www.itcon.org - Journal of Information Technology in Construction - ISSN 1874-4753

INTEGRATING IFC AND NLP FOR AUTOMATING CHANGE REQUEST VALIDATIONS

SPECIAL ISSUE: **Virtual, Augmented and Mixed: New Realities in Construction**
PUBLISHED: December 2019 at https://www.itcon.org/2019/30
EDITORS: McMeel D. & Gonzalez V. A.
DOI: 10.36680/j.itcon.2019.030

Huda Dawood, Senior Research Associate
School of Computing, Engineering and Digital Technologies (SCEDT), Teesside University; h.dawood@tees.ac.uk

Jonathan Siddle
Applied Integration Ltd
jonathan.siddle@appliedintegration.co.uk

Nashwan Dawood, Professor,
School of Computing, Engineering and Digital Technologies (SCEDT), Teesside University; N.N.Dawood@tees.ac.uk

SUMMARY: *The management and the identification of design changes constitute an essential part of the of a design flow within the architecture, engineering and construction (AEC) industry, requiring the formalisation of a multi-disciplinary collaborative information modelling environment. Construction projects generate substantial amount of change information, which needs to be updated continuously throughout the process, from initial feasibility study to the decommissioning of facilities. Complications arise from the information storage in multiple incompatible file formats that can lead to the loss or omission of details.*
In addition to any unexpected changes, the mismanagement of information is another factor leading to delays and costly errors. In order to mitigate such issues, this paper proposes to integrate the Industry Foundation Classes (IFC) data model and Natural Language Processing (NLP) to validate and visually identify the result of change requests. The system is developed using C# by 1) integrating angular framework with ASP.NET (Active Server Pages) to create a dynamic single web page and 2). Using X-BIM toolkit that supports IFC format to read, create and visualise the BuildingSmart Data Models (aka IFC Models). This approach enables a web-based platform capable of generating reports and visual previews, highlighting the differences between IFC files throughout the design processes.
A web-based system prototype allows users to compare subsequent versions of IFC design models in terms of additions, modifications and deletions. The prototype uses NLP to intelligently identify the changes that have been made as it compares newer and older versions of the same model, making this information available to designers and 3D modellers.
Prospective work will focus on the application of artificial intelligence (AI) to automate the implementation of changes within the construction models.

KEYWORDS: *3D models, IFC, NLP, AI, Web Services for construction.*

REFERENCE: *Huda Dawood, Jonathan Siddle, Nashwan Dawood (2019). Integrating IFC and NLP for automating change request validations. Journal of Information Technology in Construction (ITcon), Special issue: 'Virtual, Augmented and Mixed: New Realities in Construction', Vol. 24, pg. 540-552, DOI: 10.36680/j.itcon.2019.030*

1. INTRODUCTION AND RATIONALE

The digitisation of the construction industry via the building information modelling (BIM) lead to the evolution of multi-disciplinary collaborative design involving engineers, clients and architects throughout the construction process.

Design flow changes occurring in each discipline will have a subsequent impact on the process flow of other disciplines. There is a lack the awareness of the *domino effects* a design change request can have on the design of other interconnected building sections. Moreover, considering the long period taken during the design and planning phase, a large number of contributors are involved in specific detailing of the construction design, focusing on a different part of the model, leading to design errors. Ultimately, any undetected design changes (i.e. changes carried out incorrectly or incorrect identification of design errors after a change) will predominately result in increased financial costs of reworks and project overruns during the construction phase.

The cost of errors and reworks in construction varies widely by project. The financial and economic impact of these errors can account for up to 80% of the project costs (GIRI, 2016). Reworks resulting from clients' design changes or design consultant errors have been identified as the primary factor contributing to time and cost overruns (McDonalds, R, CCM, 2015). Furthermore, Love *et al.*, 2008, reported that 70% of the total amount of rework experienced in the construction and engineering mainly contributed to design induced reworks. However, with the emergence of building/project, information modelling and clash detections identifying errors and design changes became easier, thus reducing reworks.

Therefore, it is paramount that any changes or modifications within the multi-disciplinary design team are well tracked and managed, and requires and integrated recording, storage and retrieval of design changes process flow.

In fact, the top five BIM benefits cited by contractors are, *fewer errors and omissions, less rework and lower construction costs* as concluded in the key findings of the smart market report on the business value of BIM for construction in major global markets (Bernstein *et al.*, 2014).

Considerable amount of the research is focused on clash detection and conflicts between different objects from multi-disciplinary building information models. A number of software tools such as Solibri Model Checker, MagiCAD, Navisworks, Tekla Structures and Tekla BIMsight address these issues (Volk, Stengel and Schultmann, 2014). These tools support BCF (Building Collaboration Format) XML schema, which was introduced by Solibri, Inc, and Tekla in 2009. This allows workflow communication between BIM software tools. However, these software packages tend to be costly and cannot be used by all stakeholders

Our research is based on tracking design changes by using a developed web-based system prototype that allows users to compare different IFC design models in terms of addition, modifications and deletions. The benefits of using our web-based NLP integrated system are as follows:

- The system does not depend on BCF hence there is no need to use BIM collaboration software or propriety solutions,
- It is cloud based solution and
- All stakeholders can make changes to the model including non-experts.

Section 2 of this paper describes the main areas of research area on change managements and tracking design changes within a multi-disciplinary BIM environment. Section 3 describes the research methodology. Sections 4 & 5 portrays the developed 'ArchiTrack' Change Management System and explains how to create a change request. Finally, discussion, conclusions and future work are found in Sections 6 & 7.

2. LITERATURE REVIEW

Research on design changes is mainly conducted in three main areas; Design changes managements using predominantly graph theory; code compliance checking and queries; and IFC (Industry Foundation Classes) versioning. In addition, other research work falls in a combination of these areas.

There is substantial research on design change management, which involve graph theory based model for the identification of possible *domino effect* of making changes preceding their implantation in design projects (Isaac and Navon, 2013; Mattern and König, 2018). Moayeri et al, 2017, developed a model to aid owners and their

agents consider the impact of design changes by visualising the design-change and generating a detailed report for each envisioned design change and its ripple effects. Pilehchian et al, 2015, characterise design changes in an ontology to represent changed component attributes, dependencies between components, and change impacts using a graph-based approach. This method focus on alternative design changes and impact, however, it does not target changes, which is not a natural result of model development such as the removal of indefinable design errors.

