2019 EC³
2019 European Conference on Computing in Construction
Chania, Crete, Greece
July 10-12, 2019

MODELLING CONSTRUCTION SCHEDULING CONSTRAINTS USING SHAPES CONSTRAINT LANGUAGE (SHACL)

Ranjith K. Soman[1,2]
[1] Centre for Systems Engineering and Innovation, Imperial College London, UK
[2]The Alan Turing Institute, London, UK

.

Abstract

This paper presents a new approach for modelling construction scheduling constraints using Shapes Constraint Language. Current modelling approaches focuses on modelling precedence and discrete constraints at master planning or phase planning level and lacks the ability to model complex constraints at look ahead planning level. Proposed modelling approach addresses this limitation. Precedence constraints, discrete resource capacity constraints, disjunctive constraints and logical constraints are modelled using shapes constraint language for a simple lifting problem in this paper. The modelled constraints were tested, and the constraints model was able to identify the violations effectively and produce a validation report.

Introduction

Digitisation in the construction sector has led to generation of massive amounts of data from projects (Anumba et al., 2000; Whyte et al., 2016). Often these data are siloed in disconnected databases (domain or vendor specific) (Dave et al., 2016) and methods to integrate the siloed data has become highly relevant research problem in the construction sector. Incompatible data structures used during different phases of a construction project is found to be a major contributor to the existence of data silos (Čuš-Babič et al., 2014). To address this problem, researchers have proposed use of Linked-data technologies and Resource Description Framework (RDF) to represent information modelled in the construction sector (Beetz et al, 2009; Pauwels and Terkaj, 2016). Linked-data technologies offer a common environment for linking and sharing data across heterogeneous sources and domains without being limited by the scope of underlying schemas of the source data (Zhang and Beetz, 2016). Linked-data technologies are built following the principles of Description Logics (DL) which is a subset of first-order logic used extensively for representing structured knowledge (Baader et al., 2003). RDF is a standard model/format for data interchange based on the linked-data principles stated by Berners-Lee (2006). RDF enables data merging even when the underlying schemas are different and supports the evolution of schemas.

Researchers have used linked-data based approaches to address the limitations of Building Information Modelling in terms of interoperability, linking across domains and logical inferencing related to *the product information in BIM* (Liu et al., 2016; Pauwels et al., 2011; Pauwels et al., 2017; Quattriniet al., 2017; Terkaj et al., 2017; Zhang and Beetz, 2015, Zhang and Beetz , 2016) and for *facility management* (Kim et al., 2018; Lee et al., 2016; Terkaj et al., 2017). However, there has been insufficient attention to the construction stage, leaving issues of BIM during the construction stage relatively unexamined by researchers that use linked-data.

4-Dimensional Building Information Model (4D BIM) integrates the time dimension into the BIM models and researchers have used this concept and integrated it into the construction workflows to measure the performance (Huhnt et al., 2010; Subramanian et al., 2000). However, the data in these 4D models are not detailed to the process level information and constraints. For instance, Han and Golparvar-Fard (2017) have stated that the current methods cannot document field issues for further analysis as the 4D BIM's "Model Breakdown Structure typically does not match operational details or require creating complicated namespaces which, without visual representations, are difficult to communicate" (p. 1733). In the same context, Giretti et al. (2012) had stated that they had to decompose the tasks into sub-tasks to determine causal relationships among the involved variables so the whole progress could be estimated since there is no linear dependence between the resources employed at every hour and the work progress. The level of detail of 4D BIM is often limited to the master planning or phase planning level and lacks the level of detail and linkage required in the six weeks look ahead level. The 4D models lack the information on the relationships between the activities/ components/resources and the constraints governing them. This limits the application of automatic schedulers using artificial intelligence.

This paper tries to address this limitation, related to the lack of process level constraint codification, by modelling the process level constraint information using Shapes Constraint Language (SHACL), based on linked-data technology. The rest of this paper is divided into five sections: background, modelling constraint information

using SHACL, implementation, results and discussion, and conclusions.

