Article

Framing and Evaluating the Best Practices of IFC-Based Automated Rule Checking: A Case Study

Soroush Sobhkhiz [1], Yu-Cheng Zhou [2], Jia-Rui Lin [2,*] and Tamer E. El-Diraby [1]

[1] Department of Civil and Mineral Engineering, University of Toronto, Toronto, ON M5S 1A1, Canada; s.sobhkhiz@mail.utoronto.ca (S.S.); tamer@ecf.utoronto.ca (T.E.E.-D.)
[2] Department of Civil Engineering, Tsinghua University, Beijing 100084, China; zhouyc19@mails.tsinghua.edu.cn
* Correspondence: lin611@tsinghua.edu.cn

Abstract: This research reviews recent advances in the domain of Automated Rule Checking (ARC) and argues that current systems are predominantly designed to validate models in post-design stages, useful for applications such as e-permitting. However, such a design-check-separated paradigm imposes a burden on designers as they need to iteratively fix the fail-to-pass issues. Accordingly, the study reviews the best-practices of IFC-based ARC systems and proposes a framework for ARC system development, aiming to achieve proactive bottom-up solutions building upon the requirements and resources of end-users. To present and evaluate its capabilities, the framework is implemented in a real-life case study. The case study presents all the necessary steps that should be taken for the development of an ARC solution from rule selection and analysis, to implementation and feedback. It is explained how a rule checking problem can be broken down into separate modules implemented in an iterative approach. Results show that the proposed framework is feasible for successful implementation of ARC systems and highlight that a stable data standard and modeling guideline is needed to achieve proactive ARC solutions. The study also discusses that there are some critical limitations in using IFC which need to be addressed in future studies.

Keywords: Automated Rule Checking (ARC); Building Information Modeling (BIM); Industry Foundation Classes (IFC); proactive design; smart design

check for updates

Citation: Sobhkhiz, S.; Zhou, Y.-C.; Lin, J.-R.; El-Diraby, T.E. Framing and Evaluating the Best Practices of IFC-Based Automated Rule Checking: A Case Study. *Buildings* **2021**, *11*, 456. https://doi.org/10.3390/buildings11100456

Academic Editor: Junbok Lee

Received: 16 September 2021
Accepted: 1 October 2021
Published: 3 October 2021

1. Introduction

Automated Rule Checking (ARC) is the practice of developing tools that can automatically capture and check a set of rules against a model or design. Research into ARC goes back to the 1960s when Fenves pioneered the idea with a decision table [1]. The advent of Building Information Modeling (BIM) facilitated ARC by providing access to a considerable amount of information regarding a project in a single model. In addition, Industry Foundation Classes (IFC) promoted open BIM, creating an interoperable data platform by formalizing modeling and exchange of data. This common data standard enabled interoperability in the AEC industry and allowed BIM systems to be a platform for collaboration and data exchange. In general, ARC approaches mostly consist of all or some of the following stages: rule interpretation, model preparation, rule execution, reporting, and correction [2]. Several technical and non-technical challenges must be dealt with for automating these processes. Substantial research has been conducted to address these challenges, especially on the technical side. One trend in recent years has been applying new innovations in other fields in the automation processes of ARC, particularly in rule interpretation [3]. For instance, natural language processing (NLP) has helped in automated codification of rules or ontological approaches have been used to develop a formalized machine-readable taxonomy of rules [4,5].

However, the non-technical aspects, the business processes and stakeholder engagement, have received less attention. Given that construction projects are usually conducted in a fragmented environment, this might become a critical issue. A building is usually designed by several designers and engineers from different backgrounds (e.g., architecture and mechanical) [6]; and progress is often monitored by code experts who make sure that the design satisfies regulatory requirements. This means that the automation of rule checking should consider the views of several parties. If designers are not adequately involved in the development of an ARC system, developers might become overly focused on technical issues rather than the user experience side of the product. Currently, research is mostly focused on how to translate rules or implement rule-checking but does not address the question of "how to engage all the different stakeholders in the development process". The result of this approach has been uncoordinated progress and implementation, where ARC solutions do not really meet the industry's needs. Practitioners would prefer small, dedicated checkers to minimize or replace specific manual tasks accurately and reliably, rather than complex, all-encompassing systems [7].

However, in recent years, research is starting to answer why the extent of automation is relatively low in construction projects. More often than not, we see the stakeholder engagement as the main issue. For instance, in a recent review by Hamidavi et al. (2019), a number of experts were asked to outline existing challenges on design automation. Interestingly, the top issues identified by experts were not the technical ones and were the lack of coordination and misinterpretation. A key takeaway from their work is that "automation without supervision brings out wrong data" [8]. It is crucial to enable a collaborative environment between different disciplines in the early stages. This is especially important in the case of developers and designers as they do not speak the same language. The study also highlights that automation should allow some room for creativity, which means it should support the preliminary design stages and help designers to choose the best option to take forward for detailed design [8]. The preliminary stage is where most of the discrepancies and mistakes in hand calculations occur. Additionally, a mistake in the early stages will have a much higher impact on the whole project than one made in the final stages.

Currently, the extent of ARC applications in real projects, particularly in the private sector, is very low. One explanation can be that most of these efforts focus on the technical side of ARC, whereas there is a need to improve the practicality of these solutions and showcase the business value of ARC [7]. Currently, ARC solutions are intended for rule verification, whereas they should be more oriented towards designer needs [7]. Most ARC systems are designed to validate models in post-design stages and are useful for post-design applications such as issuing e-permits. As discussed above, to achieve a practical automation solution, ARC systems should be more involved in the preliminary stages. In practice, designers need to know the regulatory requirements in advance before designing a model (e.g., what should be the dimensions of windows); whereas the current ARC systems mostly check whether these regulations are met in a design model (e.g., whether the dimensions of windows are correct or not). Checking rules after the design is completed can guarantee the accuracy and completeness of the model. However, this design-check-separated paradigm may impose a burden on the designers because the fail-to-pass issues will result in rework, likely in an iterative way. As a result, we observe a relatively higher adoption of ARC in public sectors, where ARC is used for permits, than in private sectors. ARC systems need to be proactively providing feedback for design-specific issues during the design process, rather than passively approving or disapproving design decisions. Design decisions in the early design stages have the most profound effects on construction projects [9]. As a result, the incorporation of expert feedback in these stages will result in decreasing the discrepancies that lead to design or even construction reworks.

Studies show that ARC systems, designed based on user requirements, not only succeed in terms of rule checking but also result in additional benefits originated from cultural shifts and changes in practice. For instance, if ARC is designed and implemented successfully, it might result in wider adoption of BIM, which then improves many other aspects of design and construction [10]. Furthermore, users working with well-designed

ARC systems are more likely to gain a better tacit understanding of the codes (rules) [11]. Such studies highlight the need for research on the business and governance side of ARC applications. To this end, this research first provides a review of ARC applications. The review helps identify the main challenges associated with ARC applications. Next, based on industry best practices, a framework is proposed to categorize the main processes, stakeholders, and their interrelations in each step. The framework aims to bring ARC development as close as possible to the design and engineering team, and by doing so, result in a bottom-up approach, building upon the requirements and resources of end-users. The proposed approach is then implemented in a real project to showcase how it can coordinate different stakeholders to address development challenges. Finally, this research takes a critical standpoint and uses the review and case study to highlight the key limitations in ARC, and proposes a path for future research.

2. Literature Review

ARC has been extensively studied in the past decades. In this section, these studies are analyzed. The traditional and recent applications of ARC are reviewed, and the main issues of ARC are outlined.

2.1. Automated Rule Checking

Decision tables are the first method used for ARC applications and were first studied in the 1960s. Several studies applied this approach and added more complex knowledge-based aspects to it. For instance, Garret and Fenves [12] developed a knowledge-based standard processor for the automated design of structural components. Delis and Delis [13] developed a knowledge-based expert system for automatic fire code checking. Their proposed method includes a set of rules in IF-Then format. In fact, the follow-up approaches on Fenves's work have almost unanimously used IF-Then approaches for rule representation [14,15]. More sophisticated logic-based methods were later developed with different approaches such as object-oriented modeling, question and answer interfaces, MVD based approaches, visual programming language, and so on.

Garret and Hakim [16] presented a modeling methodology for design standards based on object-oriented modeling in which design standards are linked to the rules applicable to them. Yabuki and Law [17] combined the object-oriented paradigm with predicate logic to represent and process building codes. In another work, Kerrigan and Law [18] developed a formal infrastructure for regulatory information management by designing a document repository called REGNET, built upon an XML framework. The proposed regulation assistant system (RAS) uses regulation metadata in a web interface communicating with compliance checking systems. Model View Definitions (MVD) applies a set of rules to check the compliance of BIM data and can automatically assure that BIM has all the required data. Therefore, MVDs can be used in a variety of ARC applications. For instance, Pinheiro et al. [19] developed a standardized method based on a MVD that defines a subset of IFC related to building energy performance simulation. The proposed system can check on HVAC objects, operating schedules, controls, and simulation parameters. Finally, visual programming languages were used for rule checking mainly to address insufficient transparency and promote user incorporation. For instance, Preidel and Borrmann [20] proposed using a flow-based visual code checking language for ARC. The system is represented by graphs using elementary nodes having higher interpretation capability.

