Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 112 (2022) 73–78

www.elsevier.com/locate/procedia

14th CIRP Conference on Intelligent Computation in Manufacturing Engineering, Gulf of Naples, Italy

Enabling automated checking of information in factory planning with ontologies – a case study

Matthias Ebade Esfahani[a,*], Peter Burggräf[a,b], Tobias Adlon[a], Stephan Matoni[a]

[a]*Laboratory for Machine Tools and Production Engineering (WZL) of RWTH Aachen University, Campus-Boulevard 30, 52074 Aachen, Germany*
[b]*Chair Of International Production Engineering and Management, University of Siegen, Paul-Bonatz-Str. 9-11, 57076 Siegen, Germany*

* Corresponding author. Tel.: +49 241 80-22054. *E-mail address:* M.Ebade-Esfahani@wzl.rwth-aachen.de

Abstract

Planning a manufacturing factory involves a high number of planners who deliver their domain-specific planning information. In Building Information Modeling, a central entity needs to check the information from different planning domains for coherence to each other. Especially for non-geometrical information, this is a manual, time-consuming and error-prone approach. The paper at hand therefore aimed at developing an expert system that automates the decision making in the checking process of factory planning information. On the basis of a case study, we have applied knowledge engineering methods from the Semantic Web and calculated an F_1 *score* for validating our approach.

© 2022 The Authors. Published by Elsevier B.V.
This is an open access article under the CC BY-NC-ND license (https://creativecommons.org/licenses/by-nc-nd/4.0)
Peer-review under responsibility of the scientific committee of the 15th CIRP Conference on Intelligent Computation in Manufacturing Engineering, 14-16 July, Gulf of Naples, Italy

Keywords: Factory planning; Building Information Modeling; Ontologies; Expert systems; Automated checking

1. Introduction

Factory planning is the indivisible planning of a value-adding core – that is, production and logistics – and its surrounding building. The already multi-disciplinary process of traditional construction consequently becomes more complex with the addition of further interfaces.[1] This causes many iterations in the design process, which in turn require centralised quality checks to ensure consistency of the individual designs. In order to organise these highly networked design processes, the Aachen Factory Planning Approach was developed, comprising planning modules to support project management. For each of the planning modules, the required input information as well as the output information to be generated are transparently displayed [2]. However, even using the Aachen Factory Planning Approach, central quality checks still have to be performed manually, and planners deliver their results in 2D, 3D or sometimes even sketched on paper.

Building Information Modeling (BIM) has been introduced into the construction industry to address exactly this issue. Through consequent 3D-modeling and informational annotation of the design objects, it aims to manage design information more efficiently, to enhance clarity of the designs, and to perform central quality checks automatically [3]. But apart from the fact that BIM is rarely used in factory planning projects, many quality checks still have to be carried out manually in factory planning. BIM mainly supports the identification of geometric clashes. In factory planning, however, non-geometric and thus informational "clashes" are of particular concern – for example within the interface of production design and MEP (Mechanical, Electrical, Plumbing).[4] The central authority of a BIM-manager or BIM-coordinator is responsible for checking the various individual design deliveries against each other. Yet, the required range of competencies for this can hardly be efficiently covered by a human resource or a planning team. The corresponding checking process, especially in factory planning, should hence become automated. [5]

Beyond purely geometrical checks, there are some approaches that aim at automating rule checking. El-Gohary et al. [6, 7] and Hjelseth et al. [8] have used Natural Language Processing to automatically generate rules out of regulations

2212-8271 © 2022 The Authors. Published by Elsevier B.V.
This is an open access article under the CC BY-NC-ND license (https://creativecommons.org/licenses/by-nc-nd/4.0)
Peer-review under responsibility of the scientific committee of the 15th CIRP Conference on Intelligent Computation in Manufacturing Engineering, 14-16 July, Gulf of Naples, Italy
10.1016/j.procir.2022.09.047

This is a resupply of March 2023 as the template used in the publication of the original article contained errors. The content of the article has remained unaffected.