Background

This section provides background knowledge on scheduling constraints and Shapes Constraint Language which are used extensively in the subsequent sections of this paper.

Scheduling constraints

Constraints are base concepts for scheduling as they inform what can or cannot happen, or in other words what can or cannot be scheduled at a point of time (Dechter and Cohen, 2003). From a functional point of view, construction schedules should ensure that the following does not happen:-

i) Defy the laws of physics,

ii) Assign resources (Machines, crew etc.) that are not available or suitable to activities, and

iii) No two construction processes should happen at the same space at the same time.

These are formalized as scheduling constraints into precedence, discrete resource capacity and disjunctive constraints by researchers (Morkos, 2014; Niederliński, 2011).

Precedence constraints make sure that the principles of physics are not violated. This constraint is the basis of the Critical Path Method (Darwiche et al., 1988; Donget al., 2013). These include constraints such as roof must be cast after all the columns are cured, the third floor should be built after the second floor etc. These constraints don't validate or resolve the availability of resources.

Discrete resource capacity constraints are used to solve constraints related to the availability and assignment of resources. They are called discrete, because the resource may be available in numbers more than on, but in discrete integer values. This constraint ensures that resources are allocated according to availability. This constraint also ensures that a discrete resource is not allocated to two activities at the same time. There is abundant research on this constraint in areas such as resource levelling and resource constrained scheduling (El-Rayes and Jun, 2009; Hu and Flood, 2012).

Disjunctive constraints are constraints which states that two activities cannot happen at the same time. These can be used to model spatial constraints (Baykan and Fox, 1997), safety constraints etc. For example, there should not be any activities in the vicinity of an excavation.

In addition to the above constraints, Logical constraints are needed to govern resource assignments. Although cranes are designed to lift components, cranes have a limitation on their capacity. It can only lift up to a certain weight. These constraints should also be coded as constraints to make the schedule meaningful.

Shapes Constraint Language (SHACL)

Shapes Constraint Language (SHACL) is a data modelling language developed by the W3C working group to model constraints against which RDF data could be validated (Knublauchet al., 2017). This addresses the limitation in linked-data technologies to define structural constraints in an RDF graph due to the inherent Open World Assumption in linked-data (Ekaputra et al., 2016).

A SHACL processor has two inputs. The first input into a SHACL processor is the data graph (data in RDF format) containing the data to be validated against constraints. The second input is the shapes graph which contains the definition of constraint against which data is to be validated. Constraints in SHACL are called shapes. There are two types of shapes, a node shape and a property shape. Node shapes declare constraints on a node (for example, a class). Property shape declares constraints on the attributes of a node through a path property. When a data graph is validated against a shapes graph, a validation result is produced. The validation report is an RDF graph describing the conformance with the constraints. It details the nodes that have passed the constraints and the ones that failed (Gayoet al., 2017).

SHACL-SPARQL is an advanced feature of SHACL which contains all the functionalities of SHACL Core and in addition the expressive power of SPARQL-based constraints and an extension mechanism to declare new constraint components. SPARQL is a semantic query language to query the information from the RDF based heterogeneous data. It features an SQL-like syntax and can be used to query RDF triples that are maintained in local files or triple stores (Harris et al., 2013).

Modelling constraint information using SHACL

This section will describe the modelling of constraints explained in the previous section in the Shapes Constraint Language. The modelling of constraints in this section is based on a simplified lifting process as shown in Figure 1. There are three classes modelled for representing this process as shown in Figure 2. The classes have attributes which are data properties and object properties. Data properties are the attributes giving the information such as start date, the weight of module etc. Object properties are the attributes which define the relationship of the object with another object.

1. Process: It is the parent class and instances of this class stores information related to the task including the start and end dates as data properties, resources, module association, precedence and disjunctive constraints as object properties.
2. Crane: Instances of this class stores the information related to the crane. For the current paper, the scope of this class is limited to one attribute crane capacity modelled as a data property, in order to demonstrate the logical constraint.
3. Module: Instances of this class stores information related to the module. Module weight is the only attribute data property used.

