

# Towards Seamless Digitization in OPC UA

Václav Jirkovský<sup>1</sup><sup>a</sup>, Petr Kadera<sup>1</sup><sup>b</sup>, Marek Obitko<sup>2</sup><sup>c</sup> and Ondřej Flek<sup>2</sup>

<sup>1</sup>Czech Institute of Informatics, Robotics, and Cybernetics, Czech Technical University in Prague, Prague, Czech Republic

<sup>2</sup>Rockwell Automation R&D Center, Prague, Czech Republic

{vaclav.jirkovsky, petr.kadera}@cvut.cz, {mobitko, oflek}@ra.rockwell.com

**Keywords:** Motion Control, Drive, OPC UA, Ontology, SQWRL, Validation.

**Abstract:** The advent of Industry 4.0 has brought about the need for seamless communication and integration among diverse industrial automation systems. While current standards such as OPC UA address syntactic interoperability, they fall short in addressing semantic heterogeneity, which poses a challenge to the true understanding of exchanged data. The existing approaches, which rely on textual information models such as the OPC UA companion specifications, suffer from possible ambiguity and difficulties in development and validation. This research aims to overcome these challenges by exploring the development of an explicit, user-friendly, and machine-readable model specification for the conditional implementation of Rockwell Automation drives' motion axis attributes within OPC UA. The paper delves into suitable formalisms for model specification, taking into account user interaction and expressivity, and proposes mechanisms for model validation and future extensions. These efforts pave the way for enhanced semantic interoperability in industrial automation, thus contributing to the advancement of Industry 4.0.

## 1 INTRODUCTION

The domain of industrial automation has experienced significant changes and paradigm shifts over time. Although some changes were introduced quietly, others represent revolutions in the field, such as the invention and implementation of the initial programmable logic controller (PLC) in the 1970s (David, 1995). On the other hand, Industry 4.0 (Dalenogare et al., 2018) is not an ex post named industrial revolution, but an initiative planned in advance. The fourth industrial revolution (Industry 4.0) represents the integration of vertical and horizontal manufacturing processes and product connectivity, which can help companies achieve higher industrial performance.

The integration of all manufacturing processes and resources is made possible by increasing digitisation at all levels. However, devices and systems are produced by different companies with varying data models. Therefore, standardisation is necessary to reduce differences in syntax and meaning, ensuring easy and seamless integration and communication (Neumann, 2007).


Fortunately, there are several verified standards


(Profinet, CIP, etc. (Lin et al., 2013)) within the industrial automation domain for robust communication. One of the mature standards is the OPC UA (Leitner and Mahnke, 2006), which also provides advanced functions such as subscriptions, function calls, and an object-orientated approach for information modelling. However, these standards are capable of dealing only with syntactic heterogeneity but not with semantic heterogeneity. That is, while such standards provide means of ensuring messages can be exchanged between devices, there are no means of ensuring that the message content is understood with the highest confidence.


Awareness exists of the lack of appropriate tools and approaches to deal with semantic heterogeneity. Thus, various strategies have been employed depending on the communication standard and OEM. However, most are based on specifications of information models in natural language in textual form. An example of the aforementioned approach is the OPC UA Companion Specifications<sup>1</sup>. The companion specifications are developed for two main reasons:

- To publish domain-specific information models.
- Standardise the use of OPC UA in specific environments.

<sup>1</sup><https://opcfoundation.org/about/opc-technologies/opc-ua/ua-companion-specifications/>

<sup>a</sup> <https://orcid.org/0000-0002-1744-0753>

<sup>b</sup> <https://orcid.org/0000-0002-1747-6473>

<sup>c</sup> <https://orcid.org/0000-0001-7639-3312>

The companion specifications for the OPC UA are often called “Industry standard models” and should enable interoperability at the semantic level.

Unfortunately, the aforementioned approaches together with the “Industry standard model” have an important imperfection. They are written in textual form in natural language. It causes three main obstacles to easy integration and interoperability of systems:

- Semiotic heterogeneity (different interpretations by different users).
- Difficult development in comparison to an explicit specification in machine-readable form.
- Validation of the models developed.

