OntoUML and OntoObject-Z_based Ontologies: Application to Computer Communication Protocols

Mohamed Bettaz

Faculty of Information Technology
Czech Technical University in Prague
Czech Republic

The Czech Technical University in Prague

Czech Technical University in Prague

The biggest technical university in Central Europe. 310 years of tradition. 8 faculties and 6 university institutes.

Source: https://www.linkedin.com/school/cvutpraha/mycompany/

Centre for Conceptual Modelling and Implementation (CCMi)

OntoUML community portal

OntoUML and OntoObject-Z

- OntoUML* is a **profile** for the Unified Modeling Language (UML).
- OntoObject-Z is a **descriptive**** language inspired by OntoUML.
- OntoObject-Z is an **extension** of the Object-Z notation.
- Both OntoUML and OntoObject-Z are backed by (**IT**) ontologies (Ullman triangle, **shared conceptualization** between stakeholders and IT professionals).

* Conceptual modeling course at FIT (OntoUML, BPMN, DEMO, RDF/OWL)

** A language "suitable" for the specification of problem domain (as opposed to prescriptive languages used for solution domain). Risk of *bias of models* - (similarity with *bias of data* and *bias of algorithms* in ML).

An Example of an “Incomplete” Specification (in OntoUML)

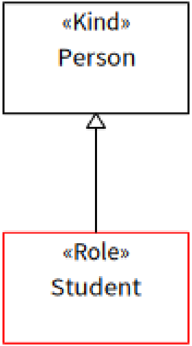

Figure 1: Role without relational dependency - illustration.

An Example of a “Complete” Specification (in OntoUML)

Figure 2: A role with a relational dependency - illustration.

An Example of an "Inconsistent" Specification in OntoUML

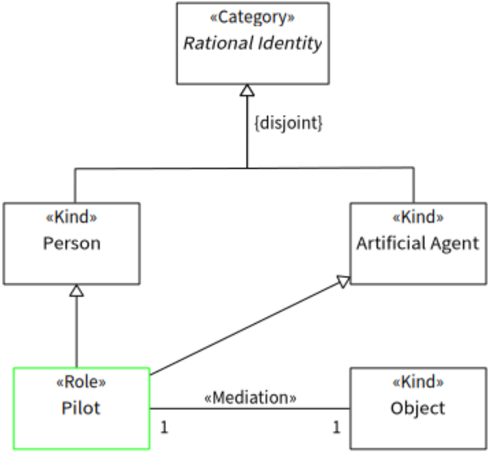

Figure 3: A type with more than one identity - illustration.

Goal

- Build OntoUML ontologies and implement them using OntoObject-Z specifications.
- **Conceptualize** these ontologies with RDF (Resource Description Framework) and OWL (Web Ontology Language).
- Take advantage from the "connection" between **RDF** and related stuff, and symbolic AI (**KRR** - Knowledge Representation and Reasoning systems).
- Illustration with Computer Communication Protocols (routing protocols).

Motivation

- Ability to model highly complex domains (such as computer communication protocols).
- OntoUML and OntoObject-Z are backed by the Unified Foundational Ontology - **UFO** (which "modifies" the semantics of UML (Object-Z) and makes it more "precise").
- The construction of ontologies for **computer communication protocols** is motivated by our interest in domain knowledge, a concept known from the **software engineering** discipline.
- Implementing OntoUML models with OntoObject-Z specifications:
  1. describe ontologies in a **formal** descriptive language.
  2. address most of the phases of the **SDLC** (Software Development Life Cycle) in a language equipped with a **refinement method**.
  3. express **specifications** and **constraints** on them in a **single language** (evolutionary approach vs discrete approach).

A Partial Taxonomy of UFO

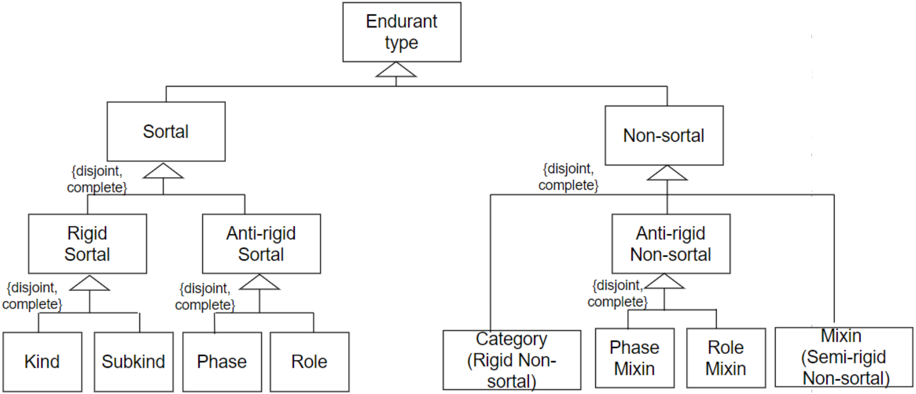

Figure 4: Source (Guizzardi et al. 2021).