Even though object relationships are not explicitly defined in module and crane classes, the inherent linking in the OWL applies the relationship defined in the process class to the crane and module classes.

Logical constraint

In the current problem setting, there is a simple logical constraint which is to be satisfied while scheduling. The Lifting capacity of the crane should always be higher than the weight of the module. The constraint is modelled as shown below.

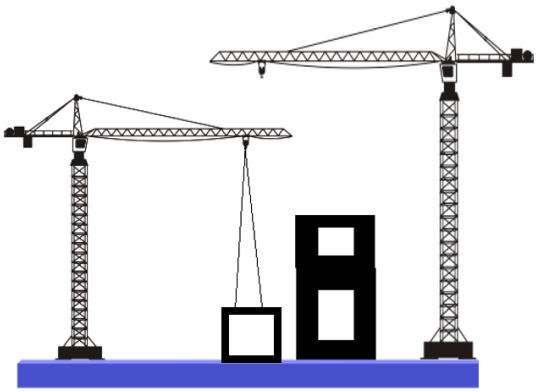

Figure 1: Lifting process

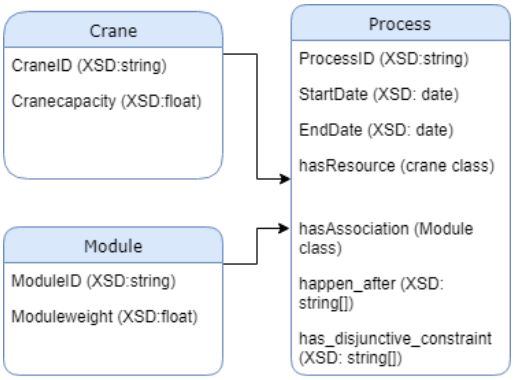

Figure 2: Class diagram for lifting process

```
OntProcess:Process
  rdf:type rdfs:Class ;
  rdf:type sh:NodeShape ;
  rdfs:subClassOf owl:Class ;
  sh:sparql [
     sh:message "Crane capacity should be greater than module weight" ;
     sh:prefixes <http://semanticprocess.x10host.com/Ontology/OntProcess> ;
     sh:select """SELECT $this
WHERE {
     $this  rdf:type OntProcess:Process.
     $this OntProcess:hasResource ?crane.
     $this OntProcess:hasAssociation ?module.
     ?crane OntProcess:Cranecapacity ?cc.
     ?module OntProcess:Moduleweight ?mw.
        FILTER (?cc <= ?mw).
}""" ;
   ]
```

The constraint is applied to the class Process as a node shape as it related to both module and the crane. Therefore, during validation, all the instances of '*OntProcess:Process*' (OntProcess is the prefix defined in the ontology for the IRI), which has crane as a resource and module as an association are checked for this logical constraint. Instances of '*OntProcess:Process*' which do not have crane and module assigned to it will be ignored from this check. Targeting the instances which have assignment is achieved by using SHACL-SPARQL constraint. The SELECT query is used to access the attributes of instances of Crane and Module class. From all the process instances with the module and crane assignment. The *FILTER (?cc <= ?mw)*. statement filters the instances where crane capacity is less than module weight. In the case of violation of the constraint, the compiler would provide a message to the user as defined in the *sh:message* which in the current case is *"Crane capacity should be higher than module weight"*.