BIM and IFC improved ARC applications by allowing computers to access a considerable amount of information regarding a project in an interoperable way. For instance, Motamedi et al. integrated facilities management (FM), Knowledge Management (KM), and BIM to help technicians identify FM problems and detect failure root causes [21]. Motawa and Almarshad presented a knowledge-based BIM system that utilizes information modeling techniques to provide access to information required for maintenance works. In their study, a case-based reasoning model was used for capturing knowledge [22]. Luo and Gong developed a BIM-based code checking system to support deep foundation construction projects. They developed a library of knowledge checking and a standard of required information for these projects [23]. Martins and Monteiro developed LicA, a BIM-based ARC application for water distribution systems [24]. The system is based on Portuguese regulations. Patlakas et al. presented a BIM-based framework for ARC applications in the structural design domain. To compress the complex engineering calculations into one single equation, they implemented multi-dimensional data fitting into a BIM-based system [25].

With less attention required to assure compliance with design codes, researchers started to use ARC to include more complex aspects, such as safety and energy performance, into design processes. With all the limitations in the design stages (time and budget), this line of research gained attention in the past few years. For instance, Sulankivi et al. investigated the possibility of identifying safety hazards in the construction schedule by developing automated safety rule-checking on top of a BIM system [26]. The platform was used to detect and eliminate fall-related hazards. Cooke et al. integrated the management of occupational health and safety risks into the design process by developing a decision support tool called ToolSHeD [27]. This method deployed argument trees to represent the reasoning of experts to assess the risk of falling from height during roof maintenance work. Some of these works improved the qualitative factors of BIM-based design projects. For instance, Choi et al. developed an automated system, called InSightBIM-Evacuation, to allow designers to check the evacuation regulation compliance of BIM data within the scope of high-rise buildings [28]. Cheng and Das [29] presented a BIM-based web service framework to provide an integrated platform for ARC and building energy simulation, which allowed users to update a building model in a distributed manner.

Recently, the complexity of rule interpretation has encouraged several researchers to work on the semantics of ARC. Some studies proposed the use of ontologies for formalizing and semantically representing building codes. For example, Yurchyshyna and Zarli proposed an ontology-based approach for formalization and semantic organization of conformance requirements in construction [30]. They developed a formal representation of requirements via SPARQL queries based on an IFC-based conformance checking ontology. Beach et al. utilized regulation and domain ontologies and RASE markup techniques to allow domain experts to specify their regulatory compliance systems without the need for extensive software development [31].

Researchers also focused on developing ontologies to link the IFC data model to domain knowledge or improving the schema of the IFC itself. IfcOWL is an attempt to enrich the semantic aspect of BIM-based systems. It is an ontology written in the Web Ontology Language equal to the schema of IFC, which enables the representation of IFC attributes in the Resource Description Framework (RDF) format. The main contribution of ifcOWL to ARC applications is the advanced query language that can be used with the RDF format (e.g., SPARQL language). As a result, retrieving information from BIM files, which is a critical requirement for ARC, becomes easier [32]. Pauwels et al. identified the expression range limitations of IFC and demonstrated how the deployment of semantic web languages (namely, ifcOWL) could overcome these limitations [5].

In addition to semantic web technologies, recent studies are also investigating Natural Language Processing (NLP) for automating the formalization of specifications into rules. NLP techniques focus on analyzing information inside natural languages, such as text classification, part-of-speech tagging, chunking, etc., which benefits the automation of rule transformation. For instance, Salama and EI-Gohary utilized machine learning and NLP techniques to automatically classify the different documents and parts of documents into predefined categories to prepare them for further text analysis and rule extraction [4]. To enable a fully automated ARC, Zhang and EI-Gohary proposed a framework that mainly utilizes information Extraction (IE) and Information Transformation (ITr) for supporting ARC [33]. They also proposed a semantic, rule-based NLP approach for automated information extraction from construction regulatory documents, using a set of IE rules and conflict resolution (CR) rules [34]. By combining IE, pattern-matching-based ITr, and logic reasoning into a unified system, they presented an approach for a fully automated ARC system [35]. It should be noted that additional studies are required to investigate the limitations of these systems in wide-scale implementations.

In summary, ARC solutions provide numerous advantages, the most important of which is improving designer productivity [36]. ARC helps designers ensure that their work complies with the existing codes and as a result reduces rework for fixing non-compliant design errors. As a result, designers can focus on more important matters such as comparing different design scenarios. However, productivity is not the only benefit of ARC adoption. Given that an ARC system captures all the necessary steps needed for checking a code, it allows designers to easily understand the codes and rules they must comply with. This is especially important for cases where several rules need to be combined together. A major benefit is that designers who do not have regulatory background on the rules, do not have to spend countless hours discovering what each rule means and how to map it to other rules. This in turn improves communication between designers and other sectors such as code experts.

However, ARC comes with its own issues and disadvantages as well. To begin with, developing an ARC system requires trained experts who understand regulatory requirements and design processes, and are equipped with technical development skills. A successful development requires the participation and engagement of parties that are by nature fragmented and distanced. This might induce substantial initial investments which may not be feasible given the already tight budget that design and construction projects usually deal with. However, the challenge is not only limited to initial developments; maintaining ARC systems can become challenging too. Given the increasing technological advances in the AEC industry, an ARC product might become obsolete too soon. For instance, a change in the data formats that an ARC system is based on, might impose significant redevelopments in the design of the whole system. Additionally, developing an ARC system often requires certain assumptions based on the target rules. These assumptions might not be in line with other rules or even conflict with them. Consequently, expanding the system to include other rules is not easily done and the system becomes non-scalable. Another consequence might be that the ARC system might be entirely useless for a different region with different regulatory assumptions, and the system becomes non-generalizable.

2.2. Traditional Application of ARC: Validation

Because of the increasing demand for the efficiency of permitting processes, there has been a push for e-governance for years. As a result, traditionally, BIM-based ARC is mainly used for e-permitting processes and has progressed to be a tool for validation and post-design checking. Today, we have many national platforms that have implemented e-permitting systems based on ARC tools, a few of which are highlighted in the following.

CORENET, short for COnstruction and Real Estate NETwork, was initiated in 1995 by the Singapore Ministry of national development. The intention behind developing CORENET was to optimize the interactions between all parties involved in a building project [37]. CORENET consists of three platforms: e-Submission, e-PlanCheck, and e-Info [38]. e-Submission is for project submission and document approval, e-PlanCheck automatically checks regulations against the submitted project, and e-Info is a repository for construction-related information in Singapore. The e-PlanCheck module was introduced in 2002 for quality control of designs, which included code compliance checking [20]. The platform is implemented on top of the FORNAX library (in C++), the main component of the e-PlanCheck module. It was developed by a private company called novaCITYNETS [39]. FORNAX is a BIM-based system and can be considered one of the most successful and earliest works of ARC in regulatory compliance. Its objects include additional attributes that allow it to generate extended IFC views, necessary for code checking [2].

BCAider is a non-BIM system developed by CSIRO to support the compliance of designs against the Building Code of Australia (BCA) [40]. In another project, CSIRO collaborated with the University of Sydney to develop a BIM-based version, called DesignCheck.

The system uses IFC to transfer 3D CAD models into a DesignCheck internal model, which allows description mapping to building codes that are encoded into object-based rules using Express language. DesignCheck is also equipped with a semantic interpretation to support design performance verification and has an interactive reporting interface that offers a variety of viewing options. Initially, the system was used to support compliance with disability access codes [41].

For checking occupant circulation rules of the US court design guide, Lee et al. [42] developed an approach as a plug-in on top of a Solibri model checker (SMC) called universal circulation network (UCN), where length-weighted graphs are used to model the distance between buildings. Some earlier works pioneered automated occupant management in the area of accessibility, such as that of [43]. In their work, Han et al. (2002) proposed a hybrid approach combining perspective-based provisions and performance-based methods to support compliance analysis for accessibility regulations of the Americans with Disabilities Act (ADA) [43].

ByggSøk and SMART code are also examples of public ARC projects. ByggSøk aims to deliver better and more efficient public services and improve industrial competitiveness through a nationwide standardization of zoning proposals and building applications [44]. SMARTcode is being developed by the International Code Council (ICC), serving as a platform for rule checking and providing methods of translation from written language rules to computer code [2]. All these projects, with more than 40 years of history, are a testament to a global need for ARC. Nevertheless, there are some limitations that prevent the widespread adoption of ARC in the private sector. The next section discusses possible issues with ARC that potentially prevent its widespread adoption.

2.3. Challenges of ARC

ARC challenges range from data incompleteness and black-box solutions to non-scalable system rule conflictions. Here we summarize the most critical challenges and focus the rest of the article on possible solutions to these issues. In order to properly implement ARC, information extraction is arguably the most critical step. Every rule requires a specific set of information to be checked. For instance, rules related to the energy performance of a building require information about the HVAC systems. As a result, the level of available information in the design (or design details) has a significant impact on how successful ARC is. However, several factors, from design complexity to design errors, can contribute to an incomplete design environment with insufficient data. This will adversely affect the implementation of ARC. It is important to simplify the ability to find the required information for a feasible ARC. Even with complete designs with complete data, BIM hides the information in its complex and layered data model. Users typically have limited abilities to conduct meaningful queries and retrieve the data out of BIM. Consequently, we find different studies trying to develop more simplified data schemas for querying the BIM data [45].

