Linked Open Simulations: An Ontology-Based Approach for System Dynamics Models on Insight Maker

Laryza L. Mussavi[1], Benno Kruit[1]

[1]Vrije Universiteit Amsterdam, De Boelelaan 1105, 1081 HV Amsterdam, The Netherlands

Abstract
This thesis introduces a System Dynamics ontology, SDMOnto, for the System Dynamics models (SDMs) available on the modeling platform Insight Maker (IM). The aim was to capture the metadata associated with these models and automate their translation to RDF using a programming approach. This involves integration with SDMOnto to enhance model interoperability and promote the reusability of cross-domain SDMs. SDMOnto consists of 24 classes, 14 object properties, 11 data properties, and 129 pre-defined individuals. Three test cases (models) were used to develop and evaluate the ontology's capability to store SDMs from IM as intended, and to identify any inconsistencies within the ontology. We found that each test case generated corresponding triples and individuals without resulting in any inconsistencies, indicating a successful conversion. In addition to the main research, we proposed a preliminary implementation for the integration of external, tabular data and its units of measure based on CSVW (CSV on the Web) standards. This implementation also generated consistent conversions. However, further inspection revealed inaccuracies, which require refinements in the implementation and more testing to prove feasibility. Since this paper is a first attempt at a standardized representation for SDMs, we recommend that future research focuses on evaluating the scalability and reusability of the ontology to address any issues that may arise on a larger scale.

1. Introduction

This paper presents an RDF-based ontology aimed at improving the integration and reusability of System Dynamics models available on the Insight Maker platform. Insight Maker (IM) is an online application that allows users to create, share, and adjust models across various domains [1]. While the System Dynamics models (SDMs) on IM cover many domains, they lack modularity. Consequently, data cannot be easily reused and integrated among models, and certainly not across modeling platforms, limiting the dissemination of knowledge. A formal representation in RDF is meant to generate a linked data representation of the Insight Maker models that will make it easier to inspect, analyze, and work with the models using automated methods. Because translating these models to RDF becomes a tedious task for large or numerous models, our goal is to automate this process using a programming approach.

Existing models may rely on numerical, tabular datasets that contain either hypothetical data, or real data obtained from an external source. It becomes difficult to ensure that models use updated tables as this data evolves. Moreover, there is currently no option in IM to automatically add information about the external data source. If we can also integrate these sources using RDF, it becomes easier to update, connect, and combine data from different models that use external data, leading to more efficient model analysis and development.

As a result, we not only want to capture the underlying structure of a model and automate its translation to RDF, but also propose an extension to the ontology that allows for the integration of metadata associated with any external data sources. Although a complete version of this extension is beyond our scope, a preliminary implementation is presented to showcase feasibility. Ultimately, the ontology described in this paper will facilitate cooperation and encourage the reuse of models across different platforms and domains by providing a foundation for a standardized framework for publishing and distributing the System Dynamics models on Insight Maker.

EKAW 2024: EKAW 2024 Workshops, Tutorials, Posters and Demos, 24th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2024), November 26-28, 2024, Amsterdam, The Netherlands.
l.l.mussavi@student.vu.nl (L. L. Mussavi)

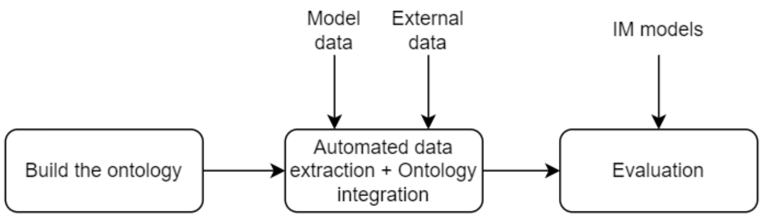


Figure 1: High-level overview of the development process

Our main goal can be summarized by the research question: "How can we enhance the interoperability and accessibility of System Dynamics models created using Insight Maker?"

This is supported by two sub-questions:

1. "Which ontological constructs are most suitable for developing an RDF-based ontology to represent SDMs within Insight Maker?"
2. "Can data source mapping enhance the interoperability and effectiveness of System Dynamics models represented in RDF, enabling dynamic data retrieval and updates from external sources, including unit verification?"

The next sections will address these questions using the following structure: Section 2 discusses the relevance of the topic and its importance in filling the currently existing gap in research by presenting background information and related work. Section 3 introduces the methods used to build the ontology, translate IM models to RDF, and a proposed method for the extraction of external data source and unit integration. Section 4 describes the results of our main research and this preliminary implementation. Finally, the discussion in Section 5 evaluates the research set-up and the pitfalls and benefits of the approach taken, followed by a summary and suggestions for future research in the Conclusion. Figure 1 contains a high-level overview of the practical process behind this structure.

2. Background information and Related Work

2.1. System Dynamics

System dynamics (SD) is a method that focuses on capturing the structure and behavior of complex, dynamic systems[1]. Within SD, the concept of *feedback systems thinking* plays an important role in interpreting models that are based on so-called *causal loop diagrams* (CLDs). These are a type of system dynamics model (SDM) consisting of feedback loops that influence the flow through the model, either postively or negatively [2]. However, when the scenario we want to model becomes more complex, reaching the point of not being solvable with the currently available mathematical knowledge, only a step-by-step solution in the form of a simulation may be possible [3]. Qualitative representations such as CLDs are insufficient when making a simulation model, meaning a more complex and algebraic structure is required, which introduces *stock-flow diagrams* and *differential equations.* A stock-flow diagram is another type of SDM representing stocks regulated by flows that vice versa depend on the condition of a stock. This regulation takes place under the influence of differential equations that can range from relatively simple to extremely complex [4].

The SDMs on IM are primarily based on stock-flow diagrams with differential equations. However, challenges of these simulations include the management of heterogeneous data that requires expert knowledge and the inability to exchange data or work together in a bigger system without manual linking. Consequently, connecting SDMs becomes a time-consuming task, which can be costly in a fast-paced work environment. To remove these limits, ontologies provide a solution capable of handling and combining the diverse types of information they hold—both written and numerical [5].

[1]A collection of components that work together for a general goal

2.2. Ontologies for Knowledge Integration

An ontology, in the context of information sciences, refers to a formal representation of a common set of concepts and the relationships between them that make up a schema. They are important building blocks of knowledge graphs and the Semantic Web as they define and reason about the semantics of entities and their relations. Ontologies can help improve interchangeability and reusability of heterogeneous information, making them a well-suited model to achieve improved interoperability [6]. Furthermore, they address the following challenges of modeling and simulations and interoperability as described by [7] : 1) Semantic Inaccessibility, 2) Logical Disconnectedness, and 3) Consistency Maintenance.