Other research efforts are in the area of code compliance checking in the AECO (Architecture Engineering Construction Operation) domain and with the emergence of widely accepted IFC open standards for Building Information Modelling (BIM) data interchange, this has been made possible to resolve (Uhm et al., 2015; C. Zhang et al, 2014). , assessed this method as limited as it relies on the use of hard-coded, proprietary rules for representing regulatory requirement. Instead, they proposed a new unified automated compliance checking using NLP, EXPRESS data processing and logic reasoning but this methodology is also limited as it relies on quantitative requirements or semantic data. However, despite interest in this area, there is still a lack of holistic approaches to address the problem of automating compliance checking (Dimyadi et al, 2016).

Further research work conducted by Jalyzada et al, 2015, tackles the challenges of integrating object versioning through IFC extensions to represent the history of changes in any object in the model. However, the approach is dependent on a prototype program, which is integrated as an API in Revit, thus the user is restricted of using Revit software.

Our current research describes the development of a tool, ArchiTrack, which is designed to aid in the problems of change management by providing a recording of track changes and reporting across building/IFC model versions throughout development. This paper aims to present a prototype system to support automated change request validation. New features building on the core ArchiTrack system that explores the automatic validation of change requests using natural language and IFC files are presented. This concept is explored using examples of changing assets within a building. Natural Language Processing (NLP) is included as a core component of the system to provide a fluid interface for change requirement entry. Additionally, change request information has the potential to come from a variety of sources such as email, memos and formal requirement documents. Allowing input in natural language supports all of these inputs and reduces the time required to parse the records manually. This feature has the added advantage of allowing all stakeholders to participate.

Previous and Related Work included systems that have been developed to support the automatic validation of requirements using IFC files and pre-defined rules (Eastman *et al.*, 2009; Nguyen & Kim, 2011; Tan, et al., 2010; Li, Cai & Kamat, 2016). A major issue with these works is the requirement to parse rules manually, although work has started to address the following points:

- Anything that generates comparisons/change tracking
- Text-to-scene system in mapping natural language to visualization, link this system to text-to-scene?
- Use of shallow natural language to extract key domain knowledge

Other NLP application research are on information extraction and information retrieval. Paredes-Valverde *et al.*, 2015, used Natural Language user interface to allow non-expert users for querying linked data however, there prototype only detects the noncompliance instances but does not adjust information. Wu et al., 2019, used a natural language based intelligent retrieval engine for the BIM object database and Revit Modelling but found some limitation to their method including not all required objects could be retrieved from different disciplines.

3. RESEARCH METHODOLOGY

This research can be best described as a *problem-solving research* in which a problem from practice is identified and academic research and industrial practices in the field are combined to formulate a novel solution.

The original research was based on developing an automated platform for propagation and implementation of mechanical and electrical planning design changes in construction projects. This was later expanded to identifying any design changes throughout the information flow requirement, i.e., target the global domain of Architecture, Engineering and Construction (AEC) regarding the digitalisation and automation of information across construction project life-cycles. More specifically, in this research, a web-based system/tool is investigated and implemented by using an alternative to the BCF workflow that is currently used in practice. The motives were i)

not to rely on expensive software or proprietary solutions, ii) can be used on a mobile devise and iii) allow the participation of all stakeholders.

The paper is structured as follows: research literature review and research analysis; technology analysis and benchmarking; development of a methodology for data-driven building design platform; prototype development, implementation and testing. Currently, the prototype development is in the testing stage and is continuously being modified to match various use case studies. A use case study of *automating change request validation* using IFC and NLP is being explored in this paper.

4. THE ARCHITRACK CHANGE MANAGEMENT SYSTEM

The ArchiTrack system is a web-based application currently in development designed to aid users working on construction projects by automating the identification of changes between different versions of the same building model. The interface of the system allows users to set up a new ArchiTrack construction project, to which they can then upload different versions of building models as IFC files, generate file previews and produce a list of changes between any two model versions.

The system is developed using a combination of the Angular framework for front-end development (i.e. user interface) with the server component being developed using C♯ and ASP.NET. The ASP.NET Core App is intended to be used for data access, authorisation and other server-side concerns (Heilsberg.A, Torgersen M, wiltmmuth s., 2011). The angular app, residing in the ClientApp subdirectory, is intended to be used for all user interface concerns. The server component makes use of the xBIM library (Lockley et al, 2017) to interact with IFC files. This allows the system to work with native IFC objects directly, benefiting from implicit semantic information. The xBIM library features allow IFC models to be viewed directly in the web browser. The application was designed to be an accessible web-based system allowing users' access from as many devices as possible to use the application without needing to install any local software.

One of the key features of the system is being able to generate a report of changes between IFC file versions. This is achieved by leveraging features provided by xBIM. Furthermore, via the library, objects and object properties can be extracted from an IFC file and compared to those extracted from another IFC file to provide a list of objects that have been added, deleted or changed between the two IFC file versions. This is compiled into a report that users can view showing the individual object changes and associated properties. Combinations of Global Object Identifiers and other unique markers are used to keep track of objects between IFC file versions.

The details of IFC file comparisons are stored, thus saving time in re-generating the changes on every viewing and allows the details for instant reuse in other areas of the system, such as validating change requests. The collected information includes details of objects; type of change applied to them (added, deleted or modified) along with semantic information such as properties, coordinates and any associated spaces or levels.

Figure 1 shows an overview of the main ArchiTrack user interface with the "File Comparison" tab selected. Figure 1: ① displays a list of user-created projects (used to group related IFC files and comparisons). Tabs highlighted in Figure 1:② allow users to switch between generated comparisons and uploaded IFC files. The area in Figure 1: ③ lists generated comparisons - clicking on a comparison will take users to a detailed report showing objects that have been deleted, added or changed (grouped by level). Clicking the preview button (Figure 1:④ display a 3D model in the browser highlighting changes. Figure 1:⑤ shows an example comparison between two building versions; colours are used to designate the type of change that has occurred to objects (red for deleted, green for added or blue for changed).

