Formal verification of ontology transformation for distribution network information model based on meta-model theory

Boya Qin1, Dong Liu1, Yiming Lu2

1Key Laboratory of Control of Power Transmission and Conversion, Ministry of Education, Department of Electrical Engineering, Shanghai Jiao Tong University, Minhang District, Shanghai, People’s Republic of China
2China Electric Power Research Institute, State Grid Corporation of China, Haidian District, Beijing, People’s Republic of China

Abstract: This study presents a formal description and verification method of the ontology transformation process of the distribution network information model based on the meta-model theory. By defining the dynamic change structures and model mapping constraints of the ontology transformation process of distribution network, a formalised specification for describing the key attributes of model updating and model mapping processes was constructed. The model updating and model mapping instances were specified using model checking verification tool SPIN. The method fundamentally guarantees the standardisation and reliability of the model ontology transformation process achieved by the meta-model.

1 Introduction

The expanding smart distribution grid requires the information model of a distributed network to transform to keep up with the development of gradually updating business modes. Procedures such as the transmission of model changes and mapping of heterogeneous models can be achieved by a reliable media model structure provided by the meta-model theory. This kind of structure is not only capable of reflecting whole information of model changes but also serves as a model of superior level defining mapping specification which achieved the combination of distributed data structure and semantics [1–3].

Traditional methods for the verification of programming logic are generally based on testing and simulation. This kind of method shares the innate defect that it is only capable of specifying errors but not proving the correctness of the logic process. This defect is due to the fact that examples used in testing and simulation usually cannot cover all possible situations perfectly. Furthermore, the operating of most systems is influenced by the external environment and possesses the characteristic of uncertainty. As a result, the verification of the programme is relatively difficult and expensive. To ensure the correctness and reliability of the logic process, a group of scientists in computer science believes that adapting formal methods in the verification and analysis process of the logic process is one of the most important measures of guaranteeing the safety and credibility of systems [4–6].

Formal methods have been used to verify whether the information model of a smart distributed network possesses expected semantic features [7–9]. Using underlying logic and semantics derived from ontology rules and propositions, formal definitions and logical descriptions have been constructed for elements in the information model such as objects, logic, and scenes [10]. In general, the formal modelling and verification method of an information model for the distribution network describes the static rules and semantic characteristics of objects in the electric power system.

On certain occasions, the model needs to be changed to meet new demands. For example, in different business systems when one system model changes and needs to transmit to another system, the latter system needs to be adjusted accordingly to achieve the synchronisation between the two systems, and the unity of distributed data structure and semantic. The static semantics of power system objects are not enough in the verification problem of model updating and model mapping. Therefore, a detailed media model needs to be established to reflect all the information on model changes. Being compatible with the existing model management methods, the latter model should be able to identify the updated information correctly and make the corresponding changes in data structure and storage schema [11].

When two heterogeneous models are merged, a higher-level model needs to be established between models with different modelling methods and formal protocols to define the mapping specifications and to realise the conversion of semantics and data between different models [12–15].

Aiming at guaranteeing the safety and credibility of model transformation processes described by the meta-model theory, the formal method is applied as the verification procedure. The motivation of this research is to construct the formal verification method for model transformation which can ensure the logical correctness of model updating and model mapping processes.

In this paper, we introduce the formal method into meta-model theory, expecting to verify the correctness and credibility of model transformation procedures such as updating and mapping. Model updating and model mapping processes are essential to model transformations in the model-driven architecture. Model updating mainly refers to the adjustment of the original model structure while model mapping focuses on the conversion from one model to another with bidirectional conversions. In consideration of these features, we construct the formal axiomatic system including formal definitions and verification rules. Formal verifications create descriptions for operating elements of the meta-model such as objects, attributes, relationships, accessibility, and type discrimination. Verification rules mainly express the basic logical rules which must be satisfied when conducting operations on model elements, such as transitivity of updating, reversibility, and reflexivity of mapping.

