Automation in Construction 120 (2020) 103369

Contents lists available at ScienceDirect

Automation in Construction

journal homepage: www.elsevier.com/locate/autcon

Linked-Data based Constraint-Checking (LDCC) to support look-ahead planning in construction

Ranjith K. Soman[a,b,*], Miguel Molina-Solana[c,d], Jennifer K. Whyte[a,b]

[a] Centre for Systems Engineering and Innovation, Imperial College London, United Kingdom
[b] The Alan Turing Institute, London, United Kingdom
[c] Data Science Institute, Imperial College London, United Kingdom
[d] Department of Computer Science and Artificial Intelligence, Universidad de Granada, Spain

ARTICLE INFO

Keywords:
Linked-data
Shapes constraint language (SHACL)
Constraints
Dynamic constraint-checking
Scheduling constraints
Linked-data based Constraint-Checking (LDCC)

ABSTRACT

In the construction sector, complex constraints are not usually modeled in conventional scheduling and 4D building information modeling software, as they are highly dynamic and span multiple domains. The lack of embedded constraint relationships in such software means that, as Automated Data Collection (ADC) technologies become used, it cannot automatically deduce the effect of deviations to schedule. This paper presents a novel method, using semantic web technologies, to model and validate complex scheduling constraints. It presents a Linked-Data based Constraint-Checking (LDCC) approach, using the Shapes Constraint Language (SHACL). A prototype web application is developed using this approach and evaluated using an OpenBIM dataset. Results demonstrate the potential of LDCC to check for constraint violation in distributed construction data. This novel method (LDCC) and its first prototype is a contribution that can be extended in future research in linked-data, BIM based rule-checking, lean construction and ADC.

1. Introduction

There is great potential to develop data-driven approaches to support look-ahead planning in construction, but these approaches require constraints to be codified into a machine-readable format. Look-ahead planning is a crucial phase in construction planning and control where the construction program for the next six weeks is developed [1]. In look-ahead planning, it is important to ensure that all prerequisites are complete and that constraints violations are removed for each work package before it is released to weekly work plans [1]. When look-ahead planning is done manually for complex projects, it is difficult to track and monitor the prerequisites and identify constraint violations [2,3]. Automated Data Collection (ADC) technologies, such as Global Positioning Systems (GPS) [4,5], Radio Frequency Identification (RFID) [6,7], computer vision [8,9], and other sensor-based data collection methods [10,11], help to automate tracking and monitoring prerequisites for work packages. Prerequisites are the conditions necessary for a work package to be executed such as availability of resource and completion of prior tasks. Constraint violations happen when the prerequisites are not complete. Although there has been research focusing on automation of prerequisite tracking, research concerning the development of data-driven methods to identify constraint violations have received insufficient attention.

The knowledge about construction processes is predominantly codified as precedence relationships in the form of master plans and phase plans in 4D Building Information Model (BIM) and other scheduling software systems. However, it is often the case that the model breakdown structures in these knowledge bases do not match the work breakdown structures for construction site activities [12] as they only contain a subset of the relationships between activities and resources [13]. Also, construction scheduling constraints such as disjunctive constraints and discrete resource capacity constraints [14,15], are not codified in these knowledge bases. In addition, detailed construction information (such as look-ahead plans and weekly plans) are not at all incorporated into these systems but communicated to the relevant personnel in the form of method statements [16].

The combination of the sensor data from ADC and the semantic relationships of the activities (including constraints) embedded in the information models enable data-driven constraint checks to ensure that work packages are ready for construction. However, complex construction constraints are not usually modeled in conventional information models because they are highly dynamic and span multiple domains. And thus, despite having the potential to support look-ahead planning in construction, the current lack of detail and semantics in the

* Corresponding author at: Centre for Systems Engineering and Innovation, Imperial College London, United Kingdom.
E-mail address: ranjithks17@imperial.ac.uk (R.K. Soman).

https://doi.org/10.1016/j.autcon.2020.103369
Received 17 November 2019; Received in revised form 14 July 2020; Accepted 16 July 2020
0926-5805/ © 2020 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/BY-NC-ND/4.0/).

information models limits the application of data-driven decision support systems in this area.

There is an emerging research trajectory on codifying this detailed construction process information and then using automated methods to generate construction programs [17–19]. Although these methods present a robust framework for the codification of construction process information, they do not address inherent challenges related to the interoperability associated with construction datasets. The first challenge is that the required data remain distributed in multiple heterogeneous datasets [16,20,21] over multiple servers, creating problems of interoperability. The second challenge is that the construction constraints are dynamic and span multiple domains [22], requiring a constraint evaluation method that could traverse through heterogeneous data stored on different servers and domains to check for constraint violations. These challenges limit the codification of detailed construction process information and semantics into a machine-readable form, thereby limiting the opportunities offered by data-driven decision support systems, such as BIM based rule-checking systems, to assist look-ahead planning.

This paper aims to develop a linked-data based constraint-checking (LDCC) method to define and check complex and dynamic scheduling constraints in construction. Linked-data is a set of design principles for publishing data in a structured format so that the data can be connected and then interpreted by machines [23]. Linked-data based methods are beginning to be used to address problems associated with interoperability issues in construction datasets [24]. Publishing the construction process information following linked-data principles as a Resource Description Framework (RDF) graph ensures semantic interoperability. This can be achieved using ontologies to define taxonomies and services to enable sharing and reusing domain knowledge [25,26]. In this paper, construction scheduling constraints are modeled to validate and detect violations in construction process information in the form of RDF graphs using a data modeling language called the Shapes Constraint Language (SHACL) [27]. The proposed method ensures that semantic interoperability is retained, enabling constraint-checking systems to infer from multiple databases without human intervention [28]. The novelty of this paper lies in addressing the stated challenges by leveraging the opportunities offered by linked-data to develop a dynamic constraint-checking method to assist look-ahead planning. The LDCC method developed in this paper is of value to researchers working on the lean construction, BIM-based rule-checking and linked-data, as it demonstrates how BIM based rule-checking can be used in combination with linked-data to develop decision-support for look-ahead planning.

The rest of the paper is organized as follows. Section 2 explains the prior work on BIM based rule-checking and introduces the linked-data applications to address data siloes, on which the current research is based. Section 3 outlines LDCC, our proposal that combines a linked-data based approach using SHACL with BIM based rule-checking to model and validate cardinality, logical, precedence, disjunctive and discrete scheduling constraints. Section 4 presents a prototype web application implementing this LDCC method, covering the user interface, processing web application, storage and data. Section 5 discusses the results of evaluating it using an OpenBIM dataset and how the LDCC method provides an important first step towards linked-data based rule-checking for construction scheduling constraints by checking for construction constraint violations. Section 6 presents the conclusions and suggests future research directions to extend the LDCC method through work on linked-data, data science, lean construction and ADC.

2. Related work

To efficiently develop the construction program in look-ahead planning, the activities that are planned for the next six weeks are identified from the master plan, followed by identification of constraint violations for the identified activities and removing them to make sure they are ready. The activities whose constraints violations are completely removed are then scheduled for the next six weeks [29]. Look-ahead planning is often done manually [30]. This results in errors in identification and removal of constraint violations [31], thus resulting in scheduling tasks, that are not ready to be executed [3]. To support look-ahead planning, researchers have used BIM to visualize the status of the different work packages [32], improve collaboration among project participants focusing on communicating constraints [33] and generate work-orders [34]. Although, there is further potential for automating the identification of constraint violation using BIM based rule-checking when detailed construction information is stored in BIM, this is a research area that has received insufficient attention.

This section discusses the prior research on BIM based rule-checking and linked-data applications to provide a background for the development of LDCC method. Section 2.1 introduces the concepts of BIM based rule-checking and discusses prior work on automating rule-checking on BIM models. However, these studies have not addressed the challenge of connecting across diverse sources of data in the construction sector while performing BIM based rule-checking. Section 2.2 describes how this challenge has been studied in detail by researchers working on linked-data, and presents SHACL, a linked-data based data constraint modeling language.

2.1. BIM based rule-checking systems

A rule-checking system used to define, deploy, execute, and monitor the results of the rule-checking process is known as a Business Rule Management System (BRMS) [35]. Knowledge in a BRMS is separated into logic (the rules that need to be satisfied) and data (to define the particular model to be checked). Data refer to the particular model that needs to be checked against the rules defined in the logic. Data are case-specific, whereas logic is generalizable and can be used across cases. Therefore, by separating logic from data, it is possible to manipulate both independently, as people who model the data may or may not be the ones who define the rules [36].

To integrate the BRMS concepts with BIM to perform rule-checking on BIM, Eastman et al. [37] provides four functionalities to perform BIM based rule-checking. These are: 1) *Rule interpretation:* Textual rules are converted into a machine-interpretable format; 2) *Building model preparation:* Information in the form of models is converted into a format required for the execution of the rules; 3) *Rule execution:* The prepared model is checked against the defined machine-interpretable rules; and 4) *Reporting:* The results of the rule-checking process are communicated to the user or raised to another system. These functionalities need to be present in the constraint checking method proposed in this paper.

Each of these functionalities may be fully automated or semi-automated to improve speed and reduce errors. Solihin et al. [38] described methods to automate each of the four steps. To automate rule interpretation, they suggested distilling the rule content from written text using tools (e.g. LegalRuleML [39], RASE (requirement, applicability, selection, and exception) markup [40]) and transcribing the rule content into a rule language (e.g. Drools [41]). To automate building model preparation, existing data models are transformed into the required knowledge representation using converters (e.g. conversion of Industry Foundation Classes (IFC) to linked-data formats that extend that functionality using ifcOWL to process rules in the Semantic Web Rule Language (SWRL) [42]). Rule execution is automated using BRMS. The results of rule-checking are taken from the BRMS and reported to the user after analysis.

Depending on how inferences are made from data, BIM based rule-checking systems can be classified into four classes according to its complexity [43]. The complexity is regarded as Class 1 when the rule-checking system queries entities and attributes against a value. For Class 2 complexity, the rule-checking system need to calculate/derive additional values for rule execution. In rule-checking systems classified as Class 3, the data structure of the information model needs to be

extended to perform rule execution. Finally, new model objects are created from the existing data sets to perform rule execution in rule-checking systems with Class 4 complexity. Thus, to perform constraint checking on distributed construction data, a rule-checking system with a complexity of Class 3 is needed as the data structure needs to be extended to link data distributed in multiple servers.

There is increasing interest in BIM based rule-checking to increase model quality and to provide decision support in the construction sector. Existing proposals use different methods, such as using rule engines [44], query languages [45], rule languages [46] and machine learning methods [47] for BIM based rule-checking. Prior research in this area has extended knowledge on using BIM based rule-checking for ensuring code compliance [48], construction safety [49], and for aiding design verification [46] and construction site planning [50]. These studies have assumed that relevant information is available as a BIM model in a single repository. However, this assumption may not hold true in the case of construction information. Therefore, the BIM based rule-checking methods focusing in the construction stage should be able to check for violation in data distributed over multiple databases. To address the issue of construction information siloes, attention is now turned to the prior research on the use of linked-data to model the knowledge required for rule-checking by exploiting the potential of linked-data to link across domains to create semantic information models [51,52].

2.2. Linked-data and constraint-checking

What is the relevance of linked-data based constraint-checking in look-ahead planning? Construction information remains fragmented across multiple silos, restricting the application of BIM based rule-checking tools [22,51,53]. In the case of look-ahead planning, construction constraints are complex and require information from heterogeneous sources. Consider the case of modeling a constraint for a lifting activity, which comprises of a module being delivered to a construction site by a truck. The module is then stored in a laydown area and finally lifted by a crane and installed at the designated location. Despite being a simple process, there is a need to integrate information from multiple sources to perform constraint-checking for this process. Crane parameters are obtained from crane subcontractors, truck parameters are obtained from truck subcontractors, and site conditions are obtained from the Building Information Model (BIM) residing in the main contractor server. All this information is required to check whether a crane is available for lifting the module, whether a laydown area is available to store the module, and whether preceding activities at the installation location are complete. Hence, the constraint-checking system should have the capability to traverse over data distributed over multiple subcontractor servers and make logical inferences to determine whether the modeled activity could be performed. Linked-data technologies offer opportunities to link across heterogeneous datasets to perform checks on the data.

