EMF model (Fig. 4). This approach allows an EMF model to be created from (and therefore split across) several RDF sources. The deserialisation process requires the RDF representation of an EMF model to be valid, but it does not require it to be a complete model. The deserialiser can partially load models when only some RDF sources are available, which enables an incomplete or broken EMF model in an EMF-RDF resource to be loaded and worked on. An EMF model may be incomplete because it is still being created, so it is missing elements. If the deserialiser cannot access some RDF resources, it will fail to create model elements. An EMF model would be considered broken if missing elements are marked as mandatory in the modelling language (e.g. with non-zero minimum cardinalities in the EMF meta-model).

The EMF-RDF resource can be configured to use Jena's reasoner to apply one or more OWL schemas to the RDF data models, before deserialisation is performed. A schema could be used to infer statements from an RDF model, to enrich some information before producing the EMF model representation. While this is not a critical feature in the initial development of the EMF-RDF resource, it is a facility that could be useful in the long-term objective of creating a virtual open-world model.

A simple case study example from the EMF-RDF resource Git repository is used to demonstrate the design and development of the EMF-RDF resource. The case study RDF data model is presented in Listing 1 in Turtle format, and the EMF Ecore metamodel (modelling language) in Listing 2. The EMF-RDF resource can be used with the Eclipse EMF IDE: it is possible to edit and view the models, or create visualisations with tools like Picto [15].

A. Loading EMF models from an EMF-RDF resource

The EMF-RDF resource deserialisation process creates an EMF model instance from a set of RDF statements. This process produces bidirectional mappings between RDF resources and EMF EObjects. The EMF-RDF resource is loaded against an EMF Ecore metamodel, like a normal EMF resource. Only the RDF statements relating to the modelling language will create EObjects (Fig. 5). The deserialiser matches namespaces/names in the modelling language to RDF namespaces and rdf:type statements; users do not need to provide mapping information. The other RDF statements are accessible through Jena APIs, by reusing the above mappings to obtain the RDF resource backing a given EObject.

In Fig. 5, the EMF-RDF resource is being viewed from the perspective of a modelling tool looking at an EObject (square box, thisObject). The black arrows connect the EObject to the RDF nodes (rounded boxes) from which the deserialiser

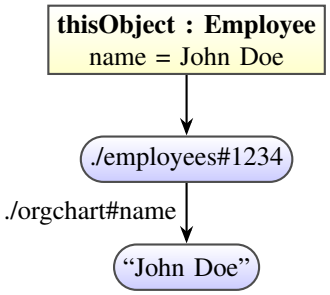

Fig. 5: EMF EObject view of an employee's RDF resource

```
1 @prefix academic: <http://york.ac
  ↪  .uk/emf-rdf/examples/academic#> .
2 <employees#1234>
3   a orgc:Employee;
4   orgc:name "John Doe";
5   academic:orcid "0000-0000-2222-1111".
```

Listing 3: Turtle statement added for John Doe's ORCID

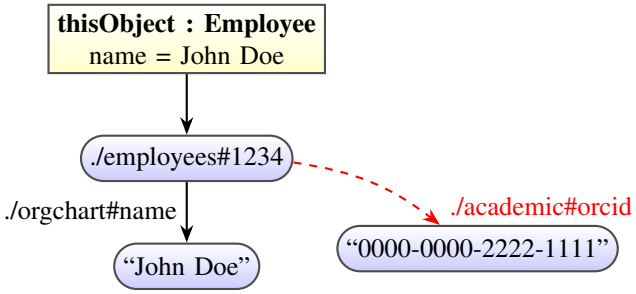

Fig. 6: The red arrow indicates RDF information outside of the modelling language that is accessible via the RDF escape

derived the EObject.

The EMF model of John Doe only shows the features described in the modelling language. However, the RDF model for John Doe can include additional information outside of the modelling language. John Doe's ORCID[4] identifier has been added on line 5 of Listing 3. The ORCID can be seen in Fig. 6: a red dashed arrow indicates that a related RDF node exists outside of the modelling language in Listing 2. This additional information was accessed by using the EMF-RDF API for mapping between EMF EObjects and their backing RDF resources.

