A Methodological Framework for Ontology-Driven Instantiation of Petri Net Manufacturing Process Models

Damiano Arena, Dimitris Kiritsis

A methodological framework for ontology-driven instantiation of Petri Net manufacturing process models

Damiano Arena, Dimitris Kiritsis

École Politechnique Fédérale de Lausanne,
SCI-STI-DK ME, Station 9, CH-1015 Lausanne, Switzerland
{damiano.arena, dimitris.kiritsis}@epfl.ch

Abstract. In the last decade, the interest and effort towards the use of ontology-based solutions for knowledge management has significantly increased. Ontologies have been used in manufacturing to provide a formal representation of the domain knowledge in a way that is machine-understandable. However, despite the ability to formally represent the elements of a domain and their relations, ontologies themselves do not provide any kind of simulation and systems behaviour analysis capabilities.
Manufacturing system knowledge may be translated into specific executable models by exploiting experience and human logical deduction. This can be also achieved using ontologies and semantic reasoning.
The framework presented in this work, therefore, aims to explore a W3C standard for inference rules, such as Semantic Web Rule Language (SWRL), and OWL ontology models to transform elements of a Knowledge-Base (KB) into Petri Net (PN) primitives. The combination of semantics and mathematical modelling techniques applied to the analysis of a simple automated assembly station highlights the existence of modelling patterns and the effectiveness of inference rules to automatically instantiate PN-based manufacturing system models.
As results, the inference rules-driven instantiation of a semantically enriched PN model has two positive consequences: (i) the axioms upon which the manufacturing system ontology is built are easy-to-reuse; (ii) the semantics-based bridging of the analysed domains shows the possibility of further enriching the KB with both qualitative and quantitative assessment capabilities.

Keywords: Ontology; Semantic Interoperability; Petri Net

1 Introduction

In the last decades, the manufacturing domain gained several benefits from the application of ontologies, such as the possibility to communicate in an unambiguous manner, the common terminology and semantic alignment, and an information infrastructure in which data are provided in a computational

way [1]. The main applications of manufacturing domain ontologies were summarized by [2]. Among them, [3] introduces an interesting ontology-based framework to support integration of data in the Product Life Cycle (PLC). Beginning Of Life (BOL), Middle Of Life (MOL) and End Of Life (EOL) knowledge aiming to support the different phases of the PLC, from its conceiving and design to the disposal. As far as the design and production phases of a product are concerned, these use BOL knowledge (e.g. technical documentation, process/product requirements, etc.) to define the requirements specification in terms of activities and resources. However, process designing may need the employment of – quantitative or qualitative – analytics techniques that ensure the feasibility and reliability of the production process. Despite the ability to formally represent the elements of a domain (i.e. manufacturing process) and their inter-relations, ontologies themselves do not provide any kind of system behaviour analysis or simulation capabilities. In this regard, Petri Net is a general purpose graphical and mathematical modelling language that is used to describe a large variety of different systems [4]. PN formalism has clear semantics which unambiguously defines the structure and behaviour of each model, hence, forming the foundation for its formal analysis. It is based on very few primitives that allow the explicit description of both states and actions of the modelled system, which is moreover built upon true concurrency, instead of interleaving [5].

Generally, the development of a manufacturing model leverages modelling experience and logical deduction of the domain-specific elements from the real case to be represented. Such logical deduction can be also achieved through semantic reasoning, which is one of the pillar on which ontologies are based on [6]. Recently, ontology-based model transformation became a well-established approach adopted to generate and validate models according to a specific domain of analysis [7]. However, the transformation and mapping from domain abstract models (technology-independent) to technology-dependent models, such as PNs, is nowadays a crucial issues [8].

The paper is structured as follows: Section 2 presents the proposed 3-step approach of this research work. Section 3 introduces the ontologies design and their semantic alignment towards the achievement of a robust theoretical framework that allows ontology-driven instantiation of PN primitives. Semantics-based simulation and analysis of the inferred PN model is therefore described in Section 4.