Pauwels et al. [24] surveyed the application of linked-data in the AEC domain and found that researchers have used linked-data to improve interoperability, to link across domains, and to generate logical proofs and inference, thus demonstrating the use of linked-data for connecting construction information. The main principles of publishing information using linked-data are to 1) use Uniform Resource Identifiers (URI) to name the resources (both concepts and properties), 2) use Hypertext Transfer Protocol (HTTP) URI so that the modeled resources can be accessed using the internet, 3) provide more information about the resources using standards (RDF and SPARQL) when the URIs are looked up, and 4) provide links to the related URIs so that the resource can act as a starting point to explore the related data spaces [54]. These

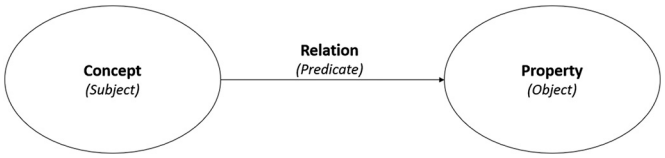


Fig. 1. RDF triple.

principles enable linked-data technologies to link and share data across heterogeneous sources without being limited by the scope of underlying schemas of the source data [55]. Researchers have attempted to use linked-data to address the limitations of BIM related to structuring and querying product information [55–57], facility management [58,59], and solving interoperability [60]. Pauwels et al. [61] combined IFC and semantic web technologies to develop a semantic rule-checking environment for BIM.

Following linked-data terminology, information artifacts are modeled as “concepts” and are related to associated “properties” in the form of a “statement.” Each statement has three resources (subject, predicate, and object) modeled as a directed labeled graph (See Fig. 1). The statement is called an RDF triple. The graph structure formed by applying the logical AND operator to a series of RDF triples is referred to as an RDF graph [23]. The semantics of an RDF graph can be further improved using RDF vocabularies and ontologies. The RDF schema vocabulary is one such vocabulary that enables the specification of classes, subclasses, data types, labels, etc. [62]. OWL is a more expressive way to improve the semantic structure of an RDF graph [63] and has RDF concepts as a subset. The graphs constructed using OWL are termed OWL ontologies, which can be used as vocabulary when constructing more complex RDF statements. Thus, RDF resources can be defined under a specific class in an OWL ontology. This ensures that new information can be inferred through a reasoning process.

To perform complex reasoning on this linked-data, existing rules languages can be used, or a dedicated rule language can be built to define custom rules and use them in a rule-based reasoning process. Some of the rule languages that are already available are SWRL [42], RIF [64], and the Notation3Logic language [65]. These languages facilitate rule-based reasoning.

Related to construction, Gómez et al. [66] explored the augmentation of semantic BIMs with capabilities to manage imprecision and vagueness. Through fuzzy ontologies and reasoning they proposed solutions to several AEC tasks: cross-domain knowledge linking (e.g. with partial concept inclusions and graded relationships), imprecise BIM queries (e.g. by using linguistic labels and imprecise topological relations) and fuzzy parametric modeling (e.g. by means of fuzzy axioms and maximization of their degree of fulfillment).

The focus of the current work is on consistency checking and not so much in other aspects of reasoning, such as deriving new knowledge. Specifically, we aim to identify constraint violations in construction schedules and thus validate the data about a construction process (modeled as an RDF graph) against a set of constraints. The SPARQL Inferencing Notation (SPIN) [67] is a SPARQL-based constraint language that offers a way to perform constraint-checking with closed-world semantics, leveraging the power of the SPARQL ASK and SPARQL CONSTRUCT queries. It links class definitions with SPARQL queries to capture constraints that formalize the behavior of the classes. In addition, SHACL, a data modeling language to model such constraints recommended by W3C [68], is a successor to SPIN. Like SPIN, SHACL is a language for validating RDF graphs against a set of conditions [68]. However, SHACL offers greater flexibility than SPIN in terms of defining the targets for constraints. While SPIN is limited to classes,

SHACL can be applied to classes or a set of nodes defined by different target mechanisms, including custom targets using SHACL advanced features.

The use of SHACL to define construction scheduling constraints means that a dedicated constraint-checking language does not need to be developed from scratch. Validating an RDF model with a set of constraints using SHACL addresses the inherent issue of the open-world assumption in RDF graphs, as SHACL offers the opportunity to define structural constraints [69]. Moreover, SHACL has two components:

- *Data graph*: contains the data to be validated.
- *Shape graph:* contains the constraints against which resources in the data graph are validated (called shapes).

This follows the BRMS principle of dividing knowledge into logic and data. Data graphs and shape graphs may be coded as a single RDF graph or multiple RDF graphs because the triples in the data graph are validated against triples in the shape graph. There are two types of shapes in SHACL: the node shape and property shape.

- *Node shapes* are constraints that act on subject resources (or concept instances) of a specific type (the type can be defined while modeling the constraints) in the data graph.
- *Property shapes* are constraints that act on the predicate resources (or attribute instances). SHACL SPARQL is an advanced feature of SHACL that contains all the features mentioned above and offers the expressive power of SPARQL to target specific instances of a subject resource satisfying certain conditions (flexible in defining the conditions) [70].

When a data graph is validated against a shape graph, the resources in the data graph are checked against the applicable shapes in the shape graph. Then, a validation report (in the form of another RDF graph) is produced, which details the resources that have passed or violated the constraints.

The SHACL-based constraint-checking is ideal for validating construction scheduling constraints (which are distributed across multiple servers), as SHACL uses the power of linked-data to traverse across constraints distributed across heterogeneous data sources and to validate them.

3. Linked-data based construction constraint-checking

To support the modeling of constraints spanning domains, this paper puts forward LDCC, our proposal for linked-data constraint-checking in construction. It addresses the problem of data fragmentation by linking across domains to check against a set of constraints, defined in the SHACL language. In addition to describing the architecture to perform model-based checking for construction process information, we developed a web application implementing LDCC and validated it using a public dataset (Schependomlaan Open BIM dataset [71]), enriched with supporting data. This section presents the architecture and ontology used for LDCC and how construction scheduling constraints are modeled in SHACL. Section 4 describes the prototype implementation and testing.

3.1. Linked-data based constraint-checking architecture

The architecture for checking the dynamic constraints is shown in Fig. 2. The LDCC method is intended to be used at look-ahead planning meetings to aid decision making. Detailed construction information shared during look-ahead planning meetings (including subtasks and constraints) will be the input to LDCC, linked to the 4D BIM, modeled as an RDF graph, and checked for constraint violations using the SHACL processing engine. This method for identifying constraint violation acts as a decision support tool during look-ahead planning meetings, helping the engineers to make decisions during these meetings. To provide decision support functionalities, there are three layers in this architecture:

1 User interface (UI) layer: This layer is for interaction with the user. In an effort to make the tool work along with the existing work practices [72], we designed the UI layer as an increment to a current 4D BIM UI with three major inputs from the user. The first input is a BIM model with embedded planning information (planning data at the master planning/phase planning level). The second input contains information required to enrich the planning data (including subtasks for main tasks/activities in master planning, renewable and nonrenewable resources needed for executing the tasks, etc.) The third user input includes information on constraints and the relationship between activities, subtasks, resources, approvals, etc. The output from this layer is the feasibility report of activities scheduled using the information from the input layer.
2 Processing layer: This layer prepares the data for processing, processes it and generates a validation report following three steps. The first step involves extracting the planning data from the BIM model (using a parser and then converting it to an RDF graph) and enriching it with the detailed data input from the user (adding triples related to subtasks and resources). This enrichment should include describing the data based on a suitable ontology automatically, such as LinkOnt (explained in Subsection 3.2) as in this paper. This step results in the generation of a data graph in RDF for the look-ahead window. The second step involves creating a shape graph from the constraint information taken as input from the user. This step also imports shapes generated in previous meetings into the shapes graph. The shape graph is generated using SHACL vocabulary. The third step involves validating the data graph against the shape graph to identify constraint violations. The resources in the data graph containing planning information from the BIM and information added from the user during the look-ahead planning meetings are validated against corresponding shapes (constraints) in the shape graph. A validation report is generated as an output of this step.
3 Ontology layer: This layer holds the schema for different data (e.g., the BIM data, resource data, or SHACL). The schemas are stored in a repository on the web and published as per the principles stated by Berners-Lee [54]. As this layer is accessed during the execution phase, the latest updates to the ontologies and constraint relationships introduce dynamicity in the LDCC architecture. Schemas are referenced and not hardcoded.

This architecture enables the separation between the data graph and shape graph, supporting re-evaluation with new information. In any case, constraints are not hardcoded into the system but are modeled and edited during look-ahead planning meetings. Modeled constraints apply to the data in the data graph, enabling the capture of knowledge generated during meetings to be codified into a machine-readable format and reused. For example, if constraints (e.g., lifting capacity) related to a resource (e.g., crane) for an activity (e.g. lifting of modules) are discussed in a look-ahead planning meeting and codified as a constraint using the LDCC prototype, a SHACL shape is generated for the resource (e.g., a SHACL shape containing information for the lifting operation, such as *lifting capacity* < *weight of the module*). Because the shape is generated for this resource, the resource is validated against that shape each time the resource is used. Hence, if the same resource is assigned

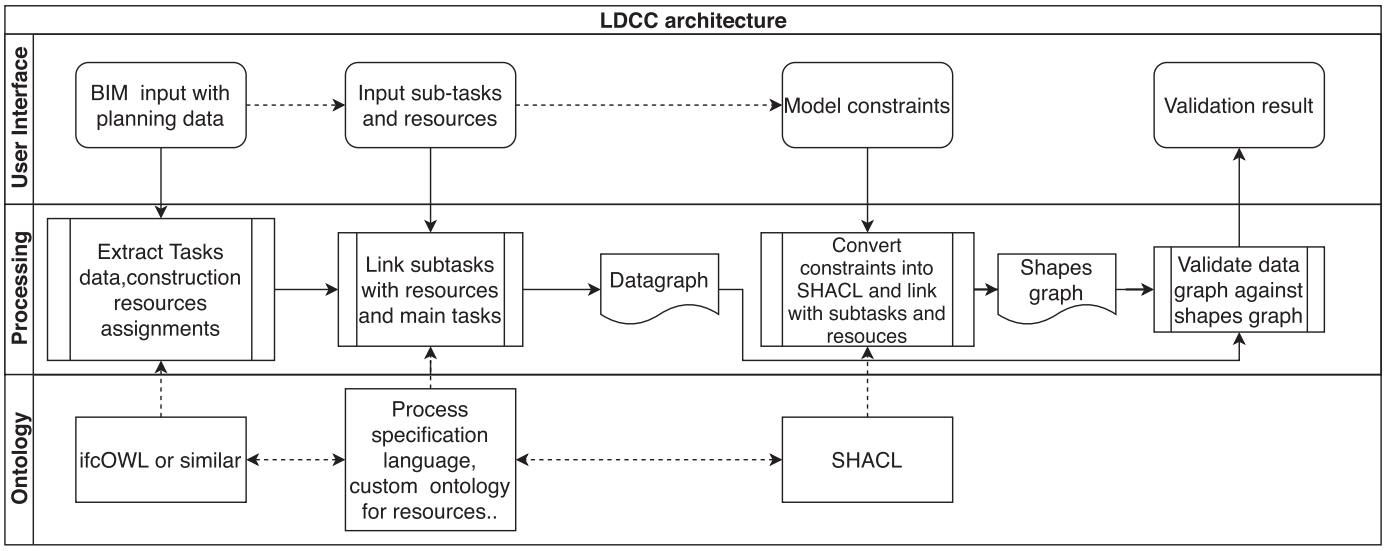


Fig. 2. LDCC architecture.

to a similar activity (lifting another module) in a subsequent meeting, the resource (crane) is validated against the previously created shape (lifting capacity < weight of the module) for the new activity, until the shape is deleted from the shape graph. This means that constraints are always automatically reused in further meetings even when they are not explicitly mentioned. However, if the nature of the activity changes completely, a new constraint needs to be modeled.

3.2. Ontology for linked-data based dynamic constraint-checking

This section details the data sources, naming strategy, and associated ontologies used for supporting LDCC. For LDCC to work, the data need to be published following the linked-data principles. The tasks involved in publishing information using linked-data include identifying and obtaining access to the data source, analyzing it, defining the resource-naming strategy, developing the ontology, transforming the data source, and linking with another dataset [73]. Within the scope of the objectives of this paper, the generation of the linked-data is to support the look-ahead planning tasks and check the look-ahead schedule by traversing across multiple datasets. Therefore, the sources of data include BIM (including the construction site information), construction process data, resource information, the approval information, etc. The next step is to analyze the data sources used to determine the relevant information necessary for performing look-ahead scheduling, including the geometry of the construction site and resources, operating requirements of the resources, and project management information [30]. Regarding the naming strategy for the entities, we chose the "hashtag" strategy to avoid redirection issues associated with the "slash" format. The relevant data are hosted within the respective servers [74]. Existing ontologies are reused for the linking data across domains in the proposed approach. Particularly, we use IfcOWL to capture the model information, LinkOnt (a custom ontology detailed below) to capture the resource information and links, and SHACL to model constraints.

To perform dynamic constraint codification, we require several classes from *IfcOWL* (*ifcTask, ifcProduct, ifcResource* and *ifcTaskTime*) to capture the project management and model information. *ifcTask* describes a construction activity, and *ifcResource* describes the associated resources required to complete an activity. In addition, *ifcProduct* represents the abstract concept of an entity related to the spatial or geometric context, whereas *ifcTaskTime* captures the time-related information for the activity to be performed.