B. Editing EMF models in EMF-RDF resources

Having deserialised the RDF to extract an EMF model, an EMF-compatible tool can be used to edit the EMF representation of the model. As changes are made to the EMF representation, EMF's notification system triggers a process to produce the same changes to the RDF representation. This low-level change-by-change synchronisation keeps the RDF and EMF representations aligned; only the RDF statements relating to the changed EMF model features are updated.

Loading a model from an EMF-RDF resource into Eclipse IDE, we could edit John Doe's name and save the model without losing the additional RDF statement containing his ORCID (line 5 of Listing 3). Editing the EMF representation of his name would produce an EMF notification that removes the old RDF statement (line 4 of Listing 3) and replaces it with a new one containing the new name.

Approaching the problem using this synchronisation method eliminates the need for complex model transformations to avoid data loss. A transformation may need to account for multiple changes and the order changes occurred, in addition to merging RDF statements outside of the modelling language. This synchronisation approach might be considered as a type of transformation; however, it is performed at an atomic level as change notifications arrive in sequence.

[4]https://orcid.org

```
1 @prefix academic: <http://york.ac
  ↪  .uk/emf-rdf/examples/academic#> .
2 <employees#1234>
3   a orgc:Employee ;
4   orgc:name "John Doe";
5   a academic:Academic;
6   academic:name "John Doe";
7   academic:orcid "0000-0000-2222-1111".
```

Listing 4: Academic statements added to John Doe

```
1 @namespace(uri="http://york.ac
  ↪  .uk/emf-rdf/examples/academic")
2 package academic;
3
4 class Academic {
5     attr String name;
6     attr String orcid;
7 }
```

Listing 5: EMF metamodel for new Academic class

Apache Jena has a similar notification infrastructure to EMF, which could be used to detect changes to an RDF model. Processing Jena's notification of changes to the RDF statements could be used to update the EMF representation of a model without deserialising all the RDF statements. In future work, we envisage using Jena's notifications to update the EMF representations in an EMF-RDF resource, enabling bi-directional synchronisation with the virtual open-world model.

C. Handling multi-type RDF resources

There may be more than one rdf:type statement for an RDF node representing an element. In EMF every element (EObject) is an instance of a single type (EClass). When a model contains an element with multiple types, the EMF-RDF resource produces an EObject instance for each type of the RDF node that is present in the EMF metamodel. It is worth noting that when the EClasses have subtype relationships (e.g. Employee and Developer), our approach chooses the most specific type for deserialisation (e.g. Developer).

Listing 4 shows a Turtle file where John Doe has been extended with more information, becoming an Academic (line 5) in addition to an Employee (line 3).

Listing 5 shows a new metamodel with a single Academic EClass, which is registered with the EMF-RDF resource for

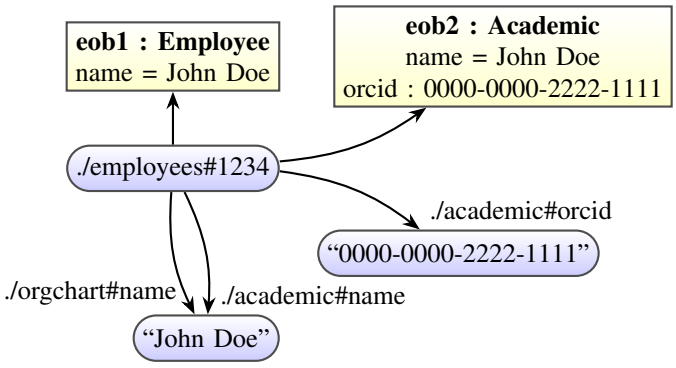

Fig. 7: RDF nodes with multiple types are represented with multiple EObjects in the EMF model

deserialisation. The URI on line 1 maps to the prefix in line 1 of the Turtle file in Listing 4, indicating to the deserialiser that these things are the same. RDF predicates map to EMF feature names (names of EAttributes/EReferences).

A view of the EMF-RDF resource from the perspective of the './employees#1234' node is presented in Fig 7. John Doe's single RDF node has produced two EObjects, one representing him as an Employee and another as an Academic. In this example, we repeat his name in two statements with different predicates (lines 4 and 6, in Listing 4) so his name is present in both EObject instances. While RDF is incredibly flexible, RDF statements can be very precise; without the 'academic:name' statements, his name would not appear in the Academic EObject.

