A Reasoning-Driven Architecture — a Pragmatic Note on Metareasoning

Lothar Hotz and Stephanie von Riegen

HITeC e.V., University of Hamburg, Germany,
email: {hotz, svriegen}@informatik.uni-hamburg.de

Abstract. In this paper, we present a reasoning-driven architecture and its realization through configurators. By providing metamodels that formally describe a knowledge-representation language, reasoning on domain models is enables, i.e. reasoning on the metalevel. One main realization mean for a reasoning-driven architecture is the mapping between conceptual descriptions used in domain models and instance descriptions on the metalevel. This aspect is focussed in the following sections.

1 Introduction

For knowledge-based tasks, like constructing or diagnosing a specific car periphery system, a strict separation is made into *domain model*, consisting of *concepts*, which covers the knowledge of a certain domain and a *system model*, consisting of *instances*, which covers the knowledge of a concrete system or product of the domain. Concepts represent sets of instances and instances represent individual objects in a real world application. The domain model and the system models are represented with a *knowledge-modeling language* which again is interpreted, because of a defined semantic, through a *knowledge-based system*. Examples are a terminology box (TBox) as a domain model representing e.g. knowledge about an animal domain; and an assertional box (ABox) as a system model representing e.g. a specific animal. Such systems provide reasoning methods for concepts and/or instances. For example, rule-based systems provide inferences on instances, i.e. monotonic inferences, if the knowledge-representation language is based on the Rule Interchange Format RIF BLD [1] or non-monotonic inferences if it is based on the Rule Interchange Format RIF PRD [2]. Description Logic reasoners like PELLET or RACER provide TBox reasoning on concepts (like concept subsumption) and ABox services on instances (like instance checking) (see www.w3.org/Submission/owl11-tractable/ or [3]).

Configurators, like [4] or [5], provide consistency checks on the configurator knowledge base representing the domain model (also called *configuration model*), i.e. on concept descriptions, and model construction facilities on instances for creating systems models (also called *configurations*) [6, 7]. For this task, configurators typically combine a broad range of inference mechanisms like constraint solving, rule-based inference, taxonomical reasoning, and search mechanisms.

Originally, configuration systems are applied to the task of configuring physical systems like ranger drilling machines [8], drive systems [9], railway interlocking stations

[10], elevators [11], or computer systems [12]. Because of the general applicability of knowledge-based systems and especially configurators also domains beyond pure hardware can be handled, like configuring services [13, 14], configuring in e-commerce environments [15], configuration in software product lines [16], constraint-based model parsing [17], or scene interpretation [18].

Following this line of generality, in [19] a first insight is provided, how configuration systems can be applied to itself, i.e. how they can infer *about* concept descriptions and instance descriptions of a domain. The basic application behind is to formally infer about the knowledge itself, i.e. to enable *knowledge reflection*. Thus, to formalize the knowledge engineering process. A main mean for enabling such applications is given by considering concepts as instances which enables instance-related inference techniques to be applied to concepts. A prerequisite is the ability of representing concepts of a domain as instances of a *metamodel*. If the metamodel is represented with a knowledge-representation language inferences on concept descriptions, i.e. on the domain model are enabled. In [19] such an *inference-enabling* metamodel is provided for the configuration language CDL (Component Description Language) [20]. Similarly, the Ontology Definition Metamodel (ODM) [21] for the Web Ontology Language OWL provides a metamodel for a knowledge-representation language. However, ODM is not represented with a knowledge-based representation language but with the Unified Modeling Language [22]. Thus, ODM is not considered here as an inference-enabling metamodel.

In this paper, as a further step in the direction of a reasoning-driven architecture (see Section 2), we provide a realization of such an architecture with configurators and analyze the mapping between concepts and instances in principal (see Section 3).