These issues also apply to SDMs to some extent, but could be overcome by proposing a formal representation in RDF that covers their basic underlying structure. RDF (Resource Description Framework) is a standard data model which is used in the creation of ontologies, recommended by W3C for data interchange on the web [8]. It is a formal machine-processable language that underpins the semantic structure of an ontology by enabling the description of any defined *instances*[2] and their logical relations in the form of triples [9]. A standardized ontology would not only allow for models to be translated to the same RDF format, but, vice versa, also make for greater accessibility across various domains as models are integrated into the same knowledge base, thereby enhancing model interoperability and accessibility [10].

2.2.1. Related Ontologies

Several studies [11, 12, 13] have explored and proposed ontologies for improving interoperability across multiple systems or domains. A notable example is CDOnto for the Internet of Things (IoT), a result of the desire for a method able to describe common and generic IoT data [14]. *Alignment* and a *contextual approach* form the basis of CDOnto. Alignment is the task of identifying the overlapping characteristics between concepts to enable meaningful data interchange. The contextual approach implies that domains (i.e. contexts) have common concepts and independent domain data. Consequently, CDOnto is built on global concepts and links that form the top-level ontology, while instantiating them with domain-specific information on a local level.

Attempts such as the ones above have yet to be made for the field of system dynamics modeling. Similar to CDOnto, creating an ontology for SDMs containing only the fundamental elements that make up their structure would be a well-suited approach to enhance integration. In the context of this paper and thus with respect to SDMs, they could be interpreted as follows:

- *Alignment* – Identifying the underlying structure of a System Dynamics model on Insight Maker, i.e. the main components defined by SD-specific terms. This is the overlapping characteristic between all SDMs regardless of their domain.
- *Global vs. Local representations* – The idea that each SDM is translated to RDF as its own model entity, while specific model data will be translated to instances in RDF and connected to the corresponding model.

2.3. System Dynamics Models in Insight Maker: Key Concepts and Attributes

To design an ontology for SDMs available on IM and address the task of alignment, it is important to identify the key concepts and attributes that make up the basic structure of these models.

The main building blocks of all SDMs are *primitives*. These consist of the earlier mentioned *stocks* and *flows*, with the addition of *variables* and *converters*. A Variable can be a constant or a dynamic value. Converters are data transformers that allow users to map inputs to outputs, much like a graphical function. These outputs are then used to influence the values of other primitives in the model. Table 1 contains an overview of the primitives, their uses, and their symbolic representation on IM. For an example of what the SDMs look like, see fig. 2 which will be discussed as part of section 3.1.

[2]Also referred to as *entities* or *individuals*

Table 1
Overview of IM key primitives as presented by [15]

Primitive	Use	Example usages	Symbol
Stock	To store a material. Equivalent to a state variable in a differential equation	A lake that stores water. A bank account that stores money. A population that stores people	▭
Flow	To move material between stocks. Equivalent to a derivative term in differential equations	A river into a lake. Withdrawals from a bank account. Deaths in a population	⇒
Variable	To store a parameter value or dynamically calculate a feature of the model	The flow rate for a river. The interest rate for a bank account. The death rate for a population	⬭
Converter	To load tabular data. To carry out a non-parametric transformation function	A schedule of flow rates by time of year. A non-parametric function relating population death rate to population size	⬡

Models in IM have links (arrows) between primitives to visually present that they are related. However, contrary to this visual connection, primitives primarily influence each other through the use of *mathematical expressions* that tie them together into one dynamic system and determine the output of the simulation. A mathematical expression can be one of two kinds: 1) a constant or 2) a more complex, dynamic expression, a differential equation. The latter may consist of direct numerical or custom pre-named constants, custom functions or those built-in in IM's simulation engine [16] , or the values of other primitives.

Another important component is the *unit of measure*. Since IM simulations depend on numerical values, specifying units for each primitive is "a great way to clarify conceptualization, disambiguate terminology, guard against common formulation errors, and check the model for dimensional consistency" [17]. While current SDMs on IM specify and validate units within the models themselves, external tabular data may encompass multiple units that are now being excluded.

2.4. Tabular Data

In the context of SDMs, semantic accessibility plays its parts when using external data. Currently, there is no way for the SDMs on IM to determine and define the metadata associated with its original source. Consequently, information becomes limited in this part of the model and thus also no possibility to easily extract and reuse this data to link or create other models. As can be done with the basic structure of an SDM, ontologies would make this integration of tabular data easier by presenting them in a common RDF format. It remains an ongoing topic in research, however, how to obtain and convert tabular data to RDF. Scholars like [18, 19] have proposed implementations that align with the guidelines in CSV on the Web (CSVW) [20]. CSVW is another W3C recommended standard used to describe the content of CSV files, capturing their metadata [21]. It offers users a method to transform CSV to RDF. We believe this could serve as a viable solution to fill the current gap in semantic accessibility of external tabular data. Hence, our goal is to combine CSVW standards with the reuse of existing unit ontologies such as QUDT [22] to explicitly define and standardize the units used in external datasets. This ensures consistency and clarity when integrating data from different sources.

3. Methodology

Using the key concepts and attributes introduced in section 2.3, this section discusses the design approach and experimental setup.

To keep our design focused and avoid obscurity, two assumptions were made regarding the level of detail to include in our ontology:

1. Links are considered ambiguous because of their visual purpose. Different parts of a model are expected to be connected through mathematical expressions or Stock-Flow relations.

2. Since our research focuses on the symbolic representation of these expressions, we only consider the custom pre-defined or built-in constants in a differential equation, and not the numerical values. However, we do include numerical stand-alone constants, as we expect a primitive expression to have at least some value.

3.1. Building the Ontology

The next step is to build the ontology in *Protégé*, an open-source editor that adheres to W3C standards [23]. Ontologies promote data sharing through linked data, often by reusing existing ones. However, since no previous system dynamics ontology exists, our challenge will be to build one from scratch. We will do this by following widely accepted principles in ontology development, as discussed by various researchers (e.g., [6]).

We have determined the scope of the ontology, namely the underlying structure of System Dynamics Models on Insight Maker. The subsequent stages of the development process are explained below.

3.1.1. Test Models

The following three models (test cases) were selected to develop and evaluate the ontology:

1. The Predator-Prey model (PP) [24]
2. The Climate Change model (CC) [25]
3. The Emergency Department Story model (ED) [26]

Each model was chosen to cover distinct domains and contains at least one instance of every primitive type (stock, flow, variable, and converter) so that their translation to RDF can be assessed across multiple cases and domains. Figure 2 presents the test cases. Throughout development, the CC model served as a reference to confirm that the design included the identified key concepts and attributes. CC contains all necessary elements and incorporates external data. Together with PP and ED, it was used to check for inconsistencies in the final evaluation of the ontology.