LinkOnt is the ontology we propose to support dynamic constraint-checking. This ontology introduces classes that are missing from IfcOWL but are required to do constraint-checking. Because *ifcTask* does not define the levels of the task (activity, subactivity, and sequence), a new class *TaskLevel* class is used to define the task level with three subclasses: *MainActivity, SubActivity,* and *Sequence*. To capture the URI of the geometry and the operating requirements of resources, *ifcResource* is linked with the *Resource* class using its *Type* property. The *Resource* class has a *ResourceProp* property that links it to the *ResourceType* class (which holds the information about the resource) and *OpRequirement* (which holds the requirements about the operating conditions of that resource). To link a resource to a *SubActivity,* a property *HasResource* is used. In summary, the *LinkOnt* ontology includes the following classes (Fig. 3): *TaskLevel, MainActivity, SubActivity, Sequence, Type, HasResource, Resource, ResourceProp, ResourceType,* and *OpRequirement*.

3.3. Modeling scheduling constraints using Shapes Constraint Language (SHACL)

Scheduling constraints ensure that the activities in the construction program are semantically linked to each other. This section describes how these scheduling constraints can be modeled as SHACL shapes to perform linked-data based constraint-checking. The constraints ensure that relationships are captured between the activities themselves, the activities and the workspace, and the activities and resources assigned to them. Constraints are a key concept in scheduling problems, as they describe what is possible and what can be scheduled at a point in time [75]. There are different types of construction scheduling constraints. Cardinality constraints ensure that the relevant values are present in a model for computations to be performed on it. Logical constraints cater to construction site constraints and constraints related to resources and

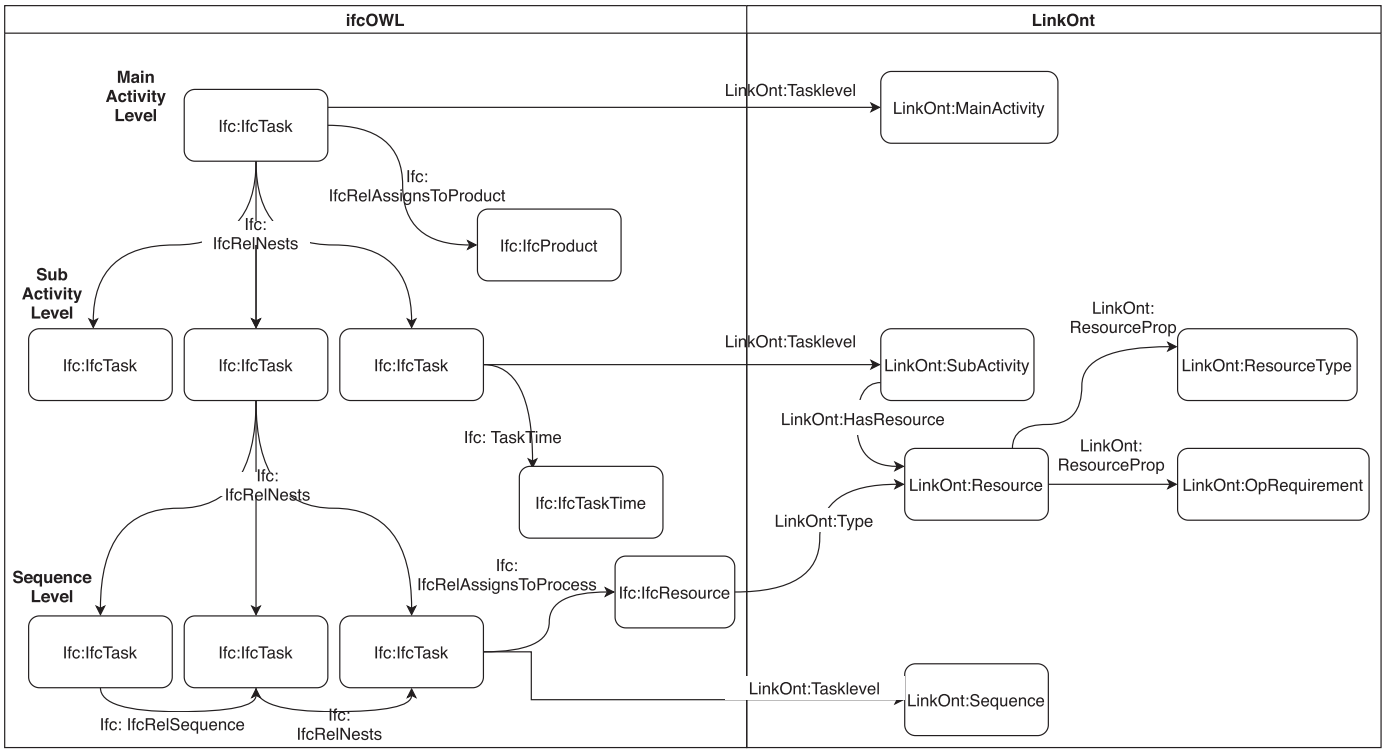


Fig. 3. Schematic of the LinkOnt ontology, showing its classes and relationships between them.

workspaces [76]. In addition, construction schedules must ensure that resources that are not available or suitable are not assigned and must guarantee that no two construction processes take place at the same time at the same location in an unshared workspace. These three construction scheduling constraints have been formalized into precedence, disjunctive, and discrete constraints [14,15]. The following subsections describe how these scheduling constraints are modeled as SHACL shapes.

3.3.1. Cardinality constraints

Cardinality constraints (which are part of SHACL core) represent restrictions on the number of values for a given property. This is important in scheduling, as different properties required different property values. For example, all scheduled activities must only have one start and one end date. Listing 1 contains the SHACL shape implementing this cardinality constraint. Similar cardinality constraints can be defined for other properties.

Listing 1. Cardinality constraints SHACL shapes.

```
1. Ifc:hasTaskTime
2.  rdf:type sh:PropertyShape;
3.  sh:targetClass Ifc:IfcTask ;
4.  sh:path Ifc: taskTime_IfcTask ;
5.  sh:minCount 1;
6.  sh:maxCount 1.
7.
8. Ifc:hasDates
9.   rdf:type sh:PropertyShape;
10.  sh:targetClass Ifc:taskTime_IfcTask ;
11.  sh:path Ifc:ScheduleStart_IfcTaskTime,
   Ifc:ScheduleFinish_IfcTaskTime;
12.   sh:minCount 1;
13.   sh:maxCount 1
```

In Listing 1, lines 1–6 ensure that each target class *Ifc:IfcTask* in the data graph has a property *Ifc:taskTime_IfcTask*. This is defined in such a way that the SHACL processing engine identifies all the classes defined in target class defined by *sh:targetClass* (in this case *Ifc:IfcTask*). Once the target class instances are identified in the data graph, the processing engine searches for the property shape (because the current property is defined as a property shape as shown in line 2 – *rdf:type sh:PropertyShape*) and the path for the property shape (as shown in line 4: *sh:path Ifc: taskTime_IfcTask*). In addition, *minCount* and *maxCount* are applied on that property path (in this case *Ifc:taskTime_IfcTask*). Similarly, lines 8–13 ensure that each target class *taskTime_IfcTask* has at least and at most one start and end date defined by *Ifc:ScheduleStart_IfcTaskTime* and *Ifc:ScheduleFinish_IfcTaskTime* properties.

3.3.2. Logical constraints

Logical constraints are used to represent the site constraints that need to be satisfied for an activity to take place. These constraints might relate to space availability, resource parameters, and activity relations. For example, the lifting capacity of a crane should always be higher than the weight of the module to be lifted. Listing 2 defines the *less than* constraint as an SHACL shape. These shapes can be reused when the crane is used to support another activity.

Listing 2. SHACL shape for Less than constraint.

```
1. LinkOnt:SubActivity
2.  rdf:type sh:NodeShape ;
3.  rdfs:subClassOf owl:Class ;
4.  sh:sparql [
5.   sh:message "Constraint LESS THAN is violated" ;
6.   sh:prefixes < http://purl.org/ConstructLinkOnt/LinkOnt > ;
7.   sh:select """SELECT $this
8. WHERE {
9.   $this rdf:type LinkOnt:SubActivity.
10.    $this LinkOnt:HasResource ?<Resource1>.
11.    ?<Resource1> LinkOnt:ResourceType <Resource1Type>.
12.    $this LinkOnt:HasResource ?<Resource2>.
13.    ?<Resource2> LinkOnt:ResourceType <Resource2Type>.
14.  …..?<Resource1> LinkOnt:OpRequirement ?<Resource1Prop>.
15.  …..?<Resource2> LinkOnt:OpRequirement ?<Resource2Prop>.
16.     FILTER (?<Resource1Prop> <= ?<Resource2Prop>).
17.  }""" ; ];
```

Listing 3. SHACL shape for precedence constraint.

```
1. Ifc:IfcTask
2.  rdf:type sh:NodeShape ;
3.  rdfs:subClassOf owl:Class ;
4. sh:sparql [
5.   sh:message "Constraint Precedence condition is violated" ;
6.   sh:prefixes < http://www.buildingsmart-tech.org/ifcOWL/IFC4#> ;
7.   sh:select """SELECT $this
8. WHERE {
9. ?Sequence a Ifc:IfcRelSequence.
10.  ?Sequence Ifc: relatingProcess_IfcRelSequence ?predecessor.
11.  ?Sequence Ifc: relatedProcess_IfcRelSequence $this.
12.  ?predecessor Ifc: taskTime_IfcTask ?predecessortime
13.  $this Ifc: taskTime_IfcTask ?successortime
14.  ?successortime Ifc: ScheduleStart_IfcTaskTime ?start
15.  ?predecessortime Ifc: ScheduleFinish_IfcTaskTime  ?end
16.  FILTER( ?start <?end)
17.  }""";];
```

To model this constraint, we need to use SHACL SPARQL, an advanced feature of SHACL that adds the expressive flexibility in targeting focus nodes. In Listing 2, the aim of the shape is to detect constraint violations where the activities have two resources (< *Resource1* > line 10, and < *Resource2* > line 12) of two types (*Resource1Type* > line 11, and < *Resource2Type* > , line 13) and operating requirements (< *Resource1Prop* > line 14, and < *Resource2Prop* > , line 15). The shape checks whether the value of < *Resource1Prop* > is less than < *Resource2Prop* > . The constraint is modeled as a node shape and applied to the *LinkOnt:SubActivity* (Listing 2, lines 1–4). The *LinkOnt:SubActivity* is related to resources assigned to it (Listing 2, lines 9, 10, and 13). Therefore, all the assertions of *LinkOnt:SubActivity,* which has resources of type < *Resource1Type* > and < *Resource2Type* > , are checked for the condition < *Resource1Prop* > *is less than* < *Resource2Prop* > . The assertions of *LinkOnt:SubActivity* which does not have the particular resource type are ignored. The properties < *Resource1Prop* > and < *Resource2Prop* > reside in the respective resource servers. The SELECT queries filter all the *LinkOnt:SubActivity* assertions with the said resource associations and filter the *LinkOnt:SubActivity* assertions where the condition is violated. This raises the message *sh:message "Constraint LESS THAN is violated"* to the user.

3.3.3. Precedence constraints

Precedence constraints validate the sequence of activities. They are the basis of network-based scheduling methods, such as the critical path method [30,77]. An example of a precedence constraint is that the roof must be cast after all the columns are cured. Note that these constraints do not check the assignment and availability of resources. The constraint is mathematically defined as follows:

Start date of the successor activity ≤ *End date of preceding activity.*

In IFC4 [78], an *IfcTask* class is associated with the details of the activities and has the *IfcRelSequence* as an inverse attribute. *IfcRelSequence* has attributes that define precedence relations. *RelatingProcess* defines the predecessor, and *RelatedProcess* defined the successor activities. The successor and predecessor activities have the *taskTime_IfcTask* class as an attribute to hold the start and end times of the activities as described in Section 3.3.1.

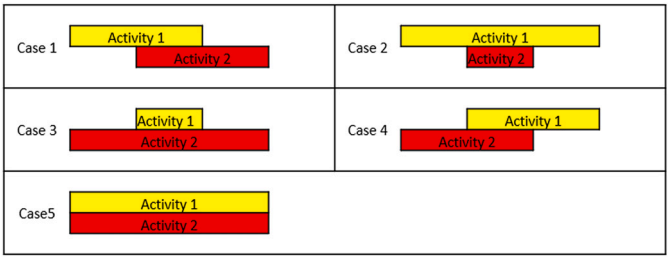


Fig. 4. Conditions for overlap of two activities.