Formalization of UFO Concepts Using Modal Logic

The example of rigidity and anti-rigidity (Guizzardi et al. (2021), Pergl (2024)):

- Rigidity: $R^{+}(T) =_{def} \Box(\forall\, x : T(x) \Rightarrow \Box(T(x)))$
  - Rigidity means inability to adapt to change.
- Anti-rigidity: $R^{-}(T) =_{def} \Box(\forall\, x : T(x) \Rightarrow \Diamond(\neg T(x)))$
  - Anti-rigidity means ability to adapt to change.

OntoObject-Z

- Introduced in two previous papers (Bettaz (2024), Bettaz and Maouche (2023)).

Object-Z Specification Signature

Definition 1 (Baumeister, Bettaz, Maouche and Mosteghanemi, 2015):
An Object-Z specification signature is a pair $\Sigma = (S, F)$, where,

- $S = C \cup T$ is the disjoint union of **class** names and primitive **types**.
- $F = B \cup R$ defines a family of operation symbols $B_{c \to t}, (c \in C, t \in T)$ representing **basic** attributes, and a family of operation symbols $R_{c \to 2^C}$ $(c \in C)$ representing **reference** attributes.

Note: **Reference** attributes are those used to capture **relationships** between Object-Z classes.

OntoObject-Z Specification Signature

Definition 2 (Bettaz and Maouche 2023):
Let Σ be an Object-Z specification signature, and $ST = (ST_c, ST_r)$, a pair, where ST_c is a set of **UFO class stereotype names** and ST_r a set of **UFO relationship stereotype names**. An OntoObject-Z specification signature is defined by the pair (f, g), where:

1. $f : (C \cup T) \rightarrow (C \times ST_c \cup T)$, and
2. $g : (B \cup R) \rightarrow (B \cup R \times ST_r)$, such that
3. $\forall\, c \in C, f(c) \in C \times ST_c, \forall\, t \in T, f(t) \in T$, and
4. $\forall\, b \in B, g(b) \in B, \forall\, r \in R, g(r) \in R \times ST_r$

Function 2 and expression 4 indicate that **reference attributes are stereotyped** by relationship stereotype names.

OntoObjectZ Metamodel

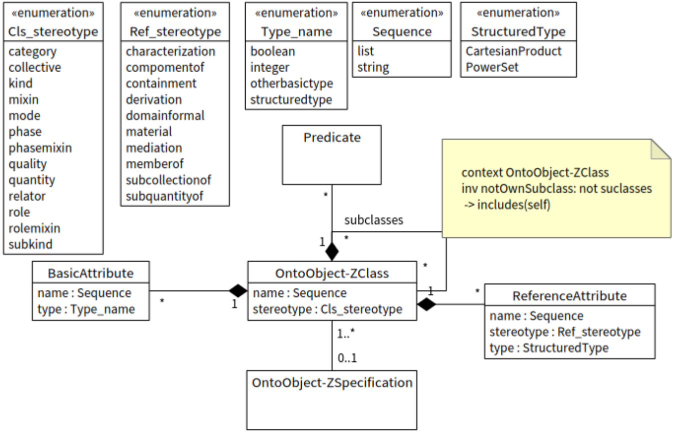

Figure 5: An OntoUML metamodel for OntoObject-Z.

EBNF

<OntoObject-Z_specification> ::= {OntoObject-Z_class}+
<OntoObject-Z_class> ::= <header> <state>
<header> ::= <header_of_sortal_class> | <header_of_nonsortal_class>
<header_of_sortal_class> ::= <<<cls_stereotype>>><cls_name> [{<cls_name>}]
<header_of_nonsortal_class> ::= <<<cls_stereotype>>> <cls_name> : abstract
<cls_stereotype> ::= category | collective | kind | mixin | mode | phase | phasemixin | quality | quantity | relator | role | rolemixin | subkind
<state> ::= [<declaration>] [<predicate>]
<declaration> ::= {<basic_attribute> | <reference_attribute>}
<basic_attribute> ::= <<<cls_stereotype>>> <cls_name> : <bas_attribute_name> : <type_name>
<type_name> ::= boolean | integer | . . .

EBNF (Cont'd.)