2 The Reasoning-Driven Architecture

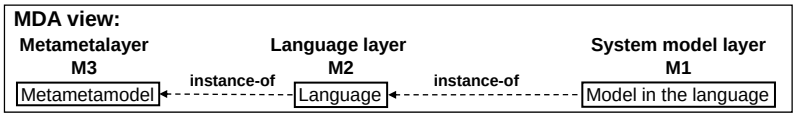

Fig. 1. Model-Driven Architecture View

For defining the Reasoning-Driven Architecture (RDA), we borrow the notion of layers from the Model-Driven Architecture (MDA) [23–25, 22]. In MDA, the main task is to specify modeling facilities that can be used for defining models (*metamodeling*), see for example [22]: "A metamodel is a model that defines the language for expressing a model". MDA provides four layers for modeling (see Figure 1, *MDA view* for three of them): $M2$ is the language layer represented by a metamodel, which is realized by (or *is a instance-of*) a metametamodel located on the $M3$ layer. The language is used for creating a model of a specific system on the $M1$ layer. The system model represents a system which is located in the reality ($M0$ layer not shown in the figure for brevity) [26]. Please note, that each layer contains elements which are instances of classes of the layer above. Typically a specific implementation in a tool ensures that a system

model on $M1$ conforms to a metamodel on $M2$ and a metamodel on $M2$ conforms to a metametamodel on $M3$.

Following [27], we distinguish in Figure 2 between a *linguistic* and an *ontological instance-of* relation. Additionally, we explicitly name the internal *implementation instance-of* relation as such, which is a simple UML type-instance relation in [27]. The linguistic instance-of relation is originated in the notion of classes and objects known from programming languages.

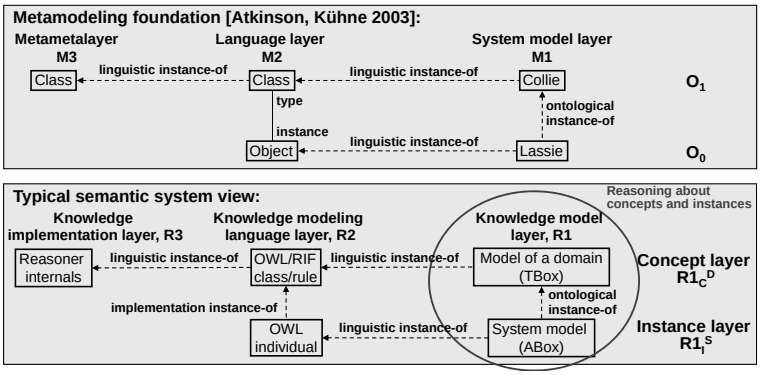

Fig. 2. Atkinsons Metamodeling and semantic system view. The MDA view is refined when applying CDL with its domain and system model on $R1$ ($R1_C^D$, $R1_I^S$).

In the Reasoning-Driven Architecture, the realization of each layer is a knowledge-based system consisting of a knowledge model and separated inference methods used for reasoning about the layer below it. For the RDA approach, we use $R1$, $R2$, $R3$ which roughly correspond to $M1$, $M2$, $M3$ in MDA, respectively. Ri stands for "Reasoning Layer i" (see Figure 2, *semantic system view*). We separate $R1$ in several reasoning layers, because for knowledge-based systems one single model on this layer is not sufficient. This is due to the in the Introduction already mentioned separation of domain model and system model. $R1$ consists of a domain model specified with concepts and constraints (denoted by $R1_C^D$) represented with a knowledge modeling language (e.g. Web Ontology Language (OWL) and a rule description with the Rule Interchange Format (RIF) [2]) and knowledge instances (denoted $R1_I^S$) representing the system model. Furthermore, corresponding reasoning facilities of the knowledge modeling language allow to reason about entities on layer $R1$ (see Figure 2, *semantic system view*).

In Figure 3, the semantic system view is applied to CDL. In this case, the implementation instance-of relation is provided through the instantiation protocol of the underlying implementation language Common Lisp and its Common Lisp Object System (CLOS) [28, 29] (classes are instances of the predefined metaclass `standard-class`). Beside the instance layer, Figure 3 depicts concept definitions (`define-concept`) of CDL and their linguistic instance-of macroexpansion to `defclass` of CLOS. The ontological instance-of relation represents relationships between knowledge elements, in CDL between concepts and instances. Here, the inference techniques of CDL, as there

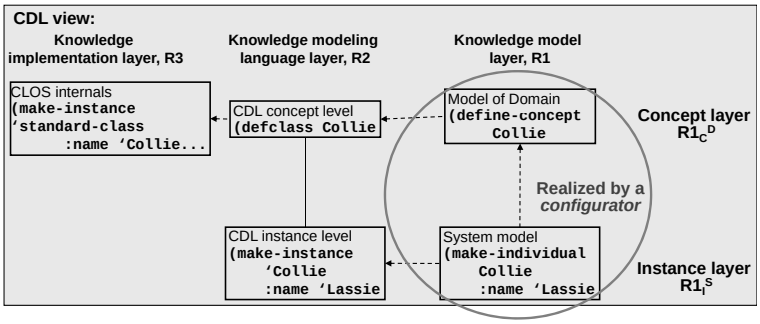

Fig. 3. Component Description Language View

are mainly taxonomical reasoning and constraint processing, are used for infering about concepts and instances on $R1_C^D$ and $R1_I^S$. Thus, the RDA is similar to the metamodeling foundation in [27] except it provides reasoning techniques on domain and system models.