Precedence constraint

Precedence constraints may be used in the current problem to define which processes have to be completed so that a new process could take place. Therefore, precedence is an object property defined on the processes. The constraint is mathematically defined as:

$$\textit{Start date of current activity} \leq \textit{End date of preceding activity}$$

```
OntProcess:Process
  rdf:type rdfs:Class ;
  rdf:type sh:NodeShape ;
  rdfs:subClassOf owl:Class ;
sh:sparql [
     sh:message "Precedence condition is violated" ;
     sh:prefixes <http://semanticprocess.x10host.com/Ontology/OntProcess> ;
     sh:select """SELECT $this
WHERE {
$this  rdf:type OntProcess:Process.
$this owl:happen_after ?process2.
$this owl:hasStartDate ?sd1.
          ?process2 owl:hasEndDate ?ed2.
FILTER( ?sd1 <?ed2)
}
```

The above code defines the precedence constraint in SHACL-SPARQL. During validation, the compiler searches for all the instances of class Process with a property happen after as represented in the statement *$this owl:happen_after ?process2* . The start dates of the current process (*$this*) are compared with *?process2* to check whether the precedence constraint is satisfied or

violated. If it is violated, the user receives a message as defined in *sh:message.* All the instances of the class Process without an *owl:happen_after* property is ignored. If there are multiple precedence assignments, all the processes are checked for constraint violations.

Disjunctive constraint

Disjunctive constraints may be defined in this problem setting to make sure that one process shouldn't take place while another process is happening. There are five cases in which this might happen. Consider 2 processes, *Process1* and *Process 2* . The five cases where these two processes overlap are:-

i. *SD1<SD2<ED1<ED2;*
ii. *SD1<SD2<ED2<ED1;*
iii. *SD2<SD1<ED1<ED2; and*
iv. *SD2<SD1<ED2<ED1.*
v. *SD1=SD2 && ED1=ED2*

where *SD1*- start date of *Process 1, SD2* - start date of *Process 2, ED1* – end date of *Process 1* and *ED2* – end date of *Process 2.* This constraint can be represented as the following constraint.

(SD2<ED1 && SD1<ED2)||(SD1=SD2 && ED1=ED2)

This is modelled in the SHACL as follows.

```
OntProcess:Process
  rdf:type rdfs:Class ;
  rdf:type sh:NodeShape ;
  rdfs:subClassOf owl:Class ;
sh:sparql [
     sh:message "Disjunctive constraint is violated " ;
     sh:prefixes
<http://semanticprocess.x10host.com/Ontology/OntProc
ess> ;
     sh:select """SELECT $this
WHERE {
          $this  rdf:type OntProcess:Process.
          $this owl:has_disjunctive_constraint ?process2.
          $this owl:hasEndDate ?ed1.
          $this owl:hasStartDate ?sd1.
          ?process2 owl:hasEndDate ?ed2.
          ?process2 owl:hasStartDate ?sd2.

FILTER((?sd2<?ed1&&?sd1<?ed2)||(?sd1=?sd2      &&
?ed1=?ed2))
}""" ;
    ]
```

As given in the previous definitions, the constraint is modelled as a node shape on the Process class. The processes which are disjunctive to the current is assigned to the current process using the *has_disjunctive_constraint* property. All the instances of this process with this assignment are checked for the violation using the above constraint representation inside the SPARQL query as shown in (*FILTER((?sd2<?ed1&&?sd1 <?ed2)||(?sd1=?sd2 && ?ed1=?ed2)))*. As before, the instances without this assignment are ignored.

Discrete constraint

Discrete constraints are required in this problems setting to ensure resources (cranes) are not assigned to two processes at the same time. Since this is a constraint related to the resource, the constraint is modelled as a node shape on the class *Crane* as given below.

Pairwise comparison feature of SPARQL is used to model this constraint. As a first step, all the processes which are assigned the particular resource is shortlisted into pairs in the statements:

?process1 OntProcess:hasResource $this.
?process2 OntProcess:hasResource $this.