The main concepts of SDMs introduced in section 2.3 are used to create classes. To determine the class hierarchy, concepts are clustered according to their purpose in the model. IM models can be converted into an XML file containing all attributes related to their primitives. Data and object properties are selected by reviewing these attributes and their relevance to the project. Since the aim is to standardize System Dynamics models, we aim to keep the ontology abstract, adding only domain and range constraints to test entity relation consistency.

Next, we want to describe the models in RDF using the proposed ontology. There is currently no way to automatically extract RDF triples from the models. As this becomes a nearly impossible task to do by hand for large amounts of data, we want to automate the conversion using a programming approach. Our implementation leverages JavaScript and Python to extract the information and convert it into RDF. We used JavaScript to collect all the data, which was then processed in Python to achieve the desired format and integrate it into the ontology.

Insight Maker offers its users a code-interface simulation engine built on the *simulation* package which runs on JavaScript (JS) [27]. Users can create models directly using the package, but also import and run existing models built on IM. The latter makes it particularly useful for this project since some of its methods can be used to obtain the data that will be translated to RDF. As a result, we wrote a JS script that extracts and saves instances and their attributes in TSV (tab-separated values) files. These files formed the basis of the triple generation in Python with the RDFLib library. RDFLib enables importing the SDM ontology, creating an RDF graph of an IM model, and integrating this graph into the ontology [28].

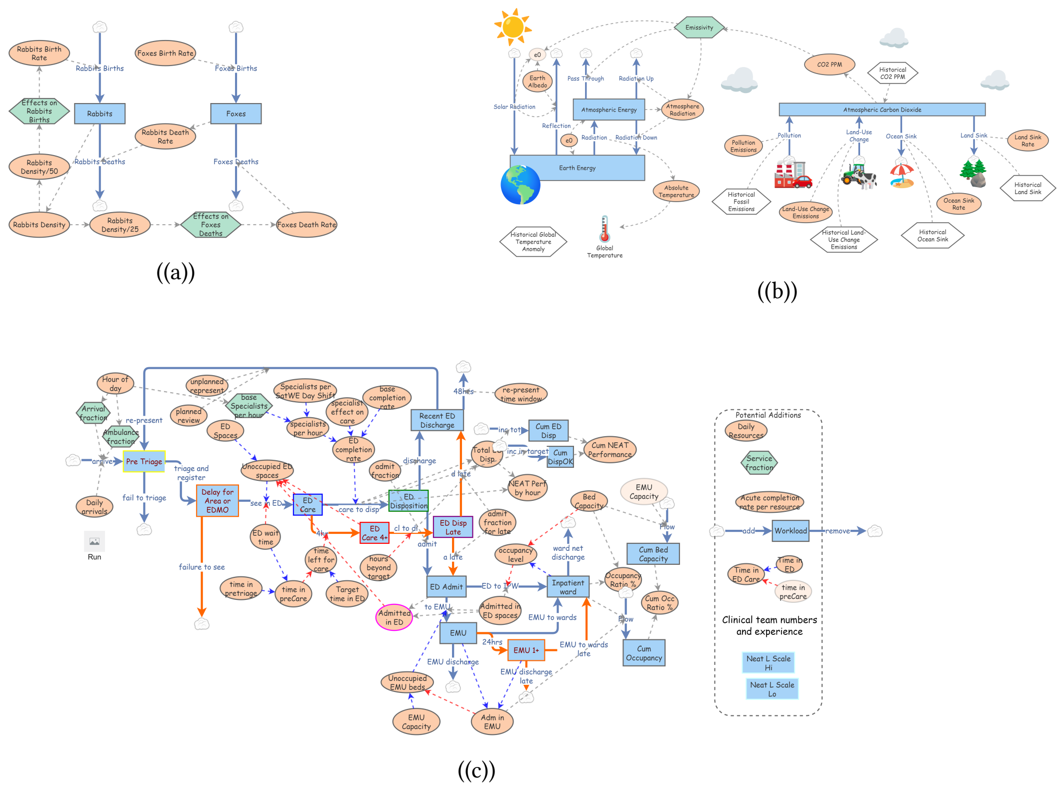


Figure 2: (a) The Predator-Prey model (b) The Climate Change model and (c) The Emergency Department Story model. For larger figures, see Appendix.

3.2. A Preliminary Implementation of External Data Integration

It is possible to incorporate external, tabular data in the Insight Maker models using the converter primitive, but as mentioned before, there is no option to describe the metadata associated with these tables. To describe external data sources as entities with the advantage of better integration, we propose a method using the tool *CoW* [29]. CoW enables the import of CSV files and conversion to JSON (metadata) or RDF according to CSVW standards. The advantage of CoW is that it provides its users with a web service or GUI that makes it easy to convert CSV to RDF.

While IM is only able to specify units within models themselves, external tabular data can encompass multiple units as well. CoW will allow us to accommodate multiple units that may be present in external tables. The idea is to add units defined in QUDT to the generated JSON files. If we can convert the data to RDF, it will become possible to define external data entities and link them to the corresponding converter in the model through the developed ontology. This would enrich the ontology with data that cannot be automatically extracted using only the IM simulation package.

Note that this preliminary implementation will not be included in the evaluation of the working ontology in the next section. Instead, we will present its results separately, focusing on its ability to generate triples and integrate units that accurately represent the original source. This approach ensures that the objective of our research remains consistent and focused.

After completion of the ontology, conversion scripts, and optional integration of external data, the goal is to validate the ontology against the three test cases from IM and check for any inconsistencies.

4. Results

The following sections present the final ontology: *SDMOnto* (System Dynamics Model Ontology). First, a comprehensive overview of SDMOnto will be given, presenting the main classes and relations. This will be followed by a validation of the RDF translation script combined with the integration into SDMOnto using a selection of IM models.

4.1. SDMOnto: Conceptualization

SDMOnto consists of 24 classes, 14 object and 11 data properties (predicates), and 129 individuals[3]. Table 2 contains an overview of all the classes, their hierarchy in terms of subclass relations, and a brief description of their use.