2 Meta-model layer structure

The meta-model generally refers to the semantics, methods, and processes that construct the model. It stores, exchanges, and transforms the completed model data and is referred to as the model of models [16]. The meta-model defines the meta-class structure and logical constraints, so it has the capability of defining the model structure and constraints. Construction of the model refers to expressing the dynamic structure of a single model, i.e. to describe the model update process. Constraining the model refers
The distribution network information model is based on the public system, and is synchronized with the model update [19]. The semantic and structure of data are changed in the model layer, ultimately affecting the targets in an entity device or require multiple entity devices to cooperate to accomplish specific control objectives. There is a fundamental difference between the model structure and semantics caused by modeling theory.

The physical characteristics and application requirements of the power grid both require that a large number of heterogeneous models be merged and the model isolation caused by different modeling methods be eliminated to construct an interconnected model as a whole [11]. Model-level operations can only solve the model semantic differences, but not fundamentally achieve model integration. Therefore, the meta-model for mapping heterogeneous models needs to be defined [12], and relevant protocol needs to be founded to guide the integration and transformation of model and data (Fig. 2).

3 Formal definitions and rules

Model updating and model mapping are essentially model-driven system transformation processes [20]. The updating is to adjust the model on the existing structure, and the mapping is to change the model from one structure to another conserving the bidirectional conversion capability [21]. The meta-model is the framework modelling the model, namely the methods, steps, and rules applied in the transformation.

Definition. Meta-model: The meta-model of the distribution network model M is MM. The meta-model for model updating is MMφ, with the corresponding instance mmφi, representing the ith update of model M. The meta-model for model mapping is MMp, the corresponding instance is mmψi, representing the mapping from model M to model Nj:

\[ \text{MM def } \bigcup_{i=0}^{n} \text{mmφ}_i \cup \text{MMp} \]  

\[ \bigcup_{i=0}^{n} \text{mmφ}_i \] is the set of past updates of model M. \[ \bigcup_{j=1}^{m} \text{mmψ}_j \] is the set of mappings of model M to m models.

The formal rules about model updating and model mapping are listed as follows:

Rule-A. Model updating: According to the definition of instance mmφi for model updating meta-model MMφ, model M transforms into the state after the ith updating processes. It still conforms to the formal rules of the original model M.

\[ \forall (\text{mmφ}_i(\text{Traf}(M^{i-1}, \text{mmφ}_i), M)) \]  

M is the model. \( \text{mmφ}_i \) represents the ith update of M and is one of the instances of MMφ. \( M^{i-1} \) is the model after \((i-1)\) updates. Traf refers to the model transforming method, \( \text{Traf}(M^{i-1}, \text{mmφ}_i) \rightarrow M^{i} \) the method determining the agreement degree of model M.

Unlike model updates, model mappings do not change the model itself, but rather define the syntax and structure details of the transformation from one model to another through meta-model instances [22–24].

Rule-B. Model mapping: The part to be mapped in the distribution network model M conforms to the formal rules of model N after the mapping process mmψi from M to N, vice versa:

\[ \forall j(\text{mmψ}_j(\text{Mapp}(N_j, \text{mmψ}_j), N_j)) \]  

\[ \forall j(\text{mmψ}_j(\text{Mapp}(N_j, M, \text{mmψ}_j), M)) \]

M is the model. \( N_j \) is the jth model to be mapped with model M. \( \text{mmψ}_j \) is the instance of meta-model MMp for model mapping from model M to model Nj. Mapp is the model mapping method. \( \text{mmψ}_j \) is the method determining the agreement degree of model Nj. \( \text{mmψ}_j \) is the method determining the agreement degree of model M.

Inference: Self-mapping for the model always meets the specification [25], namely

\[ \text{mmψ}_i(\text{Mapp}(M, M, \text{mmψ}_i), M) \]
mmV is the meta-model of self-mapping.

Definition. Model version update: The model version corresponding to the ith model update meta-model instance mmVi is

\[ V \text{def}(\text{major, minor}) \]

which is compatible with processInfo of the meta-model. major and minor refers to the major and minor versions of the model, suggesting the extent of difference between model versions. Ver(Vi) returns the corresponding mmVi, meaning the model version is Vi after the update of mmVi.