Another crucial factor for the successful implementation of ARC is the knowledge of designers on the implementation processes. Designers need to know what to expect from the system and what is expected from them. Current ARC practices lack transparency as the process is mostly hard-coded checking routines. This functionality is known as black-box development [46,47]. A black-box solution is usually difficult to scale as users are not aware of the development process and developers are unfamiliar with design best practices. Scaling such solutions requires high-level programming expertise and, therefore, limits the ability of users to add/change the rules. As a result, most ARC systems are hard to maintain, as any change in the rules necessarily means a change in the ARC system. For instance, it happens quite often that a number of different rules about the same subject are not easily map-able. In fact, rule conflict is itself a major problem in ARC applications. Now, if users and developers are not in close collaboration (which is often the case), developers who are not familiar with the context of rules will find it very difficult to scale and maintain an ARC solution.

Such complex challenges originate from the lack of semantics in rules and have led researchers to use complex systems such as ontology-based systems and NLP to enrich models with semantics. These systems are hard to scale and use by the typical users in the AEC industry (engineers and architects), who are not often familiar with such concepts; therefore, their industrial scalability and implementation remain a challenge. Additionally, these systems are rarely usable in pre-design stages and are most useful in ex-post analysis. The increasing complexity of ARC systems potentially results in the lack of practical solutions for industrial problems. The consequence is that the extent of ARC applications in real projects will remain very limited.

Nevertheless, the main challenge of the adoption of ARC is not the technological side but its governance and business management side [48]; socio-organizational issues are causing considerable resistance to change in the industry. Experience with other technologies such as BIM demonstrates that the complex, fragmented structure of the industry makes it difficult to make a new technology usable. Even after the technical issues are resolved, the main challenge is convincing the industry to use it. Issues such as lack of trust in the new technology, variations in practitioners' skills, and lack of training, understanding, and awareness prevent major changes in the AEC industry, which tends to adopt traditional practices more frequently [49]. Today, many studies have advanced the technical capabilities of ARC; however, sufficient demand from the industry is not observed, which is particularly important since we now know that private clients are one of the main change agents in the AEC industry [50]. Therefore, the next frontier of research should focus on overcoming the inertia that exists in adopting ARC.

3. Methodology

This section proposes a framework for the development of ARC systems using best practices identified in the literature, and expert experiences. The framework is proposed with the consideration of challenges identified in Section 2.3 and aims to resolve them as much as possible. However, existing practices and technologies might not be adequately capable of addressing ARC development requirements. As such, it is critical to evaluate the proposed framework in the context of a real-life project and identify its limitations. The main purpose is to explore: (1) the challenges of ARC projects, (2) whether the proposed framework can establish a guideline for solving these challenges, and (3) provide a reference for future studies and implementations of ARC solutions. Figure 1 displays the key steps of the study.

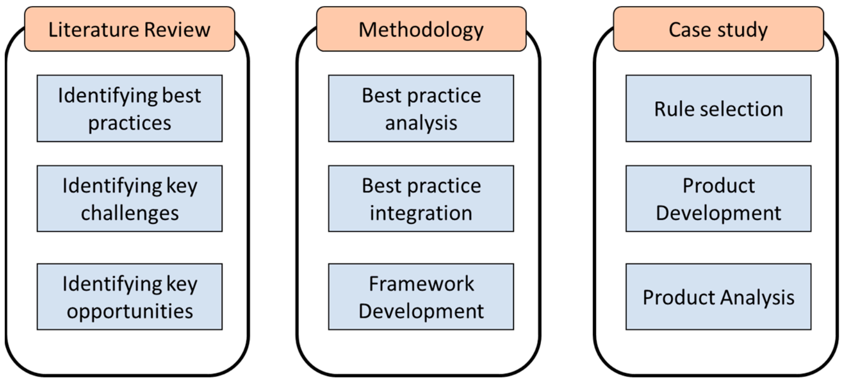


Figure 1. The overall methodology of the research.

Following the discussion provided in Section 2.3, the proposed framework must follow the requirements outlined in Table 1.

Table 1. Key requirements for developing an ARC product.

No.	Requirement	References
1	It should avoid developing black-box solutions. Designers and code experts should be aware of all processes and be dynamically involved in developing them.	[7,8,46–48,50]
2	Hard coding should be minimized to avoid hard to maintain systems.	[31,48,49]
3	Information exchange should be managed at three levels of information modeling, model exchange, and information query systems. The role of each participant in providing and retrieving information in each phase must be clear.	[5,21,22,32,33,45]
4	The final system should enable proactive design.	[20,48]

Each rule encompasses several hidden semantics that must be captured in order to be properly automated. When translating these rules into algorithms, a close collaboration between developers and code experts should be established. As such, an expert panel is needed to:

- Clarify expectations (e.g., what can/should the ARC be used for).
- Define the requirements (e.g., how much change in users' practices can be expected).
- Establish the baseline: understanding the current framework of checking a rule, the roles and responsibilities of designers and code experts, best practices, and common mistakes in checking a rule.

Having these considerations in mind, Figure 2 displays the proposed framework in this study. The framework is designed to engage the expert panel in an iterative bottom-up design process ensuring that all parties have an effective role in the process. Each iteration consists of three steps: analysis, implementation, and testing/feedback. Each iteration ends with evaluating the implemented system, resulting in a set of feedbacks analyzed in the next iteration. The outcome of each iteration is a sub-system of a more complex system developed in the next iteration. Feedback from users, developers, and test cases will be used to determine the path of the next iteration, resulting in a bottom-up approach. As demonstrated, the framework divides parties into two categories of experts and developers. The experts encompass architects, engineers, code experts, and experienced consultants either from the industry or academia. Developers include the backend developers responsible for developing algorithmic approaches to automated rules, and front-end and user experience designers. The level of engagement of each group in each of the development steps is displayed in Figure 2.

Given that developers and experts usually do not have similar perspectives on the rules, a close collaboration between developers and code experts should be established when translating rules into algorithms. It is critical for the developers to be aware of the baseline status of the project. In other words, developers need to be aware of how experts deal with rules prior to automation. In the analysis step, the experts analyze the requirements for system implementation and determine what considerations the development team should consider for rule translation and implementation. As for the rule selection, the expert panel determines the rules most valuable for automation, with the following criteria:

- Rules that are used repeatedly in a project.
- Rules that are checked by different users/designers throughout the project (requires different experts to communicate and exchange results and, therefore, is susceptible to error).
- Rules that require extensive calculations (and are, therefore, error-prone).
- Rules that incur high costs and delays if implemented incorrectly.
- Rules that have complex structures (such as having other rules embedded within them).

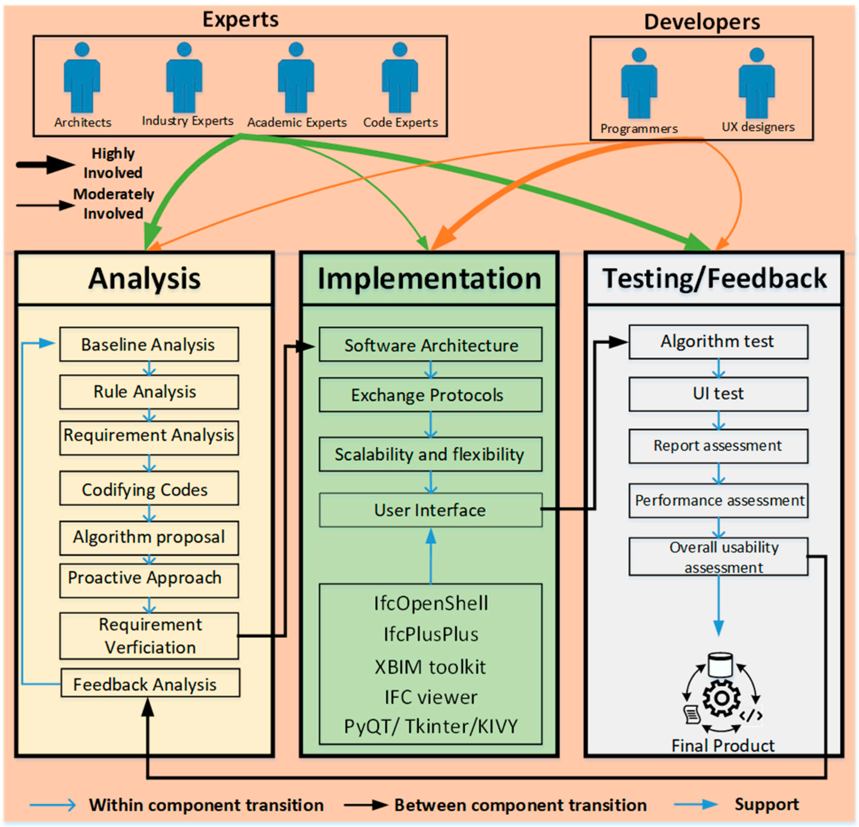


Figure 2. The proposed framework for developing proactive ARC systems.

After the rules with the highest automation value are identified, the expert panel thoroughly analyzes them. In this step, the main purpose is to achieve a common understanding of the meaning of the rules and how they are implemented. The designers and code experts will explain the best practices, hidden semantics, and the information they require in the beginning design stages to the development team. This will help the development team to better design an algorithm that captures the semantics of the rules and the best practices for checking them.