written in text. These approaches are generally applicable in the field of factory planning but do not solve the problem of checking design deliveries against rules that are not yet formulated in a standardised way. [5] However, several researchers have proposed the use of ontologies for the automated checking in factory planning or at least of general BIM design deliveries. Büscher et al. have developed a Virtual Production Intelligence platform (VPI) which integrates and checks factory planning information using a previously created ontology [9]. Lee et al. have used ontologies to develop data exchange requirements for BIM projects [10]. Encouraged by these research insights, we have developed a framework based on Semantic Web technologies and standards from the international BIM organisation buildingSMART to express informational dependencies in a machine-readable manner and thus to enable automated checking of design information in factory planning [4]. The Semantic Web extends the existing World Wide Web (WWW) by enhancing terms and concepts with their respective meaning (one word can have different meanings, e.g. pool, nail). This enables computers to unambiguously read, understand, and interlink data. [11]

Our whole research is based on Design Science Research (DSR) [12]. The core of this DSR-methodology is that the actual knowledge gain can only be achieved by iterating our framework. This is where this paper sets in. The objective of this study is to validate this already developed framework by implementing it for the first time in a real project environment aiming at automated checking of design deliveries in BIM-based factory planning. This first iteration is an essential step to transform our framework into an operable expert system in factory planning.

The paper is structured as follows:

After the introduction in section 1, we use section 2 to introduce our automated checking framework and its components. The core of the paper is section 3 where the implementation of our framework is described and validated using a factory planning project from WZL of RWTH Aachen University. The results of the study are discussed in section 4 before concluding the paper with a summary and an outlook.

2. Development of a framework for automated checking in factory planning

Our already developed framework consists of four essential elements. (a) An OWL-ontology (Web Ontology Language), transparently representing the domain knowledge and the relations among the various knowledge entities. (b) Real data from a specific factory planning project, exported from proprietary software. (c) A parser that transforms the existing real data into the RDF-schema (Resource Description Framework) and thus enables the instantiation of the data into the ontology. (d) A constraint language that checks the data based on a specific Information Delivery Manual (IDM). In the following, we shortly describe each element.

2.1. OWL-ontology creation

The first element of the framework is an OWL-ontology. These ontologies are always built in a Triple-structure of subject, predicate, object (e.g. *machine has_material iron*), which creates a high level of expressivity. The most important elements of an OWL-ontology are classes, individuals and properties. Individuals are concrete instances of a class. Classes thus aggregate similar objects and thereby form a hierarchy just as in object-oriented programming. As for properties, we use object properties (to connect certain individuals) and data properties (to connect individuals to concrete values). [13]

An important asset of ontologies, and especially OWL, is inference. So-called reasoners (e.g. *Pellet*) can automatically derive logical relationships, check the ontology for consistency and determine whether an individual can be assigned to a class based on certain formulated requirements. [13]

For the specific implementation of our framework, we use the open-source ontology editor *Protégé* from Stanford University.

2.2. Use of BIM data from a factory planning project conducted by WZL of RWTH Aachen University

The starting point of our research is the interaction with data and information from real factory planning projects that are planned with BIM. Therefore, the second step is the extraction of these data and information.

It should be emphasised that our framework is not tied to any particular proprietary software. Rather, we build on the possibility of an export of object-oriented component and attribute lists (e.g. in form of an Excel spreadsheet). For the instantiation of the data, it is important that the information in the Building Information Model adheres to the following specifications:

- The naming in the ontology and the model must match (e.g. a component's height needs to be named "has_height" and not only "height").
- Unless otherwise specified in the ontology, values of planning information shall always be specified in SI units or units composed of SI units.
- Every object must have its corresponding class annotated to it (e.g. a window in the Building Information Model requires the value 'window' in the planning information class) so that the model information can be mapped to its equivalents in the ontology. This class annotation should only be created for those objects that need to be integrated into the ontology for the automated checking process.
- All model objects that are relevant for the automated checking must have unique names.

2.3. Parsing and instantiation of the BIM data into the ontology

OWL-ontologies are built in a Triple-structure (see section 2.1). This data structure does not match the structure of the exported component and attribute lists (e.g. Excel spreadsheet).

This is a resupply of March 2023 as the template used in the publication of the original article contained errors. The content of the article has remained unaffected.

The exported data must therefore be parsed. *Protégé*'s plug-in *Cellfie* can be used for that purpose. It transforms the information from the Excel spreadsheet and directly inserts it into the selected ontology by employing user-specific instantiation rules.

