Smart Grid model verification method

Artem Voinov¹ and Ilya Senokosov²
¹Penza State University, 40, Krasnaya st., Penza, 440026, Russia
²Cryptosoft, 3, Lermontova st., Penza, 440026, Russia
E-mail: voj49@yandex.ru

Abstract. This paper considers a method for verification and subsequent modification of the Smart Grid during design stage. For automated use, it is required to create a network template, the elements of which would be present in any particular case of implementation. This role in our method is performed by the Smart Grid ontology. Verification for the network created by designer is carried out through comparison with abstract ontology model. When differences are found (e.g., lack of desired control system), the solution search occurs among the rules in Prolog language for this situation, and the Smart Grid model created by designer is translated into Prolog description. If rules are found, then they are applied, modifying the model created by designer. If not – return the model to designer for improvement. The last stage is the processed ontology model translation back to initial description. This method is considered on the example of abstract network model.

1. Introduction
Currently, the Fourth Industrial Revolution ideas (Industry 4.0) [1] are increasingly being introduced. The main goal of Industry 4.0 is the cyber-physical systems (CPS) [2] implementation into mass production. CPS consists of two parts: physical devices and smart part, which takes over the system control. One of CPS representatives is the Smart Grid system [3] (smart electrical networks).

Modern electrical networks contain not only traditional electric power sources but also alternative (solar panels, wind turbines, etc.). This circumstance led to the fact that the electrical network architecture has shifted from centralized to distributed, as alternative sources are located remotely from each other in most cases. The network control system allows changing energy direction in the network for supply and demand adjustment. This is achieved through the use of bi-directional flow in network, enabling the end consumer to act as a source as well.

In the Smart Grid, the IED (Intelligent Electronic Device) is responsible for control and management, which is an embedded microcomputer with the ability to communicate over Internet protocols (particularly TCP/IP).

The Smart Grid designing is carried out over a long period, and as a sequence there are a lot of published articles on its work principles description. Let's consider the works that are most close to our research direction. Paper [4] represents the basic capabilities of the Smart Grid. Article [5] considers monitoring, detection and classification of faults in the Smart Grid. An overview has been compiled for various faults based on the most important databases, which can be used for future developments. Article [6] discusses the issues related to the accuracy improvement for corruption location detection methods in smart networks using large amounts of data from IED. This will enable quick correction for the
network corruption and electric supply resumption. Paper [7] represents a new smart short-circuit detection sensor, which enables the possibility to update its settings during real-time operation. Based on the represented smart sensor, it is possible to create a reliable Smart Grid overcurrent protection. Paper [8] suggests a formal method for control system generation from environment graphical description (represented using the Control Availability Graph), considering the developer’s solutions. The generated system application leads to reduction in energy consumption. Paper [9] represents an algorithm for automatic synthesis of realistic distribution groups of medium voltage devices. Use of this automatic generation enables the distributed network testing and validation. Article [10] suggests a new approach to the distributed networks creation, based on publicly available data on consumer demand, substations number, their location and specifications, as well as the installed equipment estimated cost. In the reviewed scientific researches there is no description for the verification process of existing Smart Grid, as well as options generation to gaps correction.

2. Motivation
The main goal of this paper is to create a verification method, based on the result of which the automated generation of control and overcurrent protection occurs. The method is created on the example of automated check of the control schemes presence and correctness. The control schemes mean both HMI and system using smart electronic devices (e.g. microcomputers or FB). Automated HMI generation was proposed in paper [11]. The overcurrent protection generation is similar to the control schemes generation because the source network modification at physical level occurs, i.e. new devices are added. Modern Smart Grid is quite complex, therefore errors search and their correction, as well as implementation of new elements that will correctly interact with existing elements, requires numerous resources and man-hours. For the system new elements testing, their implementation and work not in the whole network, but only in an isolated part occur. In the case of successful and stable operation of the test sample, implementation into the general network occurs. Testing consumes time, resources and case of errors detection, the network is checked and modified from scratch. Automation for the new elements embedding process will significantly increase the commissioning speed for the Smart Grid updated version. The network reliability is also improved due to the elements automated embedding without which the network will operate, but critical errors will not be handled. Consequently, the network will be unstable and dangerous over the long run. For Smart Grid reliability control, an ontological description of abstract network is used, that contains description for all basic elements that should be present in any Smart Grid. If any element is not found, the network is coded in Prolog for the network automated modification. In order to do this, the rules, used by designers when creating the network, should be coded as predicates, and the network itself should be represented as a set of facts. The rules application to sets of facts will enable the changes automatic introduction. This method disadvantage is the requirement for long-term and detailed testing of initial set of rules. The testing time is compensated when using complete set of rules to multiple networks, because testing and manual modification for each network will cumulatively consume more time than the rules compilation. The set of rules profitability is directly proportional to the number of networks to which it is used.