Rule-C. Multi-updating rule [26, 27]: Given any two versions of the model, Vi and Vj, then the updating process from Vi to Vj is \( \bigcup_{i=0}^{j-1} \text{Ver}(V_i + i) \)

Rule-D. Elimination rule: In the ith update mmVi of a model, a certain object o is eliminated. In the jth update mmVj, object o is eliminated. Then it is considered that this object is not added

\[ \text{Add}(o, mmVi) \land \text{Del}(o, mmVj) \rightarrow (\neg \phi_M(o, M)) \]

Add is the two-dimensional parameter method to determine whether o is added to mmVi. Del is the two-dimensional parameter method to determine whether o is deleted from mmVj. \( \phi_M \) judges whether object o belongs to model M. o represents the object, attribute, or logic.

Rule-E. Cover rule [28]: In the ith update mmVi of a model, an initial value \( \lambda_i \) is assigned to a certain attribute a. Later, in the jth update mmVj, another value \( \lambda_j \) is assigned to a. Then the attribute a is eventually equal to \( \lambda_j \).

\[ \text{Set}(a, \lambda_i, mmVi) \land \text{Set}(a, \lambda_j, mmVj) \rightarrow \text{Init}(a, \lambda_j) \]

Set is the three-dimensional parameter method judging whether attribute a is assigned with value \( \lambda \) in the instance mmVi. Init is the two-dimensional parameter method judging the initial value of a.

Rule-F. Progressing rule: In the ith update mmVi of a model, a certain object o is moved. In the jth update mmVj, object o is moved to another location. Then the object is moved from the original location to the location after mmVj

\[ \text{Move}(o, \theta_i, mmVi) \land \text{Move}(o, \theta_j, mmVj) \rightarrow \text{Post}(o, \theta_j) \]

The move is the three-dimensional parameter method judging whether o is moved to position \( \theta \) in instance mmVi. The post is the method judging whether o is moved from the initial position to the ultimate position.

Fig. 3 Feeder model in CIM12v2

Rule-G. Safety deletion rule: A class can be safely deleted only when all the attributes are concluded and all logical relationships with other classes are deleted [21].

For any specified two versions of the model, the updating mode can be inferred from the combination of several intermediate processes \( \bigcup_{i=1}^{n} \text{mmVi} \) satisfying the formal inference rules A to G.

Rule-H. Direct mapping: An object o in model Ni can be directly mapped into an existing object in another model M, which is called direct mapping

\[ \forall j \left( o \in N_j \rightarrow \exists o \in M \right) \]

Rule-I. Indirect mapping [29, 30]: For object \( o_{N_j} \) in model \( N_j \), if there is no directly mappable object in model M, but there exists a conceptionally consistent object \( o_{M_i} \)

\[ \forall j \left( o_{N_j} \in N_j \land o_{M_i} \in M \right) \rightarrow \left( \exists o_{M_i} \in \Delta_{Mapp}(M, N_j, mm_{N_j}) \right) \]

Conceptionally consistent means identical under the modelling semantics, judged by index set describing the properties of modelling objects [16].

M is the model. \( N_j \) is the jth model to be mapped with model M. \( o_{N_j} \), \( o_{M_i} \) are the objects in \( N_j \), \( M \). \( \Delta_{Mapp} \) is the fusion model. \( \phi_{M_i} \) is the corresponding object in the fusion model. mmVi is the instance of meta-model MM for model mapping from model M to model \( N_j \). Mapp is the model mapping method. \( \phi_{N_j} \) is the method determining the agreement degree of model \( N_j \). \( \phi_{M_i} \) is the method determining the agreement degree of model M.

4 Case study

We use the model updating process of the CIM and the model mapping process between CIM and non-CIM models as the exemplifications for this research.

The meta-model theory focuses on the semantics, methods, and processes concerning the changes and fusions of models. The meta-model is the model of models. Formal methods are used to create formal models and specifications for the target model, perform verification, and use the result to guide the revise of the target model.

Both meta-model theory and formal methods regard models as research objects. Hypothetical as they may seem, these cases are the actual research objects of meta-model theory and formal methods and have been used in practical engineering projects guiding model revises and ensuring logical correctness.