Precedence constraint requires the expressive power of SPARQL to determine the activities that have precedence relations associated with them. Hence, SHACL SPARQL is used. Listing 3 presents the SHACL SPARQL definition of the precedence constraint. It is assigned as a node shape on *IfcTask* (Listing 3, lines 1–3). During the constraint-checking stage, all the assertions of *Ifc:IfcRelSequence* in the data graph are filtered (Listing 3, line 9). The corresponding activities *Ifc:relatingProcess_IfcRelSequence* (predecessor) and *Ifc:relatedProcess_IfcRelSequence* (successor) are then identified and stored to? *predecessor* and *$this,* respectively (Listing 3, lines 10–11). The *taskTime_IfcTask* assertion (which contains the time attributes) is stored to? *predecessortime* and? *successortime* (Listing 3, lines 12–13). The end date in? *predecessortime* and the start date of? *successortime* are then stored to? *start* and? *end,* respectively (Listing 3, lines 14–15). A filter function of the SPARQL is used to identify instances where? *start* is less than? *end* (Listing 3, line 16). If the start date of the related process happens before the end date of the relating process, a constraint violation is detected and a *sh:message "Constraint Precedence condition is violated"* is raised. This constraint can be reused across all the problems where the IFC process data are defined using IfcOWL following IFC4.

3.3.4. Disjunctive constraints

Disjunctive constraints ensure that processes do not overlap. Thus, if a disjunctive constraint is assigned to two activities, they should not happen in parallel. These constraints support the modeling of spatial constraints [79] and safety constraints (e.g., two activities corresponding to a unique space should not take place simultaneously) among others.

To mathematically define this constraint, consider two activities: *Activity 1* with start time *StartTime_1* and end time *EndTime_1* and *Activity 2* with start time *StartTime_2* and end time *EndTime_2.* The five cases to check whether or not they overlap are as follows (they are graphically represented in Fig. 4):

Case 1: *StartTime_1* < *StartTime_2* < *EndTime_1* < *EndTime_2;*
Case 2: *StartTime_1* < *StartTime_2* < *EndTime_2* < *EndTime_1;*
Case 3: *StartTime_2* < *StartTime_1* < *EndTime_1* < *EndTime_2;*
Case 4: *StartTime_2* < *StartTime_1* < *EndTime_2* < *EndTime_1; and*
Case 5: *StartTime_1* = *StartTime_2* && *EndTime_1* = *EndTime_2.*

These constraints can be represented as a single statement:

(*StartTime_2* < *EndTime_1* && *StartTime_1* < *EndTime_2)*||
(*StartTime_1* = *StartTime_2* && *EndTime_1* = *EndTime_2)*

Listing 4. SHACL shape for disjunctive constraint.

```
1. Ifc:IfcProduct
2.  rdf:type sh:NodeShape ;
3.  rdfs:subClassOf owl:Class ;
4. sh:sparql [
5.   sh:message "Constraint Disjunctive condition is violated" ;
6.   sh:prefixes < http://www.buildingsmart-tech.org/IfcOWL/IFC4#> ;
7.   sh:select """SELECT $this
8. WHERE {
9. ?activity1 a Ifc:IfcTask.
10.  ?activity2 a Ifc:IfcTask.
11.  ?activity1 Ifc: IfcRelAssignsToProduct $this
12.  ?activity2 Ifc: IfcRelAssignsToProduct $this
13.  ?activity1 Ifc: taskTime_IfcTask ?activity1time.
14.  ?activity2 Ifc: taskTime_IfcTask ?activity2time.
15.  ?activity1time Ifc: ScheduleStart_IfcTaskTime ?start1.
16.  ?activity1time Ifc: ScheduleFinish_IfcTaskTime ?end1.
17.  ?activity2time Ifc: ScheduleStart_IfcTaskTime ?start2.
18.  ?activity2time Ifc: ScheduleFinish_IfcTaskTime ?end2.
19.
20.  FILTER (!(?activity1=?activity2)).
21.  FILTER( (?start2<?end1 && ?start1<?end2)||( ?start1=?start2 &&
   ?end1=?end2)).
22.  }""";];
```

Listing 4 demonstrates a disjunctive constraint modeled as a SHACL shape. For this example, a node shape is applied to the *IfcProduct* class, which replicates the space in which an activity is performed (Listing 4, lines 1–3). An example of a disjunctive constraint is that no more than one activity must happen on a particular object *$this* of class *IfcProduct*. During the constraint-checking stage, all *IfcTask* elements related to *IfcProducts* are filtered using the expressive power of SHACL SPARQL and are stored into two activities:? *activity1* and? *activity2* (Listing 4, lines 9–12). To ensure that the activities are not the same, a filter operator is used (Listing 4, line 20). The start and end times of? *activity1* and? *activity2* are then stored to? *start1,*? *end1,* and? *start2,*? *end2,* respectively, through a set of code similar to the ones mentioned in Section 3.3.3 (Listing 4, lines 13–18). Then,? *activity1* and? *activity2* are checked for overlaps using a filter operator (Listing 4, line 21). If there is an overlap, the *sh:message* "*Constraint disjunctive condition is violated*" is shown (Listing 4, line 5).

3.3.5. Discrete constraint

Discrete constraints model the resource availability constraints in construction scheduling, which ensures that the allocation of resources is made according to the availability at any time point. This constraint is frequently used in research studies on resource-constrained project scheduling problems [80,81]. Because this constraint is predominantly applied to resources, the IfcOWL classes *Ifc:IfcResource, Ifc:IfcRelAssignsToProcess, Ifc:taskTime_IfcTask, ScheduleStart_IfcTaskTime,* and *Ifc: ScheduleFinish_IfcTaskTime* are used to define it.

Listing 5. SHACL shape for discrete constraint.

```
1. Ifc:IfcResource
2.  rdf:type sh:NodeShape ;
3.  rdfs:subClassOf owl:Class ;
4.  sh:sparql [
5.   sh:message "Discrete constraint for the resource not satisfied" ;
6.   sh:prefixes < http://www.buildingsmart-tech.org/IfcOWL/IFC4#> ;
   sh:select """SELECT $this
7. WHERE {
8. SELECT ?Resource (count(distinct ?activity1) as ?count)
9. {
10.  ?Resource a Ifc:IfcResource.
11.  ?activity1 Ifc:IfcRelAssignsToProcess ?Resource.
12.  ?activity2 Ifc:IfcRelAssignsToProcess ?Resource.
13.
14.  ?activity1 Ifc: taskTime_IfcTask ?activity1time.
15.  ?activity2 Ifc: taskTime_IfcTask ?activity2time.
16.  ?activity1time Ifc: ScheduleStart_IfcTaskTime ?start1.
17.  ?activity1time Ifc: ScheduleFinish_IfcTaskTime ?end1.
18.  ?activity2time Ifc: ScheduleStart_IfcTaskTime  ?start2.
19.  ?activity2time Ifc: ScheduleFinish_IfcTaskTime  ?end2.
20.
21.  FILTER (!(?activity1=?activity2)).
22.  FILTER( (?start2<?end1 && ?start1<?end2)||( ?start1=?start2 &&
   ?end1=?end2)).
23.  }
24.  FILTER (?count > <No of resource>)
25.  }""" ;
26.   ] ;
```

Again, SHACL SPARQL is used to model the shape graph for discrete constraints (see Listing 5), following a logic similar to that of the previous disjunctive modeling constraint, although now the constraint is defined as a node shape on the *IfcResource* class instead of *IfcProduct* class (Listing 5, lines 1–3). Unlike the disjunctive constraint, there is a possibility for several activities that use a particular resource type to be executed in parallel as long as it does not violate the resource constraint (i.e., when more than one resource is available for multiple resource types). Hence, a count is taken for the number of activities that use a type of resource, and the measured count should be less than the total number of resources of that type available at any point in time.

To achieve this, we use the expressive power of SHACL SPARQL defined as two nested SPARQL queries. The inner query (Listing 5, lines 9–24) identifies two activities that use the particular resource at a point of time. This loop is similar to the SPARQL query defined in Section 3.3.4. The *IfcTask* elements with a relation assigned to *IfcResource* are filtered using the expressive power of SHACL SPARQL. Then, these are stored into two activities:? *activity1* and? *activity2* (Listing 5, lines 12–13). To ensure that the activities are not the same, a filter operator is used (Listing 5, line 22). The start and end times of? *activity1* and? *activity2* are then stored into? *start1,end1* and? *start2,*? *end2,* respectively, through a code similar to the one in Section 3.3.3 (Listing 5, lines 15–20). Then,? *activity1* and? *activity2* are checked for overlaps using a filter operator (Listing 5, line 22). The outer query (Listing 5, lines 7, 8, and 25) counts the number of instances of activities using resources in parallel at a point in time, and these are then compared with the total number of resources at that point. If the count is greater than the number of resources, a constraint violation is detected, and the user is presented with a message "*Discrete constraint for the resource not satisfied*" (Listing 5, line 5).

4. Linked-data based constraint-checking implementation

After the description of the LDCC method and how constraints are modeled as SHACL shapes in the previous section, this section presents its implementation (through a web application) to validate the construction scheduling constraints in a look-ahead planning environment. We tested the 4D model from the Schependomlaan OpenBIM dataset [71]. It contains construction schedule information at the master planning level, and its activities were enriched by adding further subtasks and constraints. We manually checked the results from the LDCC application to confirm it was able to identify constraint violations.

4.1. Prototype web application

The LDCC architecture, discussed earlier in Section 3.1, was implemented as a prototype web application, with its functional details shown in Fig. 5. The information required by the developed software is imported from an existing BIM model and enriched with user inputs. Users can interact with the web application through the front-end

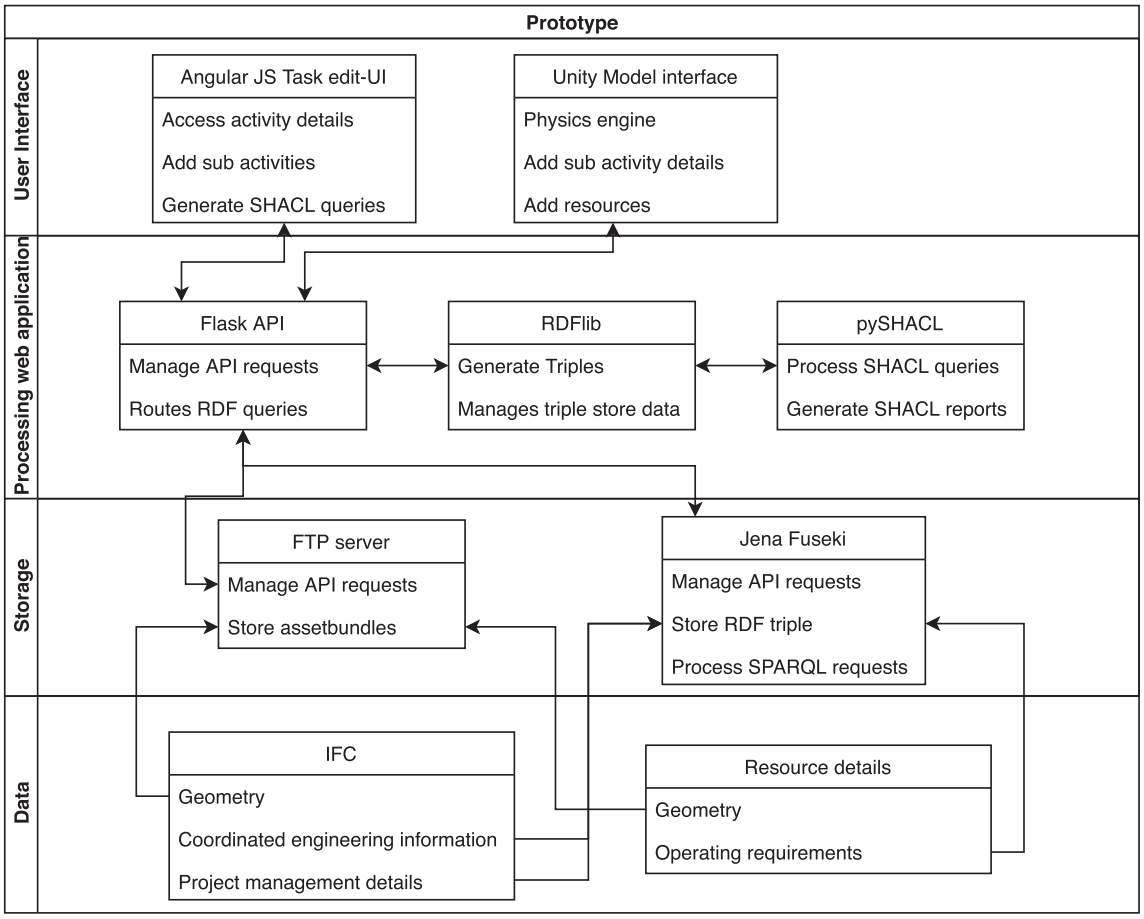


Fig. 5. Functional details of the software implementing LDCC.

which contains two UIs (further elaborated in Section 4.1.1, see Fig. 6):

1) Model interface – to interact with the geometric model; and.
2) Task-edit interface – to interact with a Gantt chart to view task information, and to further enrich the imported data with user inputs.