The research presented in this paper is motivated by demands for a suitable explicit and user-friendly specification of conditional implementation related to Rockwell Automation drives’ motion axis attributes. Conditional implementation defines conditions (e.g. usage of specific devices or values of certain motion axis attributes) in which the motion axis attribute is valid and should be presented within the OPC UA information model. Research questions which will be answered in this paper are:

- What is a suitable formalism for model specification? Concerning user-friendly interaction and expressivity.
- What mechanism should be used for model validation?
- Reusability and extensions of the model.

This paper is organised as follows: First, related work with respect to modelling formalism and their expressivity, and validation capabilities, is presented. Next, conditional implementation of motion axis attributes is introduced, followed by presentation of proposed solution of Daplex-based specification together with SWRL-based validation. Then, the proposed solution is demonstrated on the examples of conversions of MAA conditions into SWRL rules. Finally, the conclusions and future work are presented.

## 2 RELATED WORK

Hardware quality and quality engineering are a very complex issue, which is studied in a separate branch of study with detailed elaborated theory. However, the simple definition of the term Quality from (Bruel et al., 2021) may be reused: “Quality means that the hardware does the right things and does them correctly.” The “things” are known as hardware requirements. While (Bruel et al., 2021) deals with hardware

systems, the same approach can be applied, for example, to software systems.

The overall hardware quality is highly dependent on the quality of these requirements. They define the system to satisfy the user needs. Furthermore, the requirements allow for the validation of the candidate implementation against the definition. On the other hand, quality is not dependent only on requirements, but also on formal specifications of the system. The difference or at least nuance may be perceived in the meaning of these terms:

- requirements — properties of system parts relevant to users and environment,
- specification — technical properties of system parts.

This work is focused on the formal specification of the system and, more precisely, on the formal specification of the information models. However, the distinction between requirements and specification is not important, and formalisms suitable for specification may also be exploited for formal specification of requirements. Next, the focus of the presented work is on explicit knowledge, that is, the knowledge specified outside a (control) code of an application, in contrast to implicit knowledge hardcoded within a code.

The following paragraphs present the selected and possible approaches for dealing with the formal specification of information models with respect to user-friendliness and possible validation support. The OPC UA companion specifications are omitted from these approaches. The important effort of the OPC Foundation was to specify the OPC UA information models to ensure seamless interoperability. Unfortunately, the set of OPC UA specification documents is written in natural language rather than as a formal and explicit specification.

**Formal Specification Languages (FSL).** (Pang et al., 2016) are used to specify the design requirements of instrumentation and control systems. Their main purpose is to provide the possibility of formalising design requirements written in natural language. FSLs are typically based on templates and patterns to assist in requirements formalisation. Thus, the template may be perceived as a preformatted textual representation of semi-formal requirements. The template consists of: *fixed keywords*, and *attribute placeholders*. And when a template is instantiated, then its placeholders are substituted by concrete values. The following listing illustrates an FSL template example:

```
<system> shall <action>
```

where <system> and <action> are placeholders. Every pattern/template has rigorous semantics, fixed keywords, and attribute placeholders. However, there

could be obstacles during the application of templates across various domains, or inconsistencies may arise during the addition of new templates.

An approach that provides the opposite solution to the “hardcoded” solution for knowledge and information capture uses **ontologies**. Ontologies are used for the description and serialisation of a given conceptualisation (a mental image of some domain). Specifically, the term ontology may be defined using the well-known definition provided by Thomas Gruber (Gruber, 1995) — an ontology is “*explicit specification of a shared conceptualisation.*”

There are various formats/languages for serialisations of ontologies, and the format selection has direct impact on utilisable expressivity, as well as possible additional tools such as editors, reasoners, data storage, etc. Therefore, developers must consider the intended purpose of an ontology. Today, Semantic Web technologies (RDF (Lassila et al., 1998), RDFS (Brickley et al., 2014), and OWL (McGuinness et al., 2004) formats, in particular) are widely used for ontology development, mainly because of scaleable expressivity and also availability of additional (and supporting) tools.