In a real RDF model, we might use an ontology to relate the Academic and Employee names (e.g. using OWL's `equivalentProperty`). However, for this example, we used more than one statement to explore the situation where editing a model requires changing more than one statement relating to the same information. When editing John Doe's name via the EMF model, the mapping of multiple RDF classes to multiple instances of EObjects creates an interesting problem. The RDF statements are precise, so editing his name in this example could change different statements depending on the modeller's intent. For example, the modeller may want to change John Doe's Academic name and have his Employee name remain unchanged, or may want to change both statements to maintain consistency.

Editing the RDF directly requires the modeller to decide which statements to change. However, if the editor uses an EMF tool to edit the EObjects, then the EMF-RDF resource needs to be told which statements the editor intends to change. A literal interpretation of editing the Academic EObject could be taken, which would only update the Academic name RDF statement. Alternatively, it may be that the Employee EObject is the only place a name should be editable, in which case edits to the Employee instance should update all statements about John Doe's name. We intend to look at various alternatives to specify this behaviour, such as honouring OWL's `equivalentProperty` if specified, or providing configuration options for the EMF-RDF resource.

D. Separating and combining model information

As mentioned earlier and shown in Fig. 4, the EMF-RDF resource can load multiple RDF data models, combining them with a graph union and applying reasoning. This can be useful when a large body of information sometimes needs to be separated into more manageable chunks. Breaking information into chunks might be required because of hardware resource limitations, or there is a need to abstract and remove details for a global view of the information.

There is an increasing need for organisations to handle sensitive information more carefully for compliance and security reasons. One approach to managing information security is to use the least amount of information required for any given task or process. To do this, we might divide the information into

```
1 @base <http://my.org/> .
2 @prefix rdf:
  ↪  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
  ↪  .
3 @prefix rdfs:
  ↪  <http://www.w3.org/2000/01/rdf-schema#> .
4 @prefix orge: <http://york.ac⌋
  ↪  .uk/emf-rdf/examples/orgchart-extra#> .
5
6 <employees#1234>
7   a orge:ExtendedEmployee ;
8   orge:salary 40000 .
```

Listing 6: Separate RDF data models can be used to hold restricted information

```
1 @namespace(uri="http://york.ac⌋
  ↪  .uk/emf-rdf/examples/orgchart-extra")
2 package orgchart_extra;
3
4 import "http://york.ac⌋
  ↪  .uk/emf-rdf/examples/orgchart";
5
6 class ExtendedEmployee extends orgchart.Employee
  ↪  {
7         attr int salary;
8 }
```

Listing 7: An extended EMF metamodel is needed to show the restricted information in an EMF model

separate files based on the level of sensitivity and secure them appropriately. Then the smaller files containing the required information would be recombined for a task/process when needed under the appropriate security conditions.

RDF statements can be divided across multiple locations, and each location can be secured as needed. This makes it possible to separate individual statements containing sensitive information from less sensitive information at a very fine-grained level. Using a graph union operation under appropriate security conditions, sensitive information can be combined with non-sensitive information easily; this is possible with the EMF-RDF resource.

In this example, John Doe's salary information is stored in

```
1 dataModels:
2   - myorg.ttl
3   - myorg-extra.ttl
```

Listing 8: EMF-RDF resource configuration file listing the RDF data models to be unionised to create an EMF model

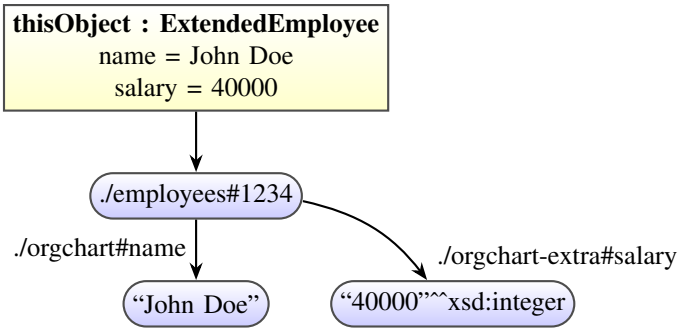

Fig. 8: Using the extended EMF modelling language and additional RDF data model, the restricted information can be accessed through the EMF model representation