Table 2
Classes and class hierarchy in SDMOnto

Class	Subclass of	Use
Model		Class for IM models
Primitive		Class for IM primitives
Symbol		Class for symbolic representation that may appear in a mathematical expression
Expression		Class for mathematical expressions
DataSource		Class for external data sources
Stock	Primitive, Symbol	Class containing the stocks in a model
Flow	Primitive, Symbol	Class containing the flows in a model
Variable	Primitive, Symbol	Class containing the variables in a model
Converter	Primitive, Symbol	Class containing the converters in a model
Constant	Symbol	Class for any constants defined in the model
Function	Symbol	Class for any function defined in the model
BuiltInFunction	Function	Class containing all built-in functions available on IM
CustomFunction	Function	Class containing any function pre-defined by the user

Classes were based on the key concepts identified in section 2.3. These included: the main primitives (stocks, flows, variables, and converters), units, and mathematical expressions. Two more superclasses were derived to account for the role of these concepts: Primitive and Symbol. The classes Stock, Flow, Variable, and Converter are all of the type Primitive but are also considered Symbols when part of a mathematical expression used by another primitive (instances of the class Expression). The Symbol class was extended with the two remaining symbols that may be part of a mathematical expression: constants and functions, both custom and built-in. The latter consists of more subclasses to represent the different types of built-in functions, which were left out to maintain clarity in the overview.

Table 3 shows how instances of the classes are related to each other and to data values through object and data properties. SDMOnto includes the converter table data as a simple string object with the hasData data property. Additionally, we added the data property hasSourceURL to allow the main implementation to include links to external sources when provided by the user by creating an instance of the DataSource class with a relation to a string containing the URL. The DataSource class was also used to integrate external metadata and units using CoW, which will be evaluated separately in section 4.2.1.

A visual representation of the class relations using the object properties can be seen in the partial Tbox[4] in fig. 3. The highlighted box containing an RDF file with external illustrates that external data would be incorporated into SDMOnto through the DataSource class. Our focus, however, remains on the basic implementation in which foreign data is manually added with the hasData property and an optional source URL. For the complete Tbox of SDMOnto, see appendix A.

[3]Built-in constants and functions on IM that could already be pre-defined
[4]Visual representation containing the schema of an ontology (classes and relations using object and data properties)

Table 3
Object and Data properties in SDMOnto

	Domain	**Range**
Object Properties		
hasTimeUnit	Model	Unit
hasPrimitive	Model	Primitive
hasFlowrate	Flow	Expression
hasStart	Flow	Stock
hasEnd	Flow	Stock
hasInitialValue	Stock	Expression
hasInflow	Stock	Flow
hasOutflow	Stock	Flow
hasExpression	Variable	Expression
hasInputSource	Converter	Primitive, Time
hasDataSource	Converter	DataSource
primitiveOf	Primitive	Model
hasEquation	Primitive	Expression
hasUnit	Primitive, DataSource	Unit
Data Properties		
hasID	Primitive	xsd:int
hasName	Primitive	rdfs:Literal
hasStrExpression	Expression	rdfs:Literal
hasData	Converter	rdfs:Literal
hasTimeLength	Model	xsd:int
hasTimeStep	Model	xsd:decimal
hasTimeStart	Model	xsd:int
hasSourceURL	DataSource	rdfs:Literal

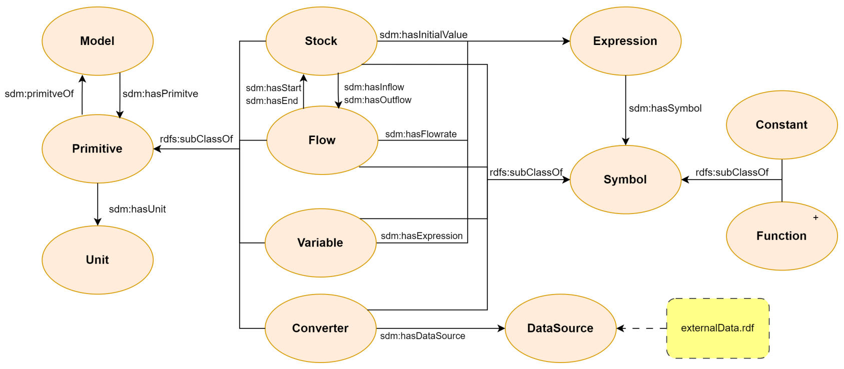


Figure 3: Partial Tbox of SDMOnto

4.2. Validating the Automated Translation to RDF

The design of the ontology combined with the translation scripts were assessed on the test cases (see section 3.1.1). Each model was transformed to RDF and linked to SDMOnto using a pipline implemented in Javascript and Python [5]. The results of their integration with SDMOnto were tested for any inconsistencies using the Hermit reasoner and manual inspection of the generated RDF graph in Protégé.

To remain clarity, we only present part of the smaller Predator Prey model after conversion to RDF, see fig. 4. Further visualizations of the conversion of all three test cases can be found in appendix B.

The main goal here was to demonstrate that the proposed ontological framework works as intended.

[5]all code is available at https://github.com/LaryzaL/Linked-Open-Simulations

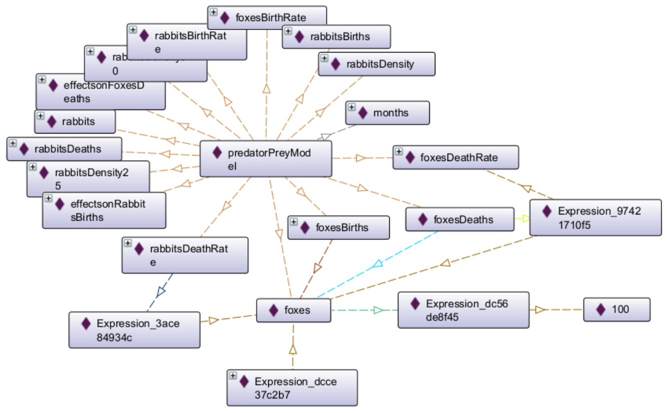


Figure 4: Partial visualization after translation of PP to RDF using SDMOnto

The results in table 4 imply a significant increase in the number of triples and unique individuals in SDMOnto after integration of the three SDMs. Through further inspection of the generated graphs in Protégé, it became evident that the conversion method was successful and models were properly converted, showing only some minor flaws in the syntax design. For example, numerical constants were defined as instances of the constant class, whereas the preferred, standardized representation would use literals. No major errors or inconsistencies were raised.

Table 4
Comparison of ontology size after Integration of IM models

Model	Triples	Unique Individuals	Growth in Triples (%)	Growth in Unique Individuals (%)
SDMOnto	391	129	-	-
PP	531	161	35.8%	24.8%
CC	740	205	89.3%	58.9%
ED	1208	309	209.0%	139.5%

4.2.1. Preliminary Implementation: Results of external data and unit integration

To perform the external data integration, we used the CC model as a test case because its creator specified the external sources they used for the converters, making it possible for us to test the preliminary method with CoW and QUDT. Table 5 shows the increase in knowledge triples and unique individuals after external data integration compared to the RDF translation of the CC model using only the hasData property (see table 3).

