A UML Profile for Functional Modeling Applied to the Molecular Function Ontology

Patryk Burek [1]*, Frank Loebe [2] and Heinrich Herre [1]

[1]Institute for Medical Informatics, Statistics and Epidemiology (IMISE), University of Leipzig, Haertelstrasse 16-18, 04107 Leipzig, Germany
[2]Computer Science Institute, University of Leipzig, Augustusplatz 10, 04109 Leipzig, Germany

ABSTRACT

Gene Ontology (GO) is the largest, and steadily growing, resource for cataloging gene products. Naturally, its growth raises issues regarding its structure. Modeling and refactoring big ontologies such as GO is far from being simple. It seems that human-friendly graphical modeling languages, such as the Unified Modeling Language (UML) could be helpful for that task. In the current paper we investigate if UML can be utilized for making the structural organization of the Molecular Function Ontology (MFO), a sub-ontology of GO, more explicit. In addition, we examine if and how using UML can support the refactoring of MFO. We utilize UML and its extension mechanism for the definition of a UML dialect, which is suited for modeling functions and is called Function Modeling Language (FuML). Next, we use FuML for capturing the structure of molecular functions. Finally, we propose and demonstrate some refactoring options for MFO.

1 INTRODUCTION

The Molecular Function Ontology (MFO) is a sub-ontology of the Gene Ontology (GO) – the largest, and steadily growing, resource for cataloging gene products. In 2000 GO contained less than 5,000 terms, in 2003 – 13,000 (Gene Ontology Consortium, 2004), in 2010 it exceeded 30,000 (du Plessis *et al.*, 2011), whereas at the beginning of 2015 its size is above 42,000 terms. The growth of the ontology leads to a suboptimal structure (du Plessis *et al.*, 2011), which motivates refactoring initiatives such as (Guardia *et al.*, 2012; Alterovitz *et al.*, 2010), besides the work of the GO Consortium itself that constantly improves and evolves GO. It turns out that modeling and refactoring big ontologies such as GO is a difficult task, the realization of which can be supported by a human-friendly representation format. The serialization formats used for machine processing of the ontologies, such as the OBO flat file format (Horrocks, 2007) or the Web Ontology Language (OWL) (W3C OWL Working Group, 2009), are not the easiest for a human user. This motivates the adoption of human-friendly graphical notations like those used in software engineering for the task of ontology representation (Kogut *et al.*, 2002; Belghiat and Bourahla, 2012) for certain purposes.

The de facto standard for graphical conceptual modeling of software systems is the Unified Modeling Language (UML) (Rumbaugh *et al.*, 2005), currently developed and maintained by the Object Management Group (OMG) (Object Management Group, 2014). UML has a big potential for various applications that go beyond software engineering, among them for modeling biological knowledge and biological ontologies (Shegogue and Zheng, 2005; Guardia *et al.*, 2012).

UML is well-suited for modeling biological systems, not at least due to the rich infrastructure and the available tools. In particular, the UML built-in extension mechanisms such as stereotypes and profiles permit the easy construction of domain- or task-specific UML dialects, e.g the OBO relations profile (Guardia *et al.*, 2012). Numerous tools for UML modeling are available on the market and can be used out of the box for visualizing biological ontologies as a whole or in part.

In the present paper we investigate if UML can be utilized for making the structure of MFO more explicit and if it can support the refactoring of MFO. We use UML and its extension mechanism for the definition of a UML dialect, called Function Modeling Language (FuML), which is suited for function modeling. Next, we use FuML for modeling the structure of molecular functions. Finally, we propose and demonstrate some refactoring options for MFO.