3 Knowledge Reflection Servers

For realizing the RDA, we introduce *Knowledge Reflection Servers* (KRS). Each server is realized with the typical concept/instance scheme. Hence, each server can be realized with a typical knowledge-based system like Description Logic systems or rule-based systems. In our case, such servers consist of multiple copies of the CDL view for representing and reason about distinct types of metaknowledge on different layers. Because each of these servers is realized with same knowledge-based facilities, i.e. CDL concepts and instances, we do not extend CDL with the notion of a metaclass, which has instances that act as classes and can again have instances (e.g. like OWL Full [30] or like MDA/UML implementations with stereotypes [27]). Through a mapping between those servers, concepts on one server are identified with instances of the next higher server. This mapping is a one-to-one (i.e. straight forward) mapping and based on a metaknowledge model [19] (see Figure 6, CDL-Server view).

In RDA, this instantiation facility is used for supporting the step from the configuration language to the domain model. By applying the configuration system to a domain model that contains every model of a language, i.e. by applying it to a *metaknowledge model* (the *Meta-CDL* [19]), the configuration of a domain model for any specific domain is supported. An example of a part of the metaknowledge model is given through the concept `concept-m` (see Figure 6, *CDL-Server view*, $R1_C^M$ and Figure 5 b). This concept represents all concepts of the layer $R1_C^D$, with all their facets, like superconcepts, relations, and parameters. A knowledge modeling language like CDL can be expressed with a knowledge modeling language like CDL because of the general applicability of the language constructs of CDL, which are based on logic [31].

For illustrating reasoning on metalayers and the one-to-one mapping, we take a biological classification as an example for multiple layers.[1] Figure 4 presents an extract of

[1] In [32], a detailed discussion of alternative modeling for biological classification is supplied.

the traditional biological classification of organisms established by Carolus Linnaeus. Each biological rank joins organisms according to shared physical and genetic characteristics. We conceive of a rank as knowledge about the next layer. The main ranks are kingdom, phylum, class, order, genus, species, and breed, which again are divided into categories. Each category unifies organisms with certain characteristics.

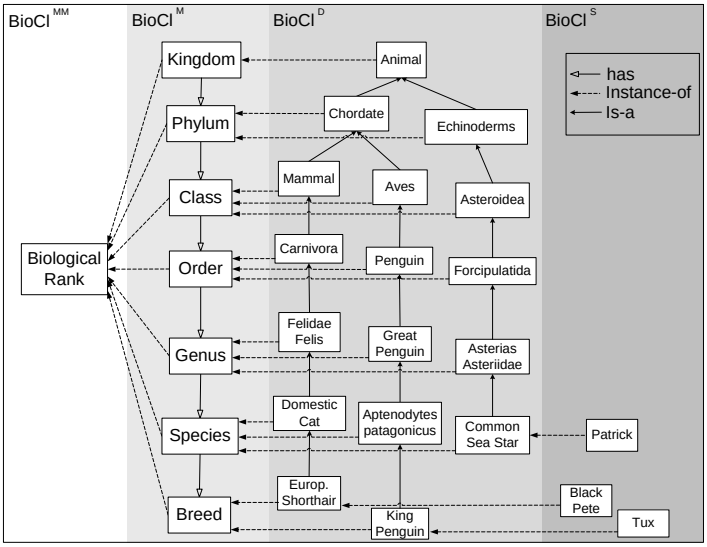

Fig. 4. Biological Classification represented with several layers. Figure inspired by [27].