<ref_attribute> ::= <<<cls_stereotype>>> <cls_name> : <<<ref_stereotype>>>
<ref_attribute_name> : $\mathbb{P}$ <<<ref_stereotype>>> <cls_name>
<ref_stereotype> ::= characterization | componentof | containment | derivation |
domainformal | material | mediation | memberof | subcollectionof | subquantityof
<predicate> ::= {<multiplicity_predicate> | <navigability_predicate> |
<is_whole_for_predicate> | <is_part_of_predicate> | isDisjoint | isComplete}
<multiplicity_predicate> ::= # <ref_attribute_name> <rel_operator> <value>
<rel_operator> ::= >= | <= | = | < | > | !=
<value> ::= 0 | 1 | 2 | ...
<is_whole_for_predicate> ::= <cls_name> isWholeFor <cls_name>
<is_part_of_predicate> ::= <cls_name> isPartOf <cls_name>
<navigability_predicate> :: = ∀ <var_name> : <cls_name> • self in <var_name> ·
<cls_name>

Running Example

Few (very) basic structural and process aspects of routing protocols.
Structural aspects in **OntoUML** and behavioural aspects in **BPMN**. In "Towards an Ontological-based CIM Modeling Framework for IoT Applications" to appear in Informatica (Bettaz and Maouche, 2024)", we used **SoaML** (used usually in the modeling of **Cyber-physical systems**).

- Static vs dynamic routing
- Autonomous systems.
- Centralized vs decentralized algorithms.
- Generating routing tables.
- Service of a routing protocol.
- Routing protocol.
- Implementation of a protocol service.

Static vs Dynamic Routing

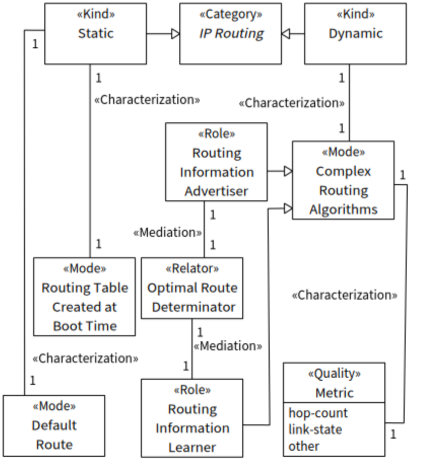

Figure 6: Static vs dynamic routing.

Static vs Dynamic Routing (Cont'd.)

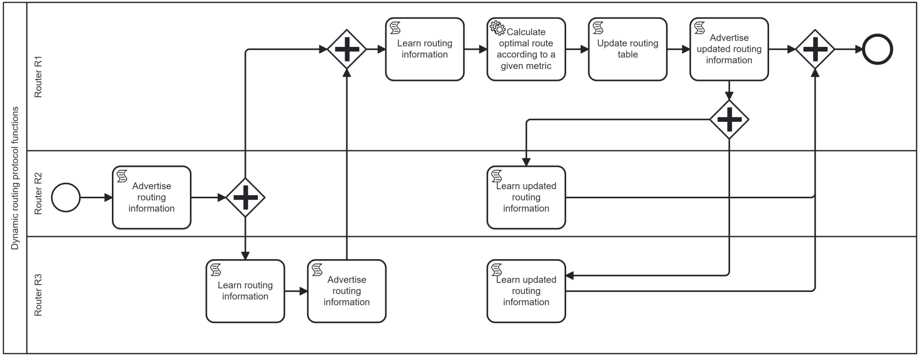

Figure 7: Related Process aspects.

On the Subject of the Whole and the Part Relationship

- “ It is as impossible to know the **parts** without knowing the **whole** as to know the whole without knowing the particular parts. ” (Blaise Pascal).
- “Mathematics may be defined as the subject in which we never know what we are talking about, nor whether what we are saying is true.“ (Bertrand Russell).
- Functional complex, Collective, Quantity, Aspects, Relator (Sortal Endurants).
- **Material** vs **Formal** relationships.

Autonomous Systems

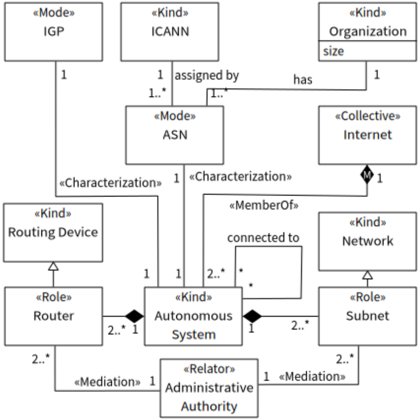

Figure 8: Autonomous systems.