2 METHODS

2.1 Molecular Function Ontology

Like all GO terms, functions in MFO are specified by id, name, natural language definition and an optional list of synonyms. For instance, the function of catalyzing carbohydrate transmembrane transport is specified by id: GO:0015144; name: *carbohydrate transmembrane transporter activity*; definition: catalysis of the transfer of carbohydrate from one side of the membrane to the other; synonym: sugar transporter. Additionally, for each function its relations with other concepts can be captured. The semantics of the relations that are used for this purpose is provided by serialization languages such as the OBO flat file format or OWL, and/or by the OBO relations ontology (RO) (Smith *et al.*, 2005). In particular, functions in MFO are organized into a hierarchy by means of the is_a link from RO; furthermore, they are linked with processes by the part_of relationship from RO; and in some cases they have relations with concepts of other ontologies such as ChEBI (Degtyarenko *et al.*, 2008). For instance, GO:0015144 is linked, by means of the RO is_a relation, to its parent functions GO:1901476 *carbohydrate transporter activity* and GO:0022891 *substrate-specific transmembrane transporter activity*, by means of the RO part_of relation to the process GO:0034219: *carbohydrate transmembrane transport*, and by means of the RO transports_or_maintains_localization_of to CHEBI:16646: *carbohydrate*.

From the above we see that the semantics of functions in MFO is provided to a large extent by informal natural language expressions and partially by relations with other concepts.

*To whom correspondence should be addressed: patryk.burek@imise.uni-leipzig.de

2.2 Intensional Subsumption

We propose defining the notion of function subsumption, which is a backbone of MFO, upon an intensional interpretation of the is_a relation. Typically, in the field of ontology engineering the extensional aspect of the is_a relation is stressed; in OWL, for instance, A is a subclass of B if every instance of A is an instance of B. The same interpretation is used in RO, where is_a is defined by the reference to the sets of all instances (extensions) of the concepts. According to this understanding the is_a relation is often called extensional subsumption, in contrast to its intensional counterpart(s), where we focus on structural subsumption (Woods, 1991). Instead of referring to instances, this type of subsumption is defined based on the structure of concepts. The latter can be understood as a composition of conceptual parts by means of various composing relations. For illustration within GO itself, GO:0005215: *transporter activity* is justified to intensionally subsume GO:0022857: *transmembrane transporter activity*, because, following (Woods, 1991), both are activities and they are (partially) defined by part-of relations, to GO:0006810: *transport* and to GO:0055085: *transmembrane transport*, resp., and the latter is subsumed by the former. Overall, the main assumption is that concepts are complex structures which can be organized into a subsumption hierarchy. The reading of intensional subsumption is similar to inheritance in object-oriented languages, where one class inherits its structure from another. That enables the structuring of classes into hierarchies.

2.3 UML Profiles and FuML

UML is a graphical modeling language founded on the explicit distinction between the static and the dynamic views of a system; it introduces thirteen diagram types, grouped into two sets: structural modeling diagrams and behavioral modeling diagrams. UML lacks constructs dedicated to function modeling as such, but it provides several build-in mechanisms that allow for an easy extension of the language. Among them are profiles.

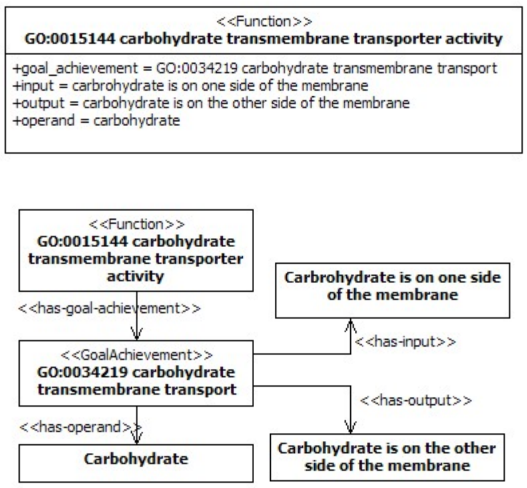
A profile is a light-weight UML mechanism, typically used for extending the language for particular platforms, domains or tasks. It specifies a set of extensions of the UML standard metamodel which include, among others, stereotypes. With stereotypes it is possible to extend the standard UML vocabulary with new model elements. A stereotype can be graphically represented by a dedicated icon, though in the most straightforward form it is represented by a stereotype name, surrounded by guillemets and placed above the name of the stereotyped UML element, cf. «Function» in Figure 1.