In the technical analysis, the development team will examine the information that is necessary for checking the rules (e.g., data tables and building elements). After analyzing data requirements, the development team assesses the processes and determines the parts/processes of the rule that would be valuable for automation. This is important for two reasons: (1) sometimes a single input from the user would facilitate the checking process significantly and help the development team to avoid designing complex algorithms, and (2) rules tend to change over time. This means that if every part of the rule is hard coded in an algorithm, once the rule changes, even slightly, the algorithm will have to change as well. As a result, the proposed framework identifies the main structure of rules for automation and leaves the remaining processes (with little automation value) to users. This way, the developed algorithm will be flexible enough to accept the possible changes that may occur in the future, without leaving too much work to the users.

Another purpose of this step is to determine the information exchange processes. The data required for checking a rule should first be modeled in some format in design models (e.g., BIM), and then accessed using a specific standard. Given that most of the previous works have been built on IFC, this research will also use this format as the data input for rule checking. As such, a common understanding between designers and developers is needed so that designers embed the right information in the right place, and developers capture that information using the right method. The main criterion for determining the former (how to embed the required information into BIM) is that the designers' works must be minimally interrupted. The main criteria for the second part (IFC export) are using the lightest model view (MVD) to capture only the necessary information, and keeping the exporting process as simple as possible (e.g., avoid asking users to set up complex export settings). Programming with IFC and extracting information from entities and attributes can be very complex. There are a number of open-source toolkits available that might facilitate this process. To name a few, IfcOpenShell (Python), IfcPlusPlus (C++), and xBIM toolkit (.Net) are the most well-known toolkits. IfcOpenShell is based on OpenCascade technology and it features other tools such as a blender importer, and xBIM has full support for visualization and geometric operations in desktop and web formats [51].

The technical analysis results in a generalized algorithm for automating the most valuable rule checking processes, and a standard for model design and exchange. Once the panel agrees on the algorithm, the development team proceeds to design software architecture and determines factors such as what input data is expected/needed, and how: data completeness will be checked, data will be stored, needed values will be extracted, which rules will be written and checked, and how the results will be reported. The proposed architecture should be scalable to accommodate future developments (in the following iterations). In addition, a user interface (UI) will be developed that provides an easy-to-understand report for designers. One key criterion is to design the UI in a way that is usable for both proactive design processes and post-design verifications.

In each iteration, after a subsystem is designed, it should be tested by the expert panel. Their feedback and the test results will be analyzed in the analysis component of the next iteration. The following includes the tests that will be conducted at this stage:

- Evaluate whether the tool correctly checks the rules (algorithm test).
- Assess the performance and usability of the tool (e.g., run time and user input).
- Evaluate the user interface and the generated report (e.g., easy to use features and adequate information in the report).

The testing procedure should include two components of post-design verification and pre-design requirement identification. The latter tests the proactivity of the system by testing how it can help designers identify the requirements of a design scenario according to a set of rules, before initiating the design. User feedback of this step will be analyzed in the next iteration and used to help the development team to improve/fix the tool. In addition, the testing process provides other valuable insights such as understanding the potential business values of the tool and identifying extension potentials to include other rules. In the following iteration, the issues that were identified in the previous step will be addressed, and the system performance is improved based on the feedback provided by the users; secondly, the next system components will be designed using the same procedure in the first iteration. The cycle will iterate until there is no other component to add and no issues in the existing components identified.

4. Case Study

The proposed framework is used to develop an ARC solution for an architectural firm located in Canada. Since the client is a private company, the priority was to decrease the time and errors in rule checking processes, especially in the design and planning phases. As a result, simplicity and functionality were the key requirements of the project. The project requirements fit well with the recommendations made in this article and provide a unique opportunity for evaluating those recommendations. The requested project deliverable was an IFC-based software prototype capable of capturing rules and automatically and proactively checking a proposed design against them.

4.1. Rule Selection and Analysis

The rule considered in this project is the code “3.7.4—Plumbing and drainage systems” from the Ontario Building code (OBC). The code determines plumbing fixture requirements and consists of over 100 complex interconnected rules. As an example, the following is a clause that determines water closet requirements for a specific class of buildings. “Except for motion picture theaters, the number of water closets required for Group A division

1 occupancies shall conform to table 3.7.4.3.B." In order to check this rule, one first needs to know what group A division 1 means. This refers to the classification system in clause 3.1.2.1 of the OBC. Once the building classification is identified, the relevant code for the building class and usage should be determined. These requirements are determined in a set of tables in code 3.7.4. However, the tables require the occupancy load to determine the number of washrooms. As such, another code is needed to determine the occupancy load of the building, based on which the number of water closets is determined. The occupancy load is determined according to OBC clause 3.1.17.1 and is calculated using the net floor area of the building.

Given that the process requires using different rules, it is prone to error. According to the designers, simple calculation mistakes such as miscalculating floor area can cause discrepancies between one designer's work and another. Further, different floors of a building and even different parts of the same floor can be used for different purposes (e.g., office or eatery). This requires using different rules for each part, which, in addition to being time consuming, will increase the chance of error. To provide a common understanding of the flow of checking the rule, designers and code experts walked the developers through the checking process and developed a general framework for how the rule should be checked. The proposed framework is displayed in Figure 3.

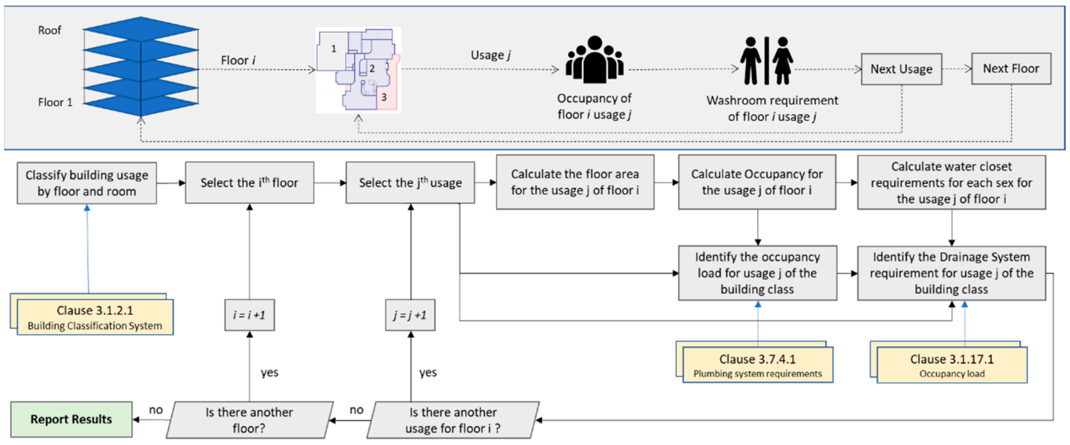

Roof
Floor 1
Floor *i*
1
2
3
Usage *j*
Occupancy of floor *i* usage *j*
Washroom requirement of floor *i* usage *j*
Next Usage
Next Floor
Classify building usage by floor and room
Select the ith floor
Select the jth usage
Calculate the floor area for the usage j of floor i
Calculate Occupancy for the usage j of floor i
Calculate water closet requirements for each sex for the usage j of floor i
Identify the occupancy load for usage j of the building class
Identify the Drainage System requirement for usage j of the building class
Clause 3.1.2.1 Building Classification System
i = *i* +1
j = *j* +1
Clause 3.7.4.1 Plumbing system requirements
Clause 3.1.17.1 Occupancy load
yes
yes
Report Results
no
Is there another floor?
no
Is there another usage for floor i ?

Figure 3. A general framework for how the rule should be checked.

4.2. *Requirement Analysis and Initial Algorithm*

This step relates to the technical assessment of rules by the development team. In this step, the development team will examine the information that is necessary for checking the rules (e.g., data tables, floor areas, and building elements). A key purpose of this step is to determine the information exchange processes. The required information should be embedded into the design models (BIM), extracted into an IFC format, and retrieved (queried) by the checking software. The requirements that are determined in the first step will play an important role. For instance, the use of the latest IFC version model views (MVD) to capture the necessary information is recommended. After analyzing data requirements, the development team assesses the processes and determines the parts/processes of the rule that would be valuable for automation. As a result, this step identifies the main structure of rules, develops a generic rule checking algorithm, and establishes information exchange standards. In this project, the following technical requirements were established:

- The rules should be checked for women's and men's washrooms separately. The results will determine other types of washrooms (e.g., universal).
- Minimal interference with the designer's usual practices.
- A simple IFC export procedure (using predefined MVDs).
- A minimum IFC file size.

- Software independency.