a separate Turtle file, away from the rest of the organisational chart (Listing 6). Only certain individuals in the organisation can access the file containing the employee's RDF node and salary information. In addition to access to the Turtle file, an extended EMF modelling language is needed that extends the Employee class with a salary attribute (Listing 7). The EMF-RDF resource is configured with two RDF data model sources as shown in Listing 8 before being accessed with an EMF-compatible tool. In Figure 8, the view of the EObject instance for John Doe with the extra salary information is shown, whereas Figure 5 is all that would be visible without the extra EMF modelling language and Turtle file. Note how the object is only exposed as an instance of the most concrete type between its two types (ExtendedEmployee).

The approach demonstrated does not address all security concerns, which are beyond the scope of the research. However, this technique might complement other security measures or be used in other ways, such as removing layers of detail from a model. These use cases create a further need for configurable behaviour in the EMF-RDF resource to direct where certain RDF statements are written when the model's EMF representation is edited.

E. Saving EMF models in EMF-RDF resources

Currently, the EMF-RDF resource assumes that OWL schemas are read-only, and Jena treats inferred RDF statements as read-only. Thus, only RDF data model statements can be changed, and each data model is saved back to its original location. The EMF-RDF resource relies on Jena to handle I/O operations to sources and to identify the format of the RDF data model; the same format is used when saving RDF data models.

An EMF-RDF resource with more than one RDF data model has several in-memory RDF data models that could be saved when an EMF modelling tool requests that the EMF model be saved. Depending on the edits made to the EMF model, changes to RDF statements could occur in any of the in-memory RDF models. Saving an RDF model to a file overwrites the original file, which may not be desirable; some RDF data models might have been loaded as a reference with no intention of them being edited. The default behaviour of the EMF-RDF resource is to save all the loaded RDF data models. We are considering adding a configuration option to mark some RDF data models as read-only, excluding them from a save operation.

How the EMF-RDF resource updates RDF statements in response to an EMF model being edited affects which RDF data model sources need to be written to on an EMF model save event. For example, the default behaviour of the EMF-RDF resource is to write all new RDF statements to the first data model when several are configured. However, when a change to an EMF model affects statements in several in-memory RDF data models, our approach currently updates the RDF statements in all of the RDF models. This approach has been taken because a native EMF model's state is typically a single instance in memory. For simplicity, the initial implementation of the EMF-RDF resource maintains consistency between the EMF model and the RDF statements by not allowing EMF model edits to create conflicting RDF statements in memory.

In the future, choosing the RDF data model where to create/update a given statement will likely need to be configurable option, as there are several possibilities depending on the use case. For example, we may want all statements relating to a given part of a model into a designated RDF data model. Or have statements containing sensitive information written into a protected RDF data model, and less sensitive statements going to a public RDF data model.

There are still many features and configurable options to add to the EMF-RDF resource. Work is ongoing to achieve equivalence between the EMF-RDF resource and a native EMF resource, building an automated test suite that simulates the various changes that can take place in a closed-world model. The EMF-RDF resource needs to enable the creation of a model from nothing to be a minimally complete implementation. At the moment, the synchronisation supports changing single- and multi-value EAttributes: support for EReferences and the creation/deletion of model objects is yet to be added, but should have some parallels with the EAttribute implementation.

V. Related works

There are some related works that should be acknowledged, as they present similar ideas and address problems related to the research presented in this paper. In this section, the work relating to the need for addressing flexibility and managing MDE at scale is presented first, then works that enabled some interoperability between open- and closed-world modelling are examined.

A. Addressing the flexibility of MDE modelling at scale

Flexible MDE is a term in the literature that relates to the rigidity of MDE that comes from models conforming to modelling languages. This is the model conformance relationship challenge: since a model must conform to the metamodel [16] of a modelling language, a modelling language is needed before a model can be created. A need to relax this conformance relationship between a model and modelling language was identified, and *flexible modelling frameworks* that gave modellers the freedom to create models before a modelling language existed were proposed. A modeller can develop a model approximating what they require, then work backwards from the model to produce the modelling language, and establish the stricter conformance relationships that make a model more rigid. Hili et al. demonstrated their approach with Fleximeta, a web-based flexible modelling framework [17].