The preview in Figure 1: ⑤ shows the system capability of generating a 3D comparison model as part of the detailed report detailing the generated item changes. Comparison models are generated by amalgamating geometry from two IFC files highlighting the changes. Any existing unchanged parts of the model are depicted in white, in order to highlight the elements that have been changed. The preview is intended to provide users with a quick comparison, while the detailed report is more useful for seeing the list of individual item changes. To combine these features, a preview button is included next to each object in the report. This allows users to see a specific item directly on the model itself.

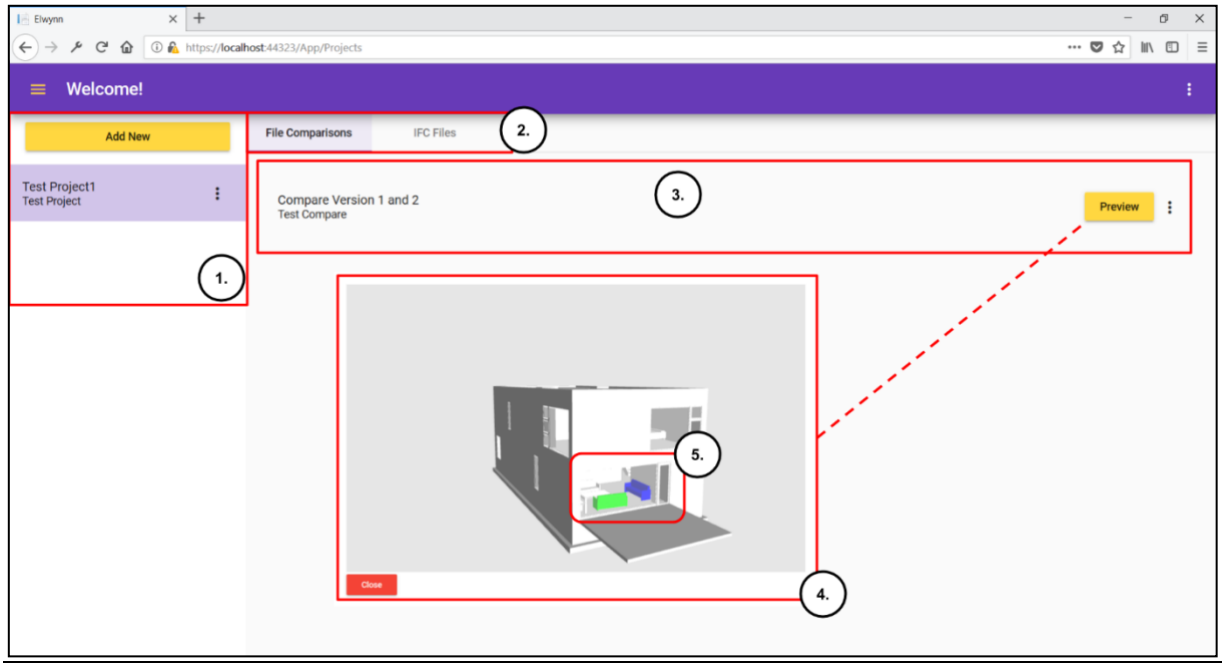

FIG. 1: Annotated screenshot from the prototype ArchiTrack software showing an overview of the main ArchiTrack interface with the "File Comparisons" tab selected. Screen capture taken from the prototype ArchTrack user interface.

The remainder of this paper will focus on the extensions made to the ArchiTrack system to improve the change management process described in section 2. Change management is a large area of research that outlines processes for handling changes within a project, aiming to improve the changes by creating a detailed record of the requests and their output (Crnkovic *et al.*, 2003). An example methodology that incorporates change management is PRINCE2 (Bentley, 2010). Within these processes, a change request is a formal record of the description of the change, usually including information such as who raised the change and if it has been completed. In software engineering, a common example of this is bug tracking software which is used to track issues in a software system; an example of such software is FogBugz (Gunderloy, 2007).

The extensions that have been developed for ArchiTrack aim to provide a way of capturing change requests as part of the overall change management process and provide an automatic method of validating their completion. Change request could potentially come from multiple sources and be completed by many individuals attached to a project. Nevertheless, we outline the core concept with a specific use case as follows:

1. Modeling : An architect develops an initial model, uploaded to ArchiTrack as IFC.
2. Change request: The architect then identifies changes that need to be made submitting a new change request to ArchiTrack and expecting the work to be completed by a contractor
3. Record update: The contractor then completes the changes and uploads a new version of the model to ArchiTrack as IFC
4. Validation: The architect can then use the ArchiTrack system to automatically validate if changes have been made
5. Verification: The architect then makes the final decision on the changes, e.g. approve changes, request more changes, etc.

An overview of this process is presented in Figure 2. This figure highlights the use case process and is designed to illustrate how users can interact through the ArchiTrack system. It also highlights the flow of information, where automated processing takes place and the information used as part of automated processing. On this figure, ArchiTrack is allocated two task routes, one for documents and one for processing tasks, in reality, these tasks are closely tied together but have been separated here to provide additional clarity.