For instance, the `Mammal` class includes organisms with glands only, thus, a downward specialisation from `Mammal` to its subcategories is depicted in Figure 4. For clarity reasons, only extracts of ranks and categories are given, for example the rank of kingdom contains more than the category animal[2]. The ranks are representing an additional layer ($BioCl^M$) above the domain model of the biological classification. The categories of the ranks form the domain model layer ($BioCl^D$) and each of them is an instance of the correspondent rank. The system model layer ($BioCl^S$) is covering specific individuals, e.g. Tux the penguin. By the given classification, the need for multiple layers becomes directly evident: It is understandable that a `King Penguin` is an instance of `Breed`. But it would be improper to denote `Tux` as a `Breed`, which would hold if `King Penguin` would be a specialization of `Breed`.

In Figure 5, we give an example for the one-to-one mapping of concepts of to instances. As usual, a domain is modelled by some knowledge-representation language. In Figure 5 (a) this is provided in CDL, however, similar concepts can be expressed with OWL. The concept `Mammal` is defined to be a specialization of `Chordate` and has a relation `has-differences` which describes a species with its differentiating characteristics. For a mammal, there should be exactly 6 characteristics where `Hair` and `Fur` are

[2] Among others it contains plants, bacteria, and fungi.

optional and the other characteristics `Glands`, `MiddleEarBones`, and `WarmBlooded` are of the indicated numbers. Additionally, the concept `Class-m` of the metamodel $BioCl^M$ is indicated. Figure 5 (b) gives a part of the metamodel and of *Meta-CDL*. The domain independent part consists of a description of a concept, i.e. `concept-m` defines what a concepts has, namely beside others a superconcept (`has-superconcept-m`) and relations (`has-relations-m`) (see also [19]). By adding domain specific concepts like `Kingdom-m`, `Phylum-m`, and `Class-m` *Meta-CDL* is extended to $BioCl^M$. Instances of concepts of $BioCl^M$ represent concept descriptions of the domain model. Because the *Meta-CDL* exactly reflects the modeling possibilities that are used for the domain model, a straight forward mapping can be realized that maps the concepts of the domain (here `Mammal`) to appropriate instances of the concepts of the metamodel (see Figure 5 c), where `Some-m` and `relation-descriptor-m` are further concepts of the metamodel). By doing so, these instances can be subject of reasoning on the metalayer, e.g. a domain-specific constraint on the metalayer can check combinations of biological classes for having specific characteristics by comparing their differences models. Thus, with this constraint it is specified that a biological class should have a unique combination of characteristics. On $BioCl^D$, these kinds of constraints may be hard to define, because they are typically not related to one specific concept but to several. Still, such constraints are usually part of some modeling guidelines, e.g. for biological classification such documents state that the definitions of biological classes should be unique.

Each layer described above is realized through a Knowledge Reflection Server in Figure 6. Each server monitors the layer below it and consists of the appropriate model and a configuration system which interprets the model. This has the advantage of using declarative models at each metalayer as well as the possibility to apply inference techniques like e.g. constraint programming at the metalayer. For example, a KRS monitors the activities during the construction of the domain model $BioCl^D$, i.e. during the domain representation phase. If e.g. a concept c of the domain is defined with `define-concept` the KRS on $R1^M$ is informed. Furthermore, a KRS

- supplies services like $check\text{-}knowledge\text{-}base$, $add\text{-}conceptual\text{-}constraint$,
- creates appropriate instances of metaconcepts of the *Meta-CDL*, e.g. $concept\text{-}m$ or $conceptual\text{-}constraint\text{-}m$,
- uses constraint propagation for checking the consistency rules,
- applies the typical model configuration process for completing the configuration, e.g. adds mandatory parts,
- checks consistency of created domain specific concepts, e.g. of $BioCl^D$,
- monitors the reasoning process, e.g. for evaluating reasoning performance, and thus, makes reasoning explicit,
- can create and use explanations,
- may solve conflicts that occur during the domain representation phase,
- may apply domain-specific metaknowledge, e.g. "ensuring specific differentiating characteristics of biological classes" with metacontraints.