We used the profile mechanism for developing a UML extension, called Function Modeling Language (FuML), aimed at supporting the modeling of functions, function ascription, and function decomposition. FuML defines 15 stereotypes for representing functions and function structure, 8 stereotypes for modeling function decomposition, subsumption and function dependencies. The full specification of FuML stereotypes is provided in (Burek and Herre, 2014). In the remaining part of the current paper we analyze how far FuML can be used for modeling and refactoring MFO.

3 ANALYSIS

3.1 Modeling Molecular Functions with FuML

3.1.1 Functions FuML enables graphical modeling of functions in a compact and in an extended form. The compact form is particularly suited for big models containing many functions, whereas the extended form is designed for visualizing the dependencies within the structure of a single function or between several functions.



<<Function>>
GO:0015144 carbohydrate transmembrane transporter activity
+goal_achievement = GO:0034219 carbohydrate transmembrane transport
+input = carbrohydrate is on one side of the membrane
+output = carbohydrate is on the other side of the membrane
+operand = carbohydrate
<<Function>>
GO:0015144 carbohydrate transmembrane transporter activity
<<has-goal-achievement>>
Carbrohydrate is on one side of the membrane
<<has-input>>
<<GoalAchievement>>
GO:0034219 carbohydrate transmembrane transport
<<has-output>>
<<has-operand>>
Carbohydrate
Carbohydrate is on the other side of the membrane

Figure 1. A FuML model of a molecular function, displayed in the compact notation at the top and in the extended form at the bottom.

Figure 1 presents an exemplary FuML model, depicting the structure of the function GO:0015144: *carbohydrate transmembrane transporter activity*. The upper part of the figure presents the compact notation, whereas the extended notation is shown in the lower part. The stereotypes utilized in the figure are discussed in the remainder of the current section.

A function in FuML is interpreted as a role that an entity plays in the context of some goal achievement, such as e.g. a teleological process. This account of functions is similar to (Karp, 2000), where a biological function of a molecule is described as the role that the molecule plays in a biological process. In this sense, the function GO:0015144: *carbohydrate transmembrane transporter activity*, defined in GO as “catalysis of the transfer of carbohydrate from one side of the membrane to the other”, depicts the catalyst role in the teleological process of transferring carbohydrate from one side of the membrane to the other. In terms of the structure we can therefore say that a function specification contains as its part a specification of a goal achievement, understood as a teleological entity which is specified in terms of a transformation from an input situation to an output situation. As presented in Figure 1, a function is depicted by a UML classifier with a stereotype «Function». It connects to its goal achievement by an association with a stereotype «has-goal-achievement» in the extended notation, whereas the compact notation utilizes the attribute goal_achievement.

3.1.2 Goal Achievements In FuML, a goal achievement (GA) x is defined as a category the instances of which are transitions from certain input situations to output situations. Input and output are defined as follows:

- The input category y of the goal achievement x is a situation category such that every instance of x is a transition starting from a situation instantiating y.

- The output y of a goal achievement x is a situation category specifying the situations in which instances of x result by transition. Every instance of x is a transition resulting in a situation instantiating y.

For example, the goal achievement *carbohydrate transmembrane transport* establishes the input category, the instances of which are situations of carbohydrate being on the one side of the membrane, and the output category, the instances of which are situations of carbohydrate being on the other side of the membrane. This means that every instance of *carbohydrate transmembrane transport* exhibits a transition from an instance of the input category to an instance of the output category, i.e. from individual situations of carbohydrate located on one side of the membrane, to individual situations of carbohydrate located on the other side of the membrane.