Once the requirements are established, the rule checking process was assessed, and key considerations were highlighted for the development team. The following considerations were identified to be embedded in the algorithm:

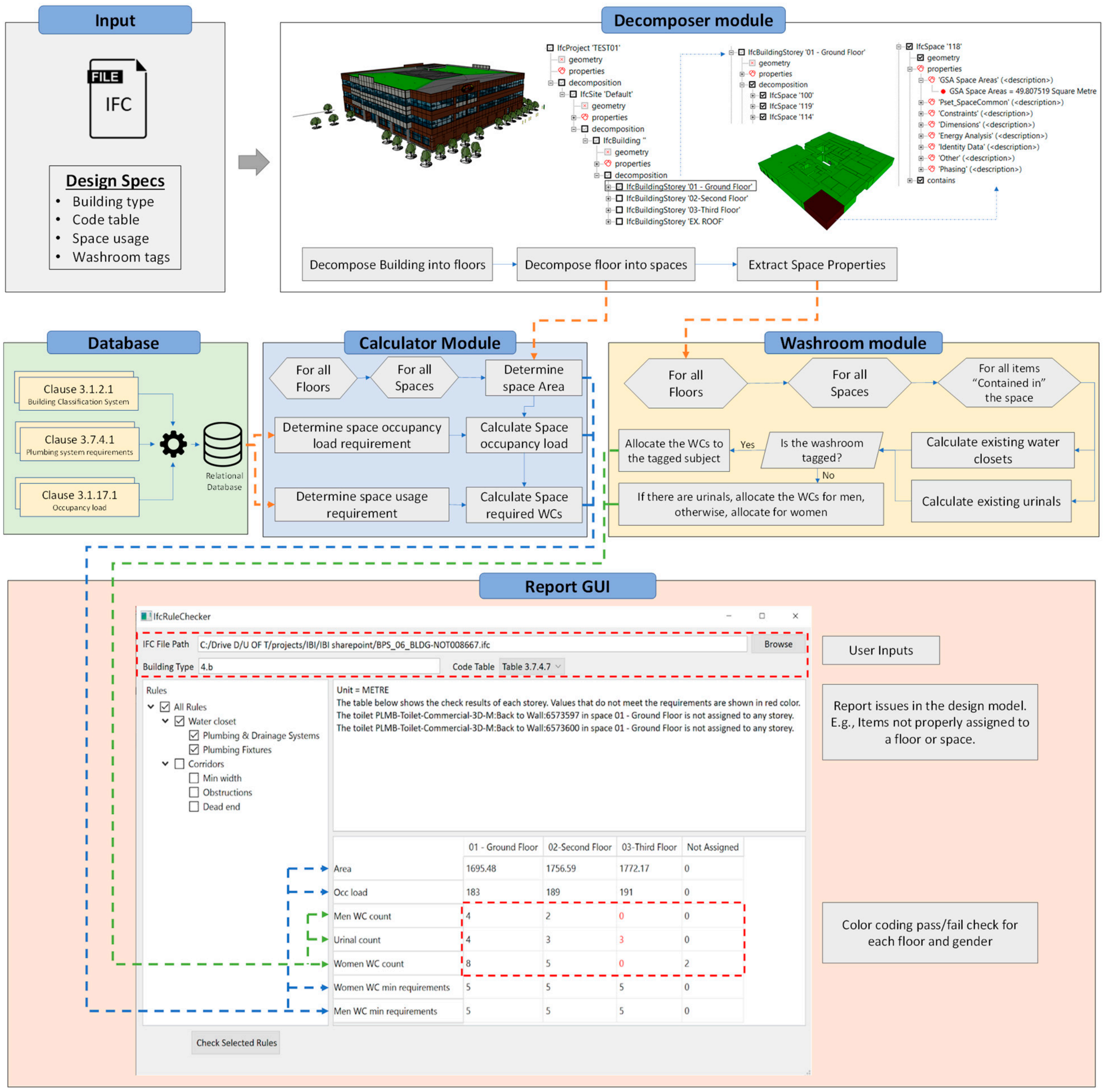
- The checking process should be able to decompose a floor into separate usage spaces so that it can account for different usages in one floor.
- For the system to be proactive, it should be able to identify the required number of water closets in advance before the design is initiated (i.e., the initial stage of the design).
- Most of the numbers are subject to change in the future. The numbers and tables should not be hardcoded, and users should be able to adjust them.
- The level of automation: Area and occupancy load calculation should be automated. Users can manually input building specific features (such as intended use).

In this project, given that only coordination and item type information is needed, the Reference View MVD is used, which has a relatively smaller size than other model views. The development team suggested a modular design of the software, in which each module is used to address a specific requirement of the final product. The proposed software architecture consisted of five main modules for (1) model decomposition, (2) occupancy load and plumbing fixture requirement calculator, (3) plumbing fixture counter, (4) rule to database convertor, and (5) rule checker and reporting module. The decomposer module is needed to break down an IFC file into floors and spaces, or a washroom module is needed to identify which spaces are used for washrooms, whether the washroom is for men, women, or other, and how many water closets/urinals are in the washroom. Figure 4 displays the modularized architecture of the proposed product by the developers. As demonstrated, the user interacts with the UI and provides preliminary information required for checking processes. The input data is then used in subsequent modules to check the rule. As displayed, the approach converts the requirements into relational databases, which are used by the checking software. Here, two main databases are used to contain the code requirement information: one for water closet requirements and the other for occupancy load. This approach provides adequate flexibility for the users to change the requirements as needed (in case of rules changing).

The proposed architecture evaluates each space by first looking for specific tags made by designers (e.g., men's washroom). However, this approach requires designers to tag and label every space which is not according to their previous practices in this project. To avoid too many interruptions, they were asked to only label a washroom that has no urinals in it, and is not intended for women. This way, tagging is reduced considerably. The software uses the "IFCContainedInSpatialStructure" relationship to automatically detect the existence of urinals in a space and assign the water closets under the men's requirements. Figure 5 provides an example of how this relationship is used in this project.

4.3. Evolutionary Development Approach

The implementation process was carried out in an iterative process (as determined by the proposed workflow), in which the software was gradually developed, tested, and refined based on the panel's feedback. In each iteration, after a subsystem was designed, it was tested by the expert panel. The evaluation included an algorithm test (whether the tool correctly checks the rules), performance test (e.g., run time), and UI test (e.g., easy to use features and readable report). The testing procedure included both post design verification applications and pre-design requirement identification. The latter tests the proactivity of the system by testing how it can help designers identify the design requirements according to a set of rules before initiating the design. In this project, the final prototype was developed in a total of six iterations. In the first step, the main purpose was to capture the main algorithm and create a basic system capable of translating the rules into code. This system lacked proactive performance. To address proactivity, the software was refined to provide a set of information (e.g., occupancy load and expected number of required water closets) based on minimum design information (such as floor perimeter). Next iterations addressed performance issues such as unit conversion, calculation errors (e.g., number rounding), and report details. In addition, a user guide was added in a "Help" tab. The product development process can be observed in Figure 6.


Input
FILE
IFC
Design Specs
Building type
Code table
Space usage
Washroom tags
Decomposer module
Decompose Building into floors
Decompose floor into spaces
Extract Space Properties
Database
Clause 3.1.2.1
Building Classification System
Clause 3.7.4.1
Plumbing system requirements
Clause 3.1.17.1
Occupancy load
Relational Database
Calculator Module
For all Floors
For all Spaces
Determine space Area
Determine space occupancy load requirement
Calculate Space occupancy load
Determine space usage requirement
Calculate Space required WCs
Washroom module
For all Floors
For all Spaces
For all items "Contained in" the space
Allocate the WCs to the tagged subject
Yes
Is the washroom tagged?
No
Calculate existing water closets
If there are urinals, allocate the WCs for men, otherwise, allocate for women
Calculate existing urinals
Report GUI
User Inputs
Report issues in the design model. E.g., Items not properly assigned to a floor or space.
Color coding pass/fail check for each floor and gender
IfcRuleChecker
Browse
Check Selected Rules

IFC File Path C:/Drive D/U OF T/projects/IBI/IBI sharepoint/BPS_06_BLDG-NOT008667.ifc

Building Type 4.b Code Table Table 3.7.4.7

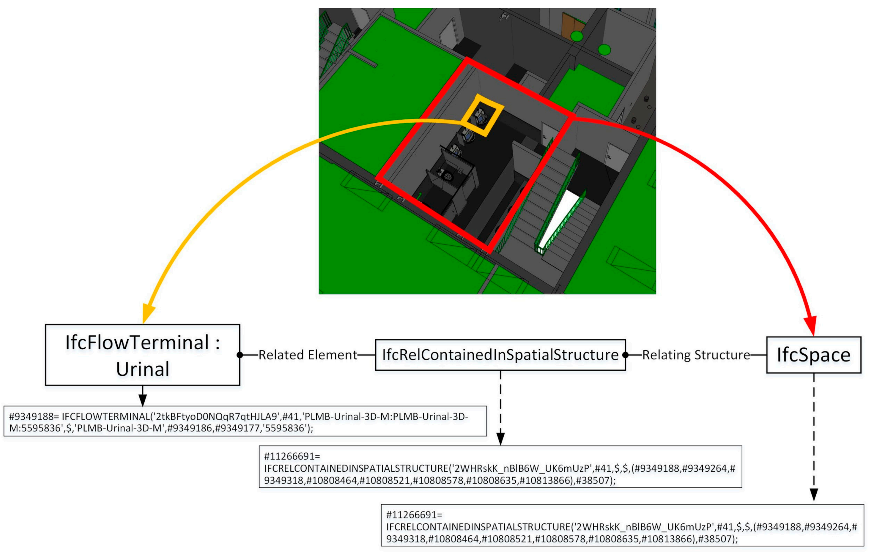
Rules
- [✓] All Rules
  - [✓] Water closet
    - [✓] Plumbing & Drainage Systems
    - [✓] Plumbing Fixtures
  - Corridors
    - Min width
    - Obstructions
    - Dead end

Unit = METRE
The table below shows the check results of each storey. Values that do not meet the requirements are shown in red color.
The toilet PLMB-Toilet-Commercial-3D-M:Back to Wall:6573597 in space 01 - Ground Floor is not assigned to any storey.
The toilet PLMB-Toilet-Commercial-3D-M:Back to Wall:6573600 in space 01 - Ground Floor is not assigned to any storey.