CDL consists of the modeling facilities concepts and constraints. Defined constraints enable checks of every combination of classes for having specific properties by comparing their models. Thus, with a constraint on $BioCl^M$ (see Figure 4) it might

a)

```
(define-concept :name Mammal
  :specialization-of Chordate
  :has-differences
   ((:type Characteristic :min 6 :max 6)
     :>
    (:type Glands :min 1 :max 1)
    (:type Hair :min 0 :max 1)
    (:type Fur :min 0 :max 1)
    (:type MiddleEarBones :min 3 :max 3)
    (:type WarmBlooded :min 1 :max 1))
  :metaconcept Class-m)
```

one-to-one mapping

b)

```
(define-concept :name concept-m
  :specialization-of named-domain-object-m
  :has-superconcept-m
   (:type concept-m :min 0 :max 1)
  :has-relations-m
   (:type relation-descriptor-m :min 0 :max inf)
  …)

(define-concept :name Kingdom-m
  :specialization-of concept-m)

(define-concept :name Phylum-m
  :specialization-of Kingdom-m)

(define-concept :name Class-m
  :specialization-of Phylum-m)
```

c)

```
(make-individual :name Mammal :instance-of Class-m
  :has-relations-m has-differences
  :has-superconcept-m Chordate)

(make-individual :name has-differences :instance-of relation-descriptor-m
  :has-left-side-m (Some-1)
  :has-right-side-m (Some-2 Some-3 Some-4 Some-5 Some-6))

(make-individual :name Some-1 :instance-of Some-m :some-of Characteristic
   :lower-bound 6 :upper-bound 6)

(make-individual :name Some-1 :instance-of Some-m :some-of Glands
   :lower-bound 1 :upper-bound 1)

(make-individual :name Some-1 :instance-of Some-m :some-of Hair
   :lower-bound 0 :upper-bound 1)
 …
(make-individual :name Chordate :instance-of concept-m)

(make-individual :name Characterisitc :instance-of concept-m)

(make-individual :name Glands :instance-of concept-m)
…
```

Fig. 5. Mapping of a concept description in CDL (a) by using a metamodel (b, also in CDL) to appropriate instances (c).

be specified what characteristics a biological class should have, also classes of different phylums are tested.

4 Discussion

Main properties of the Reasoning-Driven Architecture realized with the Knowledge Reflection Servers (KRS) as described in the previous sections are:

1. the introduction of a model on one layer that represents the knowledge facilities used on the layer below it (i.e. metaknowledge models).
2. the use of existing knowledge-based systems with their reasoning facilities on each layer, especially on metalayers. This enables reflection about knowledge.
3. the mapping of concepts of one layer to instances of the next higher layer. This approach has the potential of using more tractable instance-related inference methods instead of concept reasoning.

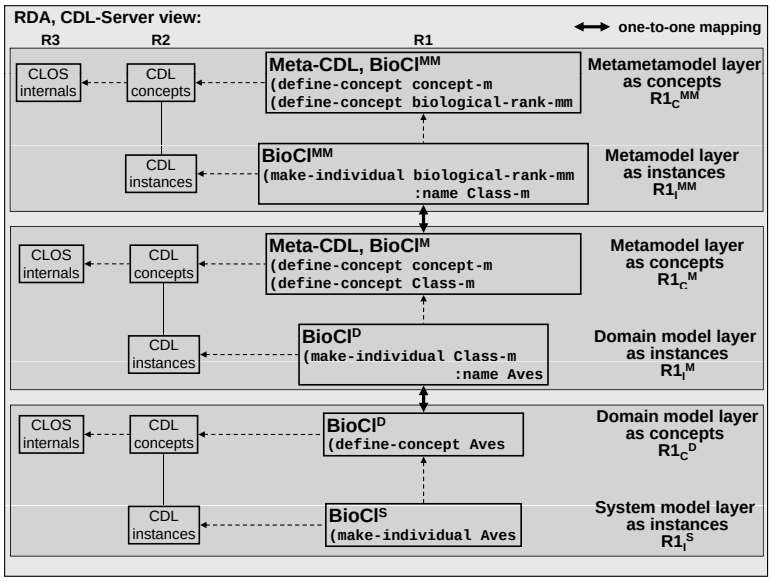

Fig. 6. Reasoning-Driven Architecture, CDL-Server view

4. the support of declarative knowledge modeling on several metalayers. This enables the modeling of knowledge and metaknowledge at the same time. Metaknowledge is typically specified in modeling guidelines. Thus, the described approach enables the modeling of modeling guidelines.