As a result of the inserted project data, the formerly general ontology becomes a project-specific ontology. [14]

2.4. Creation of checking rules with SHACL

The data of the project-specific ontology then needs to be automatically checked. Two aspects should be considered: (a) the content of the checking rules and (b) the rule language to be used. As for the content of the rules, we can refer back to an IDM that we created for factory planning projects at the WZL of RWTH Aachen University. It includes a process model and clear requirements for each of the data and information exchanges between the participants [15]. Moreover, an adequate language is needed to execute these rules. For that, we use SHACL; a so-called constraint language that was developed to formulate conditions and restrictions on RDF-based structures like ontologies. [16] SHACL needs two RDF-based elements: Firstly, a shapes graph in which shapes represent a group of conditions to be met and, secondly, a data graph which contains the data to be checked. [17]

Just like for *Cellfie*, *Protégé* provides a plug-in called *SHACL4P*, allowing the ontology creator to formulate SHACL rules in a separate tab within the same user interface.

The result of the checking process is a validation report in form of a list of violated rules. [17]

3. Depiction of the use case for demonstrating and analysing the framework

In the following section, the specific use case will be illustrated to validate our above-described framework aiming at automated checking of design deliveries in BIM-based factory planning.

3.1. Description and preparation of the use case

We applied our framework to the use case of a battery cell production in a cleanroom. At the time we conducted our validation, the project to which we applied our framework was in an advanced level of the design phase. We particularly chose this use case since cleanroom production represents an edge case where the production process sets extreme requirements for MEP in terms of air humidity, temperature and particle cleanness. [18]

As the objective of this study is to validate whether our framework is suitable for checking Building Information Models, we deliberately implemented errors in the sub-models so that we can check ex-post whether these errors are detected. These errors are, among others:

- The production machine and the workers generate an excessive heat load that cannot be sufficiently reduced by the HVAC. → *complex non-geometrical error*
- The production machine and the workers emit too many particles in total which cannot be sufficiently filtered by the air conditioning system so that the required ISO-class of the cleanroom is no longer complied with. → *complex non-geometrical error*
- The LOD (Level of Development = planning granularity) of the production machine model is too low regarding the specifications of the present planning status. → *simple non-geometrical error*
- The required connected load of the production machine is too high for the selected power connection in the cleanroom. → *simple non-geometrical error*
- The height of the production machine exceeds the height limit of the cleanroom. → *simple geometrical error*

The remaining model information is to be classified as correct (21 pieces of planning information).

We aim at validating our framework of automated checking by comparing its results to those of manual "traditional" model checking. In order to perform this comparison, an evaluation measure is needed that can capture and evaluate the accuracy of both approaches. A useful evaluation measure for the present application is the *F_1 score* which is an established method to evaluate binary classification models and their data sets. It combines *Precision* and *Recall* of a model and is therefore calculated as follows (1): [19]

$$F_1 = \frac{2*P*R}{P+R} \quad (1)$$

F_1 = F_1 score; P = Precision; R = Recall

Precision indicates in percent how much of the selected data is also correctly positive (2). [19] Transferred to our use case, *Precision* indicates how many errors that were detected are actually errors in the model. The formula is:

$$P = \frac{t_p}{t_p+f_p} \quad (2)$$

P = Precision; t_p = true positive; f_p = false positive

Recall is the percentage that describes the probability of positive data being correctly classified as positive and not going undetected (3). [19] Transferred to our application, *Recall* indicates the probability with which an error in the model is even recognised as such. The formula is:

$$R = \frac{t_p}{t_p+f_n} \quad (3)$$

R = Recall; t_p = true positive; f_n = false negative

To be able to draw a conclusion whether automated checking is superior to manual checking, a threshold value is required above which a calculated *F_1 score* can be considered acceptable. Based on the definition of the *F_1 score*, it is evident that a score of 0 is the poorest result and a score of 1 is the best result [19]. Beyond that, existing literature does not consistently offer a threshold to be reached. We as authors of this paper, however, set the following two thresholds:

This is a resupply of March 2023 as the template used in the publication of the original article contained errors. The content of the article has remained unaffected.

1) The *F_1 score* of our framework needs to be above the *F_1 score* of the manual approach (will be shown in section 3.3)
2) The threshold must be above the value that would be statistically achievable via pure coin tossing [19], i.e., per model information, the probability of being classified correctly lies at exactly 50%. Related to our use case, the odd number of information requires a determination to the safe side, which results in the following, so-called confusion matrix (see Table 1):

Table 1. Confusion matrix for calculating the threshold of the F_1 score.