	01 - Ground Floor	02-Second Floor	03-Third Floor	Not Assigned
Area	1695.48	1756.59	1772.17	0
Occ load	183	189	191	0
Men WC count	4	2	0	0
Urinal count	4	3	3	0
Women WC count	8	5	0	2
Women WC min requirements	5	5	5	0
Men WC min requirements	5	5	5	0

Figure 4. Modularized architecture of the proposed product by the developers.

Figure 7 demonstrates how the system can be used proactively. In the initial stages of design, designers will only need to determine the expected building perimeter and its type (e.g., office or residential). An empty box BIM model will be sufficient for the system to calculate the expected occupancy and the required number of water closets for the building. This way, designers can avoid erroneous and difficult calculations to determine the required water closets and the ARC system will be used in the design processes to help designers (proactively).

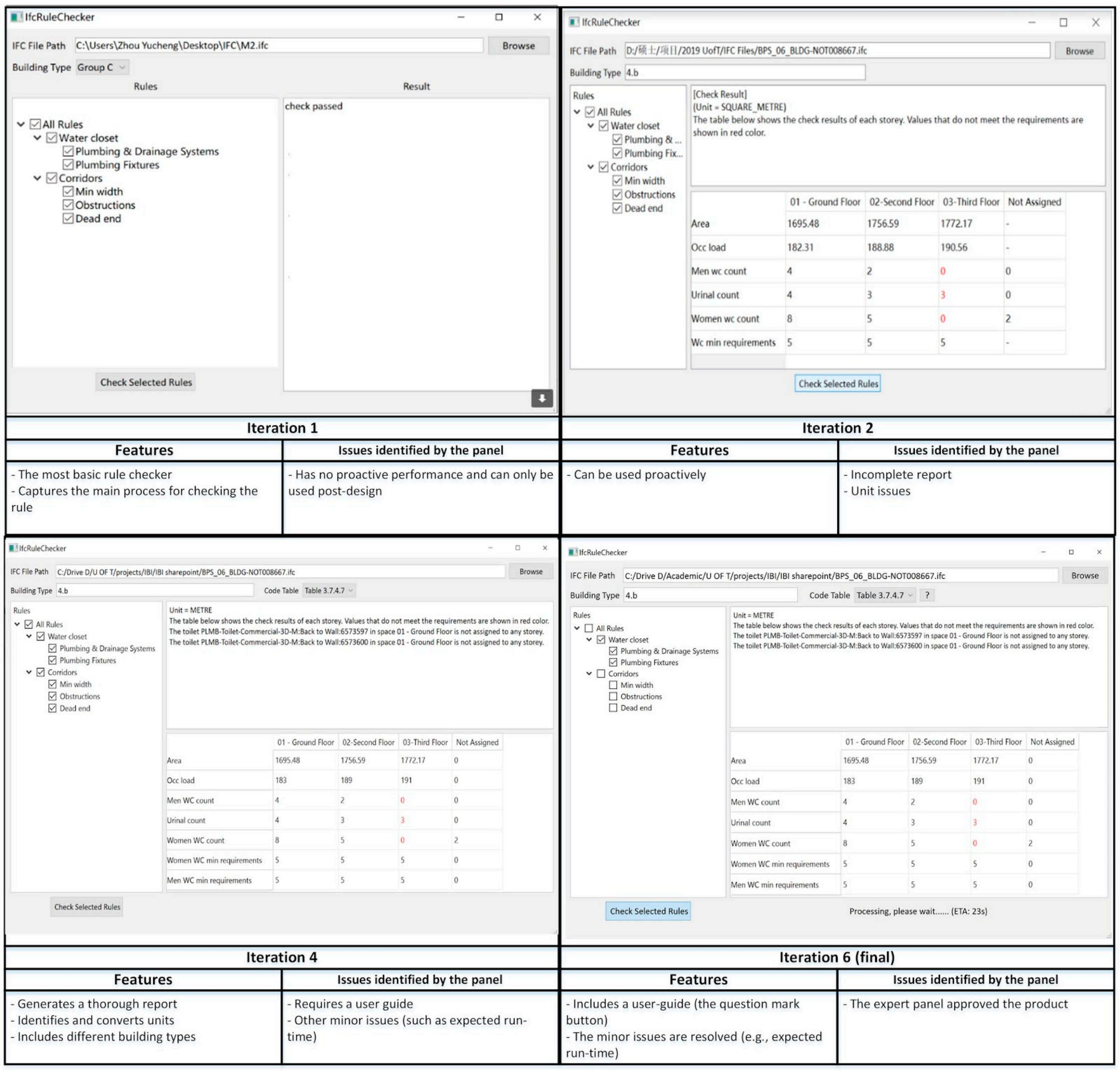


```
#9349188= IFCFLOWTERMINAL('2tkBFtyoD0NQqR7qtHJLA9',#41,'PLMB-Urinal-3D-M:PLMB-Urinal-3D-
M:5595836',$,'PLMB-Urinal-3D-M',#9349186,#9349177,'5595836');
```

```
#11266691=
IFCRELCONTAINEDINSPATIALSTRUCTURE('2WHRskK_nBlB6W_UK6mUzP',#41,$,$,(#9349188,#9349264,#
9349318,#10808464,#10808521,#10808578,#10808635,#10813866),#38507);
```

```
#11266691=
IFCRELCONTAINEDINSPATIALSTRUCTURE('2WHRskK_nBlB6W_UK6mUzP',#41,$,$,(#9349188,#9349264,#
9349318,#10808464,#10808521,#10808578,#10808635,#10813866),#38507);
```

Figure 5. An example relationship between water closets and the washrooms they are located in.



IFC File Path C:\Users\Zhou Yucheng\Desktop\IFC\M2.ifc
Building Type Group C

Rules: ☑ All Rules; ☑ Water closet; ☑ Plumbing & Drainage Systems; ☑ Plumbing Fixtures; ☑ Corridors; ☑ Min width; ☑ Obstructions; ☑ Dead end

Result: check passed

IFC File Path D:/硕士/项目/2019 UofT/IFC Files/BPS_06_BLDG-NOT008667.ifc
Building Type 4.b

Rules: ☑ All Rules; ☑ Water closet; ☑ Plumbing & ...; ☑ Plumbing Fix...; ☑ Corridors; ☑ Min width; ☑ Obstructions; ☑ Dead end

[Check Result]
(Unit = SQUARE_METRE)
The table below shows the check results of each storey. Values that do not meet the requirements are shown in red color.

	01 - Ground Floor	02-Second Floor	03-Third Floor	Not Assigned
Area	1695.48	1756.59	1772.17	-
Occ load	182.31	188.88	190.56	-
Men wc count	4	2	0	0
Urinal count	4	3	3	0
Women wc count	8	5	0	2
Wc min requirements	5	5	5	-

Iteration 1	
Features	**Issues identified by the panel**
- The most basic rule checker - Captures the main process for checking the rule	- Has no proactive performance and can only be used post-design

Iteration 2	
Features	**Issues identified by the panel**
- Can be used proactively	- Incomplete report - Unit issues

IFC File Path C:/Drive D/U OF T/projects/IBI/IBI sharepoint/BPS_06_BLDG-NOT008667.ifc
Building Type 4.b Code Table Table 3.7.4.7

Rules: ☑ All Rules; ☑ Water closet; ☑ Plumbing & Drainage Systems; ☑ Plumbing Fixtures; ☑ Corridors; ☑ Min width; ☑ Obstructions; ☑ Dead end

Unit = METRE
The table below shows the check results of each storey. Values that do not meet the requirements are shown in red color.
The toilet PLMB-Toilet-Commercial-3D-M:Back to Wall:6573597 in space 01 - Ground Floor is not assigned to any storey.
The toilet PLMB-Toilet-Commercial-3D-M:Back to Wall:6573600 in space 01 - Ground Floor is not assigned to any storey.

	01 - Ground Floor	02-Second Floor	03-Third Floor	Not Assigned
Area	1695.48	1756.59	1772.17	0
Occ load	183	189	191	0
Men WC count	4	2	0	0
Urinal count	4	3	3	0
Women WC count	8	5	0	2
Women WC min requirements	5	5	5	0
Men WC min requirements	5	5	5	0

IFC File Path C:/Drive D/Academic/U OF T/projects/IBI/IBI sharepoint/BPS_06_BLDG-NOT008667.ifc
Building Type 4.b Code Table Table 3.7.4.7 ?

Rules: ☐ All Rules; ☑ Water closet; ☑ Plumbing & Drainage Systems; ☑ Plumbing Fixtures; ☐ Corridors; ☐ Min width; ☐ Obstructions; ☐ Dead end

Unit = METRE
The table below shows the check results of each storey. Values that do not meet the requirements are shown in red color.
The toilet PLMB-Toilet-Commercial-3D-M:Back to Wall:6573597 in space 01 - Ground Floor is not assigned to any storey.
The toilet PLMB-Toilet-Commercial-3D-M:Back to Wall:6573600 in space 01 - Ground Floor is not assigned to any storey.