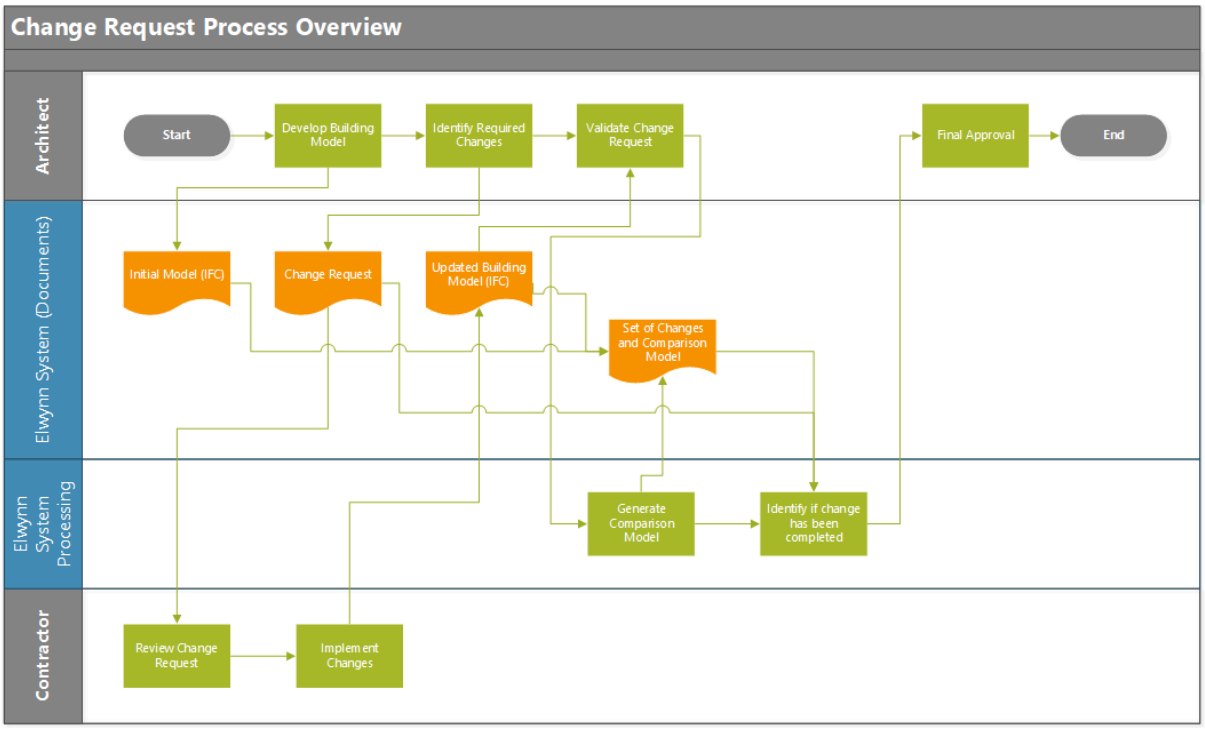

FIG. 2: Exemplar change request process overview, showing changes between an architect and contractor while interacting with the ArchTrack system.

5. CHANGE REQUEST CREATION

Automating change request creation builds upon the foundations provided by the ArchiTrack change management system through integration of a natural language component to automate the extraction of change request information from free text input. The system then uses this information to determine if requirements have been executed. Users can enter free text into the system; key information is then extracted and formalised into a change request object-structure (Minsky, 1975). The change request object structure contains properties such as the type of change request; the object associated with the change request and the location associated with the object (although this can easily be expanded and changed in future iterations). The system adopts a user-in-the-loop approach, allowing the user to edit any extracted information manually, potentially correcting any errors. For this use case study, the current version of change request validation a focus is placed on requests that refer to objects within a building, although, the plan is to consider other use case study where larger structural changes are performed in future iterations

5.1 Natural Language Processing Approach

The first step in the approach is to extract as much information as possible from the input text. To achieve this, shallow Natural Language Processing (NLP) is leveraged, which is effective at extracting domain-specific information (Åkerberg *et al*, 2003; Georg and Jaulent, 2012; Zhang and El-Gohary, 2017). In contrast with deep NLP techniques, shallow processing aims to provide a pragmatic approach, extracting key information without a deep understanding of the text. Finite State Transition Networks (FSTN) are used to analyse text, an approach inspired by the FASTUS system (Hobbs *et al.*, 1997) which, illustrated how series of FSTN applied in succession can be used to extract increasingly detailed and complex information from text.

One of the major downsides to using shallow natural language processing is that the approach can sometimes lead to brittle rules for extracting information, causing errors or extracting information that is not relevant. For this reason, leveraging the advances in statistical processing, CoreNLP (Manning *et al.*, 2014) is incorporated into the system. CoreNLP provides a feature called TokensRegex (Chang and Manning, 2014), which, allows developers

to integrate solutions that make use of FSTN. The implementation provides an interface that looks very similar to standard Regular Expressions but allows access to information provided by CoreNLP such as Part-Of-Speech (POS) Tags or word lemmatisations. This can greatly aid development as it allows FSTN to be described at a much higher level than pure lexical matches. The TokensRegex feature of CoreNLP is used to develop SUTime (Chang and Manning, 2012) library to normalise temporal expressions, which was also inspired by the FASTUS system.

Figure 3 shows an overview of the NLP approach. Successive FSTN are applied and used to annotate the input sentence with domain-specific named entities. Each stage builds on the previous: first, the nouns in the sentence are classified as objects, the type of request can then be identified using the pattern: **"*verb <object>*"**. The starting and ending locations for the object can be classified using the patterns: **"*from <object> to*"** and **"*to <object>*"** respectively. The result of this analysis is a sentence where the words are annotated with semantic information such as: the type of request, object and locations. This information is then used to populate fields in the change request form that can be used to create change requests. In the use case shown in Figure 1, this would correspond to the architect creating a new change request as part of the task "Identify Required Changes".