2 Proposed Approach

The overall approach includes three main steps: (i) Semantic modelling and manufacturing data enrichment; (ii) Ontology-driven instantiation of PN models; (iii) PN-based simulation and analysis. **Fig. 1** shows the stepwise evolution of the information flow: from manufacturing data silos to PN-based model simulation and analysis reports by leveraging semantics. In particular, the application presented in this study deals with the use of data and information regarding an automated assembly station, with specific emphasis on the reliability of its components. The development of ontologies enables the enrichment of this data in a way that it is shareable and ubiquitously interpretable by machines. Thus, we propose a semantic framework built upon two ontological models, i.e. the so-called Automated Assembly Station Ontology (AASO) and the Petri Net Ontology for Reliability modelling (PNO4R). This will therefore drive the instantiation of OWL[1]-like PN elements that resemble the design and production data semantically-enriched and stored in a KB.

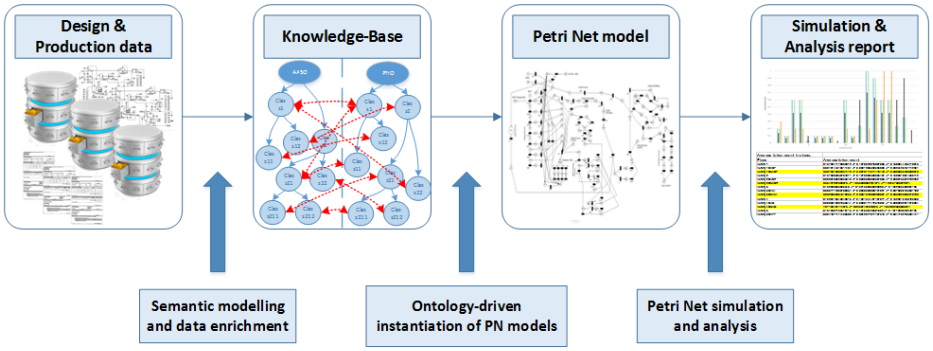

Fig. 1. Proposed Approach

Once the model is instantiated, state-of-the-art tools[2][3][4] can be used to simulate the dynamic behaviour of the PN model and analyse the results, in a way that will yield maximum insight and help with decision-making.

---

[1] Web Ontology Language https://www.w3.org/OWL/

[2] **GreatSPN Tool** http://www.di.unito.it/~greatspn/index.html

[3] **CPN Tools** http://cpntools.org/

[4] **ORIS Tool** http://www.oris-tool.org/

3 Ontology Modelling

In this section, the structures of the above-mentioned AASO and PNO4R are described in terms of common OWL ontology modelling components, such as classes, object and data properties, along with an insight on inference rules.

3.1 Automated Assembly System Ontology

The Automated Assembly System Ontology (AASO) is a domain-specific ontology developed to gather and semantically-enrich the information regarding the presented case study, i.e. an assembly station. The overall ontological structure presented in **Fig. 2** stems from an analysis of the available data regarding assembly operations, component failures and repair modes. The following classes have been, therefore, defined: *Station*, *Component*, *Activity*, *FailureMode*, *RepairMode*, *SystemState*. *Activity* refers to the operations that are performed by the *Station* while *Component* refers to all the parts that form the latter. Each *Activity* has a *SystemState* that may describe the system either before or after the occurrence of the *Activity*. *FailureMode* is the set of all the possible breakdowns that could take place, while *RepairMode* refers to the way such failures may be fixed. Eventually, *CausesOfFailure* includes the reasons that have generated a specific failure.