Guerra et al. proposed a modelling language for flexible modelling [18], extending their prior work on KITE (an Eclipse plugin based on EMF and Xtext). However, the effect on the process for developing a model is similar to Hili et al., in that a model can be created before a modelling language. Whereas the need for flexibility in these works

relates more toward the initial stages of developing a model, the conformance problems also present a problem at the later stages of development. The overlooked or unanticipated need for information to be present in a model can be costly in terms of reworking a model to include it later.

Megamodels are model whose elements are models themselves: the idea was discussed by Bézivin et al. in 2004 [19]. Megamodelling is considered to be modelling at a large global view over several smaller modelling activities, e.g. a large modelling project with several sub-projects that produce models. A megamodel, like other MDE models, would conform to a modelling language. The megamodel would capture the various model management processes on and between the smaller models, as a high-level management overview that could also enable the coordination of model management tasks as an executable model. A novel use of megamodels to understand MDE projects by Di Rocco et al. [20] provides some examples of megamodels. Using publicly available MDE projects, they reverse engineer a megamodel representation for each project as an architectural overview.

Megamodels are closely related to what we consider a virtual open-world model, which encompasses several closed-world models in a space that enables the reuse and integration of them. However, the flexibility of RDF enables the virtual open-world model to contain more than just the closed-world models. It would be possible to include considerably more information relating to the models, such as the provenance of changes and versioning or alternatives.

B. Interoperability between open- and closed-world models

Open- and closed-world modelling approaches have been available for a long time, and have grown up next to each other but with different outlooks on modelling. However, there have been times when researchers have combined them; early works examined model transformation, and more recent works directly integrate them. Transformations have limitations, which have resulted in a more integrated approach like OpenCAESAR. However, integrating the approaches can raise issues when there is a need to reuse existing models and modelling tools.

Model transformations are one approach in the literature for converting models between EMF and RDF representations. Hillairet et al. demonstrated a bidirectional mapping using a prototype developed using ATL to create a two-way bridge between EMF and RDF [4]. However, the transformation approach struggles with handling models containing information outside of the modelling language used for the EMF model. Thus, converting from RDF to EMF results in data loss when some RDF data can not be converted into an EMF model equivalent. If the same EMF is transformed back to RDF, overwriting the original RDF, then the data loss is permanent in both versions of the model.

MOF2RDF [21] is a specification published by the OMG, for mapping MOF to RDF/OWL. MOF2RDF provides a structural mapping with a limited semantic mapping and is intended to inform the creation of a model transformation process. The mapping seeks to preserve the original MOF model semantics, with all elements of the MOF model being mapped to an RDF/OWL model. The specification is clear on what it does not intend to provide, such as formal logic or model-theoretic semantics, facilities for manipulating MOF or RDF models, versioning or tracking, inference of query usage of RDF models, or containing the platform-specifics of either MOF or RDF models. To the best of our knowledge, we do not know of any implementations or usage of the specification.

EMFtriple was found on GitHub [22] and the Eclipse Marketplace [23], and it persists EMF models using RDF triples. We assume EMFtriple was created to overcome the limitations of the ATL prototypes' bidirectional transformation problems with data loss, as the intended use seems similar to that of the EMF-RDF resource. However, we could not find in EMFtriple any use of EMF's notification system that could provide live synchronisation between the EMF view and the backing RDF representation. Our EMF-RDF resource is similar to EMFtriple in that both approaches persist EMF models with RDF, but EMFtriple uses a single RDF file and does not use a graph reasoner or unions when loading RDF models. Our EMF-RDF resource exposes Jena's graph validator, reasoner and unions to enable multiple RDF schema/data models to be used when loading an EMF model from RDF sources. In addition, the EMF-RDF resource permits direct access to the RDF model, which EMFtriple does not.

openCAESAR is a Model-Based System Engineering methodology/framework that balances agility and rigor [6]. They define agility as the ability to move; this movement could be in response to changes or portability/reusability between projects. Rigour is defined as 'formalising' processes, methods or languages, relating to properties associated with terms like 'reliability' or 'safety'; things that can hinder movement by requiring additional efforts. These are the kind of desirable properties we want the EMF-RDF resource to enable.