The back-end of the web application consists of a processing web application and its associated data storage. Details of the different components are explained in this section.

4.1.1. User interface

The UI presents the imported data to the user, which can be further

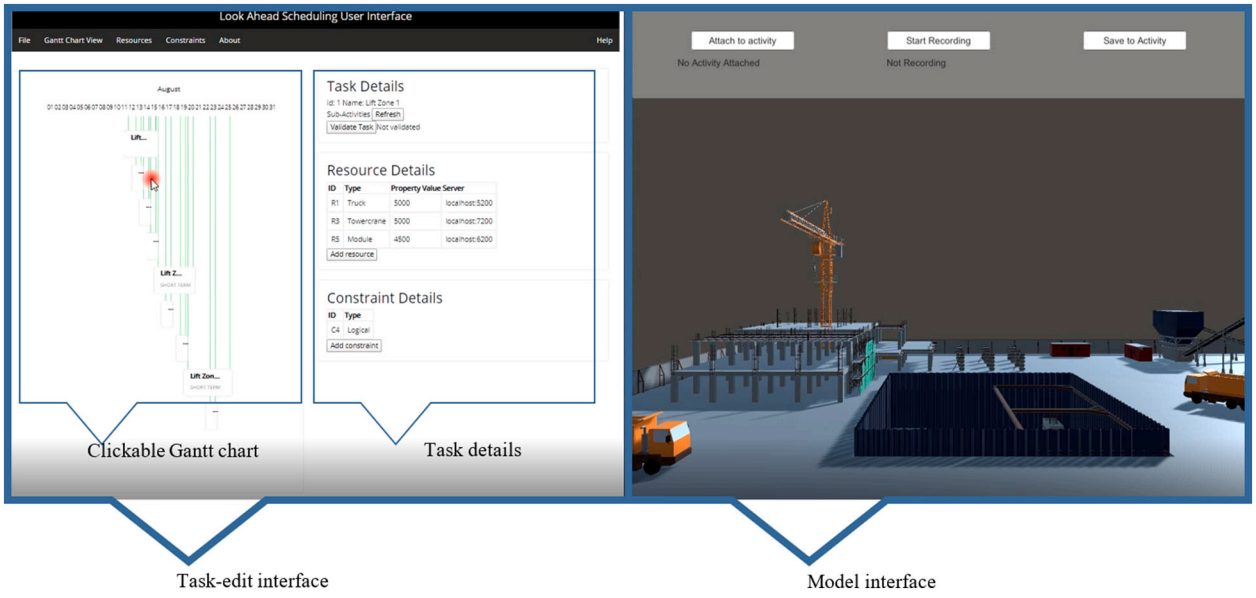


Fig. 6. User interface of the software implementing LDCC.

enriched with user inputs. The UI consists of the model interface and the task-edit interface, which were implemented separately. The model interface was developed using the Unity game engine [82] and its development is detailed in an earlier study by the authors [83]. The model interface enables 3D interaction with the construction site model (e.g., selecting the geometry of the objects, moving the resources, etc.) and can be deployed to any device, including augmented reality devices (e.g. Microsoft Hololens), virtual reality devices (e.g. HTC Vive), or a computer screen through a web browser. To interact with the model view, we used the 3Drepo.io unity plugin [84]. The Unity game engine has active physics engine which ensures that the model manipulations follow real-world physics.

The task-edit interface was developed using Angular JS [85]. It allows users to manage the project data (e.g., activities, sub activities sequences, resources, and constraints), and facilitates the enrichment of tasks with further subtasks, resources, and constraints. The task-edit interface contains a Gantt chart, generated from the planning data that was embedded in the BIM model. When a task in the Gantt chart is clicked, the right side of the interface is activated. It shows existing task details, such as task name, resource details, and added constraints. There are provisions to enrich the existing information with additional resources and constraints. Imported data as well as user inputs are transformed into data graph and shape graph using the processing web application.

Both interfaces in the UI are connected. When a task is selected in the task-edit interface, that task could be manipulated using the model interface. All the manipulations in the model interface are recorded and transformed into sequences in the task-edit interface. Details on how this interface is used to perform constraint-checking are explained in Section 4.2.

4.1.2. Processing web application

The processing web application is the main application that stands between the UI and data storage. It processes the inputs from the users, converts them to RDF graphs, acts as a SHACL processing engine, and displays validation results. The backend of the processing web application is a Flask application [86] which uses the RDFLib library [87] to process the data modeled in RDF, and validate the constraints expressed as SHACL SPARQL functions using the pySHACL library [88]. The construction information was stored as RDF triples following OWL, RDF, ifcOWL [57,89,90], LinkOnt and SHACL ontologies. The ontology layer was handled through RDFLib, which accesses the schema definitions over the internet to determine the relationships between data.

4.1.3. Storage

Two pieces of storage are required. The first one is an FTP server that holds the geometry of the resources and the construction site model. The model interface of the UI was developed using the Unity game engine and requires an AssetBundles file [91] (AssetBundles are archive files that contains Unity-specific non-code assets, such as geometry and textures, that Unity can load at run time). Asset bundles are downloaded from the FTP server during the runtime using Unity API calls to the FTP server.

A second piece of storage is used for storing specific semantic information about the construction, such as operating conditions of the resources, engineering information embedded in the model, and the project management details. These are stored as RDF triples in a Jena Fuseki server [92]. The data are accessed by the processing web application using HTTP requests. The Jena Fuseki server can also handle SPARQL requests from the processing web application to fetch data.

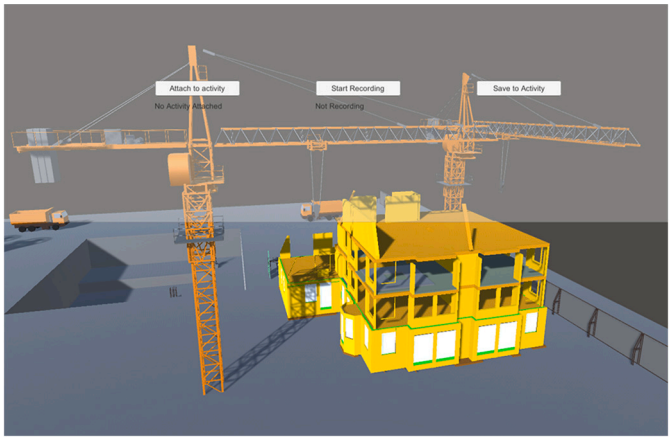


Fig. 7. Screenshot of the imported Schependomlaan dataset in an enriched site environment.

4.1.4. Data

For the application to work, two sets of data are required. The first set includes information required for managing the construction site, such as BIM, construction site characteristics, and project management details. Most of the construction site information can be obtained from a BIM exported in the IFC format. BIM contains the geometry of the structure, the coordinated engineering information, and the project management details. The second dataset required by the model includes information for resource management. This dataset has details about the resources, including their geometry and operating conditions. These need to be obtained from subcontractor servers (e.g., using an API to the subcontractor common data environment or subcontractor enterprise resource planning systems).

4.2. Prototype evaluation

To evaluate the developed application, we used the Schependomlaan OpenBIM dataset [71] as-planned model from Week 26 (Fig. 7). This evaluation enabled us to demonstrate the potential of LDCC to model and check complex and dynamic scheduling constraints in construction. Evaluation is performed by testing LDCC to define and execute complex construction scheduling constraints, such as logical constraints, precedence constraints, disjunctive constraints, and discrete constraints.

Table 1 provides the data given as input to the application to enrich the information from the Week 26 planning file of the Schependomlaan dataset. Two instances of the *IfcTask* class were enriched (referenced as Tasks T1 and T2 in this paper). Enrichment is done by clicking the task in the Gantt chart (in the task-edit interface) window shown in Fig. 6. Once a task is clicked, the details of that task are displayed in the task details pane (Fig. 8). Resources and constraints are added using that pane. During enrichment activities, S1, S2, S3, and S4 were added as

Table 1
Task details used as inputs for the evaluation.

Sub Activity	Main activity	Start date	End date	Resource assignment	Object assignment	Precedence
S1	T1	21/08/2015	23/08/2015	R1, R3	O1	
S2	T2	24/08/2015	27/08/2015	R2, R4	O2	S4
S3	T2	28/08/2015	30/08/2015	R1, R3	O2	
S4	T2	22/08/2015	25/08/2015	R2, |R4	O2	

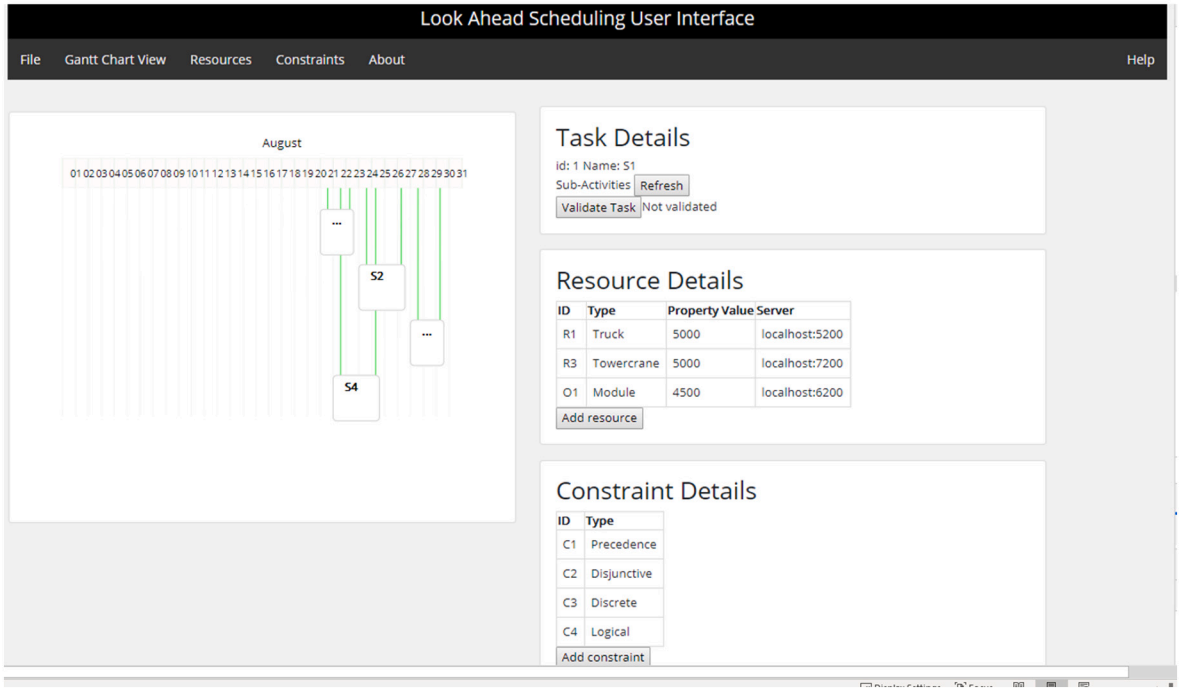


Fig. 8. Task-edit interface for the imported dataset. The data shown refer to SubActivity S1.

SubActivities to tasks T1 and T2. S1, S2, S3, and S4 are lifting activities using cranes R3 and R4 (from trucks R1 and R2) to lift modules O1 and O2 respectively. Moreover, R1, R2, and R3 were assigned operating requirement values of 5 tons, while R4 was assigned an operating requirement of 3 tons. All resources had a discrete constraint applied to them to ensure the resources are allocated as per availability. In addition, logical constraints were assigned to the resources. Two module objects O1 and O2 (with an operating requirement value of 4.5 tons) were associated with the activities S1, S2, S3, and S4, and disjunctive constraints were applied to ensure that only one subtask is performed on an object at one point in time. R1, R2, R3, R4, O1, and O2 were stored on external servers (to replicate the case of the construction, where this information is hosted in subcontractor servers). A precedence constraint was applied to activity S2 so that S2 can only start after S4. In addition to these definitions, and to demonstrate logical constraints, a simple *less than* was applied for all the resources. The less than logical constraint ensures that the value of the operating requirements of the resources is less than that of the operating requirement defined in the object associated with that activity.