Web Ontology Language (OWL) is a Semantic Web standard designed to represent rich and complex knowledge about resources. OWL is a computational logic-based language such that the knowledge expressed in OWL can be reasoned by computer programmes either to verify the consistency of that knowledge or to make implicit knowledge explicit. OWL is an important part of the W3C’s Semantic Web technology stack, which includes on lower levels RDF, RDFS, SWRL, SPARQL, etc. The OWL together with a reasoner (e.g., Pellet) is able to conduct important reasoning tasks such as consistency checking, individuals classification, etc. Thus, a subsequent validation task may also be formulated as a task for OWL ontology and a reasoner. However, the Open World Assumption as the key feature of OWL ontologies (Horridge and Bechhofer, 2011) may cause obstacles during constraints validation with the help of automated reasoning.

On the other hand, the Semantic Web Rule Language (Horrocks et al., 2004) should help overcome the aforementioned obstacle. SWRL combines OWL DL with function-free Horn logic, and therefore it allows Horn-like rules to be combined with OWL DL ontologies. Rules are of the form of an implication between a body and a head. The meaning of a rule is: whenever the conditions specified in the body hold, then the conditions specified in the head must also hold. The following listing illustrates a simple example of the rule:

$$\text{Person}(?p) \wedge \text{hasSibling}(?p,?s) \wedge \text{Male}(?s) \\ \rightarrow \text{hasBrother}(?p,?s)$$

with the meaning: all persons which have a sibling and this sibling is a man, then they must have the property hasBrother. The application of SWRL is, for example, in (Fortineau et al., 2014), where SWRL is used to express formal rules within ontology-based models with an application to the nuclear industry.

**Daplex** (query language based on functional data model) (Shipman, 1981) is a data definition and manipulation language for database systems, based on the concept of data representation called the functional data model. The focus of Daplex is on providing a “conceptually natural” database interface language. Thus, the language provides user-friendly syntax. Constraints have two parts: the quantification and the main part. The variables are quantified and specified on a given domain in the quantification part. The main part of the constraints contains predicates that should be satisfied. Furthermore, in addition to data definition exploitation of Daplex, (Martins et al., 2008) shows that Daplex provides an intuitive way of viewing and querying data in the Semantic Web.

The suitability of the Daplex language for expressing constraints was demonstrated in CoLan (Bassiliades and Gray, 1995), a high-level declarative Constraint Description Language, for an application to an Object-Orientated Database. Except for the utilisation of Daplex for constraint expressions, CoLan exploits Prolog (Clocksin and Mellish, 2003), which implements the operational semantics of the constraint. The CoLan system was dedicated to ADAM (Ellis and Demurjian, 1991) object-orientated database and there is no other application to another system.

The fulfillment of the requirements (user-friendliness, expressivity, validation capabilities, and re-usability) of the possible formalism is illustrated in Tab 1.

FSL and Daplex represent languages with user-friendly syntax which enable easy understanding about statement meanings even for non-skilled users. Both provide suitable expressivity, but FSL has no standard way for expressing many constructs such as cardinality expressions. And thus, re-usability of FSL statements may result in semiotic ambiguity. Furthermore, FSL and Daplex do not have validation capabilities. OWL and SWRL (as OWL extension) are strong in their expressivity, standardisation, and re-usability. The validation based only on the reasoning may be difficult for OWL due to Open World assumption. OWL and SWRL are not user-friendly for many applications compared to FSL and Daplex. CoLan provides a user-friendly solution due to the use of Daplex. The expressivity and validation capabilities

Table 1: Overview of existing solutions presented in Sec.2.

	User-friendly	Expressivity	Validation	Re-usability
FSL	X	x	—	x
Daplex	X	X	—	X
OWL	—	X	x	X
SWRL	—	X	X	X
CoLan	X	X	X	x

ties (as the result of the combination of Daplex and Prolog) are sufficient for many possible industrial applications and scenarios. However, the reusability of this approach is limited due to the dedicated application to ADAM database and the implicitly captured process of validation and utilisation of Prolog.