At the heart of openCAESAR is the OML [24] ontology; their perspective of creating a modelling language is that of modelling a vocabulary that is used to build models [25]. The openCAESAR framework is implemented using EMF, but uses an ontological approach to modelling. However, there can be difficulties moving to openCAESAR from existing practices, which is identified as an open challenge in Elassar et al.'s paper [6]. We are aware that the openCAESAR community is working on improving interoperability and integration: our EMF-RDF resource may help to connect openCAESAR with existing EMF-based MDE tools. The EMF-RDF resource seeks to maintain compatibility with the original EMF-based tools, and can accommodate additional RDF statements beyond those related to the EMF model.

VI. Conclusion and future works

The EMF-RDF resource presented in this paper is open-source and publicly available from GitHub[5]. We are creating an open-source implementation that we hope reduces some

[5]https://github.com/epsilonlabs/emf-rdf

technology barriers between open- and closed-world modelling approaches. MDE enables the creation of modelling languages and tools to use them, enabling complexity to be managed through leveraging different levels of abstraction, which may assist open-world modellers. However, MDE comes at a cost of rigidity caused by the conformance relationship challenge [16], which ontological approaches do not have. The EMF-RDF resource enables EMF-based models to be persisted in an RDF form that retains compatibility with their original tools. It also allows access to the RDF model, which may contain information outside of the modelling language of the EMF model.

The EMF-RDF resource is under active development, with the aim to enable the creation of an EMF model from nothing. Currently it supports reading models and saving back changes to model element attributes, but we aim to support every editing operation in EMF and be fully compatible with existing EMF tooling. This is a big task, and many possible edge cases could be challenging. However, we are confident that our synchronisation-based approach will enable the exchange of models between EMF and RDF without the data losses of prior approaches that were based on model-to-model transformations. When combined with an MDE toolset like Epsilon [11], which can work with many different types of closed-world models, the potential cases for the EMF-RDF resource greatly increase.

When the EMF-RDF resource supports most of the EMF features, a model copy operation could be used to convert a native EMF model resource into an EMF-RDF resource. With an EMF model in an RDF form, it is possible to perform open-world operations like a bundle closure, which enforces the difference between model features/classes. When the EMF-RDF resource is sufficiently complete, our research will pivot to explore new usage scenarios enabled by EMF models in RDF form, leveraging open-world techniques such as Jena's graph unions and reasoners, which could enable new ways of working with existing EMF models.

REFERENCES

[1] Dassault Systèmes, "MagicGrid book of knowledge." [Online]. Available: https://discover.3ds.com/magicgrid-book-of-knowledge

[2] J. G. Süß, S. Swift, and E. Escott, "Using devops toolchains in agile model-driven engineering," *Software and Systems Modeling*, vol. 21, no. 4, p. 1495–1510, Aug. 2022.

[3] H. Alfraihi and K. Lano, "Trends and insights into the use of model-driven engineering: A survey," in *2023 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*, Oct. 2023, p. 286–295, doi:10.1109/MODELS-C59198.2023.00058.

[4] G. Hillairet, F. Bertrand, and J.-Y. Lafaye, "Bridging EMF applications and RDF data sources," in *Proceedings of the 4th International Workshop on Semantic Web Enabled Software Engineering, SWESE*, Oct. 2008. [Online]. Available: https://hal.science/hal-00385823

[5] M. Milanović, D. Gasevic, A. Giurca, G. Wagner, and V. Devedzic, "On interchanging between OWL/SWRL and UML/OCL," in *Proceedings of 6th Workshop on OCL for (Meta-) Models in Multiple Application Domains (OCLApps) at the 9th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems (MoDELS), Genoa, Italy*, Feb. 2006.

[6] M. Elaasar, N. Rouquette, D. Wagner, B. J. Oakes, A. Hamou-Lhadj, and M. Hamdaqa, "openCAESAR: Balancing agility and rigor in model-based systems engineering," in *2023 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*, Oct. 2023, p. 221–230, doi:10.1109/MODELS-C59198.2023.00051.