	Planning information incorrect	Planning information correct
Recognised as incorrect by the planner	3 (true positive)	10 (false positive)
Recognised as correct by the planner	2 (false negative)	11 (true negative)
Calculated F_1 threshold	0.333	

After being determined, the *F_1 scores* of both approaches are normalised in terms of needed checking time (see section 4). We consider this to be an important step as the *F_1 score* can be significantly improved if just enough time is invested. However, due to the high time pressure in factory planning projects, error-free planning is not an end in itself but must be carried out within an appropriate time frame.

3.2. Implementation of the use case

In this section, we present the specific content of the Building Information Model to which we applied our use case and which we used to validate our framework of automated model checking. The Building Information Model of the use case comprises a battery cell production machine in a cleanroom including ancillary infrastructure (see Fig. 1). This model was created with the software *Revit* and *Inventor* from Autodesk.

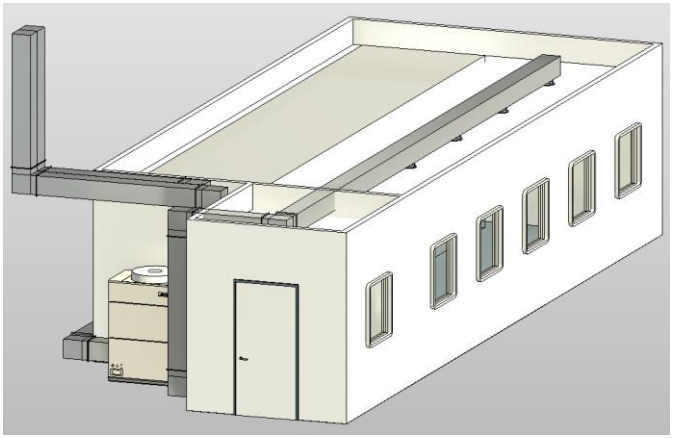

Fig. 1. Cleanroom model in *Revit*

room: name	class	name	has_Length	has_Width	has_Height	has_Weight
	Air_Handler_Unit	AirHandlerUnit	0.765	3.410	1.690	834.00
cleanroom1	Machine	ProductionMachine	10.000	2.000	2.700	16300.00
cleanroom1	POC	PowerSocket	0.075	0.080	0.080	0.20
cleanroom1	SupplyAirOutlet	SupplyAirNozzle1	0.400	0.400	0.200	0.50
cleanroom1	SupplyAirOutlet	SupplyAirNozzle2	0.400	0.400	0.200	0.50
cleanroom1	SupplyAirOutlet	SupplyAirNozzle3	0.400	0.400	0.200	0.50
cleanroom1	SupplyAirOutlet	SupplyAirNozzle4	0.400	0.400	0.200	0.50
cleanroom1	SupplyAirOutlet	SupplyAirNozzle5	0.400	0.400	0.200	0.50
cleanroom1	SupplyAirOutlet	SupplyAirNozzle6	0.320	0.320	0.200	0.45
cleanroom1	ExhaustAirOutlet	ExhaustAirNozzle1	0.200	0.060	0.200	0.40
cleanroom1	ExhaustAirOutlet	ExhaustAirNozzle2	0.200	0.060	0.200	0.40

Fig. 2. Excerpt of the exported component list of the Building Information Model

The required ontology of the interface between production and MEP had already been available as a result of previous research activities at the WZL. It contains all classes and relationships required for our use case.

As a final step, the adjusted component list was exported from the BIM software, using the plug-in *Export-Import Excel* from BIM One (see Fig. 2 for an excerpt of that list).

In a next step, a *Cellfie* script was created to parse the model information from the Excel-list into the Triple-format of the OWL-ontology. For this, specific transformation rules were created which assigned the objects and their data from the Excel-list to the appropriate classes of the ontology.

In a last step, every already assigned object and piece of data was allocated to the overarching instance of the cleanroom. An excerpt of the *Cellfie* instantiation script for the use case is shown in Fig. 3. In this excerpt, it can be observed that, inter alia, all individuals were located in column *D* of the Excel-list and that their data was documented in the columns *F* to *W*.