The use of reasoning methods in RDA is achieved by using a knowledge-representation language on a metalayer, i.e. not using e.g. the Meta Object Facility (MOF) of MDA [33], which is based on UML, for a metalayer. Other knowledge-representation languages like the Web Ontology Language (OWL) [30] could also be considered for being used on the layers. However, CDL is quite expressive, e.g. also constraints can be expressed on each layer. For realizing the KRS, even more important for us was the possibility to add server technologies to the knowledge-representation language CDL. However, by replacing the *Meta-CDL* with a metamodel for OWL (e.g. the Ontology Definition Metamodel (ODM) [21]) one could use RDA for scrutinizing the construction of domain models written in OWL.

[34, 35] and [36] present also approaches that include semantics on the metalayer, similar as the *Meta-CDL* metamodel does. By doing so, reasoning methods on each layer as well as the capability to define domain-specific extensions on the metalayer is in principal enabled. By introducing a configurator which allows the definition of procedural knowledge for controlling the used reasoning techniques, in our approach the realization of metastrategies on the metalayers can be considered. However, in [35] a new metamodeling language, called Nivel, is proposed, and by translating it to the knowledge representation language (weight constraint rule language, WCRL) a semantics is given. In our approach, on each layer the same knowledge representation language is

used, i.e. CDL. By specifiying CDL consistency rules in *Meta-CDL*, the semantics of CDL is declaratively modeled.

5 Conclusion

This paper presents a technology for using knowledge-based systems on diverse metalayers. Main ingredients for this task are models about knowledge (metamodels). Through a metamodel a straight forward mapping of concepts of an application domain to instances on the metalayer can be provided. By mapping those concepts to instances, on the metalayer instance-based reasoning methods can be applied to concepts. By combining several layers a Reasoning-Driven Architecture is provided. It enables reasoning facilities on each metalayer, opposed to the Model-Driven Architecture which focusses on transformations. The Reasoning-Driven Architecture is realized through a hierarchy of Knowledge Reflection Servers based on configuration systems. Future work will include metastrategies for conducting reasoning methods on the metalayers, a complete implementation of the servers, and a inference-enabling metamodel for OWL.

References

1. Boley, H., Kifer, M.: A guide to the basic logic dialect for rule interchange on the web. IEEE Transactions on Knowledge and Data Engineering **99**(RapidPosts) (2010)
2. Kifer, M.: Rule interchange format: The framework
3. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.: The Description Logic Handbook. Cambridge University Press (2003)
4. Hollmann, O., Wagner, T., Günter, A.: EngCon: A Flexible Domain-Independent Configuration Engine. In: Proc. ECAI-Workshop Configuration, Berlin, Germany (August 21-22 2000) 94 pp
5. Günter, A., Hotz, L.: KONWERK - A Domain Independent Configuration Tool. Configuration Papers from the AAAI Workshop (July 19 1999) 10–19
6. Buchheit, M., Klein, R., Nutt, W.: Constructive Problem Solving: A Model Construction Approach towards Configuration. Technical Report TM-95-01, Deutsches Forschungszentrum für Künstliche Intelligenz, Saarbrücken (January 1995)
7. Hotz, L., Neumann, B.: SCENIC Interpretation as a Configuration Task. Technical Report B-262-05, Fachbereich Informatik, University of Hamburg (March 2005)
8. Yang, D., Dong, M., Miao, R.: Development of a Product Configuration System with an Ontology-Based Approach. Comput. Aided Des. **40**(8) (2008) 863–878
9. Ranze, K., Scholz, T., Wagner, T., Günter, A., Herzog, O., Hollmann, O., Schlieder, C., Arlt, V.: A Structure-Based Configuration Tool: Drive Solution Designer DSD. 14. Conf. Innovative Applications of AI (2002)
10. Falkner, A., Fleischanderl, G.: Configuration requirements from railway interlocking stations. In: Proc. of the Configuration Workshop on 17th International Joint Conference on Artificial Intelligence (IJCAI-2001), Seattle, Washington (August 6 2001)
11. Marcus, S., Stout, J., McDermott, J.: VT: An Expert Elevator Designer that uses Knowledge-based Backtracking. AI Magazine (1988) 95–112
12. Soloway, E., al.: Assessing the Maintainabiliy of XCON-in-RIME: Coping with the Problem of very large Rule-bases. In: Proc. of AAAI-87, Seattle, Washington, USA (July 13-17 1987) 824–829