4.1 Model updating

As an example of the semantic transformation of the model updating process, we select the Feeder model in different CIM versions for analysis. The operations are carried out according to the meta-model theory, such as the increase and decrease of classes and attributes, the evolution of the relationship among objects, the localisation and extension, and the simplification of the scene. The formal method is used to validate the logic process, which confirms the correctness and reliability of the model updating process.

(i) Model update process

(a) CIM12v2: The Feeder is the sub-class of collection, in parallel with the network data set. The circuit class is used to represent the Feeder and the circuit section class is added converges to the circuit, as shown in Fig. 3. Circuit aggregates the power system resource class, which is associated with the substation class.

(b) CIM14v11: The circuit class is not only a collection of data but also a logical device container, so it is modified as a subset of the equipment container. Circuit section class directly inherits from the identified object. Considering that the description and function of the circuit are very similar to that in the class Feeder, the circuit
class is replaced by the Feeder class and included in the Wires Ext package of IEC 61968, as shown in Fig. 4.

(c) CIM15v20: The CIM replaces the Feeder class with the circuit class. The problem is that the combination of the circuit and circuit section coincides with the combination of Line and ACLine segments. Therefore, the circuit and circuit section are deleted. Relevant concepts are uniformly described by Line and ACLine segment, as shown in Fig. 5.

(ii) Model updating meta-model
(a) CIM12v2 to CIM14v11: The Feeder class is added to the model. The generalisation logic between the Feeder and equipment container is established, together with the association with the circuit section. The source of this association is circuit section with base 0,…, ∞, and the terminal is Feeder with base 1,…, ∞, as shown in Fig. 6.

(b) CIM14v11 to CIM15v20: The Feeder class is deleted from the model. Line and ACLine segment are combined as a substitute for the Feeder. Since the Feeder class contains attributes that are logically linked to the other two classes, deleting the Feeder class includes removing the logical relationships with other classes, as shown in the following model updating meta-model (Fig. 7): unlike adding an object in CIM12v2 to CIM14v11 updates, deleting an object must first delete its attributes and relationships. Therefore in Fig. 7, the steps of deleting attributes and relationships need to be done separately instead of embedding in the deletion of the class. Removing the Feeder attributes and relationships is similar to the Feeder addition mentioned above, the meta-model instance is not duplicated here.

(iii) Formal verification: The update of the Feeder model from CIM12v2 to CIM15v20 can be considered as the combination of two sub-updating processes: from CIM12v2 to CIM14v11 and from CIM14v11 to CIM15v20.

By Rule-C
\[ \text{mm}_{\text{u1}}(V_{12}, V_{13}) = \text{mm}_{\text{u1}}(V_{12}, V_{11}) \cup \text{mm}_{\text{u1}}(V_{11}, V_{12}) \]  \hspace{1cm} (12)

By Rule-D, \( \forall i \)
\[ \text{Add}(\text{Attr}(i, A_{\text{Feeder}}), \text{mm}_{\text{u1}}(V_{12}, V_{11})) \land \text{Del}(\text{Attr}(i, A_{\text{Feeder}}), \text{mm}_{\text{u1}}(V_{11}, V_{12})) \rightarrow (\neg \phi_{\text{M}}(\text{Attr}(i, A_{\text{Feeder}}), M)) \]  \hspace{1cm} (13)

(see (14))
\[ \text{Add}(\text{Feeder}, \text{mm}_{\text{u1}}(V_{12}, V_{11})) \land \text{Del}(\text{Feeder}, \text{mm}_{\text{u1}}(V_{11}, V_{12})) \rightarrow (\neg \phi_{\text{M}}(\text{Feeder}, M)) \]  \hspace{1cm} (15)

This indicates that the distribution network model \( M \) neither contains the attributes of Feeder nor contains any other classes which have a logical relationship with Feeder. Therefore, the class Feeder does not exist in the distribution network model \( M \). As a substitution, class Line and class ACLineSegment appear in CIM15v20.

Taking the Feeder model updating as an example in Figs. 3–5, the judgment of the model attributes and the logical relationships between the objects are established as follows.