As shown in Figure 1, an input is indicated in the extended notation by the association with stereotype «has-input», and by the input attribute of a function in the compact notation. The representation of outputs is analogous.

Typically, a transformation from an input to an output situation is a process, and then the GA can be understood as a process category. In the running example, the GA is a teleological process category, namely of carbohydrate transfer from one side of the membrane to the other. This process exhibits the causal transition from the situation of carbohydrate being on one side of the membrane to the situation where carbohydrate is on the other side of the membrane.

3.1.3 Mode of Goal Achievement In some cases the specification of a function is not reduced to a mere input-output pair, but it defines constraints on the method of function realization. For example, the molecular functions GO:0015399: *primary active transmembrane transporter activity* and GO:0015291: *secondary active transmembrane transporter activity* share the same input: solute is on one side of the membrane, and the same output: solute is on the other side of the membrane. Therefore, the pure input-output views of the functions are equal. However, they are distinct due to the way in which they achieve the goal. The former function is realized by means of some primary energy source, for instance, a chemical, electrical or solar source, whereas the latter relies on a uniporter, symporter or antiporter protein. Thus we see that the functions provide the same answer to the question on *what* is to be achieved, however they provide different answers on *how* that is realized. In order to represent this distinction, in FuML we introduce another component of function structure, called *Mode of Goal Achievement*. The mode x of the goal achievement y specifies the way in which y transforms the input to the output situation. For GO:0015399 the mode is: some primary energy source, for instance chemical, electrical or solar source, and for GO:0015291 it is: uniporter, symporter or antiporter protein. The mode is a constraint on the function realization, which does not affect the input or the output. For example, if one adds to the function of transmembrane transport the constraint that the transport should be realized by the uniporter protein then the input and the output remain unchanged. However, the function as such changes in that not every transportation process realizes it, but only those that are driven by a uniporter protein.

3.1.4 Participants Often goal achievements are expressed by action sentences of natural language and thus the results of linguistic analysis of action sentences can be applied to the analysis of the structure of goal achievements. In linguistics, the role that a noun phrase plays with respect to the action or state described by the verb of a sentence is called a thematic role (Harley, 2010). The specifications of molecular functions in MFO often contain two thematic roles – a patient (called an operand in FuML) and an actor (called a doer in FuML). An operand indicates the entity undergoing the effect of the action. We say that an operand y of the goal achievement x specifies a category y such that instances of x operate on instances of y. GO:0015144 operates on (transports) carbohydrate.

A doer is not as common in MFO as an operand. For example, in the discussed carbohydrate transmembrane transport function no doer is indicated. Typically, a doer is a part of the GA in cases where the mode of realization is provided. For instance, the functions GO:0015292 *uniporter activity* and GO:0015293 *symporter activity* both specify the mode of realization and each indicates its doer, namely the respective protein.

4 PATTERNS OF FUNCTION SUBSUMPTION

Behind functional subsumption actually various distinct relations are implicitly hidden (Burek *et al.*, 2009). FuML introduces several distinct patterns for function subsumption (Burek and Herre, 2014). In the following section we discuss the application of three of those patterns for the modeling of MFO.

In FuML the notion of function subsumption is founded on the subsumption of goal achievements. We say that the function x is subsumed by the function y if the goal achievement of x is subsumed by the goal achievement of y. Since goal achievements are quite complex entities, it is not trivial to answer the question of what it means that one goal achievement subsumes another. Here, however, the analysis of GA structure is helpful, which pertains to the intensional aspects of the corresponding GA category, as discussed in previous sections. Based on this approach one can detect various patterns of function subsumption.

4.1 Operand Specialization

Since function specifications often contain operands, it is very common to construct a hierarchy of functions on the basis of the taxonomic hierarchy of their operands. In fact, this pattern is applied frequently in MFO. Consider, for instance, the functions GO:0015075: *ion transmembrane transporter activity* and GO:0008324: *cation transmembrane transporter activity*, linked by the is_a relation in GO. The relation between those two functions is based on the relation of their operands, as cation is subsumed by ion. In FuML function subsumption by operand specialization is depicted with a dependency link with stereotype «operand-spec». The supplier of the link is the subsumed function and the client is the subsumer.