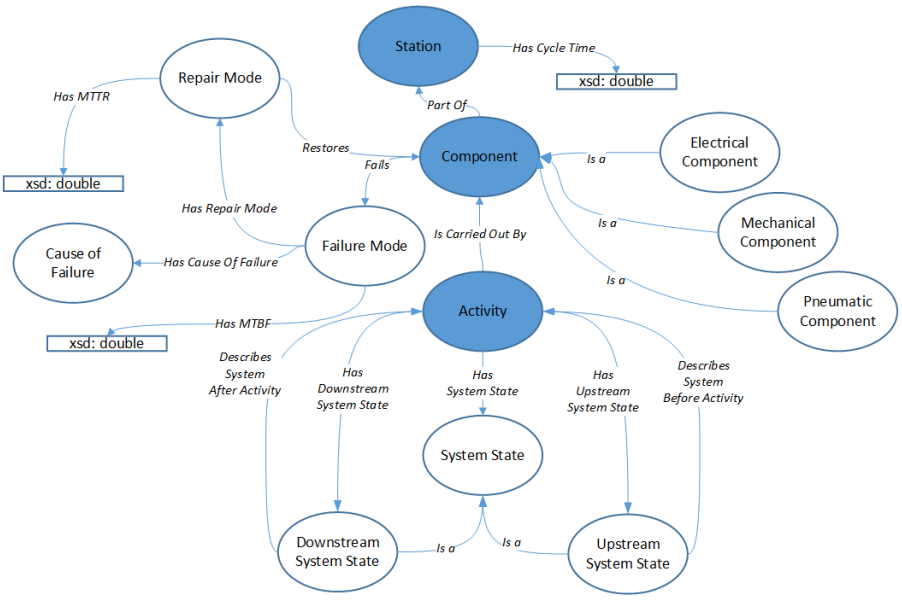

Fig. 2. Automated Assembly System Ontology

3.2 Petri Net Ontology

As far as the Petri Net Ontology (PNO) is concerned, among the various solutions proposed in literature, Gasevic and Devedzic [9] developed a UML Petri net ontology, called *Core Petri net ontology* considering both elements in common for all Petri net models and concepts that do not really exist in Petri net models to obtain more suitable synthetic concepts. Later on, the solution presented by Szymanski and colleagues [10] shows a PNO together with its upper ontology. Here, authors employed the Semantic Web Rules Language (SWRL) to express rules among primitives for instance to identify an active place (i.e. a place that contains a token) and an active transition (i.e. a transition that can fire).

This paper aims to use a hybrid approach, partly combining the research works mentioned above, in fact, here the structure of the so-called *Core Petri net Ontology* is extended with PN elements for Discrete Event Systems (DES) reliability modelling (**Fig. 3**), i.e. the PNO4R. SWRL-based rules are, then, exploited to automate the interpretation of PNO4R instances from the AASO ones.

Let us start from the introduction of the main elements: *Place*, *Transition* and *Arc*, *Initial Marking,* upon which each PN model is built. These have been defined as subclasses of the element *Net* despite [9] defines *Net* as a superclass of the *PN element*. Thus, according to the case study requirements, which will be introduced in the next section and stem from the use of Petri nets for reliability modelling and analysis, two subclasses of *Place* were identified: *ComponentPlace* and *SystemPlace*. *ComponentPlace* is a place at component level which can be either a *FailureStateComponentPlace* (if it represents the component in a failure state) or *WorkingStateComponentState* (if it represents the component when it is in available state). Conversely, *SystemPlace* includes places at system level, representing the various phases of the process.
Similarly, it is possible to further discern the *Transition* class into *ComponentTransition* and *SystemTransition*. *ComponentTransition* can be *FailureComponentTransition* or *RepairComponentTransition*, which characterizes respectively the breakdown event and the repairing activity. *SystemTransition*, instead, gathers transitions at system level describing the various activities of the system. The distinction of *Component* and *System* (**Fig. 4**) levels is due to the fact that the assembly process was discretized, then, different process

phases were identified and modelled at higher level (system level). A set of components may, therefore, fail at each one of those phases and are modelled at a lower level (component level). For these reasons, the class called *Arc* is further specialised in four subclasses according to the elements that it connects. The first one is *ComponentToSystemArc* (i.e. the arc that connects the repair transition with the system place) and the second is *SystemToComponentArc* (i.e. the arc that connects the system place and the failure transition).