### 3 PROPOSED APPROACH

The approaches presented in the previous section cannot satisfy all demanded criteria, that is, a user-friendly syntax for the formal specification, adequate expressivity, and means for potential validation.

The approach proposed in this paper is to exploit the Daplex syntax for the definition of constraints. These constraints may be easily reused and published in forms such as companion specifications.

Furthermore, SWRL is selected as a means for validation instead of Prolog in CoLan solution. In general, a constraint denotes an aspect of the information model that may be subsequently validated. Two steps have to be conducted to ensure SWRL validation:

- Create minimal ontology needed for SWRL rules application.
- Conversion of Daplex statements into SWRL rules.

Both of these steps may be performed automatically.

#### 3.1 Conversion of Daplex-Based Constraints into SWRL

There are two basic elements of the Daplex syntax which form queries, statements, and expressions. Statements restrict the validity of queries to specific instances of particular concepts and include data definition statements and FOR loops (Shipman, 1981). Expressions are specified within statements and are evaluated to a set of entities. Expressions may involve qualification, quantification, Boolean operators, and comparisons.

In general, Daplex elements have the following meaning when converted to SWRL:

- Daplex statements — Class assertions
- Daplex expressions — Object or data property assertions related to a given statement(s)

In other words, statements restrict predicates to a specific set of individuals (from the ontological perspective), e.g., “forall x in Students” — the following predicate will be evaluated against all individuals of Students concept. The translation of this statement into SWRL is:

Students(?x).

Next, expressions limit the query according to a specification. In other words, they express conditions on the properties of data and/or objects to individuals from statements. E.g., “such that x.birthdate = 2000”. The translation the above expression into SWRL is:

birthdate(?x,?xb)  $\wedge$  swrlb(?xb,2000).

In the context of this research, daplex queries have the following structure: the outer and nested part. The outer part consists of a statement and an expression — in general a Boolean predicate. The nested part expresses an existential predicate. The nested part is composed again of a statement and an expression. The nested existential predicate may be as follows:

exists y in Persons  
such that y.sex = "Female"

The existential predicate is the part of the query responsible for the aforementioned validation task, and thus restricts the validation only to a specific subset of individuals.

#### 3.2 Minimal Ontology for SWRL Rules

In the ideal scenario, there is a shared ontology for a given use case to execute SWRL rules. On the other hand, the minimal required ontology may be derived directly from the document specifying the Daplex queries (if the document is considered as a shared and verified standard/specification).

In such a case, the statements may be considered as definitions of concepts, and expressions as definitions of data/object properties. Thus, it is possible to automate the minimal ontology creation, for example, with the help of OWL API.



## 4 USE-CASE - DRIVE MOTION AXIS ATTRIBUTES

This section presents the application of Daplex-based queries conversion into SWRL on motion-axis attributes conditional implementation. The motion axis attributes may have specified condition(s) for which the given attribute is applicable/valid. There are three main categories of conditional implementation of motion axis attributes which were taken into consideration in the context of this paper. The other types are not interesting for further validation, e.g., conditions which refer to a specific version of the integrated development environment.

The three important types of condition are:

- **Dependency on Usage of a Specific Device:** for example, *Auto Sag Slip Increment* and *Auto Sag Slip Time* attributes are applicable if a position feedback device is used. The other example is represented by the *Commutation Polarity* attribute, which is applicable if a permanent magnet motor is used.
- **Dependency on the Value of Another Attribute:** for example, the *Break Voltage* attribute is applicable only if the value of the *Frequency Control Method* attribute is set to “Basic V/Hz only”. Another example is the *Bus Observer Voltage Rate Estimate* attribute, which is applicable if the value of the “*Converter Control Mode*” attribute is set to “*Bus Voltage Control*”.
- **Dependency on the Value of Another Attribute Together with a Specific Device Function:** for example, the *Bus Voltage Error Tolerance Time* attribute is applicable only if the drive is either operated in non-regenerative mode, or is used in regenerative mode and the value of the *Converter Control Mode* attribute is set to the “*Bus Voltage Control*” value.