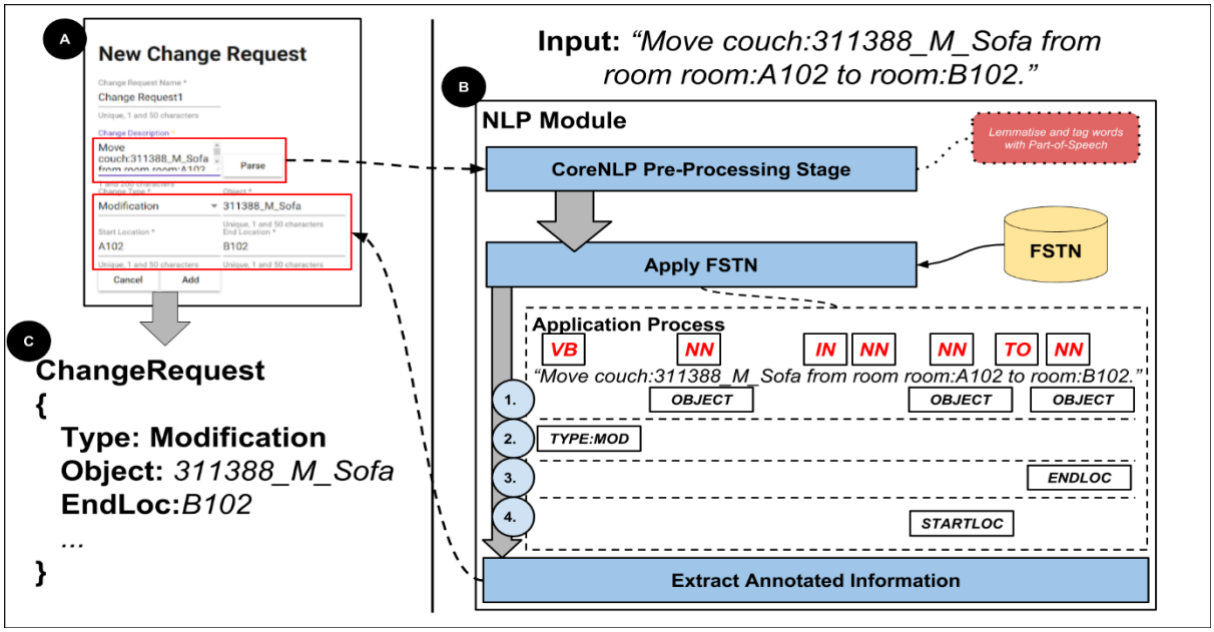

FIG 3: A) Example input as accessed form from the web-based user interface; B) Overview of the language processing component; C) Change request object extracted from input text that is also shown on the form (A) once analysis is completed.

5.1.1 FSTN Scope

A set of FSTN have been developed on input that users are expected to provide. The benefits of building this system in such a way is the ability to refine the system in the future based directly on user input and feedback.

Table 1 illustrates an example synonyms encode for different modification types within the FSTN that are used to recognise the different types of change requests, based on the three high-level types of requests that can be recognised by the system. The change request type must match one of the synonyms shown in Table 1 as well as be recognised as a verb by CoreNLP.

Table 1: Table illustrating example synonyms encode for different modification types.

Modification Type	Synonyms
Addition	Add, Create, Insert, Incorporate, Include
Modification	Alter, Adjust, Change, Refine, Revise, Move, Relocate
Deletion	Delete, Remove, Erase

Future expansion of the FSTN whilest still maintaining a robust system, the following methodology is applied Analyse new input sentences provided by users, identify instances where change requests generated from the raw input text do not match the details of the change requests submitted by users.

1. Use concordance analysis to identify new patterns that would cause the FSTN set to recognise the input as defined by the user
2. Add such new patterns to the existing set of FSTN
3. Run unit tests against existing patterns to make sure new patterns have not introduced errors
4. If errors have been introduced, either find an alternative way of integrating the new patterns or delete them.

This approach leverages the benefits of software engineering methodologies, integrating automated tests to ensure the system remains as error-free as possible even as potentially many new patterns are added to the system.

5.2 Validating a Change Request

Initial prototype evaluation was performed firstly on deletion and modification of objects and secondly on objects additions. The procedure was tested within the same design space, different design space, same level floor and different level floors. The procedure for evaluation is explained in sections 5.2.1 and 5.2.2

5.2.1 Validating deletions and modifications

Once a change request has been created, the next step is to validate the change request, analysing if the change has been completed. To achieve this, the change request is compared to a set of changes generated by ArchiTrack from two building models that have been uploaded in IFC format. From the perspective of a user, change requests are presented as a list under a tab within the main project view (shown in Figure 1:①). This allows users to see and manage existing change requests along with testing if the changes have been validated. Figure 4 shows an overview of the change validation process, highlighting a modification example. Validating a change request corresponds to the ArchiTrack System Processing task "Identify if a change has been completed" shown in Figure 2. The user interface is updated to alert the users if changes are valid or not.

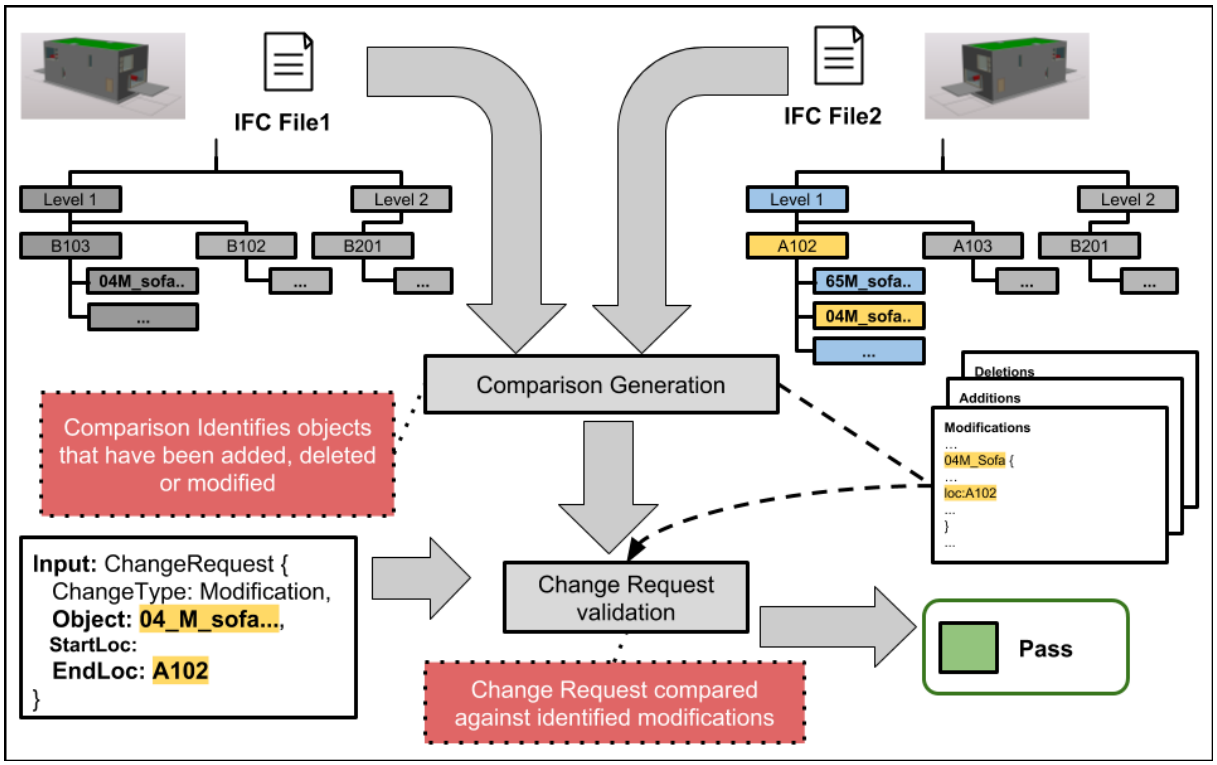

FIG. 4: An overview of the change validation process that incorporates a comparison between two IFC files.