The third subclass is *ComponentArc*, which is further divided in *PTComponentArc* (i.e. arc linking a place representing the available component to a failure transition and arc linking place representing not available components to a repairing transition) and *TPComponentArc* (i.e. arc linking failure transitions to places representing not available components and arc linking repairing transition to places representing available components).
Then, *SystemArc* is similarly divided into *PTSystemArc* and *TPSystemArc* according to the connection at system level either from place to transition or from transition to place. Eventually, *Token* is the class meant to contain the tokens that the places could contain. However, the *Marking* class has been defined outside the *Net* class, and represents the set of all possible markings of the net. In particular, its subclass *FailureMarking* identifies the failing states of the systems.

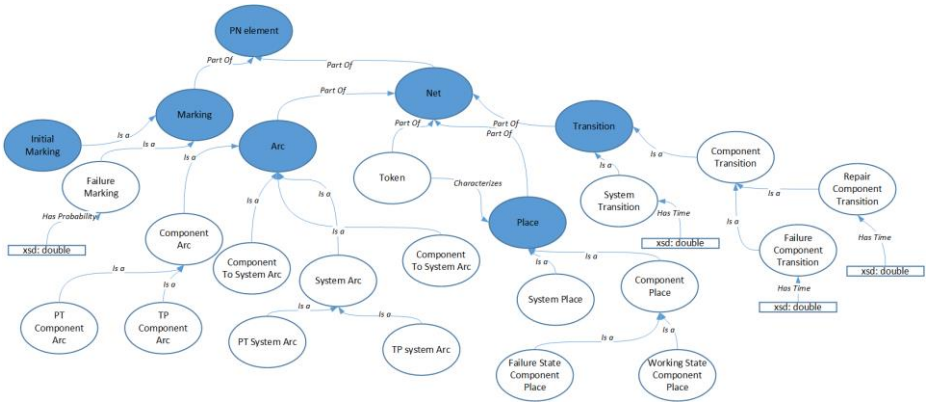

Fig. 3. Class hierarchy of Petri Net Ontology extended with DES reliability modelling elements (PNO4R)

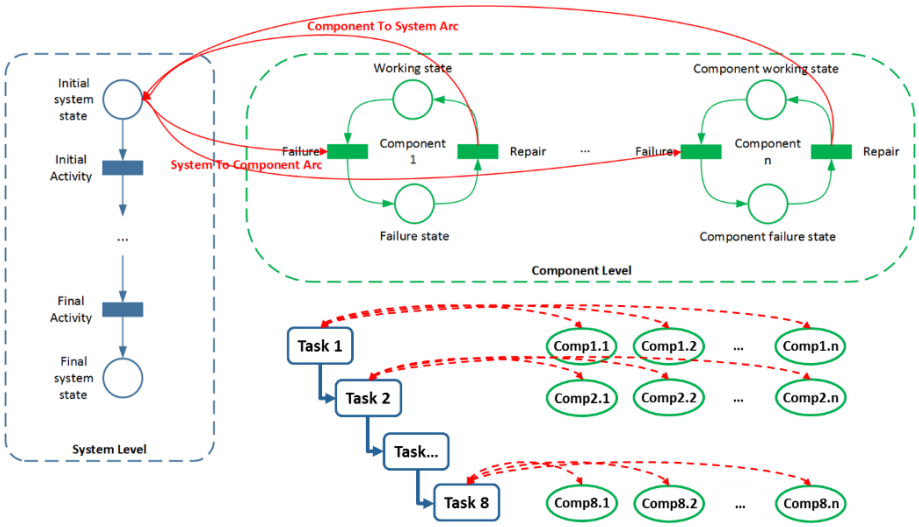

Fig. 4. Distinction between system and component levels

3.3 Semantic Alignment

Generally speaking, an alignment between two ontologies "specifies a set of correspondences, and each correspondence models a bridge between a set of ontologies entities" [11]. Applying this definition to the present study means enabling the transformation of the elements of the AASO to the respective elements of the PNO4R. To this aim, the first step has been the introduction of a hybrid concept called *Module* (**Fig. 5**), which does not belong to neither the AASO nor the PNO4R. However, this modelling patter plays a crucial role in the alignment, since it is able to mediate between the AASO and PNO4R. In particular, the *Module* represents the behaviour of a component that may fail or be repaired in different ways and with a specific probability distribution.