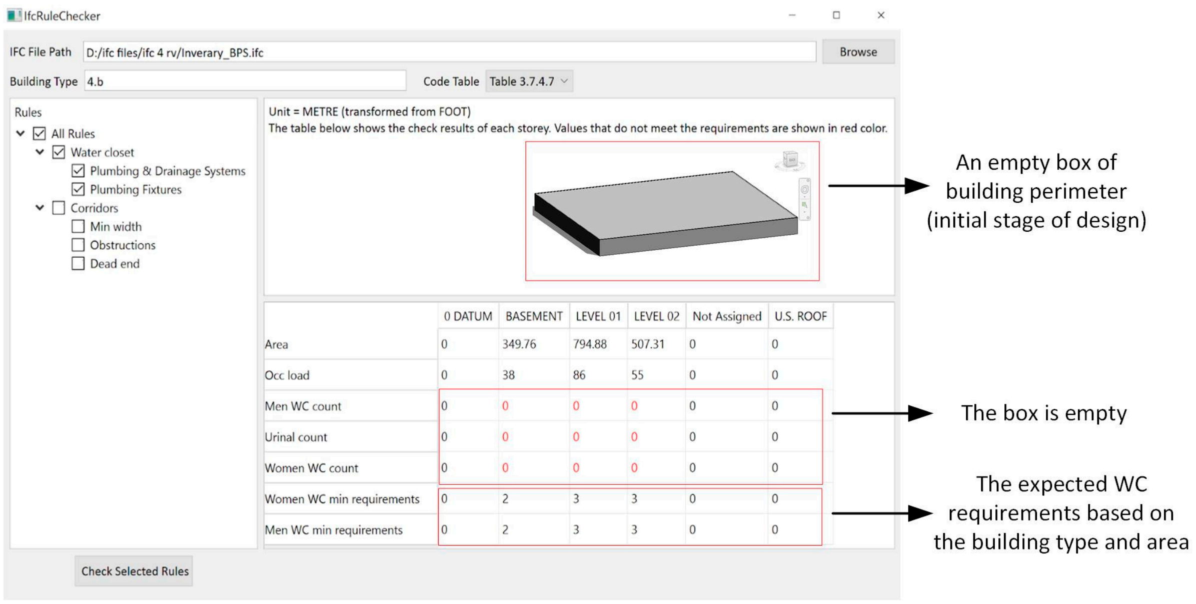
	01 - Ground Floor	02-Second Floor	03-Third Floor	Not Assigned
Area	1695.48	1756.59	1772.17	0
Occ load	183	189	191	0
Men WC count	4	2	0	0
Urinal count	4	3	3	0
Women WC count	8	5	0	2
Women WC min requirements	5	5	5	0
Men WC min requirements	5	5	5	0

Processing, please wait...... (ETA: 23s)

Iteration 4	
Features	**Issues identified by the panel**
- Generates a thorough report - Identifies and converts units - Includes different building types	- Requires a user guide - Other minor issues (such as expected run-time)

Iteration 6 (final)	
Features	**Issues identified by the panel**
- Includes a user-guide (the question mark button) - The minor issues are resolved (e.g., expected run-time)	- The expert panel approved the product

Figure 6. ARC systems, their features, and issues after iterations 1,2,4, and 6.


IfcRuleChecker
IFC File Path
D:/ifc files/ifc 4 rv/Inverary_BPS.ifc
Browse
Building Type
4.b
Code Table
Table 3.7.4.7
Rules
All Rules
Water closet
Plumbing & Drainage Systems
Plumbing Fixtures
Corridors
Min width
Obstructions
Dead end
Check Selected Rules
An empty box of building perimeter (initial stage of design)
The box is empty
The expected WC requirements based on the building type and area

Unit = METRE (transformed from FOOT)
The table below shows the check results of each storey. Values that do not meet the requirements are shown in red color.

	0 DATUM	BASEMENT	LEVEL 01	LEVEL 02	Not Assigned	U.S. ROOF
Area	0	349.76	794.88	507.31	0	0
Occ load	0	38	86	55	0	0
Men WC count	0	0	0	0	0	0
Urinal count	0	0	0	0	0	0
Women WC count	0	0	0	0	0	0
Women WC min requirements	0	2	3	3	0	0
Men WC min requirements	0	2	3	3	0	0

Figure 7. An example of using the system to promote proactive design.

5. Discussion

In the final iteration, the tool successfully worked on BIM models and a code expert from the firm approved its performance. Once the product was technically approved in the expert panel, the tool was presented to a group of five designers for a simple qualitative evaluation. Three of the designers were from Canada, one from the UK, and one from the US, all with over 5 years of experience in industrial work. In the session, the tool was showcased and then the designers were asked to participate in an open discussion on the possible advantages and challenges of the tool. Their feedback is briefly summarized in the following.

The most prominent feature that was acknowledged in the session was that the software is very useful in training the user on how to check the rules. Since the product was practically designed by designers, it has a very understandable process for another designer, who has not previously worked with the software. The proactivity of the software was also recognized as a major advantage as compared to other rule checking tools. Another advantage recognized by the audience was the ability of the software to prevent conflicts between architects in rule checking processes. Since everyone uses the same system, issues such as rule miscomprehension or calculation errors will no longer occur. The major downside mentioned by the designers was the fact that repeating the same process for a different region would require fundamental changes in the software. This is mainly because different codes use different approaches for similar issues; for instance, the American version of the same code does not calculate the occupancy load the same way. This further highlights the necessity for regulatory agencies to codify their code.

5.1. Limitations

The case study shows that a close collaboration by end-users can potentially result in successful proactive systems. Nevertheless, there were limitations as well. The main limitation can be considered the lack of a stable data standard and modeling guideline. IFC has made significant advances in terms of interoperability and data sharing, but it is not very useful by itself. EXPRESS and STEP languages do not fit well with the formal rule checking languages. As a result, even a simple query for data retrieval requires hard-coding. In addition, the large diversity of IFC formats result in case-dependent non-scalable ARC systems. This means that, eventually, an ARC system that is solely based on IFC will turn into a non-scalable product, as it might not be able to merge easily with other emerging technologies.

Furthermore, as soon as designers export an IFC file, they are disconnected from the design environment. In general, IFC toolchains push designers away from their usual design environment and prevent simple user-intuitive checks. This has two downsides. First, it complicates the process of understanding rules, and, second, it results in continuous interruptions in s designer′s workflow, no matter how fluent the process is designed. In contrast, tools such as Dynamo and Grasshopper can embed checking processes into a designer′s work environment, thus allowing them to conduct implicit checks; nevertheless, IFC remains the only software-independent solution. Perhaps, efforts in the development of ifcOWL or semantic-based solutions in general can address this problem, although this is yet to be investigated [52].

Lastly, given that this study was mostly focused on identifying critical challenges and exploring ways to address the business challenges of ARC practices, the impact of different types of rules has not been comprehensively analyzed. With this regard, Solihin and Eastman classified BIM rules based on their computational complexity [3]. The integration of their proposed classification and the proposed framework in this study should be investigated in future studies to see how adaptable the proposed framework is, and where it should be improved.

5.2. Implications

The review provided in this research implies that end-users of ARC systems are often not engaged in the development processes. The outcome is black-box non-scalable systems that cannot be easily integrated in the complex processes of a daily designer. Consequently, ARC faces several impediments when it comes to real-life implementations, especially in the private sector. These findings suggest that business requirements are just as important as the technical requirements. Herein, the integration of the different parties involved can be arguably the greatest challenge of an ARC project. For several years, IFC has been advertised as the key solution to the fragmentation issues of the AEC industry. However, the findings of this study suggest that IFC is not the silver bullet it was once considered to be. It imposes several challenges to adapt all the design processes within an IFC-based platform (Section 5.1). Given that IFC is the sole solution for a much-needed open-BIM environment, it should be used as the main information exchange technology. However, future practices should focus the encoding of the rule-checking directly into the design or common data environment (CDE) rather than into an IFC-based platform. This would be useful, in particular, if the CDE consists of a live web-based database. In other words, the rule-checking is implemented in the highest layer where information is already stored, but IFC is used as the software communication protocol. The following section further discusses the key implications for the theoretical side.

5.3. Perspectives of ARC in the Future

We are moving to an era of intelligent machines where machines understand the requirements and do the work themselves. Concepts such as smart cities, smart homes, and intelligent facilities are becoming the center of attention, and savvy companies see their business sustainability severely reliant on how much they invest in these domains. Business processes are becoming more and more automated, and rule-based systems are increasingly demanded. As a result, it can be observed that the role of technologies is no longer passive, and they are having a more active role in utilizing data-based smart systems. Technologies are now smart enough to engage in the processes prior to and during their occurrence; particularly, this can be observed in the recent advances in BIM systems. Through integration with IoT and Machine learning algorithms, BIM is becoming more intelligent, and soon it will be able to detect patterns and propose solutions all by itself. Occupants′ role is gaining more attention as it is now possible to include their behavior and preferences in the design, construction, and operation stages. At the core of such works, rule-based systems are particularly needed [48]. We can see such systems in different areas from occupant tracking systems [53] to construction site task automation [51], all of which require a set of rules to assess data and make/propose decisions.

Such technological shifts must also be reflected in how we perceive rules as a more proactive concept rather than a passive regulatory concept. ARC must be viewed not as a technical tool for quality control but as a service for automated design, on-demand analysis, and, ultimately, for establishing intelligent buildings and smart cities. ARC should be considered a form of machine learning tool, beyond an expert system. Such a standpoint pushes the value of ARC beyond just checking rules but in generating them itself. Therefore, the key value of ARC is beyond quality control, and is in its role in transforming organizations to AI initiatives, by imposing the sophisticated cultural changes necessary to develop and deploy ARC systems [48].