3. Ontological model
Initial Smart Grid model is created by designer in a special program, for example, Visual SCL. Due to model verification, ontology was created [12], which tracks errors at design stage. The onontological model contains the Smart Grid description according to IEC 61850 [13] and main elements represented in the standard. This ontology is based on the high-level CPS ontology and shown in Figure 1.

The high-level CPS elements are underlined with bold lines. The model contains both abstract concepts (Space, Rules, etc.) and specific concepts (SCL-description, IED, etc.).

The verification process lies in existence check for the elements that constitute specific concepts.
4. Method description

In this paper, the representation in the predicate language (Prolog) is used for the source network modification. The first stage of the method lies in initial SCL (System Configuration Language) file translation, which contains the Smart Grid description in accordance with IEC 61850 into Prolog description.

![Diagram of the Smart Grid network ontological model]

**Figure 1.** The Smart Grid network ontological model.

Each fact in Prolog language [14] can be one of three types:

1. A class instance with a unique name.
   A practical and convenient way to identify class elements is to specify a name based on the path to this element and the unique attribute of this instance.

2. Correspondence of attributes to class objects.
   Each class in SCL has a set of attributes, which are represented in Prolog language as a fact with two objects, constructed as shown in figure 2.
3. Definition of a class as a subclass of another class.  
Definition of one class as a subclass of another is similar for attributes relevance to class objects.  
The next stage is Smart Grid verification based on Prolog rules. Any other toolkit for work with 
predicates can be used instead of Prolog. Verification occurs on the network Prolog description. If there 
is no any required element, then 2 options are possible:  
1) Forwarding to designer for improvement.  
   In this case, the designer gets back his model with the description of required improvements to 
   comply with the ontological model.  
2) Embedding the rule which is implemented through insertion of basic variant of the lack element.  
   In this case, the designer gets back the model in an improved mode.  
The next step is a set of rules application to the Smart Grid Prolog representation. Resulting set of facts should be converted into the language in which the initial model was written (according to IEC 
61850 – this is the SCL language).  

5. Implementation example  
Take the simple electrical network shown in figure 3 as an example.  
This network describes a substation (S1) which contains 2 voltage levels (D1, E1) connected through 
a transformer. The voltage E1 is connected to three devices (Q3, Q4, Q5), each containing a current 
transformer (CTR) and a circuit breaker (CBR).
As an example, implementation way for the overcurrent protection described in paper [15] will be considered. The overcurrent protection is shown in figure 4. Embedding occurs between CTR and CBR elements. After that, the connections lying between all embedded overcurrent protection is carried out.

![Figure 4. Overcurrent protection.](image)

Represented network has an equivalent SCL description, a part of which (as follows bay Q2) is shown below:

```xml
<Bay xxy:x="10" xxy:y="45" name="Q2">  
  <ConductingEquipmentxxy:x="178" xxy:y="15" name="II" type="CTR">  
    <Terminal connectivityNode="S1/D1/Q1/B1" substationName="S1" voltageLevelName="D1" bayName="Q1" cNodeName="B1"/>
    <Terminal connectivityNode="S1/D1/Q2/L1" substationName="S1" voltageLevelName="D1" bayName="Q2" cNodeName="L1"/>
  </ConductingEquipment>
  <ConductingEquipmentxxy:x="178" xxy:y="75" name="QA1" type="CBR">  
    <Terminal connectivityNode="S1/D1/Q2/L1" substationName="S1" voltageLevelName="D1" bayName="Q2" cNodeName="L1"/>
  </ConductingEquipment>
</Bay>
```

The next stage is SCL translation into Prolog description for further embedding of the overcurrent protection.