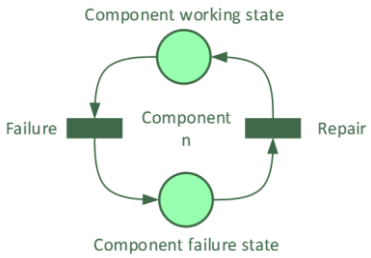

Fig. 5. Module: PN representation of a component behaviour

The instantiation process of PNO4R individuals and their properties is driven by SWRL-based rules. Starting from instances of AASO it is possible to create a congruent number of instances belonging to the PNO4R. The following relations have to be, hence, added to the semantic models in order to univocally link the elements of the two domains and avoid misinterpretations. Here, *System Level*, refers to all the discrete assembly phases and the states of the modelled system before and after each phase, while *Component Level* refers to modules resembling components that may fail at a specific state of the system.

Table 1 (left) System Level - (right) Components level owl class inter-links

System Level	
S1	Activity → SystemTransition
S2	SystemState → SystemPlace
S3	SystemState → SystemTransition
S4	SystemPlace → SystemTransition
S5	SystemPlace → SystemArc

Component Level	
C1	Module → FailureComponentTransition
C2	Module → RepairComponentTransition
C3	Module → FailureStateComponentPlace
C4	Module → WorkingStateComponentPlace
C5	Module → ComponentArc
C6	Module → ComponentToSystemArc
C7	Module → SystemToComponentArc

Due to a limited pages number, the class inter-links mentioned above cannot further described. Nonetheless, their indispensable statement will be better understood with two examples presented in the next section (S1 and C1).

4 Ontology-Driven Manufacturing Model Instantiation and Analysis: Case Study

The purpose of this section is to describe an application case, whose modelling requirements and data have been extracted from to [12]. In the latter work the authors proposed computational experiments to explore the validity of two different techniques to decompose a system. In order to reach their goal, they tested their methodology by means of a Petri net model, which resembles a part of an automated assembly shop.

At this point in time it is noteworthy to mention that the construct http://swrl.stanford.edu/ontologies/built-ins/3.3/swrlx.owl*: makeOWLThing*[5] plays a non-trivial role. This is indeed required to execute inference rules

[5] http://swrl.stanford.edu/ontologies/built-ins/3.3/swrlx.owl (last Access 2 February 2017)

through the rule engine called *Drools*[6] that create a new element according to a deduction process based on existing OWL elements. Firstly, OWL knowledge and SWRL rules will be transferred to the SWRL engine. The latter uses a reasoner library in order to infer new axioms that will be eventually translated and loaded to actual the KB. The Instantiation of owl individuals of SystemTransition class resembling owl individuals of Activity (S1) is achieved by including the following rule in the ontology

Activity(?a) ^ ActivityHasDuration(?a,?d) ^ swrlx:makeOWLThing(?st,?a) → SystemTransition(?st) ^ SystemTransitionHasDuration(?st,?d) ^ ActivityIsCharacterizedBy(?a,?st)

This should be perceived as the backbone of the inferred PN model since most of the other modelling elements will be connected to those ones. Each OWL individual inferred through this rule is, therefore, related to *SystemTransitionHasDuration*, whose data property is inherited from the *ActivityHasDuration*, and then related to the *Activity* as well. Furthermore, here there is *ActivityIsCharacterizedBy* that has been defined as the inverse functional of *SystemTransitionCharacterizes*. This way, it is possible to connect instances belonging to the two different domain ontologies in a univocal manner.