Centralized and Decentralized Algorithms

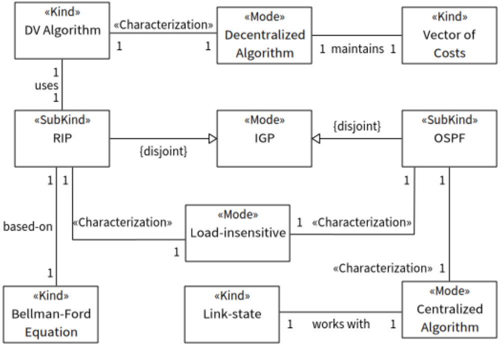

Figure 9: Centralized and decentralized algorithms.

Generating Routing Tables

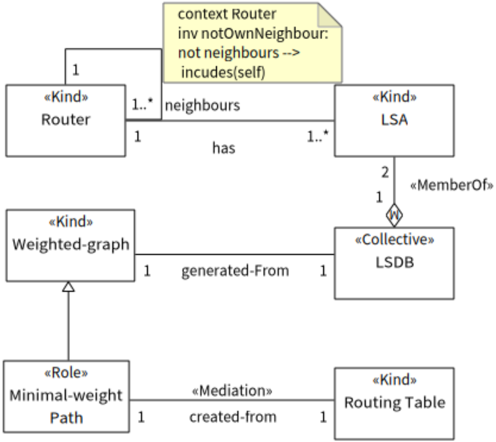

Figure 10: Generating routing tables.

Routing Protocol Service

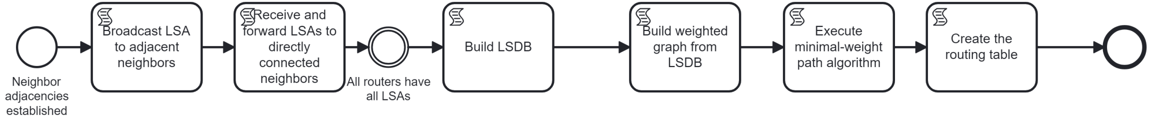

Figure 11: Process aspects.

LS routing Protocol Service Implementation with Collapsed Subprocesses

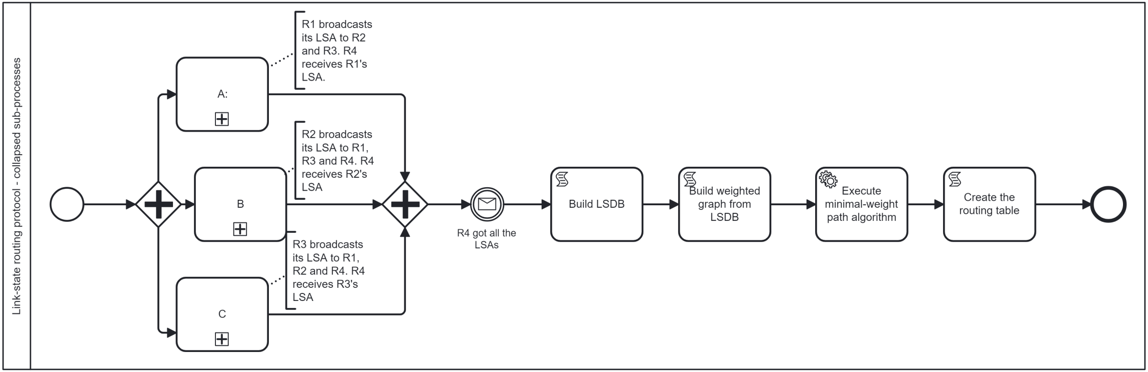

Figure 12: Router R4 creates its routing table.

Implementation of the Protocol Service (Cont'd.)

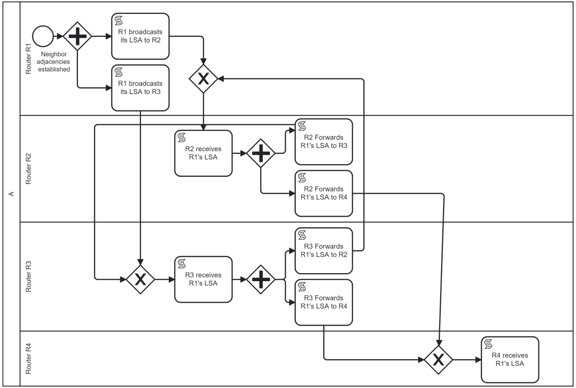

Figure 13: Subprocess A expanded.

Implementation of the Protocol Service (Cont'd.)