A prototype implementation of this interface is shown in Figure 5: Ⓐ shows the details of the change request; Ⓑ allows the user to select the IFC files to validate the change request; Ⓒ runs through the validation procedure; Ⓓ visually indicates the results of the test; Ⓔ allows the user to edit or delete the change request. A user of the ArchiTrack system can initiate submitting a change request. In the use case outlined in Figure 1, this corresponds to an architect validating a change request as part of the "Validate Change Request" task.

FIG 5: Overview of change validation interface as viewed within the ArchiTrack interface.

The validation procedure builds on the foundations of the comparison generation feature that was previously implemented. This allows the system to search the comparison details between two IFC files (which are saved after the initial comparison is generated) as we include locations of objects (as an *IfcSpace*, if available). Such a subset of information is all that is needed to validate a change in many cases. Another benefit of re-using our ability to generate comparison information is that property information for objects is extracted as part of the comparison generation process. Object properties are not currently supported within change requests, but this system will be expanded in the future.

For deletions, the process consists of searching the set of deleted objects between two IFC files for the unique object identifiers. If the object(s) cannot be found in the deletions the change request will fail, else it will pass. The starting location can be optionally included as part of the change request - this prevents any IFC file that did not originally contain the object from asserting that the change request has been completed.

For modifications, the initial procedure is largely the same: the set of changes is queried for the unique object identifier; the change request will fail to be validated if the object cannot be found. If the object is found, however, an additional step is needed. Exploiting the semantic information available in the IFC we include the space in which the object is located (if available). Currently, if a space is specified in the change request but the object location cannot be found in the IFC file – for example, not being semantically associated with a space – the change request will fail. If the start location was available, it can also be used to make sure that the object was originally located in a given space.

A problem that exists in validating change requests - resolving an object entered in plain text to an object in the 3D building model - is a non-trivial problem. For example, in the sentence: "Move the couch to the upstairs bedroom." The system would need to identify both the couch and bedroom, resolving these to one instance each if multiple instances exist. In text-to-scene systems, this problem is called *grounding* (Chang, et al., 2015). While this problem is very difficult to solve through completely automated means, a pragmatic approach can be used that allows users to enter unique object identifiers into a change request easily.

The system can use object information extracted from IFC files to provide users with a list of object names and unique identifiers. These can be inserted into the change request directly, prefixed with plain English followed a colon, to improve the readability of the actual text. This allows change requests to contain the precise information needed for automated validation, without users having to enter unique object identifiers by hand.

5.2.2 Validating additions

Validating objects that have been added is more challenging, as a unique identifier will not be available at the point a change request is added, making the grounding problem unavoidable. This is the core problem in validating additions: identifying an object in the set of changes between two IFC files that matches the object specified in the change request. These features are still being implemented, but this section will outline the methodology. The semantic information included with objects in IFC files such as the name of an object is used to match them to objects in generated comparisons – to simplify processing it is assumed that the location stated in the change request exists in the previous IFC file, avoiding the problem of having to identify a new location and a new object at the same time, although this will be an area to build upon in future versions.

The examples used to develop the methodology uses object names as those contained in the IFC reference the object type. Names of objects added in the generated change request are mapped to the names of objects in the change request, this involves splitting the name into individual words such as: “m_sofa_new-edition” becoming “m”, “sofa”, “new” and “edition”, this can then be checked for instances of the object in the change request (or synonyms for the object). Objects with the highest number of instances will be identified as matches.

For instances where multiple objects are being added to the same space, the number of objects can be included. For example, if there were two new objects of the same type and only one change request for a new object of that type, the change request will still pass. Likewise, if there were only one new object and two individual change requests, they would both still pass. To avoid these situations, the user can incorporate the quantity as part of an addition change request.

Figure 6 shows an overview of the process used to match a change request object to the set of objects that have been added. The example shows an instance where the change request will match against two objects due to the quantity being specified. To test our methodology, a small set of object synonyms covering the duplex model was developed. This allows instances where a change request used a synonym for an object to match against objects in the change set. Future development will encompass iterations of the system, resource such as WordNet could be used to look-up synonyms without having to add them to the system manually (Miller, 1995).