As described in section 2, the final step in our automated checking approach was the creation of SHACL rules that check the project-specific ontology for planning errors. In our case, the required rules for geometric and non-geometric model checking were derived from an already created IDM (see section 2.4). After all rules had been specified, they were created based on the specifications of SHACL. Following the completion of the SHACL script, a validation report was generated upon rule execution (see Fig. 4). This validation report shows the automatically detected errors in the project-specific ontology. In addition, the previously created checking rules can be used to conclude which planning information is to be classified as correct.

```
Rule
Individual: @D*
Types: @C*
Facts:
@F1 @F*(xsd:float), @G1 @G*(xsd:float), @H1 @H*(xsd:float),
@I1 @I*(xsd:float), @J1 @J*(xsd:float), @K1 @K*(xsd:float), @L1 @L*(xsd:float),
@M1 @M*(xsd:float), @N1 @N*(xsd:float), @O1 @O*(xsd:float), @P1 @P*(xsd:float),
@Q1 @Q*(xsd:float), @R1 @R*(xsd:float), @S1 @S*(xsd:float), @T1 @T*(xsd:float),
@U1 @U*(xsd:float), @V1 @V*(xsd:float), has_LoD @W*(xsd:float)
Individual: @A3
Types: Cleanroom
Facts: is_equipped_with @D23, has_number_of_persons_with_cleanroom_clothes 2,
has_ISO_cleanroom_class ISO_cleanroom_class_6, has_Length 11, has_Width 5,
has_Height 2.6
```

Fig. 3. Excerpt from the instantiation script

This is a resupply of March 2023 as the template used in the publication of the original article contained errors. The content of the article has remained unaffected.

Message
Cooling load of cleanroom too high for air handler unit
Cooling load of cleanroom too high for air handler unit
Cooling load of cleanroom too high for air handler unit
Cooling load of cleanroom too high for air handler unit
Cooling load of cleanroom too high for air handler unit
Cooling load of cleanroom too high for air handler unit
Machine amperage too high for power socket in cleanroom
Height of the machine must be smaller than the height of the room
Required LoD is not achieved
Particle concentration in the cleanroom too high, so that the ISO class can no longer be maintained
Air Handler Unit is missing in the model

Fig. 4. Excerpt from the validation report

3.3. Results

We validated our framework by comparing manual and automated checking of the model information. The process of our previously developed framework for automated checking has been described in section 2. As for the manual checking, we involved seven factory planners and provided them with information they would also normally refer to in our projects when checking such factory planning models manually. Among these were: Employer's Information Requirements (EIR), 2D construction plans of the cleanroom, several data sheets (e.g. of the production machine), a collection of formulas for calculating the process parameters in the cleanroom and planning information from the Aachen Factory Planning Approach. Apart from this, no other instructions were given for the manual checking. The analysed results of the manual checking can be seen in Table 2.

Table 2. Results of the manual checking.

Planner	Duration (min)	Precision	Recall	F_1 score
1	35:00	0.333	0.2	0.25
2	46:56	0.833	1	0.909
3	39:41	0.75	0.6	0.667
4	33:20	0.8	0.8	0.8
5	39:21	0.667	0.4	0.5
6	53:25	1	0.8	0.889
7	48:01	0.8	0.8	0.8
Average	42:15	0.74	0.657	0.688

The following distinctive characteristics of the manual checking results can be documented: The simple geometric and non-geometric errors were recognised by most of the planners, whereas the complex errors were recognised less frequently. Especially those complex errors which included formulas led to calculation and transformation mistakes. For example, only one planner calculated the actual particle concentration in the cleanroom correctly. In addition, the planners also determined some planning information to be incorrect, even though it was correct (e.g. only five supply air nozzles were detected, although six were present and required).

In contrast to manual checking, Fig. 4 shows an extract of the validation report of the automated checking approach. It is worth mentioning that the error regarding the insufficient cooling capacity of the air handler unit appears six times. This is because the calculation for the cooling load was performed six times, which can be led back to the existence of six instances of the class window in our use case. Thus, these six errors can be considered as one. Beyond that, only one piece of actually correct planning information is classified as incorrect: The existence of the instance air handler unit. A reason for this error could not yet be determined.

The component list to be exported in the BIM software, the instantiation script for *Cellfie* and the SHACL validation script were not included in the duration of the automated checking process. The required time for the checking process itself is also hardly noticeable in the automated approach. However, what has been included in the mentioned duration is the loading and execution time of various software/plug-ins. The evaluated results of the automated checking can be seen in Table 3:

Table 3. Results of the automated checking.