Firstly, the model object type is defined in the model verification tool SPIN, including all the objects that need to be verified

\[
\text{mtype} = \{ \text{Collection, Substation, NetworkDataSet, PowerSystemResource, Circuit, CircuitSection, ConnectivityNodeContainer, Feeder, EquipmentContainer, SubGeographicalRegion, GeographicalRegion, Equipment, ConductEquipment, Conductor, Line, DCLineSegment, ACLineSegment} \}
\]

By analogy, the logical specifications are constructed between Gen(c, c), Asso(c, c), Aggr(c, c) and the formal description of CIM12v2, CIM14v11, and CIM15v20 is completed gradually. On this basis, the two model updating processes CIM12v2 to CIM14v11 and CIM14v11 to CIM15v20 are described in SPIN according to the meta-models described by (12)–(15).

Finally, we put forward the logic that needs to be validated. That is after the two model updating processes, the related attributes of Feeder do not exist in the final version of the model and no class is logically associated with Feeder

\[ (\neg \phi_{\text{M}}(\text{Attr}(i, A_{\text{Feeder}}), M)) \land (\text{Log}(i, L_{\text{Feeder}}, M)) \]  \hspace{1cm} (16)

Equation (18) shows the conclusion: \( (\neg \phi_{\text{M}}(\text{Feeder}, M)) \) is logically reliable in the final version of the model.

From the verification results, we can see that no errors (errors: 0) have been found after an operation of depth 1499 (depth reached: 1499). That is after a limited but sufficient number of model tests, the verification tool has reached all possible states in the state space of the local model. Applying the rules, the static attributes and the logical relationships among the three versions of models are validated, and the rationality of the dynamic operation process in the model updating meta-model is verified.

The SPIN verification tool stops the verification run after enough times of model checking have been performed that the Linear Temporal Logic (LTL) formula is validated without deadlock, starvation, or other unsafe conditions. The conclusion
that the model satisfies the logical assertion to be validated is that in the final version of the model there are no relevant properties of Feeder and there are no classes that are logically associated with Feeder (Fig. 9).

4.2 Model mapping

For the fusion of heterogeneous models, we select two heterogeneous Bay models as an instance, as shown in Fig. 10. Matching the model elements according to their differences, the correctness of the mapping is guaranteed through the formal verification process. The result of the meta-model operation is to synchronise the data structure and logic syntax of these two heterogeneous models.

A large number of models have been put into operation using a database-like table structure, where the definitions and attributes related to all description objects are put into the same class without a reasonable abstraction and division, as shown in Fig. 10, non-CIM transformer bay model. However, CIM is based on the object-oriented UML language and uses the modelling relationships of generalisation, aggregation, and association to express the logical relationship between objects.

By Rule-H

Bay. subsName ∈ NonCIMBay → ∃Core:: Subs. name ∈ CIMBay

Bay. bayID ∈ NonCIMBay → ∃Core:: Bay. mRID ∈ CIMBay

Bay. bayName ∈ NonCIMBay → ∃Core:: Bay. name ∈ CIMBay

Bay. busCon ∈ NonCIMBay → ∃Core:: Bay. busBarConf ∈ CIMBay

For mapping objects that cannot be directly mapped to a corresponding target, the indirect mapping meta-model needs to be constructed through logical relations between the objects. Although indirect mapping is more complicated than direct mapping, it does not lose model information because of the change of mapping mode.

By Rule-I (see (23)). Thus, subsName, bayID, bayName, and busCon in the non-CIM bay model are mapped into related properties Sub.name, Bay.mRID, Bay.name, and Bay.busBarConf in the CIM bay model under direct mapping. The indirect mapping method maps the ratedV in the non-CIM bay model to related properties in both the VLvl and baseV classes in the CIM bay model.

The SPIN tool is used to verify the logical reasoning of the above model mapping process. The verification results are shown in Fig. 11. The verification results show that no errors (errors: 0) are found after the model has undergone a mapping operation of depth 3148 (depth reached 3148 times) (Fig. 12).

The correctness of the dynamic mapping process from the non-CIM bay model to the CIM bay model is verified.