Table 5
Comparison of the size of CC before and after integrating external data

	Triples	Individuals	Classes
Without external data integration	740	205	24
With external data integration	1831	221	28

It becomes evident from the results in table 5 that the use of CoW substantially increased the amount of generated triples. This again indicates that the suggested implementation works. However, after inspection of the generated RDF file and its instances, it became clear that there were some issues regarding the translation to RDF and the accuracy of these results. The main problem with the CoW-generated content is that the relations between the source and the data are not properly formulated. It is possible to link the external RDF data to the corresponding converter by defining a DataSource instance that uses the same base URI as defined in the metadata during the conversion to RDF using CoW, but it goes wrong when defining the triples associated with this external data source. QUDT is not properly imported as ontology to define the units in the tables, and triples are noted solely using annotation properties. This is an issue we were unable to fix due to time constraints and to prevent the research from losing its main focus, which is SDMOnto and the automated translation to RDF of System Dynamics models on Insight Maker.

5. Discussion

Looking back on the development and validation of SDMOnto, naturally, it did not come without its challenges.

The ontology was constructed under the assumption that visual links are ambiguous (section 2.3) and primitives are connected through differential equations. Therefore, we excluded Insight Maker models that are built solely on links in which zero values are added for each primitive. This makes it impossible to properly represent all the connections in a model using RDF. If the ontology were to take links into account, this would mean it would encompass a larger range of models.

Another pitfall was the scope of the models covered. While we looked at System Dynamics models, it is important to note that simulations on IM can also be a combination of both System Dynamics and Agent-based modeling. These models are not covered by the ontology. Therefore, future work could investigate how SDMOnto may be extended to encompass not only pure SDMs, but also those that combine System Dynamics and Agent-based components for improved interoperability regardless of their type. This could be achieved through reuse of previously developed Agent-based modeling ontologies. Furthermore, the automated translation of models to RDF depends on the use of IM's simulation package. Hence, it would require adjustments to automate the transformation of models on other platforms.

Finally, in terms of the entire work process and its validation, we mentioned that no previous work has been done on the topic, thus we had no reference framework to work and compare the performance of our proposed methods with. Therefore, the way to go about it was to work with a simple trial and error approach using the test cases and the steps defined in section 3. While this resulted in the translation of models without inconsistencies, it goes without saying that it remains a first proposal of a formal standard that has to be further assessed. To identify areas where SDMOnto or the automated scripts need improvements, it is recommended that the ontology will be continued to be tested on scalability and reusability. Using a larger and more diverse set of SDMs alongside methods such as querying based on competency questions will help evaluate the ontology's weaknesses.

Regarding the proposed integration of external data and units, it became clear that the current implementation lacks the ability to properly generate triples that effectively describe their semantics. While we were able to generate data in RDF, a considerable amount of improvements is required to turn this into a well-structured, usable format without the difficulties of having to correct everything by hand. Nevertheless, it offered a first look into the feasibility of extending models in RDF through CSVW. We continue to believe this is possible, but only with a considerable amount of additional research.

6. Conclusion

In this paper, we built an ontology targeted at System Dynamics models found on the Insight Maker platform: SDMOnto. Its creation was made up of two parts: (1) designing the ontology that describes the general underlying structure of SDMs on IM, and (2) the creation of automated scripts to transform models to RDF and integrate their data with SDMOnto. The result consists of 24 classes and 129 pre-defined individuals, linked to each other by 14 object properties and 11 data properties. The results were tested on three test cases (models) that were successfully translated to RDF, generating a substantial amount of knowledge. Though the implementation proved to be successful, the suggested approach is not without limitations. Before conclusive answers can be given about the true success of SDMOnto, other alternatives and enhancements will want to be investigated. Future work may encompass testing on a larger set of models and the application of competency questions to identify what can be improved.

We also looked at a pre-limenary implementation of the incorporation of external tabular (meta)data using a combination of CSVW and CoW. The results show a significant difference in the amount of knowledge generated, but lacks a well-formulated structure when closely inspected. Fixing this issue will require further research efforts.

References

[1] S. Fortmann-Roe, Insight Maker, https://insightmaker.com/, Accessed 30-06-2024.

[2] M. Reynolds, S. Holwell, Systems approaches to making change: a practical guide, Springer, 2020. doi:https://doi.org/10.1007/978-1-4471-7472-1_2.

[3] B. K. Bala, F. M. Arshad, K. M. Noh, et al., System dynamics, Modelling and Simulation 274 (2017) 5–12.

[4] M. Reynolds, S. Holwell, Systems approaches to making change: a practical guide, Springer, 2020. doi:https://doi.org/10.1007/978-1-4471-7472-1_2.

[5] M. H. de Boer, Exploring knowledge extraction techniques for system dynamics modelling: Comparative analysis and considerations (2023).

[6] I. S. Bajwa, A framework for ontology creation and management for semantic web, International Journal of Innovation, Management and Technology 2 (2011) 116–118.

[7] P. Benjamin, M. Graul, A framework for adaptive modeling and ontology-driven simulation (FAMOS), in: Enabling Technologies for Simulation Science X, volume 6227, SPIE, 2006, pp. 39–49.

[8] D. Brickley, R. V. Guha, B. McBride, Rdf schema 1.1. w3c recommendation, World Wide Web Consortium 2 (2014).

[9] O. Lassila, Resource description framework (rdf) model and syntax specification, w3c, http://www. w3. org/TR/REC-rdf-syntax/ (1999).

[10] M. Hofmann, Ontologies in modeling and simulation: an epistemological perspective, in: Ontology, epistemology, and teleology for modeling and simulation: Philosophical foundations for intelligent m&s applications, Springer, 2013, pp. 59–87.

[11] Q. Geng, S. Deng, D. Jia, J. Jin, Cross-domain ontology construction and alignment from online customer product reviews, Information Sciences 531 (2020) 47–67.

[12] A. Devanand, G. Karmakar, N. Krdzavac, R. Rigo-Mariani, Y. F. Eddy, I. A. Karimi, M. Kraft, Ontopowsys: A power system ontology for cross domain interactions in an eco industrial park, Energy and AI 1 (2020) 100008.

[13] M. Fumagalli, G. Bella, S. Conti, F. Giunchiglia, Ontology-driven cross-domain transfer learning, in: Formal Ontology in Information Systems, IOS Press, 2020, pp. 249–263.