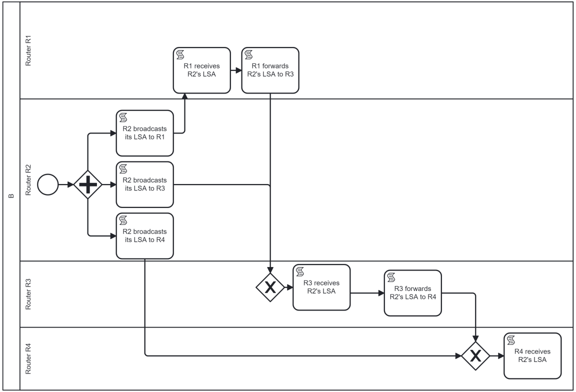

Figure 14: Subprocess B expanded.

Implementation of the Protocol Service (Cont'd.)

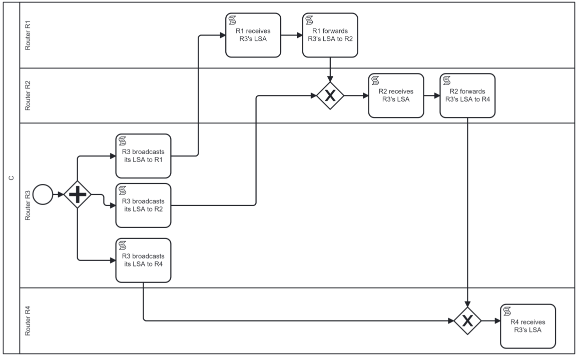

Figure 15: Subprocess C expamded.

Knowledge Representation and Reasoning Systems

- We use **RDF** (Resource Description Framework) and its standards (RDFS, OWL, SPARQL) for Knowledge Representation and Reasoning (**KRR**) (“symbolic AI” systems).
- **Knowledge Representation** is covered by RDF.
- **Reasoning** is done by the standards around RDF.
- **Terminology** related to RDF (Salihoğlu, 2024):
  - semantic web
  - inference
  - ontologies
  - knowledge graphs
  - SPARQL
  - IRIs
  - RDFS
  - OWL

Knowledge Representation and Reasoning Systems (Con'td.)

- RDF is suitable for modeling (complex) **Domains** (such as computer communication protocols and IoT systems/applications).
- **Querying** data and metadata in a seamless way.
- **Automatic logical inference** (*type or superclass of a resource? Which routing protocols are centralized or decentralized?, Which algorithms are DV or LS?*.

gUFO

- We use **gUFO** (gentle UFO: Almeida et al., 2020) as an ontology **implementation of UFO** (Unified Foundational Ontology) in **OWL** (Web Ontology Language).
- UFO is a reference ontology (giving precedence to real-world adequacy).
- As an implementation ontology, gUFO scarifies real-world adequacy to obtain computational properties.
- OWL sees only "**classes**".
- UFO(gUFO) distinguishes **different sorts of classes** (for example the Kinds)

gUFO (cont'd.)

- Using gUFO-based (domain) ontology allows for automatic **error detection** (for endurant types) - **set of rules that say what is wrong in our models**.
- for instance, **non-sortals** cannot subclass **sortals**.
- Sortals cannot specialize more than one kind.
- **Rigid types** cannot subclass (specialize) **anti-rigid types**, etc.

gUFO Conceptualization

Example cf. Figure 6 (static vs dynamic routing)

a) class Static

```
:Static rdf:type owl:Class ;
        rdfs:subClassOf gufo:Object ; # in the taxonomy of Individuals
        rdfs:subClassOf IP_Routing ; # in the taxonomy of types
        rdf:type gufo:Kind ;
        rdfs:subClassOf owl:Restriction ;
        owl:onProperty :characterizes;
        owl:qualifiedCardinality 1;
        owl:onClass :Default_Route.
```

gUFO Conceptualization(Cont'd.1)

b) class Default_Route :Default_Route:rdf:type owl:Class ;
rdfs:subClassOf gufo:Object ; # in the taxonomy of Individuals
rdf:type gufo:Mode ; # in the taxonomy of Types
rdfs:subClassOf owl:Restriction ;
owl:onProperty :characterizes;
owl:qualifiedCardinality 1;
owl:onClass:Static .