The instantiation of individuals of the owl FailureComponentTransition inferred from the individuals of the owl class Module (C1) is achieved as follows. Each Module comprises of one FailureComponentTransition and one RepairComponentTransition. The former has a data property describing the λ, which is the inverse of the Mean Time Between Failures (MTBF). The SWRL rule, hence, aims to create the owl instance and make it inherit that property:

Module(?m) ^ Failure(?f) ^ ModuleCharacterizesFailure(?m,?f) ^ FailureHasMTBF(?f,?mtbf) ^ swrlx:makeOWLThing(?fct,?m) ^ swrlb:divide(?l, 1,?mtbf) → FailureComponentTransition(?fct) ^ FailureComponentTransitionHasLambda(?fct,?l) ^ ModuleIsComposedByFCT(?m,?fct) ^ FailureComponentTransitionCharacterizes(?fct,?f)

[6] http://protegewiki.stanford.edu/wiki/DroolsTab (last Access 2 February 2017)

4.1 Simulation of the Inferred PN Model

Hereafter the data used for the simulation of the PN model. This consists of 8 System Transitions (ST), 13 System Places (SP), 16 Component Transitions (CT), 36 Component Places (CP). There is only one module linked to each SP, which means that only one component may fail in a specific phase of the process. The available values are, therefore, the processing time of each activity, the failure rate (λ), and the repair rate (μ) as it is shown in **Table 2**. Then, the initial marking of the PN has been set as all the components are available and the process is ready to start.

Generally speaking, there are two kind of simulations: continuous and discrete [13] [14]. Here we propose the evaluation of the average number of tokens in a place and the transition throughputs by simulating the inferred PN through GreatSPN simulator, which exploits the Monte Carlo Simulation technique. Given the fact that the net has been built in a way that each place can contain at most one token, this measure reflects the probability of having a token in that place. The simulation results have been obtained through a so-called Basic GSPN (no colours) simulation with the Solver GreatSPN Confidence Interval 95% - Approximation 20% - Set batch length max: 31000.

Table 2. Reliability values and processing times

Activity	Time [h]	Component	λ [1/h]	μ [1/h]
A1	0.0192	BF	0.174	3.412
A2	0.0192	BR	0.295	3.386
A3	0.0185	MI	0.073	3.439
A4	0.0185	BI	0.004	5.042
A5	0.0204	FF	0.114	3.433
A6	0.0204	FW	0.074	3.461
A7	0.0204	FC	0.074	3.378
A8	0.0204	RF	0.194	3.409
Tback	0.0196	UC	0.367	3.385

4.2 Semantics-Based Analysis of the Results

Two key assessments have been carried out on the simulation results: the probability that a failure state component place has a token and the throughputs of the system (given by the simulation tool). Beside their semantical enrichment and storage in the KB, this can be queried in order to obtain: i) the probability that a failure occurs on a specific component (**Fig. 6**); and ii) the overall system availability (**Fig. 7**);

component	ComponentHasProbabilityToFail
MI	"0.0031379900268981"^^<http://www.w3.org/2001/XMLSchema#double>
FC	"0.0036769317614786"^^<http://www.w3.org/2001/XMLSchema#double>
FW	"0.003603079152595"^^<http://www.w3.org/2001/XMLSchema#double>
RF	"0.0093411466662446"^^<http://www.w3.org/2001/XMLSchema#double>
BF	"0.0078714671473149"^^<http://www.w3.org/2001/XMLSchema#double>
BR	"1.033900632191E-4"^^<http://www.w3.org/2001/XMLSchema#double>
FF	"0.0057082925242496"^^<http://www.w3.org/2001/XMLSchema#double>
BI	"8.82587424049E-5"^^<http://www.w3.org/2001/XMLSchema#double>
UC	"0.0174963998355185"^^<http://www.w3.org/2001/XMLSchema#double>