4.2 Mode Addition

Another pattern of function subsumption, frequently met in MFO, is based on modes of goal achievement. Consider two functions, GO:0022857: *transmembrane transporter activity* and GO:0022804: *active transmembrane transporter activity*. Both share the same operand, namely substance, as well as the same input-output pair – operand is on one side of the membrane and operand is on the other side of the membrane. In this sense those functions are equal. However, they differ in that the former does not define any mode of realization, whereas the latter has the following mode defined: the transporter binding the solute undergoes a series of conformational changes. Therefore, one can say that

GO:0022804 specializes GO:0022857 by addition of a mode. We say that function x is subsumed by the function y by mode addition if x is subsumed by y and x has some mode, whereas y has no mode assigned. Function subsumption by mode addition is depicted in FuML by means of a dependency link with stereotype «mode-added». The subsumed function is the supplier of the link and the subsuming function is a client.

4.3 Mode Specialization

Subsumption of functions can be based on the mode of realization also in cases where a parent function has already a mode assigned. Consider, for instance, the function GO:0022804: *active transmembrane transporter activity* having the mode: transporter binds the solute and undergoes a series of conformational changes and the function GO:0015291: *secondary active transmembrane transporter activity* with the mode: transporter binds the solute and undergoes a series of conformational changes driven by chemiosmotic energy sources, including uniport, symport or antiport. The latter clearly characterizes particular modes of active transmembrane transport. Consequently, it seems intuitive to say that GO:0015291 specializes GO:0022804 (as is the case in GO). We call this type of function subsumption the subsumption by mode specialization and define it as follows: The function x is subsumed by the function y by mode specialization if x is subsumed by y and mode r of x specializes mode s of y. In FuML function subsumption by mode specialization is depicted with a dependency link with stereotype «mode-spec». The subsumed function is the supplier of the link and the specialized function is a client.

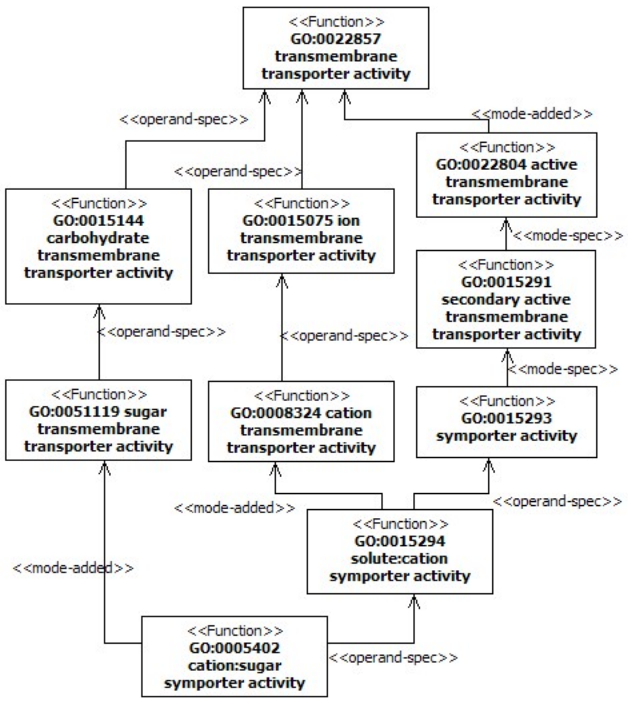
5 APPLICATION

The application of FuML to GO pursues two objectives. The first objective is the usage of FuML for establishing a semantic basis for molecular functions that supports the representation of functions in an organized way beyond the textual description. Moreover, the discussed patterns represent basic knowledge on the inter-relations between biological processes and molecular functions. The part_of relation between biological processes and molecular functions can be mapped to the has-goal-achievement association between functions and goal achievements.