From this list of pairs, identical processes are removed using *FILTER (!(?process1=?process2)).* Then the data properties start date and end date for each process in the pair is retrieved and checked for the disjunctive constraint in the statement *FILTER((?sd2<?ed1&&?sd1<?ed2)||(?sd1=?sd2 && ?ed1=?ed2)).*

The discrete constraint is necessary for all the resources. In case, the resources can have multiple assignments; this can also be defined in this constraint using SPARQL COUNT query.

```
OntProcess:Crane
  rdf:type rdfs:Class ;
  rdf:type sh:NodeShape ;
  rdfs:subClassOf owl:Class ;
  sh:sparql [
     sh:message "Discrete constraint for the resource not
satisfied" ;
     sh:prefixes
<http://semanticprocess.x10host.com/Ontology/OntProc
ess> ;
     sh:select """SELECT $this
WHERE {
           $this rdf:type OntProcess:Crane .
           ?process1 rdf:type OntProcess:Process.
           ?process2 rdf:type OntProcess:Process.
           ?process1 OntProcess:hasResource  $this.
           ?process2 OntProcess:hasResource  $this.
           ?process1 owl:hasEndDate ?ed1.
           ?process1 owl:hasStartDate ?sd1.
           ?process2 owl:hasEndDate ?ed2.
           ?process2 owl:hasStartDate ?sd2.

           FILTER (!(?process1=?process2))
           FILTER((?sd2<?ed1&&?sd1<?ed2)||(?sd1=?sd
2 && ?ed1=?ed2) )
}""" ;
    ] ;
.
```

Table 1: Instances of the process class

Process	Crane	Crane capacity (metric ton)	Module	Module Weight (metric ton)	Start date	End date	Precedence Constraint	Disjunctive constraint
Process_1	Crane_1	12	Module_1	15	01/01/2019	02/01/2019		
Process_2	Crane_2	18	Module_2	15	02/01/2019	03/01/2019	Process 1	
Process_3	Crane_2	18	Module_3	15	01/01/2019	05/01/2019	Process 1	Process 1,Process 2

Implementation

The above process was implemented using open source Python Libraries RDFLib and pySHACL in a Jupyter Notebook environment. The ontology for the process was developed using Protégé(Musen, 2015). The instances used for testing are shown in Table 1. The results of the same are presented in the next section.

Results and Discussion

The results after the validation of the data graph against shapes graph are given in Table 2. From the results in Table 2, we can identify that there are constraint violations in Process 1, Process 3 and Crane 2.

In *Process_1*, the logical constraint was violated, and the message *"Crane capacity should be greater than module weight"* was received as an output. This is an actual violation as *Crane_1* with a capacity of 12 metric ton was assigned to *Process_1* to lift *Module_1* which has a mass of 15 metric ton.

In *Process_3,* there are two violations which were reported in the messages "*Disjunctive constraint is violated"* and *"Precedence condition is violated". Process_3* was assigned a disjunctive constraint with *Process_1* and *Process_2.* However, *Process_3* starts with *Process_1* and ends after *Process_2* which is a clear violation of the disjunctive constraint. In addition, *Process_3* was assigned a precedence constraint with *Process_1*. This is again violated as the start times of both processes are the same.

For *Crane_2, "Discrete constraint for the resource not satisfied"* was received as an output. This is true as the *Crane_1* was assigned to two processes *Process_2* and *Process_3* at the same time.

Hence from the above, we can conclude that the modelled constraints were able to detect the violations effectively. Although the model represented here is very simplistic, and the violations could be easily detected without any modelling, the same method can be applied to more massive process datasets where it would be tedious to identify the violations. The data (RDF containing instances) and the shapes (RDF containing constraints) are different components linked through the ontology of the lifting process. More constraints can be added dynamically to the shapes file, and the validation is done for the whole data. This is an excellent advantage in construction to incorporate new constraints as they arise.

Also, as the constraints are coded in a machine-readable format (RDF), it is possible to do analytics and machine learning on this model too and derive insights for future automation of constraint modelling.

Conclusion

This paper describes a method to define construction scheduling constraints using Shapes Constraint Language. Scheduling constraints such as Precedence constraints, discrete resource capacity constraints, disjunctive constraints and logical constraints were modelled using shapes constraint language for a simple lifting problem in this paper. The modelled constraints were tested against a data set of lifting processes, and the violation constraints model was able to identify the violations effectively and produce a validation report.