Owing to the state space explosion problem caused by an enormous quantity of objects, attributes, and relations in different models, verification of the model mapping process is quite complicated. Under the limit of current computing resources, conducting formal verification for more specific model cases is beyond achievement yet.
Based on the meta-model theory, this paper presents a formal description and verification method for the ontology transformation of the distribution network information model. Formal description and verification have been conducted to the constructed meta-model of the distribution network information model mapping instances are verified using SPIN, a model checking tool.

In conclusion, we have successfully extended the application of the formal method from describing static rules and semantic features of power system objects to verify the dynamic model transformation process. This will enlighten the research of modelling and verification methods for the smart distribution grid and power system.

5 Conclusion

6 Acknowledgments

This work was supported by the National Key Research and Development Program of China (Basic Research: 2017YFB0903000) and the National Natural Science Foundation of China (with issue number: 51677116).

7 References

[1] Kabaae, E., Wen, L., Wang, Z., et al.: ‘An axiom based metamodel for software process formalisation: an ontology approach’, Int. Conf. on Software Process Improvement and Capability Determination, Cham, 2017.
[2] Chaunah, M.A., Babar, M.A., Sheng, Q.Z.: ‘A reference architecture for provisioning of tools as a service: meta-model, ontologies and design elements’, Future Gener. Comput. Syst., 2017, 69, pp. 41–65.
[3] González-Pérez, C., Henderson-Sellers, B., McBride, T., et al.: ‘An ontology for ISO software engineering standards: 2) proof of concept and application’, Comput. Stand. Interfaces, 2016, 48, pp. 112–123.
[4] Lynch, K., et al.: ‘Ontology-driven metamodelling validation in cyber-physical systems’ in ‘Information technology: new generations’ (Springer, Cham, 2016), pp. 1255–1258.
[5] Gabrenezer, S., Kauffman, P., Seidl, S., et al.: ‘A feature-based classification of formal verification techniques for software models’, Softw. Syst. Model., 2019, 18, (1), pp. 473–498.
[6] Hafeez, A., Musavi, S.H.A.: ‘Ontology-based verification of UML class/OCL model’, Mehran Univ. Res. J. Eng. Technol., 2018, 37, (4), pp. 521–534.
[7] Teixeira, M., Ribeiro, R., Barbosa, M., et al.: ‘A formal method applied to the automated software engineering with quality guarantees’, 2014 IEEE Int. Symp. on Software Reliability Engineering Workshops (ISSREW), Naples, Italy, Nov. 3 2014 to Nov. 6 2014.
[8] Ishigooka, T., Saisi, H., Piper, T., et al.: ‘Practical formal verification for model based development of cyber-physical systems’. 2016 IEEE Int. Conf. on Computational Science and Engineering (CSE) and IEEE Int. Conf. on Embedded and Ubiquitous, Paris, France, August 24–26, 2016.
[9] Zambran, E., Guzzardi, G.: ‘Formal definition of a general ontology pattern language using a graph grammar’. 2017 Federated Conf. on Computer Science and Information Systems (FedCSIS), Prague, Czech Republic, 3–6 September, 2017.
[10] Qin, B., Liu, D., Zhang, P., et al.: ‘Formal modeling and verification of smart distribution grid based on common information model’, China Int. Conf. on Electricity Distribution (CICED), Xian, People's Republic of China, August 10–13, 2016.
[11] Khan, A.H., Musavi, S.H.A., Rehman, A.U., et al.: ‘Ontology-based finite satisfiability of UML class models’, IEEE Trans. on Software Engineering (TSE), 2016, 42, pp. 3040–3050.