The second and the main objective of applying FuML to MFO is to explicitly document design choices and the subsumption patterns utilized implicitly in MFO. Figure 2 presents such a documentation for a fragment of MFO in terms of FuML. The patterns are indicated by stereotypes of FuML, which enables an easy-to-grasp visualization of the structure of MFO as well as the underlying design choices. One benefit of this approach is that the explicit specification of the design choices makes the ontology much more intelligible for a human user.

Furthermore, the application of FuML reveals potential of refactoring and revision of GO. For instance, the application of FuML in modeling the functions GO:0022857: *transmembrane transporter activity* and GO:0022891: *substrate-specific transmembrane transporter activity* reveals that both share similar goal achievements: transfer of an operand from one side of a membrane to the other, with input: operand is on one side of the membrane, and output: operand is on the other side of the membrane. Consequently and following FuML, a potential difference between GO:0022857 and GO:0022891 can be searched in their operands. For GO:0022857 that is a substance, whereas for GO:0022891 it is a specific substance or group of substances. Therefore, the first refactoring option would be to explicitly document the pattern of subsumption between GO:0022857 and GO:0022891 as operand specialization. The alternative refactoring option is driven by the further analysis of operands of those functions, in particular by clarifying what the difference between "a substance" and "a specific substance or group of substances" is. The answer could be found in GO:0022892 *substrate-specific transporter activity*, a parent function of GO:0022891. An operand of GO:0022892 is exemplified by macromolecules, small molecules or ions. In that case, however, it seems that functions like GO:0090482: *vitamin transmembrane transporter activity* and GO:0015238: *drug transmembrane transporter activity* should also be considered as substance specific transmembrane transport and specialize GO:0022891 by operand specialization, which is currently not the case, however.

Finally, the third possible refactoring option could be based on the assumption that the distinction between those two operands is only superficial and GO:0022891 is merely used for the organization of the function taxonomy, i.e., for grouping all functions that are distinguished by operands such as ion, alcohol, and water. According to this view, GO:0022891 would in fact be a duplication of GO:0022857, introduced into MFO only for the purpose of structuring it, but not as a specification of particular biological functions. As illustrated in Figure 2, FuML enables the replacement of GO:0022891 with an explicit specification of the design choices by stereotyped links.



<<Function>> GO:0022857 transmembrane transporter activity
<<operand-spec>>
<<operand-spec>>
<<mode-added>>
<<Function>> GO:0022804 active transmembrane transporter activity
<<Function>> GO:0015144 carbohydrate transmembrane transporter activity
<<Function>> GO:0015075 ion transmembrane transporter activity
<<mode-spec>>
<<Function>> GO:0015291 secondary active transmembrane transporter activity
<<operand-spec>>
<<operand-spec>>
<<mode-spec>>
<<Function>> GO:0051119 sugar transmembrane transporter activity
<<Function>> GO:0008324 cation transmembrane transporter activity
<<Function>> GO:0015293 symporter activity
<<mode-added>>
<<operand-spec>>
<<Function>> GO:0015294 solute:cation symporter activity
<<mode-added>>
<<Function>> GO:0005402 cation:sugar symporter activity
<<operand-spec>>

Figure 2. An MFO segment modeled with FuML.

The decision on the refactoring option, as in any modeling enterprise, is the responsibility of the modeler(s), GO developers in this case. Yet, the above analysis demonstrates how graphical languages, such as FuML, similarly as in software and systems engineering, can drive and support that task for biological ontologies such as MFO.

6 RELATED WORK

The ideas underlying the structure of functions, introduced in FuML, are the result of an analysis of the current state of the art of function modeling in software, systems and ontological engineering. For instance, the interpretation of a function in terms of a role is common not only in biological systems (Karp, 2000), but also in functional modeling in mechanical engineering (Kitamura *et al.*, 2006; Lind, 1994; Chandrasekaran and Josephson, 2000).