gUFO Conceptualization (Cont'd.2)

a-b) relationship charaterizes (between class Static and Class Default_Route)

```
:charaterizes rdf:type owl:ObjectProperty ; # in the taxonomy of Individuals
              rdf:type gufo: Characterization ; # in the taxonomy of Types
              rdfs:domain:Static ;
              rdfs:range:Default_Route .
```

gUFO Conceptualization (Cont'd.3)

c) class Optimal_Route_Determinator

```
:Optimal_Route_Determinator rdf:type owl:Class ;
                            rdfs:subClassOf gufo:Object ; # in the taxonomy of Individuals
                            rdf:type gufo:Relator .
                            rdfs:subClassOf owl:Restriction ;
                            owl:onProperty :mediates ;
                            owl:qualifiedCardinality 1 ;
                            owl:onClass :Routing_Information_Advertiser;
                            owl:qualifiedCardinality 1 ;
                            owl:onClass : Routing_Information_Learner.
```

gUFO Conceptualization (Cont'd.4)

d) class Routing_Information_Advertiser

```
:Routing_Information_Advertiser rdf:type owl:Class ;
                                rdfs:subClassOf gufo:Object ; # taxonomie of Individuals
                                rdf:type gufo:Role .
                                rdfs:subClassOf owl:Restriction ;
                                owl:onProperty :mediates ;
                                owl:qualifiedCardinality 1 ;
                                owl:onClass :Optimal_Route_Determinator;
```

gUFO Conceptualization (Cont'd.5)

e) class Routing_Information_Learner

```
:Routing_Information_Learner rdf:type owl:Class ;
                             rdfs:subClassOf gufo:Object ; # in the taxonomy of Individuals
                             rdf:type gufo:Role .
                             rdfs:subClassOf owl:Restriction ;
                             owl:onProperty :mediates ;
                             owl:qualifiedCardinality 1 ;
                             owl:onClass :Optimal_Route_Determinator;
```

gUFO Conceptualization (Cont'd.6)

c-d) relationship mediates (between class Optimal_Route_Determinator and Routing_Information_Advertiser)

```
:mediates rdf:type owl:ObjectProperty ; # in the taxonomy of Individuals
          rdf:type gufo: Mediation ; # in the taxonomy of Types
          rdfs:domain:Optimal_Route_Determinator ;
          rdfs:range:Routing_Information_Advertiser .
```

d-e) relationship mediates (between class Optimal_Route_Determinator and Routing_Information_Learner)

```
:mediates rdf:type owl:ObjectProperty ;# in the taxonomy of Individuals
          rdf:type gufo: Mediation ; # in the taxonomy of Types
          rdfs:domain:Optimal_Route_Determinator ;
          rdfs:range:Routing_Information_Learner .
```

TBD

- gUFO conceptualization for process aspects (BPMN).

Future Work

- Formalization
  - express the protocol service as a many-sorted algebra.
  - express the protocol as a many-sorted algebra.
  - show that both algebra are isomorphic (proof that the protocol implements the protocol service).
  - or . . .
  - Use RL to show that RR expressing the protocol could be derived from RR expressing the protocol service.

1 S. Assar. Model driven requirements engineering mapping the field and beyond. In Model Driven Requirement Engineering Workshop MoDRE, 2014. https://dx.doi.org/10.1109/MoDRE.2014.6890820.

2 U. Aßmann, S. Zschaler, and G. Wagner. Ontologies, meta-models and the model-driven paradigm. In Ontologies for Software Engineering and Software Technology. Springer Berlin Heidelberg, 2010. https://dx.doi.org/10.1007/3-540-34518-3 9.

3 S. Benkhaled, M. Hemam, M. Djezzar, and M. Maimour. An ontology – based contextual approach for cross-domain applications in Internet of Things. Informatica An International Journal of Computing and Informatics, 2022. https://doi.org/10.31449/inf.v46i5.3627.

4 C. H. Bernabe, V. E. S. Souza, R. de Almeida Falbo, R. S. S. Guizzardi, and C. Silva. GORO 2.0: Evolving an ontology for goal-oriented requirements engineering. In Advances in Conceptual Modeling ER, 2019. https://dx.doi.org/10.1007/978-3-030-34146-6 15.

5 M. Bettaz. Implementing OntoUML Models with OntoObject- Z Specifications: A Proof of Concept Relying on a Partial Ontology for VLANs. In 14th International Conference on Simulation and Modeling Methodologies, Technologies and Applications, SIMULTECH 2024. SciTePRESS - Science and Technology Publications, Lda, 2024. https://dx.doi.org/10.5220/0012854500003758.

6 M. Bettaz and M. Maouche. Towards a New Ontology-based Descriptive Language: OntoObject- Z. In International Conference on Contemporary Computing and Informatics (IC3I). IEEE, 2023. https://dx.doi.org/10.1109/IC3I59117.2023.10397921.

7 B. F. B. Braga, J. P. A. Almeida, G. Guizzardi, and A. B. Benevides. Transforming OntoUML into Alloy: towards conceptual model validation using a lightweight formal method. Innovations in Systems and Software Engineering, 2010. https://dx.doi.org/10.1007/s11334-009-0120-5.