When the *Validate Task* button for an activity is clicked (Fig. 9), the data associated with the activity are checked against the shapes. Additionally, other activities happening at the same time and their

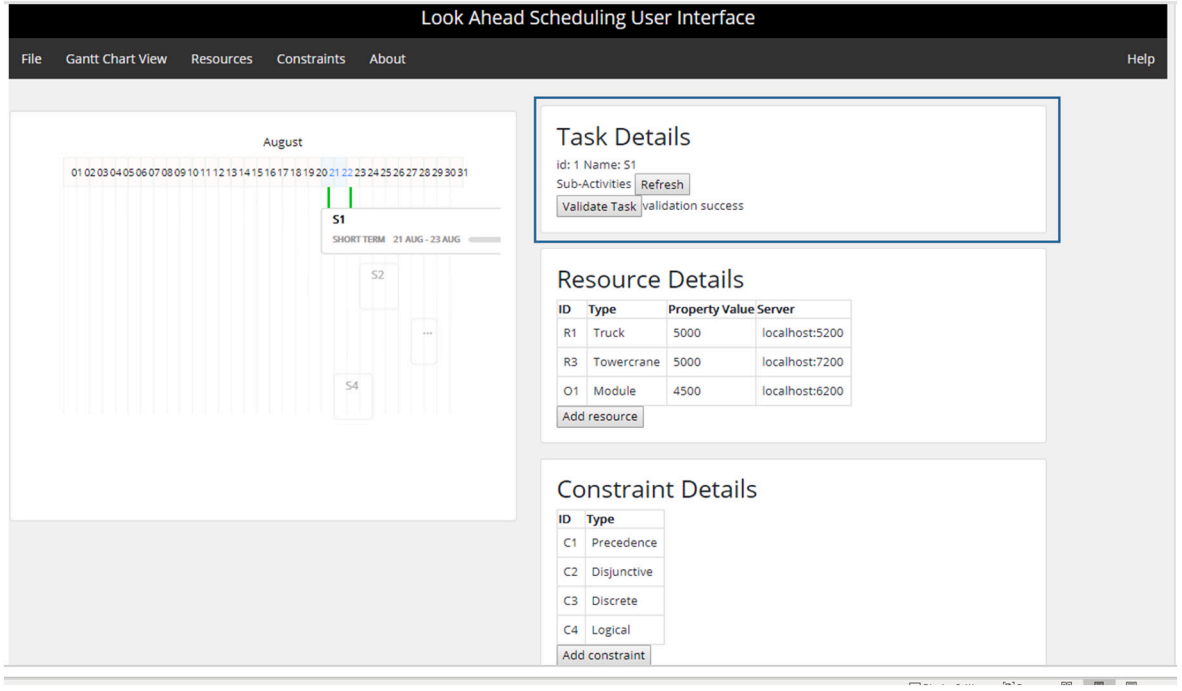


Fig. 9. Successful validation of S1 activity.

Table 2
Results after constraint-checking.

Subtask	Result
S1	No constraint violation
S2	Logical constraint violation, disjunctive constraint violation, discrete constraint violation, precedence constraint violation
S3	No constraint violation
S4	Logical, disjunctive constraint violation, discrete constraint violation

associated resources are also checked for conflicts. Table 2 shows the results from the constraint-checking process performed by the application.

The developed application was able to identify several constraint violations using information distributed over multiple servers. There was no constraint violation for S1 (Fig. 9) and S3, but S2 violates all the constraints (Fig. 10), and S4 violates discrete and disjunctive constraints. In addition, S2 and S4 overlap each other on 24 August 2015 (which violates the precedence constraint applied on S2). Moreover, S2 and S4 share resources R2 and R4, thus violating the discrete resource constraints. Similarly, S2 and S4 operate on the same object O2, violating the disjunctive constraint, as two activities cannot happen in the same space simultaneously. The logical constraint is violated for these activities, as the operating requirement value of R4 is less than the operating requirement value of O2.

5. Discussion

The proposed LDCC method follows the principles of BIM based rule-checking, and the approach for rule-checking suggested in Eastman et al. [37] (discussed in Section 2.1) where rules are translated into SHACL automatically by engaging with a UI. Construction information is converted to an RDF graph (machine-readable format) for easy execution of constraints. The SHACL constraints are then checked for constraint violation using a processing web application enabled by RDFLib and pySHACL. The results of constraint-checking are reported back to the user through the UI. While LDCC allows for automated analysis of data from ADC technologies instead of automated distilling rules directly from the textual content, rules are defined as constraints by the user. LDCC reaches Class 3 complexity, as described by Solihin and Eastman [43], because the data structure is extended and traversed to provide logical proofs for constraint-checking.

In contrast with previous studies that uses BIM based rule-checking (e.g., [93,94]), our proposal could identify all the constraint violations and inform the user about them even when the data are distributed over multiple servers. When a discrete constraint was applied to the resources (R1, R2, R3, and R4), the resources do not have any schedule data associated with them in their servers. The schedule data were stored in the activities, as shown in Table 1, and resources and objects were in separate servers making the existing methods unsuitable for modeling this complex constraint. Despite this, the LDCC query could crawl through the data to identify relevant time data from the resource assignments and make logical inferences, to identify whether there were multiple assignments during the same period. This is possible because the ontology layer is separate from the processing layer; therefore, queries need not have the know-how of the whole data structure schema but only the attributes necessary for making the traversal (also reported in prior research [24,61]).

The remaining semantics for the data traversal is handled through reasoning offered by the OWL and RDF ontologies. Though this example could have been evaluated manually, the real power of the proposed LDCC method emerges as the scale of the problem becomes larger. As the scale of the problem increases (e.g., managing the resources and space allocation of multiple trades working in a large construction site with numerous shared resources), manual methods to identify constraint violations among multiple interdependent activities and resources becomes difficult [95,96] and results in errors [3]. However, using computer-based methods such as LDCC, the scale of the problem is no longer a critical issue, as the computer can process these interdependencies since the processing mechanism remains the same, and the speed is restricted only by the processing capability of the computer.

There are multiple benefits for using the proposed method in look-ahead planning. The first benefit of LDCC is that it does not only provides automatic constraint checks during the look-ahead planning meetings, but also captures the construction knowledge generated and

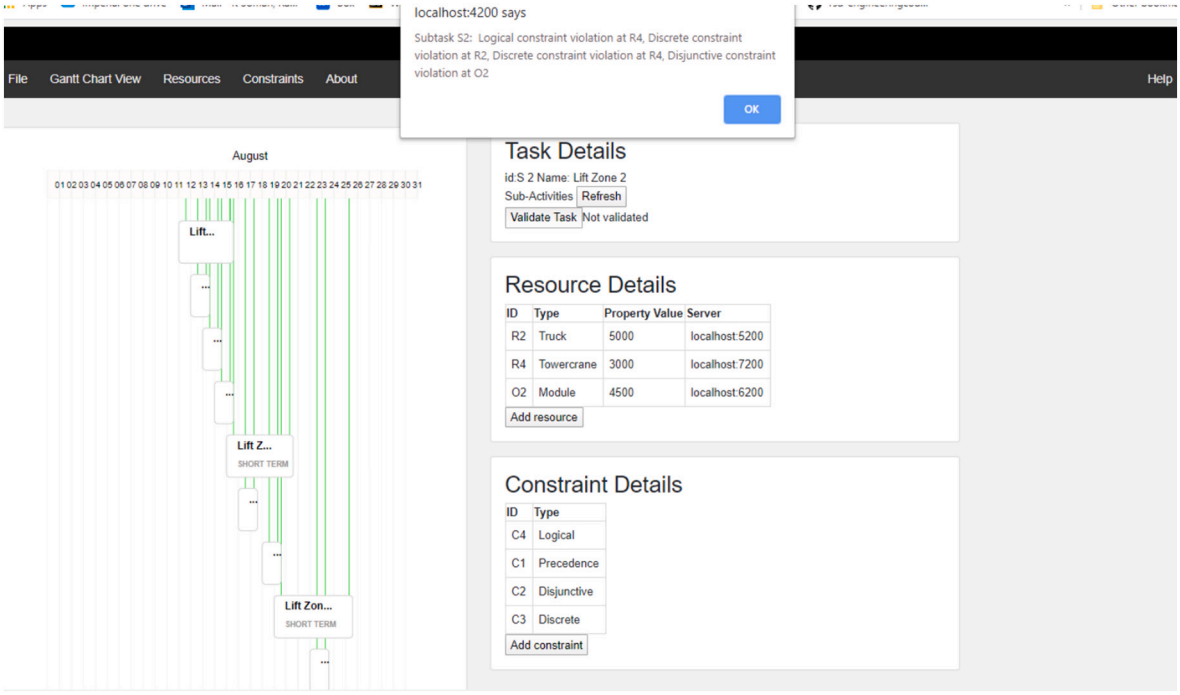


Fig. 10. Validation failure for S2 activity.

discussed in them and codifies it into a machine readable format, which can be reused later (as described in section 3.1). This reduces the need for data input after a stage of saturation, as the shape graph is populated during subsequent sessions. The second benefit of this approach is that it can accommodate the dynamicity of constraints. Data and constraints are added incrementally, and constraint checks can be performed automatically whenever there is a change in the data graph or the shapes graph. For example, if the operating requirement of equipment changes when a new operating requirement is modeled as a new shape (constraint), all the assignments of the equipment, including the ones that had already been checked for constraint violation, would be re-evaluated for the new constraint. This enables stakeholders to identify the effect of changes in the project variables as and when they arise. This feature also enables multiple parties to make changes to the codified construction information while maintaining the integrity. When linked with other communication platforms, LDCC can inform the people responsible for the activities of the constraint violations, if any. Therefore, LDCC acts as a "unit-test" environment for short-term schedule data.

In addition, the proposed approach can be combined with machine learning frameworks and discrete-event simulation tools to reschedule the processes and aid decision making during disruptions because the current state of work and the constraints are codified. Previous studies have used methods to generate optimal construction schedules addressing constraints, such as crew availability and material availability [97,98]. However, these methods require mathematical modeling of constraints to generate schedules. In addition, LDCC eases this task by taking in construction constraints spanning multiple domains and distributed over heterogeneous datasets from the user, converting them into a machine-readable format. The LDCC environment thus acts as a processing engine for simulations and machine learning algorithms, where it is cumbersome to formulate numerous constraints mathematically. For example, in the case of genetic algorithms, the LDCC environment can be used to check the fitness value of the individuals when the constraints cannot be modeled a priori. Similarly, for a reinforcement learning algorithm, LDCC can reside within the environment to evaluate the reward for the actions taken by the agent. Thus, the proposed LDCC can be used to automatically reschedule tasks to create a feasible look-ahead schedule when there are disruptions, thereby supporting look-ahead planning.

The research presented in this paper is an important first step towards linked-data based rule-checking for construction scheduling constraints and the LDCC method demonstrates the effectiveness of the approach to address the challenges existing in checking for construction constraint violations. However, previous research has reported that the ifcOWL ontology (used in our software application) is not well suited for linked-data applications [99]. In contrast with ontologies which are light and modular [100], ifcOWL is large, complex and contains unintuitive semantic constructs. Hence, it is relatively difficult to model SHACL constraints for a user who doesn't have full grasp of the IFC schema. We partly addressed this limitation in this work by defining SHACL shape snippets for common scheduling constraints (Section 3.3). These snippets can be reused through the UI. A limitation of this work is not exploring the use of semantic relationships within the geometry [55] to perform geometry based constraint-checks, such as the ones demonstrated in Schwabe et al. [93]. These geometry-based checks are possible; however, they need further investigation before they can be defined as SHACL constraints and is an avenue for future research. Also, the evaluation of the developed application implementing the framework was done using a public construction dataset as a proof of concept. The software needs to be tested further in actual look-ahead planning meetings to demonstrate the effectiveness and to understand the limitations. As discussed in the next section, this opens up avenues for future research in the area of linked-data in architecture and construction focusing on geometry-based constraint checks and simple ontologies to describe construction process information.

6. Conclusions

This paper presents a novel method using linked-data based constraint-checking to model and validate construction scheduling constraints (cardinality constraints, logical constraints, precedence constraints, disjunctive constraints, and discrete constraints). The LDCC method addresses the inherent challenges in modeling construction constraint datasets by leveraging the power of linked-data to traverse across the web of data created by linking across the domains and datasets. The prototype web application developed using LDCC architecture could identify construction scheduling constraint violations for input data distributed over multiple servers, thus, demonstrating the potential of this approach.

This paper contributes to the emerging research trajectory on using automated methods for construction scheduling by developing a decision support system to aid look-ahead planning in construction. When combined with ADC technologies, LDCC mitigates the difficulty to track and monitor the prerequisites and validate constraints in complex construction projects through data-driven constraint-checking. In addition, LDCC method offers opportunity to capture the construction knowledge created during look-ahead meetings. This reduces the need for redundant input from the stakeholders in subsequent meetings, as the stakeholder requirements and constraints are codified into a machine-readable format. Furthermore, this method acts as a "unit testing" environment for detailed construction process information. Unit testing fosters model-based collaboration among stakeholders, because every time a change occurs, one can evaluate the consequences of change to the program and check for constraint violations using the codified construction information because the collective construction scheduling input of all the stakeholders is codified as detailed construction process information and constraints.