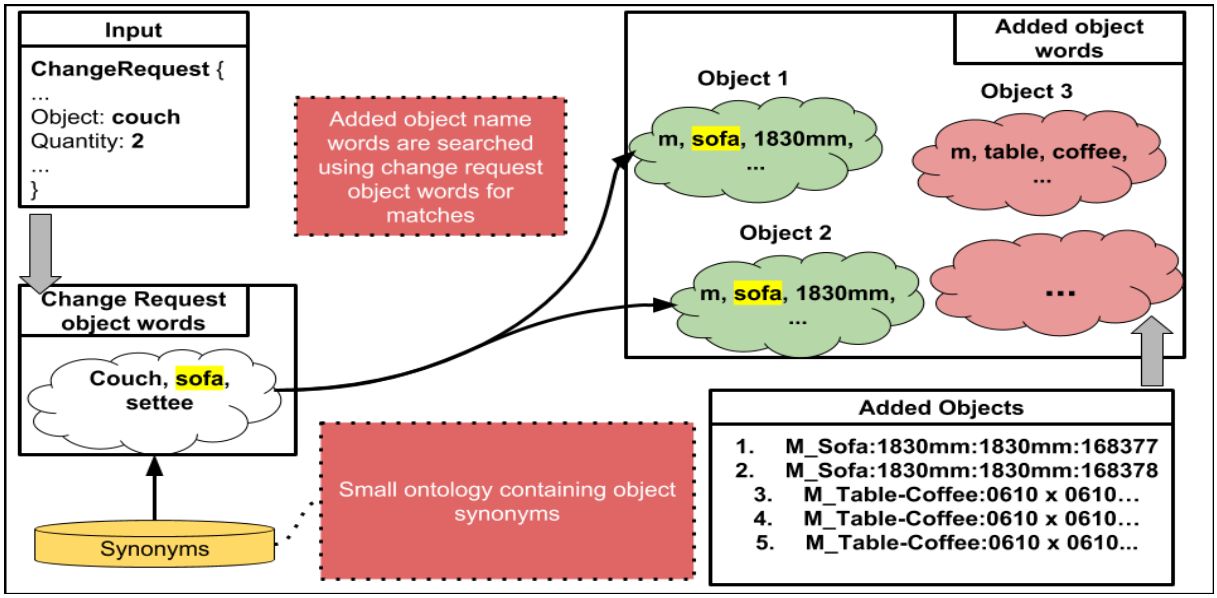

FIG 6: Overview of mapping change request objects to added objects.

6. DISCUSSION

In this research, a web-based system/tool is investigated and implemented by using an alternative to the BCF workflow that is currently used in practice. Although, there are a number of design software that follow the BCF workflow and are designed to check design compliance, design errors and changes, they need to be operated by experts, expensive and not cloud-based. Other developed research tools are mainly developed API within design software such as Auto Cad Revit hence cannot be used with other design packages. Our developed tool is a ‘standalone’ software that is not part of an add-on to an expensive design package and aim to solve real-world issues within the construction domain. The main motivation of this research was not to rely on expensive software or proprietary solutions, to facilitate the usability on-site/cloud-based using mobile devices and allowing the participation of all stakeholders.

For ease of use and portability, we are targeting a web-based application, inclusion of natural language and use of IFC as an input format to reduce the barrier to adoption. Currently, the system is focused on allowing users to compare IFC models and create change requests that can act as rules validating a change.

Validation of change requests could be used to reduce the time in checking changes (and potentially improve quality) when multiple developers are working on the same project. However, assuming these requests are kept

up-to-date, they can also be used to validate that future models are still meeting the requirements, essentially acting as Unit Tests (Jorgensen, 2013) for a project.

Evaluating individual components (such as the performance of the NLP component in extracting information) is currently a challenge due to the variations in workflow between organisations, no standardised procedure for change requests and the limited freely available data. However, our work started collaborating directly with the potential customers of the system, and additional interested industry end users are being pursued. Feedback from such collaborations will be directly incorporated into future development. This should lead to the generation of enough data to evaluate effectively the natural language component of the system, which can be further improved over time by comparing the raw text-input with the user-refined form input. In addition, once the system is available to customers, a procedure of providing general feedback will be provided.

7. CONCLUSIONS AND FUTURE WORK

The current work represents a prototype application that aims to improve automation on construction projects. Automatic change validation provides a framework that allows users to define custom rules to help users ensure consistency and quality from projects. All tested changes were validated with no errors.

Although our current prototype validation showed zero percent errors, i.e. satisfied our testing requirement, however, there is a potential to expand the change request system to incorporate more general criteria exists, e.g. when all rooms should contain a specific piece of furniture, rooms should meet specific size requirements, etc. This could lead to a system that incorporates many of the automated testing methods used in the software industry (Gamma & Beck, 2006; Smart, 2011; Jorgensen, 2013) thus, greatly reducing the time designers need to spend manually inspecting changes.

In the short-term, the goal will be to develop the change request system further, e.g. overcoming the situations when an object is not in the specified start location, which would cause a change request to fail. Resolving such issues would lead to a more granular system that provides warnings when the results may be unclear to the user or even indicates an error with the change request itself. Moreover, Object properties and space allocation are not currently supported within change requests, but future development of the prototype will endeavour to include these features.

The goal would also be to expand the natural language processing approach, with an end result to expand this approach even further to automatically extracting rules from an input specification. For this, we could leverage the use of deontic verbs, which have been shown to mark key construction information and would integrate well with our NLP system (Georg and Jaulent, 2012). Deontic verbs in English usually present the model verbs must, could, should, etc. They describe behaviour as it should be carried out and are common across many types of documents including medical, legal and guidelines.

ACKNOWLEDGEMENTS

This research was part of knowledge Transfer Partnerships with Applied Integration Limited and Teesside University, funded by Innovate UK. The authors would also like to acknowledge the contributions and continued support of all the members of KTP partnership.

REFERENCES

Åkerberg, Ola; Svensson, Hans; Schulz, Bastian; Nugues, P. (2003) 'CarSim: An Automatic 3D Text-to-Scene Conversion System Applied to Road Accident Reports', in Proceedings of the Research Notes and Demonstrations of the 10th Conference of the European Chapter of the Association of Computational Linguistics. Available at: https://lucris.lub.lu.se/ws/files/5870805/630426.pdf.

Bentley, C. (2010) PRINCE2: a practicle handbook. 3rd edn. Elsevier/Butterworth-Heinemann.

Bernstein, H. M. et al. (2014) The Business Value of BIM for Construction in Major GLobal Markets: How contractors around the world are driving innovation with Building Information Modeling, SmartMarket Report. doi: b.

Chang, A. X. and Manning, C. D. (2012) 'SUTime: A library for recognizing and normalizing time expressions.', Proceedings of the 8th International Conference on Language Resources and Evaluation, (iii), pp. 3735–3740. doi: 10.1017/CBO9781107415324.004.

Chang, A. X. and Manning, C. D. (2014) 'TOKENS REGEX : Defining cascaded regular expressions over tokens', p. 4.

Crnkovic, I. et al. (2003) 'Journal of Systems and Software: Guest editorial', Journal of Systems and Software, 65(3), pp. 169–171. doi: 10.1016/S0164-1212(02)00036-5.

Dimyadi, J., Solihin, W. and Hjelseth, E. (2016) 'Classification of BIM-based Model checking concepts', Journal of Information Technology in Construction. Department of Computer Science, 21, pp. 354–370.