The notion of goal achievement grasps the teleological character of a function, its orientation on some goal. This aspect is stressed in many approaches to function representation, e.g. (Sasajima *et al.*, 1995; Iwasaki *et al.*, 1995; Gero, 1990). In particular, defining a function in terms of input-output pairs is present in modeling technical artifacts (Borgo *et al.*, 2011; Goel *et al.*, 2009).

The mode of realization, also called the way-of-function-achievement, specifying the constraints on the method of function realization is present in (Kitamura *et al.*, 2002), among others.

To the best of our knowledge, the presented patterns of function decomposition are not collected and integrated into any other single modeling framework, though the techniques themselves are commonly used, especially in software and systems engineering, e.g. see the function-means-context link in (Bracewell and Wallace, 2001) or the decomposition with zig-zaging in (Nam, 2001).

7 CONCLUSION

In the current paper we present and discuss applications of UML and patterns for function subsumption to the modeling and refactoring of biological ontologies. In particular, we developed a UML profile for functional modeling, called the Function Modeling Language (FuML) (Burek and Herre, 2014), and apply it to the modeling and refactoring of a segment of the Molecular Function Ontology.

The application of FuML enables the systematic, graphical representation of information that is currently available in MFO mainly in the form of textual descriptions. We demonstrate that behind the extensional is_a relation, which is used for the construction of MFO, several different patterns of intensional subsumption can be determined. Modeling MFO via FuML helps in identifying these patterns and, moreover, provides the means for representing them directly in the hierarchy of molecular functions. We argue that this can help making the ontology structure more comprehensible for human users and supports communication. The claim is illustrated by an analysis and a model of an MFO fragment with FuML, from which we derive several refactoring options.

Besides proposing FuML and the particular refactoring options in this paper, for future work we consider first the continued analysis of MFO. Extending this to a larger scale may require establishing software support, e.g., for identifying subsumption pattern instances within MFO (semi-)automatically. Moreover, FuML and its methods may also be transferred to or yield new methods for common languages of biomedical ontologies, nowadays including OWL.

REFERENCES

Alterovitz, G., Xiang, M., Hill, D. P., Lomax, J., Liu, J., Cherkassky, M., Dreyfuss, J., Mungall, C., Harris, M. A., Dolan, M. E., *et al.* (2010). Ontology engineering. *Nature biotechnology*, **28**(2), 128–130.

Belghiat, A. and Bourahla, M. (2012). Automatic generation of OWL ontologies from UML class diagrams based on meta-modelling and graph grammars. *World Academy of Science, Engineering and Technology*, **6**(8), 380–385.

Borgo, S., Carrara, M., Garbacz, P., and Vermaas, P. E. (2011). A formalization of functions as operations on flows. *Journal of Computing and Information Science in Engineering*, **11**(3), 031007.

Bracewell, R. H. and Wallace, K. M. (2001). Designing a representation to support function-means based synthesis of mechanical design solutions. In S. Culley, A. Duffy, C. McMahon, and K. Wallace, editors, *Proceedings of ICED01, Glasgow, Scotland, UK, Aug 21-23*, pages 275–282.

Burek, P. and Herre, H. (2014). FuML Specification v1.0. Onto-med report, University of Leipzig.

Burek, P., Herre, H., and Loebe, F. (2009). Ontological analysis of functional decomposition. In H. Fujita and V. Mařík, editors, *Proceedings of the 8th International Conference on Software Methodologies, Tools and Techniques, SoMeT 2009, Prague, Czech Republic, Sep 23-25*, pages 428–439, Amsterdam. IOS Press.

Chandrasekaran, B. and Josephson, J. R. (2000). Function in device representation. *Engineering with computers*, **16**(3-4), 162–177.