[14] S. Benkhaled, M. Hemam, M. Djezzar, M. Maimour, An ontology–based contextual approach for cross-domain applications in internet of things, Informatica 46 (2022).

[15] S. Fortmann-Roe, Insight maker: A general-purpose tool for web-based modeling & simulation, Simulation Modelling Practice and Theory 47 (2014) 28–45.

[16] Built-in Functions, https://insightmaker.com/docs/functions, Accessed 29-06-2024.

[17] IEEE Systems, Working with units, Accessed: 2024-06-24. URL: https://www.iseesystems.com/resources/help/v1-2/Content/03-BuildingModels/Working_with_Units/Overview_Working_with_Units.htm.

[18] D. Chaves-Fraga, L. Pozo-Gilo, J. Toledo, E. Ruckhaus, O. Corcho, Morph-csv: Virtual knowledge graph access for tabular data., in: ISWC (Demos/Industry), 2020, pp. 11–16.

[19] S. H. Mahmud, M. A. Hossin, M. R. Hasan, H. Jahan, S. R. H. Noori, M. R. Ahmed, Publishing csv data as linked data on the web, in: Proceedings of ICETIT 2019: Emerging Trends in Information Technology, Springer, 2020, pp. 805–817.

[20] CSVW - CSV on the Web, https://csvw.org/, Accessed 30-06-2024.

[21] J. Tennison, G. Kellogg, I. Herman, Model for tabular data and metadata on the web. w3c recommendation, World Wide Web Consortium (W3C) (2015).
[22] QUDT, Accessed: 2024-06-24. URL: https://www.qudt.org/.
[23] M. A. Musen, The protégé project: a look back and a look forward, AI Matters 1 (2015) 4–12. URL: https://doi.org/10.1145/2757001.2757003. doi:10.1145/2757001.2757003.
[24] group3, Investigation of Predator/Prey Modal 2, https://insightmaker.com/insight/5wxPfAWq0vyL38fCh077vB/Investigation-of-Predator-Prey-Modal-2, Accessed 29-06-2024.
[25] S. Fortmann-Roe, Global Climate Change, https://insightmaker.com/insight/34ijXHsL8uVmuT2vnYv7En/Global-Climate-Change, Accessed 29-06-2024.
[26] R. Forero, Story of ED Flows without Separate Areas, https://insightmaker.com/insight/2IBvMYSwzz6LYcFHUfvCOh/Story-of-ED-Flows-without-Separate-Areas, Accessed 29-06-2024.
[27] S. Fortmann-Roe, Simulation – Simulation and modeling for Node and the browser, https://github.com/scottfr/simulation/tree/main, Accessed 29-06-2024.
[28] R. Team, rdflib 7.0.0 documentation, https://rdflib.readthedocs.io/en/stable/, Accessed 30-06-2024.
[29] R. Hoekstra, CoW: Converter for CSV on the Web, https://csvw-converter.readthedocs.io/en/latest/, 2016. [Accessed 29-06-2024].