### 4.1 Example of MAA Constraints in Daplex

In the following paragraphs, the form of MAA constraints in the Daplex-based syntax is presented.

The MAA condition of the type *Dependency on the usage of a specific device* - “The attributes *Auto Sag Slip Increment* (id 874) and *Auto Sag Slip Time* (id 875) are applicable iff a position feedback device is used.” has the daplex form as shown in Listing 1.

Listing 1: Auto Sag Slip Increment Conditional implementation

```
forall m in MAA
```

```
such that m.name="Auto Sag Slip Increment"
exists d in D
such that d.type="Position feedback
device"
```

where MAA is the set of all motion axis attributes and D is the set of all available devices.

Next, the MAA condition of the type *Dependency on the value of another attribute*:

“*The Break Voltage attribute is applicable only if the value of the Frequency Control Method attribute is set to “Basic V/Hz only.”*”

has the daplex form as shown in Listing 2.

Listing 2: Break Voltage Conditional implementation

```
forall m in MAA
such that m.name="Break Voltage"
exists m1 in MAA
such that m1.name="Frequency Control
Method" and m1.value = 0
```

### 4.2 MAA Daplex Constraint Conversion to SWRL

In the previous paragraphs, examples of MAA condition (i.e., dependence on a specific device and value of another attribute) conversions into Daplex were introduced. The conversion from Daplex to SWRL is demonstrated in the last type of condition in the following paragraphs.

The MAA condition of the third type:

“*The Bus Voltage Error Tolerance Time attribute is applicable only if the drive is either operated in non-regenerative mode, or is used in regenerative mode and the value of the Converter Control Mode attribute is set to the Bus Voltage Control value.*”

has the Daplex counterpart:

Listing 3: The Bus voltage error tolerance time conditional implementation

```
forall m in MAA
such that m.name="Bus Voltage Error
Tolerance Time"
exists (df in DFn and m1 in MAA
such that m1.name="Converter Control
Mode" and m1.value=0 and df="G")
or (df in Dfn such that df.value="N")
```

where MAA is the set of all attributes of the motion axis and DF<sub>n</sub> is the set of available device functions.

The Daplex query has two parts: the outer predicate and the nested predicate. The outer predicate *forall m in MAA such that m.name=“Bus Voltage Error Tolerance Time”* is converted into SWRL as

Listing 4: Converted outer part of the condition into SWRL body

```

forall m in MAA
Class assertion -> MAA(?x)
(such that m.name="Bus Voltage Error Tolerance Time")
Object/Data property assertion
-> name(?x,?xn) ^ swrlb:equal/contains(?xn,"Bus Voltage Error Tolerance Time")
exists m1 in MAA and df in DFn
Implication: Class assertion -> MAA(?m1) ^ DFn(?df)
(such that m1.name="Frequency Control Method" and m1.value = 0
and df.value="G")
Object/Data property assertion
-> name(?m1,?m1n) ^ swrlb:equal(?m1n," Frequency Control Method")
^ value(?m1,?m1v) ^ swrlb:equal(?m1v,0)
^ value(?df,?dfv) ^ value(?dfv,"G")

```

Figure 1: One of the output rules from Daplex to SWRL conversion.

```

MAA(?x) ^ name(?x,?xn)
^ swrlb:equal(?xn,"Bus Voltage Error
Tolerance Time")

```

The nested predicate represents the disjunction of two existential predicates *df in DFn* and *m1 in MAA* such that *m1.name="Converter Control Mode"* and *m1.value=0* and *df="G"* and *df in DFn* such that *df="N"*. The disjunction breaks down SWRL into two separate rules. The SWRL rules of the nested predicate are shown in Listing 5-6

Listing 5: Converted nested part of the condition into SWRL body (1st rule)

```

MAA(?y) ^ name(?y,?yn) ^ value(?y,?yv)
^ swrlb:equal(?yn,"Converter Control
Mode")
^ swrlb:equal(?yv,0) ^ DFn(?z)
^ value(?z,?zv) ^ swrlb:equal(?zv,"G")

```

Listing 6: Converted nested part of the condition into SWRL body (2nd rule)

```

DFn(?y) ^ value(?y,?yv)
^ swrlb:equal(?yv,"N")

```

The conversion of the outer and the nested Daplex predicates to SWRL together is illustrated in Fig. 1. The figure illustrates only the first SWRL rule concerning the regenerative drive mode. The second SWRL rule is analogous.