This paper demonstrated the capability of linked-data approach towards constraint validation for construction scheduling problem. The described methodology could be applied to a broader set of data to produce similar results. However, that is beyond the scope of this paper. Authors are exploring methods to make the modelling of constraints user-friendly by creating an intuitive UI. Also, research is being done towards integration of this method to IFC OWL for automating look ahead planning.

Table 2: Output after constraint validation

Process	Output
Process 1	Constraint Violation in SPARQLConstraintComponent (http://www.w3.org/ns/shacl#SPARQLConstraintComponent): Severity: sh:Violation Source Shape: OntProcess:Process Focus Node: OntProcess:Process_1 Value Node: OntProcess:Process_1 Source Constraint: [sh:message Literal("Crane capacity should be greater than module weight") ; sh:prefixes <http://semanticprocess.x10host.com/Ontology/OntProcess> ; sh:select Literal("SELECT $this WHERE { $this rdf:type OntProcess:Process. $this OntProcess:hasResource ?crane. $this OntProcess:hasAssociation ?module. ?crane OntProcess:Cranecapacity ?cc. ?module OntProcess:Moduleweight ?mw. FILTER (?cc <= ?mw).}")] Message: **Crane capacity should be greater than module weight**
Process 2	No Violation
Process 3	Constraint Violation in SPARQLConstraintComponent (http://www.w3.org/ns/shacl#SPARQLConstraintComponent): Severity: sh:Violation Source Shape: OntProcess:Process Focus Node: OntProcess:Process_3 Value Node: OntProcess:Process_3 Source Constraint: [sh:message Literal("Precedence condition is violated") ; sh:prefixes <http://semanticprocess.x10host.com/Ontology/OntProcess> ; sh:select Literal("SELECT $this WHERE { $this rdf:type OntProcess:Process. $this owl:happen_after ?process2. $this owl:hasEndDate ?ed1. $this owl:hasStartDate ?sd1. ?process2 owl:hasEndDate ?ed2. ?process2 owl:hasStartDate ?sd2. FILTER(?sd1 <?ed2) }")] **Message: Precedence condition is violated** Constraint Violation in SPARQLConstraintComponent (http://www.w3.org/ns/shacl#SPARQLConstraintComponent): Severity: sh:Violation Source Shape: OntProcess:Process Focus Node: OntProcess:Process_3 Value Node: OntProcess:Process_3 Source Constraint: [sh:message Literal("Disjunctive constraint is violated ") ; sh:prefixes <http://semanticprocess.x10host.com/Ontology/OntProcess> ; sh:select Literal("SELECT $this WHERE { $this rdf:type OntProcess:Process. $this owl:has_disjunctive_constraint ?process2. $this owl:hasEndDate ?ed1. $this owl:hasStartDate ?sd1. ?process2 owl:hasEndDate ?ed2. ?process2 owl:hasStartDate ?sd2. FILTER((?sd2<?ed1&&?sd1<?ed2)||(?sd1=?sd2 && ?ed1=?ed2)) }")] **Message: Disjunctive constraint is violated**
Crane 1	No violation
Crane 2	Constraint Violation in SPARQLConstraintComponent (http://www.w3.org/ns/shacl#SPARQLConstraintComponent): Severity: sh:Violation Source Shape: OntProcess:Crane Focus Node: OntProcess:Crane_2

	Value Node: OntProcess:Crane_2 Source Constraint: [sh:message Literal("Discrete constrsint for the resource not satisfied") ; sh:prefixes <http://semanticprocess.x10host.com/Ontology/OntProcess> ; sh:select Literal("SELECT $this WHERE { $this rdf:type OntProcess:Crane . ?process1 rdf:type OntProcess:Process. ?process2 rdf:type OntProcess:Process. ?process1 OntProcess:hasResource $this. ?process2 OntProcess:hasResource $this. ?process1 owl:hasEndDate ?ed1. ?process1 owl:hasStartDate ?sd1. ?process2 owl:hasEndDate ?ed2. ?process2 owl:hasStartDate ?sd2. FILTER (!(?process1=?process2)) FILTER((?sd2<?ed1&&?sd1<?ed2)||(?sd1=?sd2 && ?ed1=?ed2)) }")] **Message: Discrete constraint for the resource not satisfied**

Acknowledgements

The PhD research of the author is co-funded by Bentley systems UK and, through a Skempton Scholarship, the Department of Civil and Environmental Engineering, Imperial College London. The author acknowledges the supervisory inputs offered by Prof Jennifer Whyte and Dr David Birch; and also the guidance offered by Prof. Jakob Beetz.