Fig. 6. Protégé SPARQL Query Tab: Probability that a failure occurs on a specific component

SystemAvailability
"0.9489730440800767"^^<http://www.w3.org/2001/XMLSchema#double>

Fig. 7. Protégé SPARQL Query Tab: System availability

5 Conclusions

The conceptualization of a modelling framework based on ontologies and Petri nets has been thoroughly described. Efforts were made to clarify the interpretation of manufacturing concepts in the PN domain and vice versa, thus, paving the way towards a methodological framework to be used for ontology-based model transformation, PN-based simulation, and reliability analysis of a manufacturing system. The core part of this research concerns the development of two different models: the Automatic Assembly System Ontology (AASO) and the Petri Net Ontology for Reliability modelling (PNO4R). The latter represents an attempt of extension of an existing Petri Net ontological model with elements of reliability modelling for discrete event systems. This enabled the PN-based representation of the automated assembly station knowledge for reliability assessment. The structure of the model and the operating principles were eventually outlined. Finally, the achievement of semantic interoperability between the AASO and PNO4R enabled the SWRL rules-based instantiation of all the PN elements resembling a simple assembly

process along with its simulation and analysis, showing the potentialities of the proposed approach, beside its soundness.

References

1. Schlenoff, C., Ivester, R., Libes, D., Denno, P., S., & S. (1999). An Analysis of Existing Ontological Systems for Applications in Manufacturing and Healthcare. NIST.
2. Negri, E., Fumagalli, L., & Garetti, M. (2015). Approach for the use of ontologies for KPI calculation in the manufacturing domain. Proceedings of the XX Summerschool of Industrial Mechanical Plants "Francesco Turco", Napoli, Italy, 16th-18th September 2015, 30–36.
3. Kiritsis, D. (2011). Closed-loop PLM for intelligent products in the era of the Internet of things. Comput. Des. 43, 479–501.
4. Desel, J. and Reisig, W., (2015). The concepts of Petri nets. Software and Systems Modeling, 14(2), p.669.
5. Jensen, K., (2013). Coloured Petri nets: basic concepts, analysis methods and practical use (Vol. 1). Springer Science & Business Media.
6. Motik, B. and Rosati, R., (2010). Reconciling description logics and rules. Journal of the ACM (JACM), 57(5), p.30.
7. Roser, S. and Bauer, B., (2005), October. Ontology-based model transformation. In International Conference on Model Driven Engineering Languages and Systems (pp. 355-356). Springer Berlin Heidelberg.
8. Silega, N., Noguera, M. and Macias, D., (2016). Ontology-based Transformation from CIM to PIM. IEEE Latin America Transactions, 14(9), pp.4156-4165.
9. Gasevic, D. and Devedzic, V., (2007). Interoperable Petri net models via ontology. International Journal of Web Engineering and Technology, 3(4), pp.374-396.
10. Szymański, K., Dobrowolski, G., Koźlak, J., & Zygmunt, A. (2007). A Proposition of Knowledge Management Methodology for the Purpose of Reasoning With the Use of an Upper-Ontology. Knowledge Creation Diffusion Utilization, 8, 117–133.
11. Scharffe, F., Euzenat, J. and Fensel, D., (2008), March. Towards design patterns for ontology alignment. In Proceedings of the 2008 ACM symposium on Applied computing (pp. 2321-2325). ACM.
12. Jeong, K. C., & Kim, Y. D. (1997). Performance analysis of assembly/disassembly systems with unreliable machines and random processing times. IIE Transactions, 30(1), 41–53.
13. Balbo, G. (2007). Introduction to Generalized Stochastic Petri Nets. In *Formal Methods for Performance Evaluation* (First, pp. 1–83). Springer.
14. Wang, J. (2006). Petri Nets for Dynamic Event-Driven System Modeling. In P. Fishwick (Ed.), *Handbook of Dynamic System Modeling* (pp. 1–17).