Appendix

A. SDMOnto

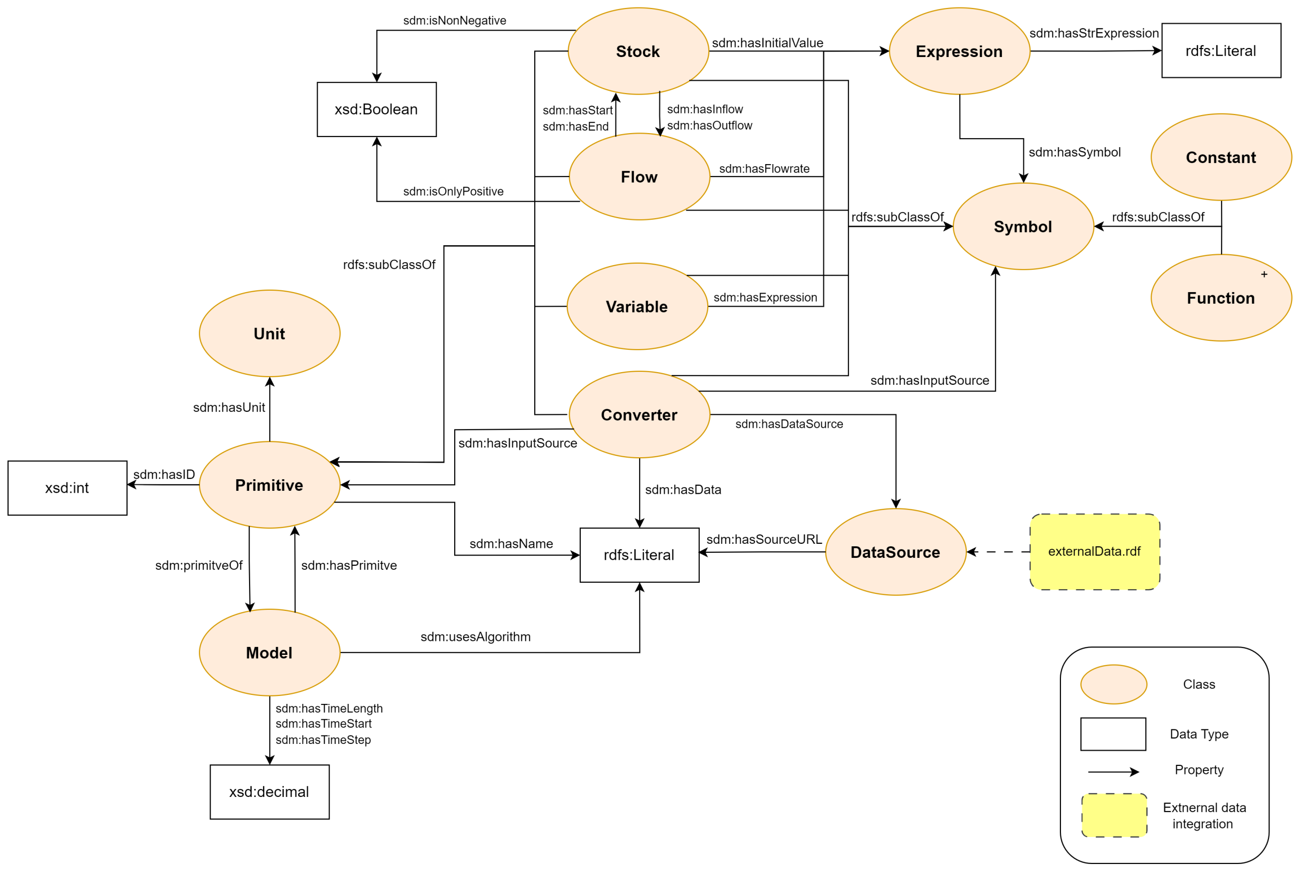


Figure 5: The Tbox of SDMOnto

B. Visualizations of PP, CC, and ED after translation to RDF using SDMOnto

B.1. Predator Prey model

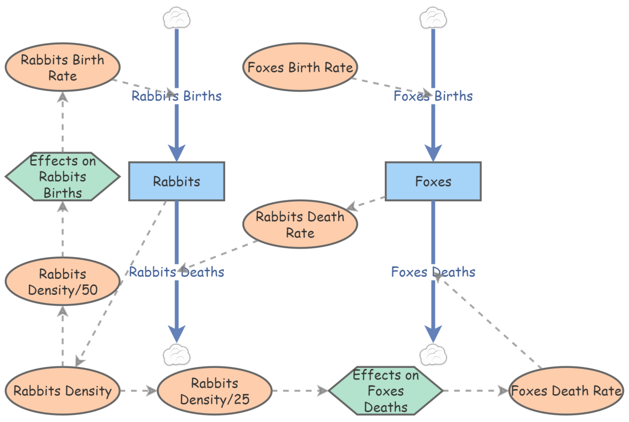
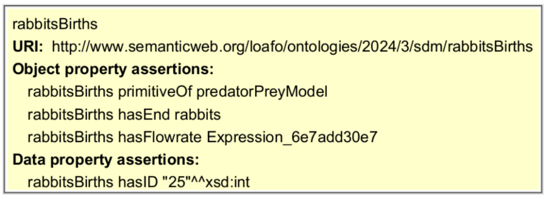
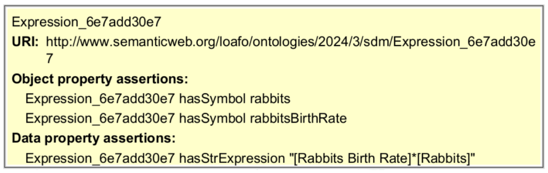


Figure 6: The Predator-Prey model



rabbitsBirths
URI: http://www.semanticweb.org/loafo/ontologies/2024/3/sdm/rabbitsBirths
Object property assertions:
rabbitsBirths primitiveOf predatorPreyModel
rabbitsBirths hasEnd rabbits
rabbitsBirths hasFlowrate Expression_6e7add30e7
Data property assertions:
rabbitsBirths hasID "25"^^xsd:int

(a).: Decription of the Flow rabbitsBirths



Expression_6e7add30e7
URI: http://www.semanticweb.org/loafo/ontologies/2024/3/sdm/Expression_6e7add30e7
Object property assertions:
Expression_6e7add30e7 hasSymbol rabbits
Expression_6e7add30e7 hasSymbol rabbitsBirthRate
Data property assertions:
Expression_6e7add30e7 hasStrExpression "[Rabbits Birth Rate]*[Rabbits]"

(b).: Decription of the Expression linked to rabbitsBirths

Figure 7: Example descriptions of PP instances in RDF

B.2. Climate Change model

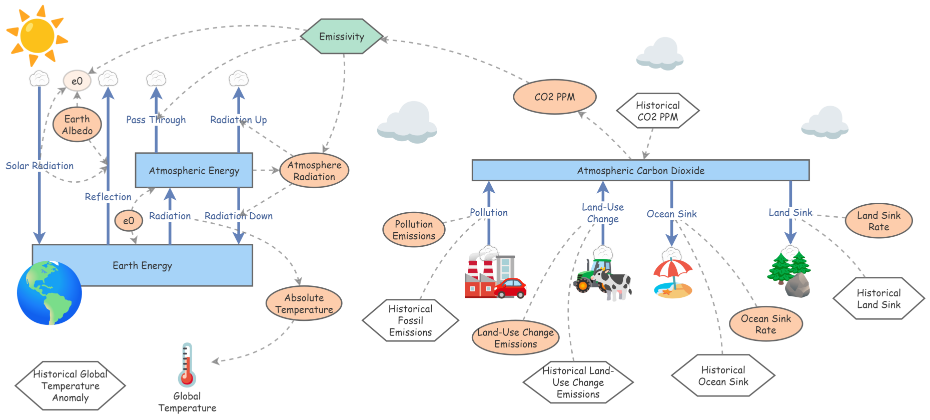
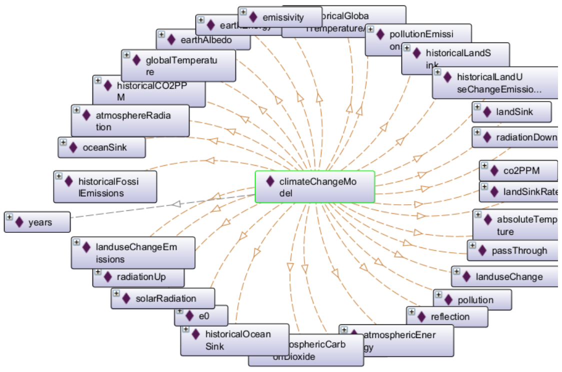
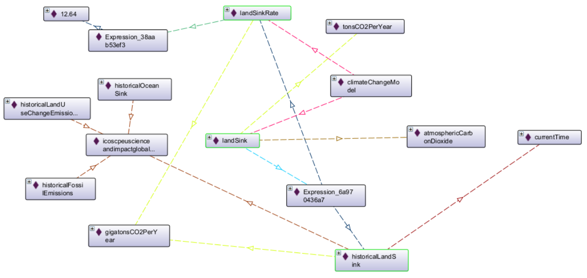


Figure 8: The Climate Change model

B.3. Emergency Department Story model

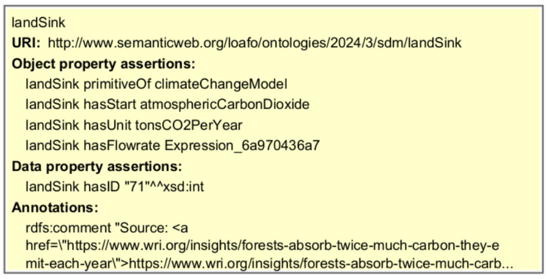


(a).: Graph of CC and its instances

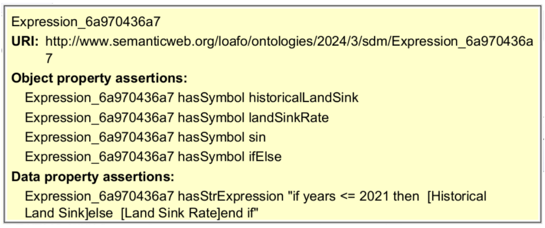


(b).: Graph of CC expanding on the Flow landSink

Figure 9: Partial visualizations of CC after translation to RDF using SDMOnto


landSink
URI: http://www.semanticweb.org/loafo/ontologies/2024/3/sdm/landSink
Object property assertions:
landSink primitiveOf climateChangeModel
landSink hasStart atmosphericCarbonDioxide
landSink hasUnit tonsCO2PerYear
landSink hasFlowrate Expression_6a970436a7
Data property assertions:
landSink hasID "71"^^xsd:int
Annotations:
rdfs:comment "Source: <a href=\"https://www.wri.org/insights/forests-absorb-twice-much-carbon-they-emit-each-year\">https://www.wri.org/insights/forests-absorb-twice-much-carb...