[12] Kheddar, D.E., Rodriguez, H.H., Kretschmer, R., et al.: ‘An exploratory experiment on metamodel-transformation co-evolution’. 2017 24th Asia-Pacific Software Engineering Conf. (APSEC), Nanjing, People's Republic of China, Dec. 4 2017 to Dec. 6 2017.
[13] El Mazzouki, N., Nikiforova, O., Lakhrissi, Y., et al.: ‘Towards a generic metamodel for model composition using transformation’, Procedia Comput. Sci., 2017, 104, pp. 564–571.
[14] García-Holgado, A., García-Peñalvo, F.J.: ‘Validation of the learning ecosystem metamodel using transformation rules’, Future Gener. Comput. Syst., 2019, 91, pp. 300–310.
[15] Karla, C., Sarkar, D., Mandal, C.: ‘Verification of parallelising transformations of KPN models’, IET Cyber-Phys. Syst., Theory Appl., 2019, 4, (5), pp. 276–289.
[16] Hinkelmann, K., Laurenzen, E., Martin, A., et al.: ‘Ontology-based metamodelling’ in ‘Business information systems and technology 4.0’ (Springer, Cham, 2018), pp. 177–194.
[17] Malavolta, I., Maccioni, H., Pellicione, P., et al.: ‘Providing architectural languages and tools interoperability through model transformation technologies’, IEEE Trans. Softw. Eng., 2010, 36, (1), pp. 119–140.
[18] Cordeiro, L.C., de Lima Filho, E.B., Bessa, I.V.: ‘Survey on automated symbolic verification and its application for synthesising cyber-physical systems’, IET Cyber-Phys. Syst., Theory Appl., 2020, 5, (1), pp. 1–24.
[19] Lu, Y.M., Liu, D., Yu, W.P.: ‘A meta-model of the CIM update process for smart grid applications’, IEEE Trans. Power Syst., 2014, 29, (1), pp. 49–57.
[20] Ren, H., Huang, J., Jiang, S., et al.: ‘Verification using counterexample fragment based specification relaxation: case of modular/concurrent linear hybrid automata’, IET Cyber-Phys. Syst., Theory Appl., 2017, 2, (2), pp. 65–74.
[21] Przigoda, N., Soeken, M., Wille, R., et al.: ‘Verifying the structure and behavior in UML/OCL models using satisfiability solvers’, IET Cyber-Phys. Syst., Theory Appl., 2016, 10, (4), pp. 276–289.
[22] Bersani, M.M., Marconi, F., Rossi, M., et al.: ‘Formal verification of data-intensive applications through model checking modulo theories’. Proc. 24th ACM SIGSOFT Int. SPIN Symp. on Model Checking of Software, Santa Barbara, CA, USA, July, 2017.
[23] Malik, Q.A., Truscan, D., Lilliu, J.: ‘Using UML models and formal verification in model-based testing’. 2010 17th IEEE Int. Conf. and Workshops on Engineering of Computer Based Systems (ECBS), Oxford, UK, 22–26 March 2010.
[24] Jared, J.R., Bocchi, Y., Fernandez, D., et al.: ‘An analysis of context-aware data models for smart cities: towards fware and eti CIM emerging data model’, Int. Arch. Photogramm. Rec. Remote Sens. Spat. Inf. Sci., 2017, 42, pp. 41–43.
[25] Luckeneder, C., Kaindl, H.: ‘A case study of systematic top-down design of cyber-physical models with integrated validation and formal verification’. Proc. 34th ACM SIGAPP Symp. on Applied Computing, Limassol, Cyprus, April 8–12, 2019.
[26] Basile, C., Canavese, D., Regano, L., et al.: ‘A meta-model for software protections and reverse engineering attacks’, J. Softw. Syst., 2019, 150, pp. 21–21.
[27] Oregoo, H., Azgomi, M.A.: ‘Modelling and evaluation of the security of cyber-physical systems using stochastic Petri nets’, IET Cyber-Phys. Syst. Theory Appl., 2019, 4, (1), pp. 50–57.
[28] Ojamaa, A., Haav, H.-M., Penjam, J.: ‘Semi-automated generation of DSL meta models from formal domain ontologies’ in ‘Model and data engineering’ (Springer, Cham, 2015), pp. 3–15.
[29] Kim, J., Park, S.O.: ‘U-Health smart system architecture and ontology model’, J. Supercomput., 2015, 71, (6), pp. 2121–2137.
[30] Rondhuan, L.B., Sleiman, H.A., Meadhia, C., et al.: ‘Multi-agent solutions for energy systems: a model driven approach’. 2017 22nd IEEE Int. Conf. on Emerging Technologies and Factory Automation (ETF), Limassol, Cyprus, September 12–15, 2017.