Duration (min)	Precision	Recall	F_1 score
2:29	0.833	1	0.909

It can be concluded from Table 3 that the automated checking approach detected all errors built into the model and that it only detected one additional error beyond that.

4. Discussion and conclusion

Based on the results shown in Table 2 and Table 3, it can be deduced that the average F_1 *score* of the manual checking approach is calculated as 0.688 and the F_1 *score* of the automated checking approach as 0.909. Therefore, the automated checking approach has achieved the better result. If we now normalise both F_1 *scores* over the required checking duration, we obtain a value of 0.016 for the manual checking and a value of 0.366 for the automated checking. The normalisation is done by dividing the F_1 *score* by the required checking duration.

Based on these results, it can be seen that both threshold values described in section 3.1 are exceeded with the automated checking approach of our framework. In addition, the comparison of both the unnormalised and normalised F_1 *scores* of the manual and automated checking shows that the automated approach can be classified as qualitatively better.

The objective of this study was to validate our framework for automated checking of planning information by implementing it for the first time in a real project environment. Since its F_1 *score* in our use case is 0.909, which is above the F_1 *threshold* and also above the manual F_1 *score*, it can be concluded that the framework does help to check different BIM-based factory design deliveries more efficiently than done manually. Our framework can therefore facilitate the often complex and time-consuming review of Building Information Models in factory planning. Moreover, the transparent,

This is a resupply of March 2023 as the template used in the publication of the original article contained errors. The content of the article has remained unaffected.

unambiguous, and machine-readable formalisation of both resources and their interdependencies enables downstream development of plug-ins for proprietary software.

It might be critically noted that also for the automated checking, a longer preparation was necessary, e.g. for the creation of the ontology itself. However, we as authors see this rather as a setup time which is needed only once in order to be able to use it within the scope of a specific project. In addition, many parts of this preparation could also be used for following similar projects, thereby shortening the required preparation time. Beyond that, manual checking also requires preparation time, for example to extract formulas, specifications and properties of a model object. Equivalent to the automated checking, this preparation time was not taken into account in the calculations either.

It could also be criticised that we did not make full use of existing, already established ontologies and existing IDMs. However, neither suitable ontologies nor adequate IDMs had existed for factory planning. Even if established ontologies and IDMs existed, an adaptation would most likely be necessary since the naming in the ontology has to match the naming of both the *Cellfie*-script and the properties in the Building Information Model. Trying to add synonyms to the ontology in *Protégé* afterwards, which are recognised by *Cellfie*, does not work (e.g. via skos:altLabel or rdfs:Label). Therefore, we had created a suitable ontology and IDM ourselves.

Apart from that, we would like to emphasise that the ontology itself was created in German language to match the language that is mostly used in German factory planning projects. For stronger integration into the Semantic Web, it could be beneficial to convert the ontology into English. For this purpose, a mapping step could be built in before the instantiation with *Cellfie*, which translates the German terms of the model into English.

The validation carried out in this paper convinces us that our framework is an expert system that brings BIM-based factory planning projects one step closer to automated planning. Nevertheless, it might be reasonable to still perform manual checks for trivial topics (e.g. “Does the factory have a roof?”) where manual checks are not error-prone and comparably time-efficient. For this reason, we would like to use further research steps to investigate the optimal balance between manual and automated checking.

References

[1] Burggräf, P., Dannapfel, M., Schneidermann, D., Ebade Esfahani, M. & Schwamborn, N., Integrated factory modelling: Using BIM to disrupt the interface between manufacturing and construction in factory planning, in: Wilde, W. P. de, Mahdjoubi, L., Garrigós, A. G. (Eds.). *Building Information Modelling (BIM) in Design, Construction and Operations III. BIM 2019*, Seville, Spain, 09.10.2019 - 11.10.2019. *WIT Transactions on The Built Environment*, WIT PressSouthampton UK, 2019, pp. 143–155.

[2] Schuh, G., Kampker, A. & Wesch-Potente, C., Condition based factory planning. *Prod. Eng. Res. Devel.*, 5 (1), pp. 89–94, 2011. DOI: 10.1007/s11740-010-0281-y.