13. Tiihonen, J., Heiskala, M., Paloheimo, K.S., Anderson, A.: Configuration of Contract Based Services. In Sinz, C., Haag, A., eds.: Configuration Workshop, 2006. Workshop Proceedings ECAI, Riva del Garda (2006)
14. Felfernig, A., Friedrich, G., Jannach, D., Zanker, M.: Semantic Configuration Web Services in the CAWICOMS Project. In: Proc. of the Configuration Workshop on 15th European Conference on Artificial Intelligence (ECAI-2002), Lyon, France (July 21-26 2002) 82–88
15. Zanker, M., Aschinger, M., Jessenitschnig, M.: Constraint-Based Personalized Bundling of Products and Services. In Tiihonen, J., Felfernig, A., Zanker, M., Männistö, T., eds.: Workshop on Configuration Systems. Workshop Proceedings ECAI, Patras (2008)
16. Hotz, L., Wolter, K., Krebs, T., Deelstra, S., Sinnema, M., Nijhuis, J., MacGregor, J.: Configuration in Industrial Product Families - The ConIPF Methodology. IOS Press, Berlin (2006)
17. Kleiner, M., Albert, P., Bézivin, J.: Parsing SBVR-Based Controlled Languages. In: MoDELS. (2009) 122–136
18. Hotz, L., Neumann, B.: Scene Interpretation as a Configuration Task. Künstliche Intelligenz **3** (2005) 59–65
19. Hotz, L.: Construction of configuration models. In Schumann, M., Kolbe, L.M., Breitner, M.H., Frerichs, A., eds.: Multikonferenz Wirtschaftsinformatik 2010. (2010)
20. Hotz, L.: Frame-based Knowledge Representation for Configuration, Analysis, and Diagnoses of technical Systems (in German). Volume 325 of DISKI. Infix (2009)
21. OMG: Ontology Definition Metamodel, Version 1.0. Object Management Group. (2009)
22. OMG: Unified Modeling Language: Infrastructure, version 2.1.1, formal/07-02-06. Object Management Group. (2007)
23. OMG: Meta Object Facility Core Specification, version 2.0, formal/2006-01-01. Object Management Group. (2006)
24. Kühne, T.: Matters of (Meta-)Modeling. Journal on Software and Systems Modeling **5**(4) (2006) 369–385
25. Hesse, W.: More Matters on (Meta-)Modelling: Remarks on Thomas Kühne's "Matters". Journal on Software and Systems Modeling **5**(4) (2006) 369–385
26. Gaŝević, D., Djuric, D., Devedzic, V., Selic, B.: Model Driven Architecture and Ontology Development. Springer-Verlag New York, Inc., Secaucus, NJ, USA (2006)
27. Atkinson, C., Kühne, T.: Model-Driven Development: A Metamodeling Foundation. IEEE Softw. **20**(5) (2003) 36–41
28. Keene, S.E.: Object-Oriented Programming in Common Lisp: A Programmer's Guide to CLOS. Addison-Wesley (1989)
29. Kiczales, J., Bobrow, D.G., des Rivieres, J.: The Art of the Metaobject Protocol. MIT Press, Cambridge, MA (1991)
30. Antoniou, G., Harmelen, F.V.: Web Ontology Language: OWL. In: Handbook on Ontologies in Information Systems, Springer (2003) 67–92
31. Schröder, C., Möller, R., Lutz, C.: A Partial Logical Reconstruction of PLAKON / KONWERK. In Baader, F., ed.: DFKI-Memo D-96-04, Proceedings of the Workshop on Knowledge Representation and Configuration WRKP'96. (1996)
32. Schulz, S., Stenzhorn, H., Boeker, M.: The ontology of biological taxa. In: ISMB. Volume 24. (2008) 313–321
33. OMG: MDA Guide Version 1.0.1, omg/03-06-01. Object Management Group (2003)
34. Asikainen, T., Männistö, T.: A Metamodelling Approach to Configuration Knowledge Representation. In: Proc. of the Configuration Workshop on 22th European Conference on Artificial Intelligence (IJCAI-2009), Pasadena, California (2009)
35. Asikainen, T., Männistö, T.: Nivel: a metamodelling language with a formal semantics. Software and Systems Modeling (2009)
36. Haase, P., Palma, R., d'Aquin M.: Updated Version of the Networked Ontology Model. Project Deliverable D1.1.5, Neon Project (2009) www.neon-project.org.