8 CCMI. OpenPonk platform. https://ccmi.fit.cvut.cz/tools/openponk/, 2023. CCMI Research Group, Faculty of Information Technology, Czech Technical University in Prague.

9 B. Costa, P. F. Pires, and F. C. Delicato. Modeling SOA-based IoT Applications with SoaML4IoT. In World Forum on Internet of Things (WF-IoT), 2019. https://dx.doi.org/10.1109/WF-IoT.2019.8767218.

10 I. C. Costa and J. M. P. de Oliveira. GO4SOA: Goaloriented modeling for soa. In International Conference on Web Information Systems and Technologies, 2016. https://dx.doi.org/10.5220/0005800902470254.

11 L. O. B. da Silva Santos, G. Guizzardi, and R. S. S. Guizzardi. GSO: Designing a well-founded service ontology to support dynamic service discovery and composition. In Enterprise Distributed Object Computing (EDOC), 2009. https://dx.doi.org/10.1109/EDOCW.2009.5332016.

12 D. Dermeval, J. Vilela, I. I. Bittencourt, J. Castro, S. Isotani, P. Brito, and A. Silva. Applications of ontologies in requirements engineering: a systematic review of the literature. Requirements Engineering, 2016. https://dx.doi.org/10.1007/s00766-015-0222-6.

13 C. Diamantini, A. Freddi, S. Longhi, D. Potena, and E. Storti. A goal-oriented, ontology-based methodology to support the design of AAL environments. Expert Systems With Applications, 2016. https://dx.doi.org/10.1016/j.eswa.2016.07.032.

14 B. Elvesæter, C. Carrez, P. Mohagheghi, A.-J. Berre, S. G. Johnsen, and A. Solberg. Model-driven service engineering with soaml. In Service Engineering Book. Springer, 2011. https://dx.doi.org/10.1007/978-3-7091-0415-6 2.

15 S. J. T. Fotso, M. Frappier, R. Laleau, A. Mammar, and M. Leuschel. Formalisation of SysML/KAOS Goal assignments with b system component decompositions. In Integrated Formal Methods (IFM), 2018. https://dx.doi.org/10.1007/978-3-7091-0415- 6

16 X. Franch, L. L ´opez, C. Cares, and D. Colomer. The i* framework for goal-oriented modeling. In Domain- Specific Conceptual Modeling: Concepts, Methods and Tools. Springer, 2016. https://dx.doi.org/10.1007/978-3-319-39417-6 22.

17 G. Giancarlo, B. B. Alessander, F. Claudenir, P. Daniele, A. J. Paulo, and P. S. Tiagoa. UFO: Unified foundational ontology. Applied Ontology, 2022. https://dx.doi.org/10.3233/AO-210256.

18 N. Guarino. Formal ontologies and information systems. In Formal Ontology in Information Systems (FOIS). IOS Press, 1998.

19 G. Guizzardi, R. de Almeida Falbo, and R. Guizzardi. Grounding software domain ontologies in the unified foundational ontology (UFO): The case of the ODE software process ontology. In Conferencia Iberoamericana de Software Engineering, 2008.

20 G. Guizzardi and G. Wagner. Using the unified foundational ontology (UFO) as a foundation for general conceptual modeling languages. In Theory and Applications of Ontology: Computer Applications. Springer, 2010. https://dx.doi.org/10.1007/978-90-481-8847-5 8.

22 R. S. S. Guizzardi and G. Guizzardi. Applying the UFO ontology to design an agent-oriented engineering language. In Advances in Databases and Information Systems (ADBIS), 2014. https://doi.org/10.1007/978-3-642-15576-5 16.

23 R. S. S. Guizzardi, G. Guizzardi, A. Perini, and J. Mylopoulos. Towards an ontological account of agent-oriented goals. In Software Engineering for Large-scale Multi-Agent Systems (SELMAS), 2006. https://dx.doi.org/10.1007/978-3-540-73131-3 9.

24 K. Hinkelmann, E. Laurenzi, A. Martin, and B. Th¨onssen. Ontology-based metamodeling. In Business Information Systems and Technology 4.0. Springer, 2018. https://dx.doi.org/10.1007/978-3-319-74322-6 12.

25 L. Kadakolmath and U. D. Ramu. Goal-oriented modeling of an urban subway control system using KAOS. Indonesian Journal of Computer Science (IJCS), 2023. https://doi.org/10.33022/ijcs.v12i3.3239.