[3] Sacks, R., Eastman, C. M., Lee, G. & Teicholz, P. M., *BIM handbook: A guide to building information modeling for owners, designers, engineers, contractors, and facility managers*, Wiley, Hoboken, New Jersey, 2018.

[4] Burggräf, P., Dannapfel, M., Ebade Esfahani, M. & Schwamborn, N., How to Improve Collaboration Efficiency in the Built Environment of Factories by Using an Integrated Factory Modelling Concept – An Expert Study. *IJDNE*, 15 (4), pp. 473–481, 2020. DOI: 10.18280/ijdne.150403.

[5] Burggräf, P., Dannapfel, M., Ebade Esfahani, M. & Scheidler, F., Creation of an expert system for design validation in BIM-based factory design through automatic checking of semantic information. *Proc. of the 14th CIRP Int. Conference on Intelligent Computation in Manufacturing Engineering (CIRP ICME '20), 14-17 July 2020, Procedia CIRP, Elsevier, in print.*

[6] Zhang, J. & El-Gohary, N., Extraction of Construction Regulatory Requirements from Textual Documents Using Natural Language Processing Techniques, in: Issa, R. R., Flood, I. (Eds.). *Computing in Civil Engineering (2012). International Conference on Computing in Civil Engineering*, Clearwater Beach, Florida, United States, June 17-20, 2012, American Society of Civil Engineers, Reston, VA 06112012, pp. 453–460.

[7] Zhang, J. & El-Gohary, N. M., Integrating semantic NLP and logic reasoning into a unified system for fully-automated code checking. *Automation in Construction*, 73, pp. 45–57, 2017. DOI: 10.1016/j.autcon.2016.08.027.

[8] Hjelseth, E. & Nisbet, N., Capturing normative constraints by use of the semantic mark-up (RASE) methodology. *Proceedings of CIB W78-W102 Conference in Sophia Antopolis*, pp. 1–10, 2011.

[9] Büscher, C., Voet, H., Krunke, M., Burggräf, P., Meisen, T. & Jeschke, S., Semantic Information Modelling for Factory Planning Projects. *Procedia CIRP*, 41, pp. 478–483, 2016. DOI: 10.1016/j.procir.2015.12.022.

[10] Lee, Y.-C., Eastman, C. M. & Solihin, W., An ontology-based approach for developing data exchange requirements and model views of building information modeling. *Advanced Engineering Informatics*, 30 (3), pp. 354–367, 2016. DOI: 10.1016/j.aei.2016.04.008.

[11] Semantic Web; World Wide Web Consortium. https://www.w3.org/standards/semanticweb/, Accessed on: 20 Apr 2021.

[12] Hevner, A. R., March, S. T., Park, J. & Ram, S., Design Science in Information Systems. *Management Information Systems Quarterly*, 28 (1), pp. 75–105, 2004.

[13] Horridge Matthew, *A Practical Guide To Building OWL Ontologies Using Protégé 4 and CO-ODE Tools*.

[14] Cellfie; Johardi. https://github.com/protegeproject/cellfie-plugin, Accessed on: 11 Apr 2021.

[15] DIN Deutsches Institut für Normung e.V., Building information models – Information delivery manual, DIN EN ISO 29481-1, Jan. 2018.

[16] Hartmann, T., Zapilko, B., Wackerow, J. & Eckert, K., Validating RDF Data Quality Using Constraints to Direct the Development of Constraint Languages. *2016 IEEE Tenth International Conference on Semantic Computing (ICSC). 2016 IEEE Tenth International Conference on Semantic Computing (ICSC)*, Laguna Hills, CA, USA, 04.02.2016 - 06.02.2016, IEEE, 2016 - 2016, pp. 116–123.

[17] Shapes Constraint Language (SHACL); Knublauch, H. & Kontokostas, D. https://www.w3.org/TR/shacl/, Accessed on: 11 Apr 2021.

[18] Korthauer, R. (Ed.), *Handbuch Lithium-Ionen-Batterien*, Springer Berlin Heidelberg, Berlin, Heidelberg and s.l., 2013.

[19] What is the F-score?; Wood, T. https://deepai.org/machine-learning-glossary-and-terms/f-score, Accessed on: 11 Apr 2021.

This is a resupply of March 2023 as the template used in the publication of the original article contained errors. The content of the article has remained unaffected.