For completeness, the definition of the SWRL head (the rule consequent) in the conversion process was not covered in the previous sections. MAA conditions determine the body of the SWRL. Subsequently, the consequent section outlines the actions to be taken once the conditions are met. There are various possibilities on how to form the SWRL head.

1. Classify the motion axis attribute under a special concept of the ontology. For instance, the *SuccessfullyValidated* concept. The corresponding head is presented as:

```
-> SuccessfullyValidate (?m)
```

2. Notify a user about the fulfilled conditions using SQWRL extension, e.g., printing to the output MAA individual and its name with the help of the select method. The corresponding head is presented as:

```
-> sqwrl:select(?m,?mn)
```

The overall process of rules specification as well as possible validation is depicted in Fig 2 and may be summarized in following steps:

- S(Q)WRL rules specification
  - Input: Daplex form of the MAA conditions
  - Definition of minimal required ontology if needed using OWL API
  - Specification of the set of S(Q)WRL rules representing requirements on MAAs using S(Q)WRL API
- Validation
  - Input: MAAs for validation in the form of Daplex existential predicates
  - Create corresponding ontology individuals using the ontology and OWL API
  - MAAs validation against S(Q)WRL rules

## 5 CONCLUSIONS

This paper addressed the challenge of semantic heterogeneity in industrial automation, particularly focussing on the conditional implementation of the attributes of the motion axis. We explored the limitations of existing approaches, such as natural language specifications, and proposed a novel solution based on Daplex and SWRL.

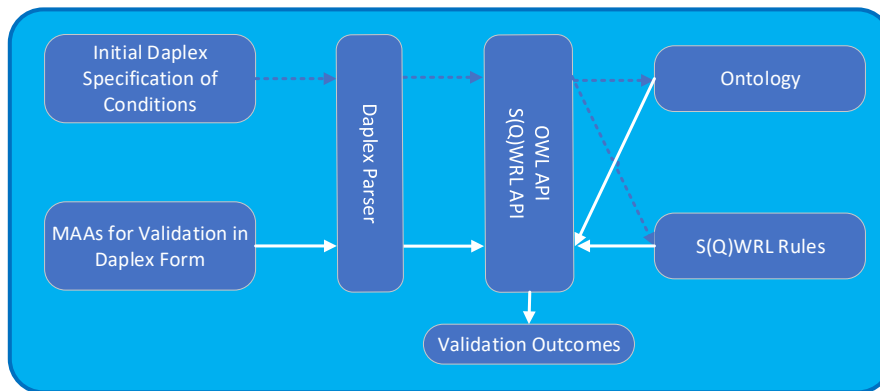


Figure 2: Rules specification and validation process overview.

Our approach leverages the user-friendly syntax of Daplex for defining constraints and the reasoning capabilities of SWRL for validation. We demonstrated the feasibility of our solution by converting Daplex-based constraints into SWRL rules and generating a minimal ontology for their execution. The use case of motion axis attributes illustrated practical application and benefits of our approach. However, the main goal of this work was not to propose the solution which is able to satisfy all possible requirements from different applications or domains. The main goal was to demonstrate that exists user friendly and explicit specification of information models, which should be shared in suitable form (e.g. a standard) to overcome the semantic heterogeneity and facilitate the interoperability.

To conclude, answers to research questions follows:

- What is a suitable formalism for model specification? OPC UA seems to be the most mature standard (in the context of Industry 4.0). This standard may be accompanied by a set of conditions in a user-friendly and explicit way, such as presented in this work to facilitate interoperability and integration.
- What mechanism should be used for model validation? The mechanism is given by the form of the specification. In the context of the presented solution, it is reasoning. The mechanism should not only validate the model using the given conditions. It should also validate that every new constraint is consistent with the constraints already defined.
- The reusability of models should be facilitated by a set of rules (“best practices”) on how to specify the model. However, the model specification with specific constraints is tailored to a given application, and thus the transferability is complicated.