Degtyarenko, K., De Matos, P., Ennis, M., Hastings, J., Zbinden, M., McNaught, A., Alcántara, R., Darsow, M., Guedj, M., and Ashburner, M. (2008). ChEBI: a database and ontology for chemical entities of biological interest. *Nucleic acids research*, **36**(Suppl 1), D344–D350.

du Plessis, L., Škunca, N., and Dessimoz, C. (2011). The what, where, how and why of gene ontology — a primer for bioinformaticians. *Briefings in Bioinformatics*, **12**(6), 723–735.

Gene Ontology Consortium (2004). The Gene Ontology (GO) database and informatics resource. *Nucleic acids research*, **32**(Suppl 1), D258–D261.

Gero, J. S. (1990). Design prototypes: a knowledge representation schema for design. *AI Magazine*, **11**(4), 26–36.

Goel, A. K., Rugaber, S., and Vattam, S. (2009). Structure, behavior, and function of complex systems: The structure, behavior, and function modeling language. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, **23**(01), 23–35.

Guardia, G. D., Vêncio, R. Z., and de Farias, C. R. (2012). A UML profile for the OBO relation ontology. *BMC Genomics*, **13**(Suppl 5), S3.

Harley, H. (2010). Thematic roles. *The Cambridge Encyclopedia of the Language Sciences*, pages 861–862.

Horrocks, I. (2007). OBO flat file format syntax and semantics and mapping to OWL Web Ontology Language. Technical report, University of Manchester.

Iwasaki, Y., Vescovi, M., Fikes, R., and Chandrasekaran, B. (1995). Causal functional representation language with behavior-based semantics. *Applied Artificial Intelligence: An International Journal*, **9**(1), 5–31.

Karp, P. D. (2000). An ontology for biological function based on molecular interactions. *Bioinformatics*, **16**(3), 269–285.

Kitamura, Y., Sano, T., Namba, K., and Mizoguchi, R. (2002). A functional concept ontology and its application to automatic identification of functional structures. *Advanced Engineering Informatics*, **16**(2), 145–163.

Kitamura, Y., Koji, Y., and Mizoguchi, R. (2006). An ontological model of device function: industrial deployment and lessons learned. *Applied Ontology*, **1**(3), 237–262.

Kogut, P., Cranefield, S., Hart, L., Dutra, M., Baclawski, K., Kokar, M., and Smith, J. (2002). UML for ontology development. *The Knowledge Engineering Review*, **17**(1), 61–64.

Lind, M. (1994). Modeling goals and functions of complex industrial plants. *Applied Artificial Intelligence: An International Journal*, **8**(2), 259–283.

Nam, P. S. (2001). *Axiomatic design: Advances and applications*. Oxford University Press, New York.

Object Management Group (2014). `http://www.omg.org/`.

Rumbaugh, J., Jacobson, I., and Booch, G. (2005). *The Unified Modeling Language Reference Manual*. Addison Wesley, Reading, Massachusetts, 2. edition.

Sasajima, M., Kitamura, Y., Ikeda, M., and Mizoguchi, R. (1995). FBRL: A function and behavior representation language. In *Proc. of IJCAI 1995*, pages 1830–1836.

Shegogue, D. and Zheng, W. J. (2005). Integration of the Gene Ontology into an object-oriented architecture. *BMC Bioinformatics*, **6**(1), 113.

Smith, B., Ceusters, W., Klagges, B., Köhler, J., Kumar, A., Lomax, J., Mungall, C., Neuhaus, F., Rector, A. L., and Rosse, C. (2005). Relations in biomedical ontologies. *Genome biology*, **6**(5), R46.

W3C OWL Working Group (2009). OWL 2 Web Ontology Language Document Overview. Technical report, World Wide Web Consortium.

Woods, W. A. (1991). Understanding subsumption and taxonomy: A framework for progress. In *Principles of Semantic Networks*, pages 45–94. Morgan Kaufmann.