Prolog description for Q2 is shown below:

```prolog
has_Bay('S1_D1','S1_D1_Bay_2').  
has_name('S1_D1_Bay_2','Q2').  
'ConductingEquipment'('S1_D1_Bay_2_ConductingEquipment_1').  
has_ConductingEquipment('S1_D1_Bay_2','S1_D1_Bay_2_ConductingEquipment_1').  
has_class('S1_D1_Bay_2_ConductingEquipment_1','CTR').  
has_Terminal('S1_D1_Bay_2_ConductingEquipment_1_Terminal_1').  
'Terminal'('S1_D1_Bay_2_ConductingEquipment_1_Terminal_1').  
has_connectivityNode('S1_D1_Bay_2_ConductingEquipment_1_Terminal_1','S1/D1/Q1/B1').  
has_Terminal('S1_D1_Bay_2_ConductingEquipment_1_Terminal_2').  
'Terminal'('S1_D1_Bay_2_ConductingEquipment_1_Terminal_2').  
has_connectivityNode('S1_D1_Bay_2_ConductingEquipment_1_Terminal_2','S1/D1/Q2/L1').
```

```xml``
Each overcurrent protection is represented in Prolog language as an abstract black box. The name of each generated instance will contain “protect” + ordinal number. The current voltage is supplied to input, and on output there are the same voltage and the control signal for CBR. For example, the first instance is abstractly declared as follows:

overcurrent_protect('protect_1').
has_value_input('protect_1','val').
has_pioc_in('protect_1','PIOC').
has_break_output('protect_1','XCBR').
has_break_in('protect_1','CBR').
has_break_out('protect_1','PTRC').

For automated network embedding carrying out, it is required to create rules in Prolog language. The add_overcurrent_protect_add rule is used to add an overcurrent protection instance (without links):

add_overcurrent_protect_add(NameProtect):-
variableX(X),concat('protect_',X,NameProtect),assert(overcurrent_protect(NameProtect)),assert(has_value_input(NameProtect,'val')),assert(has_pioc_in(NameProtect,'PIOC')),assert(has_break_output(NameProtect,'XCBR')),assert(has_break_in(NameProtect,'CBR')),assert(has_break_out(NameProtect,'PTRC')),incement_X.

This rule returns the name of added overcurrent protection and also carries out the following actions:
variable(X) – get current overcurrent protection number,
concat('protect_','X',NameProtect) – get added overcurrent protection name,
assert(overcurrent_protect(NameProtect)) – add overcurrent protection name as a fact,
assert(has_value_input(NameProtect,'val')) – add input for voltage,
assert(has_pioc_in(NameProtect,'PIOC')) – add input to get the triggered overcurrent protection from another protection element,
assert(has_break_output(NameProtect,'XCBR')) – add output signal of the circuit breaker triggering on this protection element,
assert(has_break_in( NameProtect,'CBR')) – add protection triggering signal,
assert(has_break_out(NameProtect,'PTRC')) – add signal of the overcurrent protection triggering on this protection element

increment_X– incrementing current number of overcurrent protection.

Then, it is required to determine the place in scheme, where the overcurrent protection will be embedded. Since each element of the Smart Grid has an absolute path, it is required to determine such path to each embedded element. The rule_path rule is used for this purpose. After that, terminals should be added on the paths found, which will link the network elements. The add_overcurrent_terminal_add rule is used to add terminals. Main rule rule_main is responsible for implementing one overcurrent protection between CTR and CBR. The rule is recursively executed until all required overcurrent protection is added. The rule_path, add_overcurrent_terminal_add and rule_main rules are shown below:

rule_path(NameCBR,Path):-
has_ConductingEquipment(BayNumber,NameCBR),has_name(BayNumber,BayName),has_Bay(VoltageNumber,BayNumber),has_name(VoltageNumber,VoltageName),has_VoltageLevel(SubstationName,VoltageNumber),concat3(SubstationName,'/',VoltageName,Newstring1),concat3('/',BayName,'/',Newstring2),concat(Newstring1,Newstring2,Path).

add_overcurrent_terminal_add(NameCBR,NameTerminal):-
variableY(X),concat('new_terminal_',X,NameTerminal),assert('Terminal'(NameTerminal)),rule_path(NameCBR,Path),concat3(Path,'P',X,RealPath),assert(has_connectivityNode(NameTerminal,RealPath)),incement_Y.

rule_main:-
'ConductingEquipment'(NameCBR),
has_class(NameCBR,'CBR'),not(used_CBR(NameCBR)),has_Terminal(NameCBR,TerminalCBR_1),has_Terminal(NameCBR,TerminalCBR_2),TerminalCBR_1\==TerminalCBR_2,'Terminal'(TerminalCBR_1),has_Terminal(NameCTR,TerminalCTR_1),has_Terminal(NameCTR,TerminalCTR_2),TerminalCTR_1\==TerminalCTR_2,'Terminal'(TerminalCTR_1),has_connectivityNode(TerminalCTR_1,NodeCTR),has_connectivityNode(TerminalCBR_1,NodeCBR),NodeCTR==NodeCBR,retract(has_Terminal(NameCBR,TerminalCBR_1)),retract(has_Terminal(NameCTR,TerminalCTR_1)),retract(has_connectivityNode(TerminalCBR_1,NodeCBR),retract(has_connectivityNode(TerminalCTR_1)),retract(has_connectivityNode(TerminalCBR_1,X)),retract(has_connectivityNode(TerminalCTR_1,Y)),add_overcurrent_protect_add(NameProtect),add_overcurrent_terminal_add(NameCBR,NewTerminal_1),assert(has_Terminal(NameCTR,NewTerminal_1),assert(has_Terminal(NameProtect,NewTerminal_1)),add_overcurrent_terminal_add(NameCBR,NewTerminal_2),assert(has_Terminal(NameProtect,NewTerminal_2)),assert(has_Terminal(NameCBR,NewTerminal_3),assert(has_Terminal(NameProtect,NewTerminal_3)),assert(used_CBR(NameCBR)), rule_main.