[7] W3C, "W3C Semantic Web Standards - homepage," date last checked: June 2025. [Online]. Available: https://www.w3.org/2001/sw/wiki/Main_Page

[8] Eclipse Foundation, "Eclipse Modeling Framework," Jun. 2025, date last checked: June 2025. [Online]. Available: https://projects.eclipse.org/projects/modeling.emf.emf/

[9] RDF working group, "W3C RDF homepage," 2025, date last checked: June 2025. [Online]. Available: https://www.w3.org/RDF/

[10] W3C, "RDF - W3C Semantic Web Standards," date last checked: June 2025. [Online]. Available: https://www.w3.org/2001/sw/wiki/RDF

[11] Eclipse Foundation, "Eclipse Epsilon homepage," 2025, date last checked: June 2025. [Online]. Available: https://www.eclipse.org/epsilon/

[12] A. Hussain, W. Wu, and Z. Tang, "An MDE-based methodology for closed-world integrity constraint checking in the semantic web," *Journal of Web Semantics*, vol. 74, p. 100717, Oct. 2022, doi:10.1016/j.websem.2022.100717.

[13] W3C, "OWL - W3C Semantic Web Standards," date last checked: June 2025. [Online]. Available: https://www.w3.org/2001/sw/wiki/OWL

[14] Apache Software Foundation, "Apache Jena - homepage," date last checked: June 2025. [Online]. Available: {https://jena.apache.org/}

[15] Eclipse Foundation, "Eclipse Epsilon Picto homepage," 2025, date last checked: June 2025. [Online]. Available: https://eclipse.dev/epsilon/doc/picto/

[16] N. Hili and J.-S. Sottet, "The conformance relation challenge: Building flexible modelling frameworks," in *Proceedings of MODELS 2017 Satellite Events*, vol. 2019. CEUR-WS, 2017, pp. 418–423. [Online]. Available: https://ceur-ws.org/Vol-2019/flexmde_6.pdf

[17] N. Hili, "A metamodeling framework for promoting flexibility and creativity over strict model conformance," in *Proceedings of the 2nd Workshop on Flexible Model Driven Engineering co-located with MoDELS 2016*, vol. 1694. CEUR-WS, Oct. 2016, pp. 2–11. [Online]. Available: https://ceur-ws.org/Vol-1694/FlexMDE2016_paper_6.pdf

[18] E. Guerra and J. De Lara, "On the quest for flexible modelling," in *Proceedings of the 21th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems*. Copenhagen Denmark: ACM, Oct. 2018, p. 23–33, doi:10.1145/3239372.3239376.

[19] J. Bézivin, F. Jouault, and P. Valduriez, "On the need for megamodels," in *Proceedings of the OOPSLA/GPCE: Best Practices for Model-Driven Software Development workshop, 19th Annual ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications,(2004)*, Vancouver, Canada, Oct. 2004.

[20] J. Di Rocco, D. Di Ruscio, J. Härtel, L. Iovino, R. Lämmel, and A. Pierantonio, "Understanding MDE projects: megamodels to the rescue for architecture recovery," *Software and Systems Modeling*, vol. 19, no. 2, p. 401–423, Mar. 2020, doi:10.1007/s10270-019-00748-7.

[21] Object Management Group, "MOF to RDF Mapping," September 2021, date last checked: June 2025. [Online]. Available: {https://www.omg.org/spec/MOF2RDF/1.0/About-MOF2RDF/}

[22] G. Hillairet, "ghillairet/emftriple," Jun. 2024, date last checked: June 2025. [Online]. Available: {https://github.com/ghillairet/emftriple}

[23] ——, "Emf triple — eclipse plugins, bundles and products - eclipse marketplace — eclipse foundation," date last checked: June 2025. [Online]. Available: {https://marketplace.eclipse.org/content/emf-triple}

[24] NASA Jet Propulsion Laboratory (JPL), "Ontological modeling language v2 - homepage," date last checked: June 2025. [Online]. Available: https://www.opencaesar.io/oml/

[25] D. A. Wagner, M. Chodas, M. Elaasar, J. S. Jenkins, and N. Rouquette, *Ontological Metamodeling and Analysis Using openCAESAR*. Springer, Cham, 2022, p. 1–30, doi:10.1007/978-3-030-27486-3_78-1.