The research presented in this paper is a first step. To exploit the full potential of the LDCC method, further research is needed to extend it. Researchers working on linked-data in the architecture and construction domain can use this work reported here as a basis to develop LDCC and investigate its applicability to detect violations related to workspace constraints, construction safety constraints, and construction site layout constraints by exploiting the geometric data in the domain models. In addition, data science researchers can use the prototype developed in this paper to further investigate the potential of the use of LDCC as a simulation or processing engine to automate the generation of look-ahead schedules using optimization and machine learning. Researchers working in the area of lean construction can extend the LDCC approach to cater to the constraints in real construction sites and test its efficacy in improving the flow of work and to develop further methods for enhancing model-based collaboration in look-ahead planning. Researchers working on ADC can extend it to predict the knock-on-effects of the deviations detected by sensors to predict future disruptions. The current application implementing LDCC provides an important first step towards linked-data based rule-checking for construction scheduling constraints by checking for construction constraint violations.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This is a substantially extended and enhanced version of the paper presented at the CIB W78 Annual Conference held at Northumbria University in Newcastle UK in September 2019. We would like to acknowledge the editorial contributions of Professor Bimal Kumar of

Northumbria University and Dr. Farzad Rahimian of Teesside University in the publication of this paper. The first author's PhD research is co-funded by Bentley systems the UK and through a Skempton Scholarship, the Department of Civil and Environmental Engineering, Imperial College London. During this work, the first author was also supported by the PhD enrichment scholarship from The Alan Turing Institute. The authors acknowledge the supervisory inputs offered by Dr. David Birch and Dean Bowman and also acknowledges the guidance offered by Prof. Jakob Beetz. The second author is supported by the European Commission (H2020 MSCA-IF ga. No. 754446). The third author acknowledges the support of Laing O'Rourke and the Royal Academy of Engineering for cosponsoring her Professorship.

References

[1] G. Ballard, Lookahead Planning: The Missing Link in Production Control, in: Proceedings of the 5th Annual Meeting of the International Group for Lean Construction, Gold Coast, Australia, 1562384988, 1997, pp. 1–14.
[2] B. Dave, J.-P. Hämäläinen, L. Koskela, Exploring the recurrent problems in the last planner implementation on construction projects, in: Indian Lean Construction Conference, Chennai, India, 2015: pp. 1–9, ISBN 0706-7437.
[3] S. Lindhard, S. Wandahl, Scheduling of large, complex, and constrained construction projects - an exploration of LPS application, International Journal of Project Organisation and Management. 6 (2014) 237–253, https://doi.org/10.1504/IJPOM.2014.065258.
[4] S.N. Razavi, O. Moselhi, GPS-less indoor construction location sensing, Autom. Constr. 28 (2012) 128–136, https://doi.org/10.1016/j.autcon.2012.05.015.
[5] N. Pradhananga, J. Teizer, Automatic spatio-temporal analysis of construction site equipment operations using GPS data, Autom. Constr. 29 (2013) 107–122, https://doi.org/10.1016/j.autcon.2012.09.004.
[6] J. Majrouhi Sardroud, Influence of RFID technology on automated management of construction materials and components, Scientia Iranica. 19 (2012) 381–392, https://doi.org/10.1016/j.scient.2012.02.023.
[7] E. Valero, A. Adán, C. Cerrada, Evolution of RFID applications in construction: a literature review, Sensors. 15 (2015) 15988–16008, https://doi.org/10.3390/s150715988.
[8] S. Tuttas, A. Braun, A. Borrmann, U. Stilla, Evaluation of acquisition strategies for image-based construction site monitoring, in: International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives, 2016: pp. 733–740, doi:https://doi.org/10.5194/isprsarchives-XLI-B5-733-2016.
[9] C. Kim, B. Kim, H. Kim, 4D CAD model updating using image processing-based construction progress monitoring, Autom. Constr. 35 (2013) 44–52, https://doi.org/10.1016/j.autcon.2013.03.005.
[10] M. Ibrahim, O. Moselhi, Experimental study of wireless sensor networks for indoor construction operations, in: 31st International Symposium on Automation and Robotics in Construction and Mining, ISARC, 2014: pp. 805–812, ISBN 978-0-646-59711-9.
[11] R.K. Soman, B. Raphael, K. Varghese, A system identification methodology to monitor construction activities using structural responses, Autom. Constr. 75 (2017) 79–90, https://doi.org/10.1016/j.autcon.2016.12.006.
[12] K.K. Han, M. Golparvar-Fard, Potential of big visual data and building information modeling for construction performance analytics: an exploratory study, Autom. Constr. 73 (2017) 184–198, https://doi.org/10.1016/j.autcon.2016.11.004.
[13] A. Giretti, A. Carbonari, G. Novembri, F. Robuffo, Estimation of job-site work progress through on-site monitoring, in: 2012 Proceedings of the 29th International Symposium of Automation and Robotics in Construction, ISARC 2012, 2012, doi:https://doi.org/10.4017/gt.2012.11.02.348.704.
[14] R. Morkos, Operational Efficiency Frontier: Visualizing, Manipulating, and Navigating the Construction Scheduling State Space with Precedence, Discrete, and Disjunctive Constraints, Stanford University, PhD dissertation, 2014http://purl.stanford.edu/qz016sf9089.
[15] A. Niederliński, A Quick and Gentle Guide to Constraint Logic Programming Via ECLiPSe, Gliwice, Poland, 9788362652082, 2011.
[16] R.K. Soman, J.K. Whyte, Codification challenges for data science in construction, J. Constr. Eng. Manag. 146 (2020) 04020072, , https://doi.org/10.1061/(ASCE)CO.1943-7862.0001846.
[17] H. Moon, H. Kim, V.R. Kamat, L. Kang, BIM-based construction scheduling method using optimization theory for reducing activity overlaps, Journal of Computing in Civil Engineering. 29 (2015), doi:https://doi.org/10.1061/(ASCE)CP.1943-5487.0000342.
[18] B. Choi, H.-S. Lee, M. Park, Y.K. Cho, H. Kim, Framework for work-space planning using four-dimensional BIM in construction projects, J. Constr. Eng. Manag. 140 (2014) 04014041, , https://doi.org/10.1061/(ASCE)CO.1943-7862.0000885.
[19] H. Liu, M. Al-Hussein, M. Lu, BIM-based integrated approach for detailed construction scheduling under resource constraints, Autom. Constr. 53 (2015) 29–43, https://doi.org/10.1016/j.autcon.2015.03.008.
[20] Y. Arayici, T. Fernando, V. Munoz, M. Bassanino, Interoperability specification development for integrated BIM use in performance based design, Autom. Constr. 85 (2018) 167–181, https://doi.org/10.1016/j.autcon.2017.10.018.
[21] J.F. Tchouanguem Djuedja, M.H. Karray, B.K. Foguem, C. Magniont, F.H. Abanda, Interoperability challenges in Building Information Modelling (BIM), in: Proceedings of the I-ESA Conferences, Springer International Publishing, 2019: pp. 275–282, doi:https://doi.org/10.1007/978-3-030-13693-2_23.
[22] J. Wang, W. Shou, X. Wang, P. Wu, Developing and evaluating a framework of total constraint management for improving workflow in liquefied natural gas construction, Constr. Manag. Econ. 34 (2016) 859–874, https://doi.org/10.1080/01446193.2016.1227460.
[23] C. Bizer, T. Heath, T. Berners-Lee, Linked data - the story so far, Int. J. Semant. Web Inf. Syst. 5 (2009) 1–22, https://doi.org/10.4018/jswis.2009081901.
[24] P. Pauwels, S. Zhang, Y.-C. Lee, Semantic web technologies in AEC industry: a literature overview, Autom. Constr. 73 (2017) 145–165, https://doi.org/10.1016/J.AUTCON.2016.10.003.
[25] J. Davies, D. Fensel, F. Van Harmelen, Towards the Semantic Web: Ontology-Driven Knowledge Management, J. Wiley (2003) 9780470848678.
[26] T.A. Ei-Diraby, C. Lima, B. Feis, Domain taxonomy for construction concepts: toward a formal ontology for construction knowledge, J. Comput. Civ. Eng. 19 (2005) 394–406, https://doi.org/10.1061/(ASCE)0887-3801(2005)19:4(394).
[27] J. Corman, J.L. Reutter, O. Savković, Semantics and Validation of Recursive SHACL, in: International Semantic Web Conference (ISWC) 2018, Springer, Cham, 2018, pp. 318–336, https://doi.org/10.1007/978-3-030-00671-6_19.
[28] Y. Kalfoglou, Cases on Semantic Interoperability for Information Systems Integration, IGI Global (2010), https://doi.org/10.4018/978-1-60566-894-9.
[29] A. Mossman, Last Planner®: 5 + 1 Crucial & Collaborative Conversations for Predictable Design & Construction Delivery, Lean Construction Institute, http://bit.ly/LPS-5cc, (2013) , Accessed date: 17 November 2019.
[30] N. Dong, M. Fischer, Z. Haddad, R. Levitt, A method to automate Look-Ahead Schedule (LAS) generation for the finishing phase of construction projects, Autom. Constr. 35 (2013) 157–173, https://doi.org/10.1016/j.autcon.2013.05.023.
[31] F.R. Hamzeh, E. Zankoul, C. Rouhana, How can 'tasks made ready' during look-ahead planning impact reliable workflow and project duration? Constr. Manag. Econ. 33 (2015) 243–258, https://doi.org/10.1080/01446193.2015.1047878.
[32] R. Sacks, R. Barak, B. Belaciano, U. Gurevich, E. Pikas, KanBIM workflow management system: prototype implementation and field testing, Lean Construction Journal. (2013): pp. 19–35. https://www.leanconstruction.org/media/docs/lcj/2013/LCJ_12_004.pdf (accessed November 30, 2019).
[33] B. Dave, Developing a Construction Management System Based on Lean Construction and Building Information Modelling, University of Salford, PhD dissertation, 2013http://usir.salford.ac.uk/id/eprint/30820/ , Accessed date: 21 June 2020.
[34] M.H. Song, M. Fischer, P. Theis, Field study on the connection between BIM and daily work orders, J. Constr. Eng. Manag. 143 (2017) 06016007, , https://doi.org/10.1061/(ASCE)CO.1943-7862.0001267.
[35] S. Russell, P. Norvig, Artificial Intelligence - a modern approach, 4th edition, Prentice Hall, New Jersey, USA, 0-13-461099-7, 2003.
[36] M. Bali, J. Taylor, S. Larbi, Drools JBoss Rules 5.0 developer's guide: develop rules-based business logic using the Drools platform, Packt Pub, 2009, ISBN 1847195644.
[37] C. Eastman, J. Lee, Y. Jeong, J. Lee, Automatic rule-based checking of building designs, Autom. Constr. 18 (2009) 1011–1033, https://doi.org/10.1016/j.autcon.2009.07.002.
[38] W. Solihin, J. Dimyadi, Y.-C. Lee, C. Eastman, R. Amor, The critical role of accessible data for BIM-based automated rule checking systems, in: Lean and Computing in Construction Congress - Volume 1: Proceedings of the Joint Conference on Computing in Construction, Heriot-Watt University, Edinburgh, 2017: pp. 53–60, doi:10.24928/JC3-2017/0161.
[39] T. OASIS LegalRuleML, LegalRuleML core specification version 1.0, (2018). http://docs.oasis-open.org/legalruleml/legalruleml-core-spec/v1.0/legalruleml-core-spec-v1.0.html (accessed November 17, 2019).
[40] E. Hjelseth, N. Nisbet, Exploring Semantic Based Model Checking, in: Proceedings of the 27th CIB W78 International Conference, Cairo, Egypt, 2010, pp. 16–18 (ISSN: 2706-6568).
[41] Red Hat Inc, Drools, (2019). https://www.drools.org/ (accessed November 17, 2019).
[42] World Wide Web Consortium (W3C), SWRL: A semantic web rule language combining OWL and RuleML, (2004). https://www.w3.org/Submission/SWRL/ (accessed November 17, 2019).
[43] W. Solihin, C. Eastman, Classification of rules for automated BIM rule checking development, Autom. Constr. 53 (2015) 69–82, https://doi.org/10.1016/j.autcon.2015.03.003.
[44] X. Tan, A. Hammad, P. Fazio, Automated code compliance checking for building envelope design, J. Comput. Civ. Eng. 24 (2010) 203–211, https://doi.org/10.1061/(ASCE)0887-3801(2010)24:2(203).
[45] N.O. Nawari, Automating codes conformance, J. Archit. Eng. 18 (2012) 315–323, https://doi.org/10.1061/(ASCE)AE.1943-5568.0000049.
[46] J.K. Lee, C. Eastman, Y.C. Lee, Implementation of a BIM domain-specific language for the building environment rule and analysis, Journal of Intelligent and Robotic Systems: Theory and Applications. 79 (2015) 507–522, https://doi.org/10.1007/s10846-014-0117-7.
[47] T. Bloch, R. Sacks, Comparing machine learning and rule-based inferencing for semantic enrichment of BIM models, Autom. Constr. 91 (2018) 256–272, https://doi.org/10.1016/j.autcon.2018.03.018.
[48] J.P. Martins, A. Monteiro, LicA: a BIM based automated code-checking application for water distribution systems, Autom. Constr. 29 (2013) 12–23, https://doi.org/10.1016/j.autcon.2012.08.008.
[49] S. Zhang, J. Teizer, J.K. Lee, C.M. Eastman, M. Venugopal, Building Information Modeling (BIM) and safety: automatic safety checking of construction models and schedules, Autom. Constr. 29 (2013) 183–195, https://doi.org/10.1016/j.autcon.