To begin the rules execution, it is required to call rule_main. Resulting scheme after all rules execution and translation into SCL description will be as shown in figure 5.
In Figure 5, an instance of the overcurrent protection is represented as a triangle, with 1 input and 2 outputs. All elements of the overcurrent protection are interconnected, which enables the network deactivation at any section of the network. This is the simplest implementation for peer-to-peer overcurrent protection. In more complex overcurrent protection, the main protection element is allocated, triggering of which turns off all the network elements. A side protection element triggering deactivates only a part of the network.

6. Conclusion
This method is suitable not only for the overcurrent protection, but also for any embedded the Smart Grid scheme. Also, expanding the number of rules used will enable a larger area to be covered.

The main disadvantage is the rules creation complexity, due to which high-skilled professionals are required for the rules development, therefore, an abstract network model was selected as an example. But conception of use the rules itself will avoid most design errors and undoubtedly improve the final network.

Further, we plan to create new rules for particular cases, as well as improve the method for new elements embedding. This will enable connecting several networks with each other at once, which will be useful in the transition to the networks global intellectualization.

Acknowledgments
The development of the technological process of SS processing was carried out with the financial support of the Russian Federation President Council on Grants (scholarship of the President of the Russian Federation for young scientists and graduate students) (project No. SP-2511.2021.5).

References
[1] Lasl H, Fettke P, Kemper H G, Feld T and Hoffmann M 2014 Industry 4.0. Business & information systems engineering 6(4) 239-42
[2] Iarovyi S, Mohammed W M, Lobov A, Ferrer B R and Lastra J L M 2016 Cyber–physical systems for open-knowledge-driven manufacturing execution systems. Proceedings of the IEEE 104(5) 1142-1154
[3] IEC S, 2010 Smart Grid Strategic Group (SG3). IEC Smart Grid Standardization Roadmap
[4] Li F, Qiao W, Sun H, Wan H, Wang J, Xia Y and Zhang P 2014 Smart transmission grid: Vision and framework. IEEE transactions on Smart Grid 1(2) 168-177
[5] Rivas A E L and Abrão T 2020 Faults in smart grid systems: Monitoring, detection and classification Electric. Power Systems Research 189 106602
[6] Kezunovic M 2011 Smart fault location for smart grids. IEEE transactions on smart grid 2(1) 11-22
[7] Alonso M Amaris H Alcala D and Florez D M 2020 Smart Sensors for Smart Grid Reliability. Sensors 20(8) 2187
[8] Wojnicki I Kotulska L Sędziwy A and Ernst S 2017 Application of distributed graph transformations to automated generation of control patterns for intelligent lighting systems. Journal of computational science 23 20-30
[9] Schweitzer E Scaglione A Monti A and Pagani G A 2017 Automated generation algorithm for synthetic medium voltage radial distribution systems. IEEE Journal on Emerging and Selected Topics in Circuits and Systems 7(2) 271-284
[10] Trpovski A Recalde D and Hamacher T 2018 Synthetic distribution grid generation using power system planning: Case study of Singapore. In 2018 53rd International Universities Power Engineering Conference (IEEE: UPEC) 1-6
[11] Voinov A, Yang C W and Vyatkin V 2017 Automatic generation of function block systems implementing HMI for energy distribution automation. 2017 IEEE 15th International Conference on Industrial Informatics (IEEE: INDIN) 706-13
[12] Guarino N Oberle D and Staab S 2009 What is an ontology? Handbook on ontologies (Berlin: Springer Heidelberg) 1-17
[13] International Electrotechnical Commission 2009 IEC 61850-6: Communication networks and systems for power utility automation -Part 6 (Configuration language for communication in electrical substations related to IEDs Sec. Ed.)
[14] Clocksin W F and Mellish C S 2012 Programming in Prolog: Using the ISO standard. Springer Science & Business Media
[15] Yang C W Zhabelova G Vyatkin V Nair N K C and Apostolov A 2012 Smart Grid automation: Distributed protection application with IEC61850/IEC61499. IEEE 10th International Conference on Industrial Informatics (IEEE) 1067-1072