References

Anumba, C. J., Bouchlaghem, N. M., Whyte, J., & Duke, A. (2000). Perspectives on an integrated construction project model. *International Journal of Cooperative Information Systems*, 09(03), pp.283–313.

Baader, Franz, Diego Calvanese, Deborah McGuinness, Peter Patel-Schneider, and Daniele Nardi. (2003) eds. *The description logic handbook: Theory, implementation and applications*. Cambridge university press

Baykan, C. A., and Fox, M. S. (1997). Spatial synthesis by disjunctive constraint satisfaction. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 11(04), pp. 245–262.

Beetz, J., Van Leeuwen, J., and De Vries, B. (2009). IfcOWL: A case of transforming EXPRESS schemas into ontologies. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*:, 23(1),pp. 89–101.

Berners-Lee, T. (2006). *Linked Data - Design Issues*, Available at : https://www.w3.org/DesignIssues/LinkedData.html (Accessed January 14, 2019)

Čuš-Babič, N., Rebolj, D., Nekrep-Perc, M., and Podbreznik, P. (2014). Supply-chain transparency within industrialized construction projects. *Computers in Industry*, 65(2), pp.345–353.

Darwiche, A., Levitt, R. E., and Hayes-Roth, B. (1988). OARPLAN: Generating project plans by reasoning about objects, actions and resources. *Artificial Intelligence for Engineering, Design, Analysis and Manufacturing*, 2(3), pp.169–181.

Dave, B., Kubler, S., Främling, K., and Koskela, L. (2016). Opportunities for enhanced lean construction management using Internet of Things standards. *Automation in Construction*, 61, pp.86–97.

Dechter, R. and Cohen,D. (2003). *Constraint processing*. Morgan Kaufmann. Sanfrancisco, US.

Dong, N., Fischer, M., Haddad, Z., and Levitt, R. (2013). A method to automate look-ahead schedule (LAS) generation for the finishing phase of construction projects. *Automation in Construction*, 35, pp.157–173.

Ekaputra, F. J., and Xiashuo Lin. (2016). SHACL4P: SHACL constraints validation within Protégé ontology editor. In Proceedings of the *2016 International Conference on Data and Software Engineering (ICoDSE)* . Bali, Indonesia

El-Rayes, K., and Jun, D. H. (2009). Optimizing resource leveling in construction projects. *Journal of Construction Engineering and Management*, 135(11), pp.1172–1180.

Gayo, J. E. L., Prud'hommeaux, E., Boneva, I., and Kontokostas, D. (2017). *Validating RDF Data*. Synthesis Lectures on the Semantic Web: Theory and Technology, 7(1), pp.1–328.

Giretti, A., Carbonari, A., Novembri, G., and Robuffo, F. (2012). Estimation of job-site work progress through on-site monitoring. In Proceedings of the *29th International Symposium of Automation and Robotics in Construction, ISARC 2012*, Kochi,India

Han, K. K., and Golparvar-Fard, M. (2017). Potential of big visual data and building information modeling for construction performance analytics: An exploratory study. *Automation in Construction*, 73,

pp.184–198.