26 D. Man. Ontologies in computer science. DIDACTICA MATHEMATICA, 31(1):43–46, 2013.

27 R. Matulevicius, P. Heymans, and A. L. Opdahl. Ontological analysis of KAOS using separation of reference. In Contemporary Issues in Database Design and Information Systems Development. IGI Global, 2007. https://doi.org/10.4018/978-1-59904-289-3.ch002.

28 J. C. Nardi, J. P. A. Almeida, P. H. A. da Silva, and G. Guizzardi. An ontology-based diagnosis of mainstream service modeling languages. In International Enterprise Distributed Object Computing Conference (EDOC), 2019. https://doi.org/10.1109/EDOC.2019.00023.

29 J. C. Nardi, R. de Almeida Falbo, J. P. A. Almeida, G. Guizzardi, L. F. Pires, M. J. van Sinderen, and N. Guarino. Towards a commitment-based reference ontology for services. In Enterprise Distributed Object Computing (EDOC), 2013. https://doi.org/10.1109/EDOC.2013.28.

30 J. C. Nardi, R. de Almeida Falbo, J. P. A. Almeida, G. Guizzardi, L. F. Pires, M. J. van Sinderena, N. Guarino, and C. M. Fonseca. A commitmentbased reference ontology for services. Information Systems, 2015. https://doi.org/10.1016/j.is.2015.01.012.

31 NEMO. Goal oriented requirements ontology (GORO). https://dev.nemo.inf.ufes.br/seon/GORO.html. Research Group.

32 J. C. Nwokeji, T. Clark, and B. S. Barn. Towards a comprehensive meta-model for KAOS. In Model- Driven Requirements Engineering (MoDRE), 2013. https://doi.org/10.1109/MoDRE.2013.6597261.

33 OMG. Service oriented architecture modeling language (soaml) specification, v 1.0.1.

34 M. A. Orellana, J. R. Silva, and E. L. Pellini. A model-based and goal-oriented approach for the conceptual design of smart grid services. Machines, 2021. https://doi.org/10.3390/machines9120370.
35 I. Osman, S. B. Yahia, and G. Diallo. Ontology integration: Approaches and challenging issues. Information Fusion, 2021. https://doi.org/10.1016/j.inffus.2021.01.007.
36 R. Pergl, T. P. Sales, and Z. Rybola. Towards OntoUML for software engineering: From domain ontology to implementation model. In Model and Data Engineering (MEDI), 2013. https://doi.org/10.1007/978-3-642-41366-7 21.
37 Y. Purnomo, R. Doss, N. B. Suhardi, and N. B. Kurniawan. Consolidating service engineering ontologies building service ontology from SOA modeling language (SoaML). International Journal of Computer and Information Engineering, 2018. https://doi.org/10.1109/ICITSI.2018.8695936.
38 G. Reggio. A UML-based proposal for IoT system requirements specification. In International Workshop on Modelling in Software Engineering (MiSE), 2018. https://doi.org/10.1145/3193954.3193956.
39 C. Reginato, J. Salamon, and M. P. Barcellos. Ontology integration approaches: A systematic mapping. In CEUR Worshops. CEUR-WS.org, 2018.

40 Z. Rybola and R. Pergl. Towards ontouml for software engineering: Transformation of rigid sortal types into relational databases. In Federated Conference on Computer Science and Information Systems (FedCSIS), 2016. https://dx.doi.org/10.2298/CSIS170109035R.

41 F. M. Suchanek. OntoUML specification. https://ontouml.readthedocs.io/en/latest/, 2018.

42 M. Tabatabaie, F. A. C. Polack, and R. F. Paige. KAOS-B A goal-oriented process model for EIS. In International Workshop on Modelling, Simulation, Verification and Validation of Enterprise Information Systems (ICEIS), 2010. http://dx.doi.org/10.5220/0003016000400049.

43 S. Tueno, R. Laleau, A. Mammar, and M. Frappier. Towards using ontologies for domain modeling within the SysML/KAOS approach. In International Requirements Engineering Conference Workshops (REW), 2017. http://dx.doi.org/10.1109/REW.2017.22.

44 A. van Lamsweerde. The KAOS meta-model: Ten years after. Technical report, Universite Catholique de Louvain, 1993.

45 V. Werneck, A. de Padua Oliveira, and J. C. S. do Prado Leite. Comparing GORE frameworks: istar and KAOS. In Workshop on Requirement Engineering (WER), 2009.

46 F. Zickert. Evaluation of the goal-oriented requirements engineering method kaos. In Americas Conference on —Information Systems (AMCIS), 2010.

47 Almeida, J.P.A., Guizzardi, G., Falbo, R.A., Sales, P.S. (2020). gUFO: A Lightweight Implementation of the Unified Foundational Ontology (UFO). Technical Report.