(a).: Decription of the Flow landSink


Expression_6a970436a7
URI: http://www.semanticweb.org/loafo/ontologies/2024/3/sdm/Expression_6a970436a7
Object property assertions:
Expression_6a970436a7 hasSymbol historicalLandSink
Expression_6a970436a7 hasSymbol landSinkRate
Expression_6a970436a7 hasSymbol sin
Expression_6a970436a7 hasSymbol ifElse
Data property assertions:
Expression_6a970436a7 hasStrExpression "if years <= 2021 then [Historical Land Sink]else [Land Sink Rate]end if"

(b).: Decription of the Expression linked to landSink

Figure 10: Example descriptions of CC instances in RDF

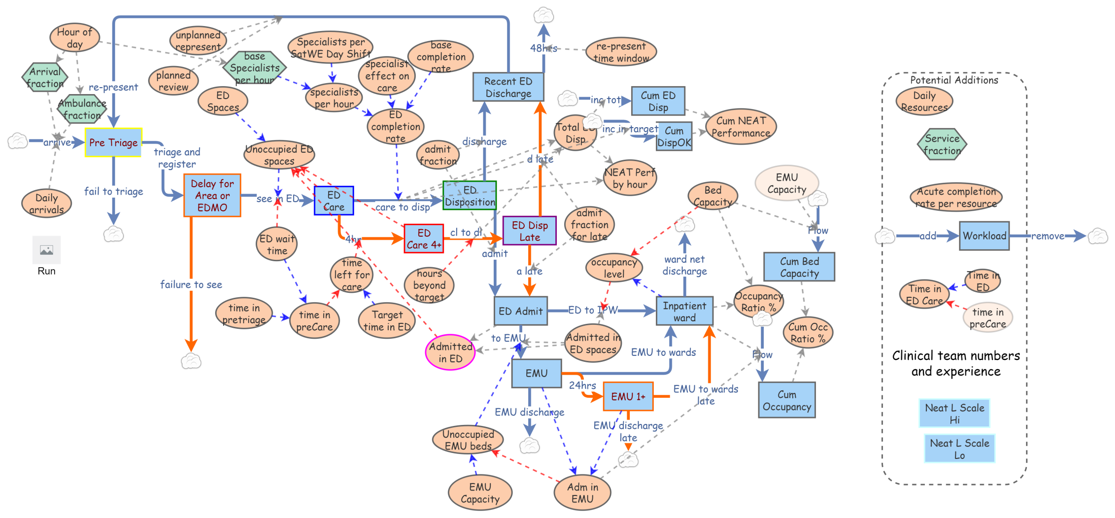
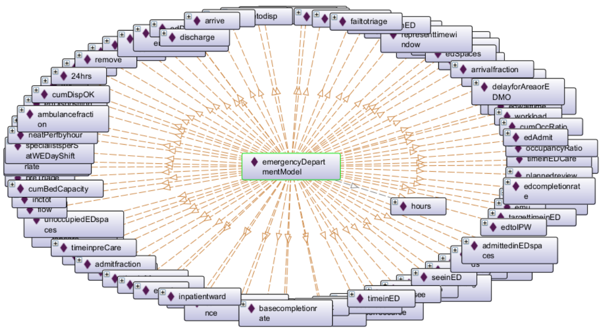
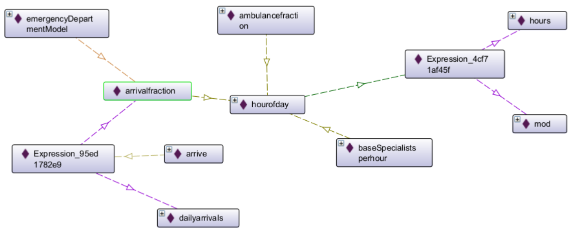


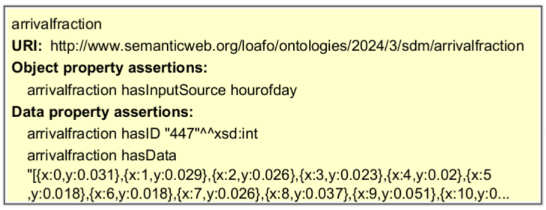
Figure 11: The Emergency Department Story model


(a).: Graph of ED and its instances

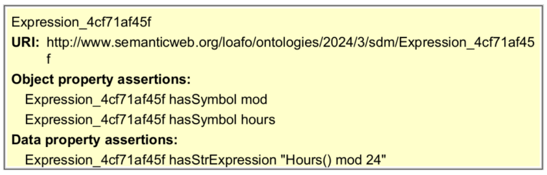


(b).: Graph of ED expanding on the Converter arrivalFraction

Figure 12: Partial visualizations of ED after translation to RDF using SDMOnto


arrivalfraction
URI: http://www.semanticweb.org/loafo/ontologies/2024/3/sdm/arrivalfraction
Object property assertions:
arrivalfraction hasInputSource hourofday
Data property assertions:
arrivalfraction hasID "447"^^xsd:int
arrivalfraction hasData "[{x:0,y:0.031},{x:1,y:0.029},{x:2,y:0.026},{x:3,y:0.023},{x:4,y:0.02},{x:5,y:0.018},{x:6,y:0.018},{x:7,y:0.026},{x:8,y:0.037},{x:9,y:0.051},{x:10,y:0...

(a).: Description of the Converter arrivalfraction


Expression_4cf71af45f
URI: http://www.semanticweb.org/loafo/ontologies/2024/3/sdm/Expression_4cf71af45f
Object property assertions:
Expression_4cf71af45f hasSymbol mod
Expression_4cf71af45f hasSymbol hours
Data property assertions:
Expression_4cf71af45f hasStrExpression "Hours() mod 24"

(b).: Description of the Expression linked to arrivalfraction

Figure 13: Example descriptions of ED instances in RDF