Harris, S., and Seaborne, A. Prud'hommeaux, E. (2013). *SPARQL 1.1 query language*. Available at https://www.w3.org/TR/sparql11-query/ (Accessed: 6 June 2019)

Hu, J., and Flood, I. (2012). A Multi-Objective Scheduling Model for Solving the Resource-Constrained Project Scheduling and Resource Leveling Problems. In Proceedings of the *Computing in Civil Engineering (2012)* Reston, USA

Huhnt, W., Richter, S., Wallner, S., Habashi, T., and Krämer, T. (2010). Data management for animation of construction processes. *Advanced Engineering Informatics*, 24(4), pp.404–416

Kim, K., Kim, H., Kim, W., Kim, C., Kim, J., and Yu, J. (2018). Integration of ifc objects and facility management work information using Semantic Web. *Automation in Construction*, 87, pp.173–187.

Knublauch, H., Allemang, D., and Steyskal, S. (2017). SHACL Advanced Features. Available at ss

Lee, D.-Y., Chi, H., Wang, J., Wang, X., and Park, C.-S. (2016). A linked data system framework for sharing construction defect information using ontologies and BIM environments. *Automation in Construction*, 68, pp.102–113.

Lin, J. J., and Golparvar-Fard, M. (2016). Web-Based 4D Visual Production Models for Decentralized Work Tracking and Information Communication on Construction Sites. In Proceedings of the *2016 Construction Research Congress*, pp. 1731–1741.

Liu, H., Lu, M., and Al-Hussein, M. (2016). Ontology-based semantic approach for construction-oriented quantity take-off from BIM models in the light-frame building industry. *Advanced Engineering Informatics*, 30(2), pp.190–207.

Morkos, R. (2014*). Operational efficiency frontier: Visualizing, manipulating, and navigating the construction scheduling state space with precedence, discrete, and disjunctive constraints*. Thesis. Stanford University.

Musen, M. A. (2015). The protégé project: A Look Back and a Look Forward. *AI Matters*, 1(4), pp.4–12.

Niederliński, A. (2011). *A Quick and Gentle Guide to Constraint Logic Programming Via ECLiPSe*.3rd ed. Gliwice :Jacek Skalmierski Computer Studio

Pauwels, P., de Farias, T. M., Zhang, C., Roxin, A., Beetz, J., De Roo, J., and Nicolle, C. (2017). A performance benchmark over semantic rule checking approaches in construction industry. *Advanced Engineering Informatics*, 33, pp.68–88.

Pauwels, P., and Terkaj, W. (2016). EXPRESS to OWL for construction industry: Towards a recommendable and usable ifcOWL ontology. *Automation in Construction*, 63, pp.100–133.

Pauwels, P., Van Deursen, D., Verstraeten, R., De Roo, J., De Meyer, R., Van De Walle, R., and Van Campenhout, J. (2011). A semantic rule checking environment for building performance checking. *Automation in Construction*, 20(5), pp.506–518.

Quattrini, R., Pierdicca, R., and Morbidoni, C. (2017). Knowledge-based data enrichment for HBIM: Exploring high-quality models using the semantic-web. *Journal of Cultural Heritage*, 28, pp.129–139.

Subramanian, P. S., Songer, A. D., and Diekmann, J. E. (2000). Visualizing Process Performance. In Proceedings of *Computing in Civil and Building Engineering*. Reston, USA1

Terkaj, W., Schneider, G. F., and Pauwels, P. (2017). Reusing Domain Ontologies in Linked Building Data : the Case of Building Automation and Control. Proceedings of the *8th Workshop Formal Ontologies Meet Industry, Joint Ontology Workshops 2017, CEUR Workshop Proceedings,* Bolazano-Bozen,Italy .

Whyte, J., Stasis, A., and Lindkvist, C. (2016). Managing change in the delivery of complex projects: Configuration management, asset information and "big data." *International Journal of Project Management*, 34(2), pp.339–351.

Zhang, C., and Beetz, J. (2015). Model Checking on the Semantic Web: IFC Validation Using Modularized and Distributed Constraints.In: Proceedings of the *32nd CIB W78 Conference* Eindhoven, The Netherlands,.

Zhang, C., and Beetz, J. (2016). Querying Linked Building Data Using SPARQL with Functional Extensions. In: Proceedings of *11th European Conference on Product and Process Modelling,* Limassol,Cyprussss