2012.05.006.
[50] K. Schwabe, S. Liedtke, M. König, J. Teizer, BIM-based construction site layout planning and scheduling, in: 16th International Conference on Computing in Civil and Building Engineering, 2016: pp. 679–686, ISBN 978-4-9907371-2-2.
[51] S. Zhang, F. Boukamp, J. Teizer, Ontology-based semantic modeling of construction safety knowledge: towards automated safety planning for job hazard analysis (JHA), Autom. Constr. 52 (2015) 29–41, https://doi.org/10.1016/j.autcon.2015.02.005.
[52] B.T. Zhong, L.Y. Ding, H.B. Luo, Y. Zhou, Y.Z. Hu, H.M. Hu, Ontology-based semantic modeling of regulation constraint for automated construction quality compliance checking, Autom. Constr. 28 (2012) 58–70, https://doi.org/10.1016/j.autcon.2012.06.006.
[53] K. Kim, Y.K. Cho, Construction-specific spatial information reasoning in building information models, Adv. Eng. Inform. 29 (2015) 1013–1027, https://doi.org/10.1016/J.AEI.2015.08.004.
[54] T. Berners-Lee, Linked Data - Design Issues, https://www.w3.org/DesignIssues/LinkedData.html, (2006) , Accessed date: 14 January 2019.
[55] C. Zhang, J. Beetz, Querying linked building data using SPARQL with functional extensions, 11th European Conference on Product & Process Modeling, 2016 9781315386904.
[56] H. Liu, M. Lu, M. Al-Hussein, Ontology-based semantic approach for construction-oriented quantity take-off from BIM models in the light-frame building industry, Adv. Eng. Inform. 30 (2016) 190–207, https://doi.org/10.1016/j.aei.2016.03.001.
[57] J. Beetz, J. Van Leeuwen, B. De Vries, IfcOWL: a case of transforming EXPRESS schemas into ontologies, artificial intelligence for engineering design, Analysis and Manufacturing: AIEDAM. 23 (2009) 89–101, https://doi.org/10.1017/S0890060409000122.
[58] K. Kim, H. Kim, W. Kim, C. Kim, J. Kim, J. Yu, Integration of IFC objects and facility management work information using semantic web, Autom. Constr. 87 (2018) 173–187, https://doi.org/10.1016/j.autcon.2017.12.019.
[59] W. Terkaj, G.F. Schneider, P. Pauwels, Reusing Domain Ontologies in Linked Building Data : The Case of Building Automation and Control, in: Proceedings of the 8th Workshop Formal Ontologies Meet Industry, Joint Ontology Workshops, http://ceur-ws.org/Vol-2050/FOMI_paper_4.pdf, (2017) , Accessed date: 25 January 2020.
[60] P. Pauwels, R. De Meyer, J. Van Campenhout, Interoperability for the design and construction industry through semantic web technology, In: Declerck T., Granitzer M., Grzegorzek M., Romanelli M., Rüger S., Sintek M. (eds) Semantic Multimedia. SAMT 2010. Lecture Notes in Computer Science, Vol 6725. Springer, Berlin, Heidelberg: pp. 143–158, doi:https://doi.org/10.1007/978-3-642-23017-2_10.
[61] P. Pauwels, D. Van Deursen, R. Verstraeten, J. De Roo, R. De Meyer, R. Van De Walle, J. Van Campenhout, A semantic rule checking environment for building performance checking, Autom. Constr. 20 (2011) 506–518, https://doi.org/10.1016/j.autcon.2010.11.017.
[62] World Wide Web Consortium (W3C), RDF Schema 1.1, (2014.) https://www.w3.org/TR/rdf-schema/ (accessed January 25, 2020).
[63] World Wide Web Consortium (W3C), OWL 2 Web Ontology Language Overview, https://www.w3.org/TR/owl2-overview/, (2012) , Accessed date: 25 January 2020.
[64] World Wide Web Consortium (W3C), Rule Interchange Format (RIF), https://www.w3.org/TR/rif-overview/, (2013) , Accessed date: 25 January 2020.
[65] T. Berners-Lee, D. Connolly, L. Kagal, Y. Scharf, J. Hendler, N3Logic: A logical framework for the World Wide Web, Theory and Practice of Logic Programming. (2008): pp. 249–269, doi:https://doi.org/10.1017/S1471068407003213.
[66] J. Gómez-Romero, F. Bobillo, M. Ros, M. Molina-Solana, M.D. Ruiz, M.J. Martín-Bautista, A fuzzy extension of the semantic building information model, Autom. Constr. 57 (2015) 202–212, https://doi.org/10.1016/j.autcon.2015.04.007.
[67] World Wide Web Consortium (W3C), SPIN - overview and motivation, https://www.w3.org/Submission/2011/SUBM-spin-overview-20110222/, (2011) , Accessed date: 6 March 2020.
[68] World Wide Web Consortium (W3C), Shapes Constraint Language (SHACL), https://www.w3.org/TR/shacl/, (2017) , Accessed date: 25 January 2020.
[69] F.J. Ekaputra, Xiashuo Lin, SHACL4P: SHACL constraints validation within Protégé ontology editor, in: 2016 International Conference on Data and Software Engineering (ICoDSE), IEEE, 2016: pp. 1–6, doi:https://doi.org/10.1109/ICODSE.2016.7936162.
[70] H. Knublauch, D. Allemang, S. Steyskal, SHACL Advanced Features, W3C Working Group, https://www.w3.org/TR/shacl-af/, (2017) , Accessed date: 25 April 2019.
[71] BuildingSmart, Dataset Schependomlaan, (2018), https://github.com/buildingSMART/Sample-Test-Files/tree/master/IFC%202x3/Schependomlaan (accessed July 13,2020).
[72] T. Hartmann, A grassroots model of decision support system implications by construction project teams, PhD dissertation, Stanford University, 2008 https://stacks.stanford.edu/file/druid:gv950qy3147/WP108.pdf (accessed April 25,2020).
[73] F. Radulovic, M. Poveda-Villalón, D. Vila-Suero, V. Rodríguez-Doncel, R. García-Castro, A. Gómez-Pérez, Guidelines for linked data generation and publication: an example in building energy consumption, Autom. Constr. 57 (2015) 178–187, https://doi.org/10.1016/j.autcon.2015.04.002.
[74] World Wide Web Consortium (W3C), Best practice recipes for publishing RDF vocabularies, https://www.w3.org/2001/sw/BestPractices/VM/http-examples/2006-01-18/, (2006) , Accessed date: 19 June 2020.
[75] R. Dechter, Constraint Processing, Morgan Kaufmann, US, 978-1-55860-890-0, 2003, https://doi.org/10.1016/B978-1-55860-890-0.X5000-2.
[76] R.K. Soman, Modelling construction scheduling constraints using Shapes Constraint Language (SHACL), in: 2019 European Conference on Computing in Construction, Chania, Greece, 2019: pp. 351–358, doi:10.35490/EC3.2019.170.
[77] A. Darwiche, R.E. Levitt, B. Hayes-Roth, OARPLAN: generating project plans by reasoning about objects, actions and resources, artificial intelligence for engineering, design, Analysis and Manufacturing. 2 (1988) 169–181, https://doi.org/10.1017/S0890060400000639.
[78] buildingSMART, Industry Foundation Classes Release 4 (IFC4), https://standards.buildingsmart.org/IFC/RELEASE/IFC4/FINAL/HTML/, (2017) , Accessed date: 30 June 2020.
[79] C.A. Baykan, M.S. Fox, Spatial synthesis by disjunctive constraint satisfaction, Artificial Intelligence for Engineering Design Analysis and Manufacturing. 11 (1997) 245–262, https://doi.org/10.1017/S0890060400003206.
[80] K. El-Rayes, D.H. Jun, Optimizing resource leveling in construction projects, Journal of Construction Engineering and Management. 135 (2009): pp. 1172–1180, doi:(ASCE)CO.1943–7862.0000097.
[81] J. Hu, I. Flood, A Multi-Objective Scheduling Model for Solving the Resource-Constrained Project Scheduling and Resource Leveling Problems, in: Computing in Civil Engineering (2012), American Society of Civil Engineers, Reston, VA, 2012, pp. 49–56, https://doi.org/10.1061/9780784412343.0007.
[82] UnityTechnologies, 3D software for architecture, engineering & construction | Unity, (2019), https://unity.com/solutions/architecture-engineering-construction (accessed June 16, 2020).
[83] R.K. Soman, D. Birch, J.K. Whyte, Framework for shared visualization and real-time information flow to the construction site, Digital Proceedings of the 24th EG-ICE International Workshop on Intelligent Computing in Engineering, 2017 9781510843455.
[84] S. Friston, C. Fan, J. Doboš, T. Scully, A. Steed, 3DRepo4Unity: Dynamic Loading of Version Controlled 3D Assets into the Unity Game Engine, in: Proceedings of the 22nd International Conference on 3D Web Technology - Web3D '17, ACM Press, New York, New York, USA, 2017, pp. 1–9, https://doi.org/10.1145/3055624.3075941.
[85] Google, AngularJS — superheroic JavaScript MVW framework, (2019). https://angularjs.org/ (accessed June 16, 2020).
[86] Flask, Welcome to Flask — flask documentation (1.1.x), (2010). https://flask.palletsprojects.com/en/1.1.x/ (accessed June 16, 2020).
[87] RDFLib, rdflib 5.0.0 — rdflib 5.0.0 documentation, (2009). https://rdflib.readthedocs.io/en/stable/ (accessed June 16, 2020).
[88] pySHACL, RDFLib/pySHACL: A Python validator for SHACL, (2018). https://github.com/RDFLib/pySHACL (accessed June 16, 2020).
[89] P. Pauwels, W. Terkaj, EXPRESS to OWL for construction industry: towards a recommendable and usable ifcOWL ontology, Autom. Constr. 63 (2016) 100–133, https://doi.org/10.1016/j.autcon.2015.12.003.
[90] P. Pauwels, T. Krijnen, W. Terkaj, J. Beetz, Enhancing the ifcOWL ontology with an alternative representation for geometric data, Autom. Constr. 80 (2017) 77–94, https://doi.org/10.1016/j.autcon.2017.03.001.
[91] Unity technologies, Assets, resources and assetbundles - Unity Learn, (2019). https://learn.unity.com/tutorial/assets-resources-and-assetbundles (accessed June 16, 2020).
[92] Apache Jena, Apache Jena Fuseki, https://jena.apache.org/documentation/fuseki2/, (2020) , Accessed date: 16 June 2020.
[93] K. Schwabe, J. Teizer, M. König, Applying rule-based model-checking to construction site layout planning tasks, Autom. Constr. 97 (2019) 205–219, https://doi.org/10.1016/j.autcon.2018.10.012.
[94] D.K.H.H. Chua, K.W.W. Yeoh, A framework for construction requirements based planning utilizing constraints logic programming, In: Proceedings of 17th Annual Conference of the International Group for Lean Construction. (2009): pp. 327–336, ISBN 9574165302.
[95] F.R. Hamzeh, Improving construction workflow- the role of production planning and control, University of California Berkley, 2009, doi:10.13140/RG.2.1.1182.9286.
[96] A.O. AlSehaimi, P. Tzortzopoulos, L. Koskela, Last Planner System: experiences from pilot implementation in the Middle East, in: Proceedings of 17th Annual Conference of the International Group for Lean Construction. (2009): pp. 53–65, ISBN 9574165302.
[97] J. Liu, M. Lu, Constraint programming approach to optimizing project schedules under material logistics and crew availability constraints, J. Constr. Eng. Manag. 144 (2018) 04018049, , https://doi.org/10.1061/(ASCE)CO.1943-7862.0001507.
[98] J. Liu, Y. Liu, Y. Shi, J. Li, Solving resource-constrained project scheduling problem via genetic algorithm, J. Comput. Civ. Eng. 34 (2020) 04019055, , https://doi.org/10.1061/(ASCE)CP.1943-5487.0000874.
[99] P. Pauwels, A. Roxin, SimpleBIM : From full ifcOWL graphs to simplified building graphs, 11th European Conference on Product and Process Modelling. (2016): pp. 11–18, ISBN 1138032808.
[100] M.H. Rasmussen, P. Pauwels, M. Lefrançois, G.F. Schneider, C.A. Hviid, J. Karshøj, Recent changes in the building topology ontology, 5th Linked Data in Architecture and Construction Workshop (LDAC 2017). (2017), doi:10.13140/RG.2.2.32365.28647.