Eastman, C. et al. (2009) 'Automatic rule-based checking of building designs', Automation in Construction. Elsevier B.V., 18(8), pp. 1011–1033. doi: 10.1016/j.autcon.2009.07.002.

Gamma, E. & Beck, K., 2006. JUnit. s.l.:s.n

Georg, G. and Jaulent, M.-C. (2012) 'A document engineering environment for clinical guidelines', p. 69. doi: 10.1145/1284420.1284440.

GIRI (2016) Get It Right Initiative Get It Right Initiative Literature Review, Get It Right Initiative.

Gunderloy, M., 2007. Painless project management with FogBugz. s.l.:Apress.

Heilsberg.A, Torgersen M, wiltmmuth s., G. P. (2011) C# Programming Language. 4th edn.

Hobbs, J. R. et al. (1997) 'FASTUS: A Cascaded Finite-State Transducer for Extracting Information from Natural-Language Text', pp. 1–25. Available at: http://arxiv.org/abs/cmp-lg/9705013.

Isaac, S. and Navon, R. (2013) 'A graph-based model for the identification of the impact of design changes', Automation in Construction, 31, pp. 31–40. doi: 10.1016/j.autcon.2012.11.043.

Jalyzada, A. J. et al. (no date) IFC EXTENSION FOR DESIGN CHANGE MANAGEMENT BIM for design change management View project Sustainable BIM project View project IFC EXTENSION FOR DESIGN CHANGE MANAGEMENT. Available at: https://www.researchgate.net/publication/331024954.

Jorgensen, P. C. (2013) Software Testing Fourth Edition A Craftsman's Approach. Available at: http://webpages.iust.ac.ir/azgomi/Courses/ST/eBook2 - Software Testing - A Craftsman,s Approach.pdf.

Li, S., Cai, H., Kamat, V. R., (2016)'Integrating NLP and Spatial reasoning for Utility Compliance Checking' American Scociety of Civil Engineers DOI: 10.1061/(ASCE)CO.1943-7862.0001199

Lockley, S., Benghi, C. and Černý, M. (2017) 'Xbim.Essentials: a library for interoperable building information applications', The Journal of Open Source Software, 2(20), p. 473. doi: 10.21105/joss.00473.

Love, P. E. D., Edwards, D. J. and Irani, Z. (2008) 'Forensic Project Management: An Exploratory Examination of the Causal Behavior of Design-Induced Rework', IEEE Transactions on Engineering Management, 55(2), pp. 234–247. doi: 10.1109/TEM.2008.919677.

Manning, C. D. et al., 2014. The Stanford CoreNLP Natural Language Processing Toolkit. 55--60, s.n

Mattern, H. and König, M. (2018) 'BIM-based modeling and management of design options at early planning phases', Advanced Engineering Informatics. Elsevier Ltd, 38, pp. 316–329. doi: 10.1016/j.aei.2018.08.007.

McDonalds, R, CCM, L. G. A. (2015) Root Causes & Sequential Cost Of Reworks. NewYork. Available at: https://axaxl.com/-/media/fff/pdfs/const_whitepaper_cost-of-rework_xl_june2013.pdf.

Miller, G. A., 1995. WordNet: a lexical database for English. Communications of the ACM, 38(11), pp. 39--41.

Minsky, M. (1975) 'A framework for representing knowledge, Psychology of Computer Vision, PH Winston, editor', The Psychology of Computer Vision, pp. 95–128. Available at: http://ci.nii.ac.jp/naid/10003043011/en/.

Moayeri, V., Moselhi, O. and Zhu, Z. (2017) 'Design Change Management Using BIM-based Visualization Model', International Journal of Architecture, Engineering and Construction. International Association for Sustainable Development and Management, 6(1). doi: 10.7492/ijaec.2017.001.

Murray, N., Lerner, A., Coury, F. & Taborda, C., 2016. ng-book 2: The Complete Book on Angular 2 (Volume 2). s.l.:Fullstack. io.

Nguyen, T.-H. & Kim, J.-L., 2011. Building code compliance checking using BIM technology. *Simulation Conference (WSC), Proceedings of the 2011 Winter*, pp. 3395--3400.

Paredes-Valverde, M.A., Valencia-Garcia, M. A, Colomo-Palacio, R, Alor-Hernandez, G (2015) 'A semantic-based approach for querying linked data using Natural Language', Journal of information Science, Volume 42, issue 6, 2015, p 851-862.

Pilehchian, B., Staub-French, S. and Nepal, M. P. (2015) 'A conceptual approach to track design changes within a multi-disciplinary building information modeling environment', Canadian Journal of Civil Engineering. Canadian Science Publishing, 42(2), pp. 139–152. doi: 10.1139/cjce-2014-0078.

Smart, J. (2011) Jenkins: The Definitive Guide. O'Reilly Media.

Tan, X., Hammad, A. & Fazio, P., 2010. Automated code compliance checking for building envelope design. *Journal of Computing in Civil Engineering,* Volume 24, pp. 203--211.

Uhm, M. et al. (2015) 'Requirements for computational rule checking of requests for proposals (RFPs) for building designs in South Korea', Advanced Engineering Informatics. Elsevier Ltd, 29(3), pp. 602–615. doi: 10.1016/j.aei.2015.05.006.

Volk, R., Stengel, J. and Schultmann, F. (2014) 'Building Information Modeling (BIM) for existing buildings — Literature review and future needs', Automation in Construction, 38, pp. 109–127. doi: 10.1016/j.autcon.2013.10.023.

Zhang, C.. Beetz, J., Vries D (2014) 'An Ontological Approach for Semantic Validation of IFC Models' proceeding of the 21st International workshop on Intelligent Computing in Engineering 2014, 16-18 July 2014, Caediff, Uk, P.1-8. Available at: https://www.researchgate.net/publication/266326240.

Zhang, J. and El-Gohary, N. M. (2017) 'Integrating semantic NLP and logic reasoning into a unified system for fully-automated code checking', Automation in Construction. Elsevier B.V., 73, pp. 45–57. doi: 10.1016/j.autcon.2016.08.027.