## ACKNOWLEDGEMENTS

This research has been supported by the Rockwell Automation Laboratory for Distributed Intelligent Control (RA-DIC), by institutional resources for research by the Czech Technical University in Prague, Czech Republic, by the OP VVV DMS Cluster 4.0 project funded by The Ministry of Education, Youth and Sports, and by the European Union under the project Robotics and advanced industrial production (reg. no. CZ.02.01.01/00/22.008/0004590).

## REFERENCES

- Bassiliades, N. and Gray, P. M. (1995). Colan: A functional constraint language and its implementation. *Data & Knowledge Engineering*, 14(3):203–249.
- Brickley, D., Guha, R. V., and McBride, B. (2014). Rdf schema 1.1. *W3C recommendation*, 25:2004–2014.
- Bruel, J.-M., Ebersold, S., Galinier, F., Mazzara, M., Naumchev, A., and Meyer, B. (2021). The role of formalism in system requirements. *ACM Computing Surveys (CSUR)*, 54(5):1–36.
- Clocksin, W. F. and Mellish, C. S. (2003). *Programming in PROLOG*. Springer Science & Business Media.
- Dalenogare, L. S., Benitez, G. B., Ayala, N. F., and Frank, A. G. (2018). The expected contribution of industry 4.0 technologies for industrial performance. *International Journal of production economics*, 204:383–394.
- David, R. (1995). Grafcet: A powerful tool for specification of logic controllers. *IEEE transactions on control systems technology*, 3(3):253–268.
- Ellis, H. J. and Demurjian, S. A. (1991). Adam: a graphical, object-oriented database-design tool and code generator. In *Proceedings of the 19th annual conference on Computer Science*, pages 496–505.
- Fortineau, V., Fiorentini, X., Paviot, T., Louis-Sidney, L., and Lamouri, S. (2014). Expressing formal rules

- within ontology-based models using swrl: an application to the nuclear industry. *International Journal of Product Lifecycle Management*, 7(1):75–93.
- Gruber, T. R. (1995). Toward principles for the design of ontologies used for knowledge sharing? *INT J HUM-COMPUT ST.*, 43(5-6):907–928.
- Horridge, M. and Bechhofer, S. (2011). The owl api: A java api for owl ontologies. *Semantic web*, 2(1):11–21.
- Horrocks, I., Patel-Schneider, P. F., Boley, H., Tabet, S., Grosz, B., Dean, M., et al. (2004). Swrl: A semantic web rule language combining owl and ruleml. *W3C Member submission*, 21(79):1–31.
- Lassila, O., Swick, R. R., Wide, W., and Consortium, W. (1998). Resource description framework (rdf) model and syntax specification.
- Leitner, S.-H. and Mahnke, W. (2006). Opc ua—service-oriented architecture for industrial applications.
- Lin, Z., Pearson, S., et al. (2013). An inside look at industrial ethernet communication protocols. *Texas Instruments, White Paper*.
- Martins, J., Nunes, R., Karjalainen, M., and Kemp, G. J. (2008). A functional data model approach to querying rdf/rdfs data. In *British National Conference on Databases*, pages 153–164. Springer.
- McGuinness, D. L., Van Harmelen, F., et al. (2004). Owl web ontology language overview. *W3C recommendation*, 10(10):2004.
- Neumann, P. (2007). Communication in industrial automation—what is going on? *Control Engineering Practice*, 15(11):1332–1347.
- Pang, C., Pakonen, A., Buzhinsky, I., and Vyatkin, V. (2016). A study on user-friendly formal specification languages for requirements formalization. In *2016 IEEE 14th International Conference on Industrial Informatics (INDIN)*, pages 676–682. IEEE.
- Shipman, D. W. (1981). The functional data model and the data languages dplex. *ACM Transactions on Database Systems (TODS)*, 6(1):140–173.