Arguably, the long-term perspective of the role of ARC systems in the AEC industry is complete design automation. As a result, it can be expected that future ARC studies will focus on reversing the traditional approach in which the design is checked against the rules into a new approach, in which designs will be automatically generated by the rules. The future ARC systems will be able to understand the requirements and rules, identify design constraints, and accordingly generate an optimal design scenario. In other words, rules will build the building themselves rather than just checking them. Of course, such a shift requires a significant cultural transition from passive ARC systems to active ones.

We argue that a proactive perspective is needed to shift from traditional quality-control ARC to future automated-design ARC; a perspective in which the role of ARC first transforms from a passive technology to a semi-active one as a proactive tool. Figure 8 demonstrates the three approaches of ARC implementations considered in this study. The AEC industry must create an incentive for private clients to demand and use ARC systems proactively in their daily projects. Proactive design approaches can ultimately set the path for the widespread adoption of ARC by (1) educating the AEC community, (2) pushing the technical edges of rule-based systems to active technologies from passive ones, and (3) increasing client awareness, and motivating them to invest in the next generation of bottom-up, AI-enabled ARC systems.

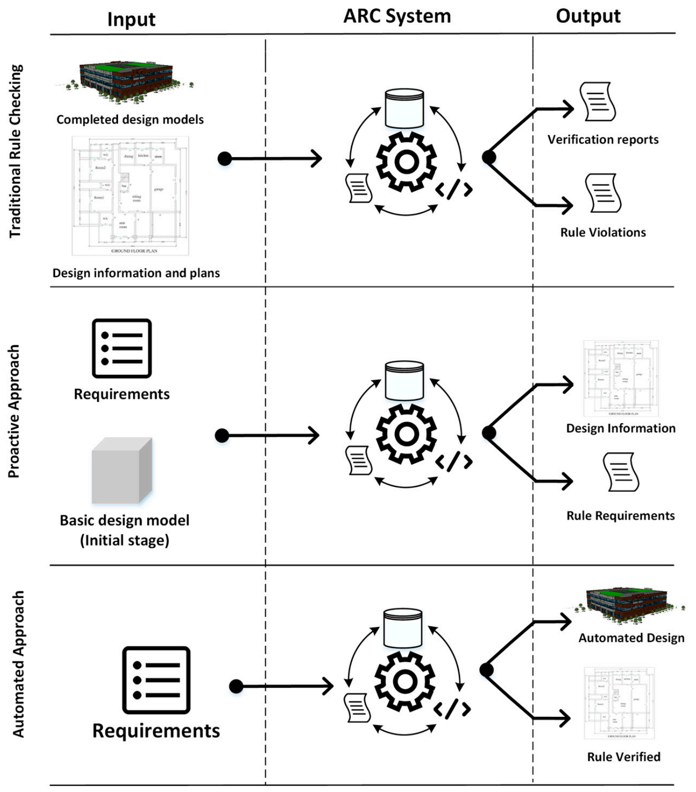


Figure 8. Three perspectives on the role of ARC in AEC applications.

6. Conclusions

In this study, the concept of IFC-based ARC and its applications in the AEC industry are analyzed. It is argued that deploying ARC systems requires sophisticated cultural changes among the AEC community, which can only be achieved through wide-scale usage of these systems. As a result, encouraging both public and private clients to adopt ARC systems as a key service in their organizations is critical. It is further discussed that because of the post-design verification nature of the current ARC systems, they mostly address the needs of public clients, and the private sector lags far behind in adopting ARC. We observe that many studies have advanced the technical capabilities of ARC; nevertheless, sufficient demand from the industry is not found. In fact, the increasing complexity of ARC systems has resulted in the lack of practical solutions for industrial problems.

As a result, arguably, the main challenge of the adoption of ARC is not the technological side, but is its governance and business management side [48]. It is not that we cannot translate rules into algorithms, it is that, with current approaches, the final product does not really serve the needs of an everyday designer. Th current design-check-separated paradigm imposes a burden on designers because, iteratively, the fail-to-pass issues will only result in rework. To this end, this study attempts to investigate the existing best practices in previous IFC-based ARC projects, and develop a framework. The framework focuses mostly on pushing ARC as close as possible to the design and engineering team. The objective is to enable a bottom-up approach, building upon the requirements and resources of end-users to achieve proactivity.

To evaluate the framework, it is implemented in a real case study to develop an ARC solution for an architectural firm located in Canada. The case study demonstrates that the proposed framework is feasible, and a close collaboration by the end-users can potentially result in successful proactive systems. The study also revealed that enabling easy understanding of rules is essential. This is especially important, given that complete automation of some rules might be difficult or impossible. In such cases, the best alternative is to enable an easy definition of rules for the end users, which the proposed framework can achieve.

Author Contributions: Conceptualization, S.S. and Y.-C.Z.; methodology, S.S.; software, Y.-C.Z.; validation, S.S. and Y.-C.Z.; formal analysis, S.S.; investigation, S.S. and Y.-C.Z.; resources, S.S.; data curation, S.S.; writing—original draft preparation, S.S. and Y.-C.Z.; writing—review and editing, S.S., Y.-C.Z. and J.-R.L.; visualization, S.S.; supervision, J.-R.L. and T.E.E.-D.; project administration, J.-R.L. and T.E.E.-D.; funding acquisition, J.-R.L. All authors have read and agreed to the published version of the manuscript.

Funding: The research was funded by the National Natural Science Foundation of China (grant number 72091512 and 51908323), and the Tsinghua University Initiative Scientific Research Program (grant number 2019Z02UOT).

Data Availability Statement: Data sharing not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Fenves, S.J. Tabular Decision Logic for Structural Design. *J. Struct. Div.* **1966**, *92*, 473–490. [CrossRef]
2. Eastman, C.; Lee, J.M.; Jeong, Y.S.; Lee, J.K. Automatic rule-based checking of building designs. *Autom. Constr.* **2009**, *18*, 1011–1033. [CrossRef]
3. Solihin, W.; Eastman, C. Classification of rules for automated BIM rule checking development. *Autom. Constr.* **2015**, *53*, 69–82. [CrossRef]
4. Salama, D.M.; El-Gohary, N.M. Semantic text classification for supporting automated compliance checking in con-struction. *J. Comput. Civ. Eng.* **2013**, *30*, 04014106. [CrossRef]
5. Pauwels, P.; Van Deursen, D.; Verstraeten, R.; De Roo, J.; De Meyer, R.; Van de Walle, R.; Van Campenhout, J. A semantic rule checking environment for building performance checking. *Autom. Constr.* **2011**, *20*, 506–518. [CrossRef]
6. Lin, J.-R.; Zhou, Y.-C. Semantic classification and hash code accelerated detection of design changes in BIM models. *Autom. Constr.* **2020**, *115*, 103212. [CrossRef]

7. Hjelseth, E.; Lassen, A.K.; Dimyadi, J. Development of BIM-based model checking solutions. In Proceedings of the 33rd International Conference of CIB W78, Brisbane, Australia, 31 October–2 November 2016.
8. Hamidavi, T.; Abrishami, S.; Ponterosso, P.; Begg, D.; Nanos, N. OSD: A framework for the early stage parametric optimisation of the structural design in BIM-based platform. *Constr. Innov.* **2020**, 20. [CrossRef]
9. Mekawy, M.; Petzold, F. BIM-based model checking in the early design phases of precast concrete structures: Learning, Prototyping and Adapting. In Proceedings of the 23rd International Conference on Computer-Aided Architectural Design Research in Asia, Beijing, China, 17–19 May 2018; pp. 71–80.
10. Beach, T.H.; Hippolyte, J.-L.; Rezgui, Y. Towards the adoption of automated regulatory compliance checking in the built environment. *Autom. Constr.* **2020**, 118. [CrossRef]
11. Clayton, M.; Fudge, P.; Thompson, J. Automated plan review for building code compliance using BIM. In Proceedings of the 20th International Workshop: Intelligent Computing in Engineering (EG-ICE 2013), Vienna, Austria, 1–3 July 2013.
12. Garrett, J.H.; Fenves, S.J. A knowledge-based standards processor for structural component design. *Eng. Comput.* **1987**, *2*, 219–238. [CrossRef]
13. Delis, E.A.; Delis, A. Automatic Fire-Code Checking Using Expert-System Technology. *J. Comput. Civ. Eng.* **1995**, *9*, 141–156. [CrossRef]
14. Rosenman, M.A.; Gero, J.S. Design codes as expert systems. *Comput. Des.* **1985**, *17*, 399–409. [CrossRef]
15. Dym, C.; Henchey, R.; Delis, E.; Gonick, S. A knowledge-based system for automated architectural code checking. *Comput. Des.* **1988**, *20*, 137–145. [CrossRef]
16. Garrett, J.H.; Hakim, M.M. Object-oriented model of engineering design standards. *J. Comput. Civ. Eng.* **1992**, *6*, 323–347. [CrossRef]
17. Yabuki, N.; Law, K.H. An Object-Logic model for the representation and processing of design standards. *Eng. Comput.* **1993**, *9*, 133–159. [CrossRef]
18. Kerrigan, S.; Law, K.H. Logic-based regulation compliance-assistance. In Proceedings of the 9th International Conference on Artificial Intelligence and Law, Edinburgh, UK, 24–28 June 2003; pp. 126–135.
19. Pinheiro, S.; Wimmer, R.; O'Donnell, J.; Muhic, S.; Bazjanac, V.; Maile, T.; Frisch, J.; van Treeck, C. MVD based information exchange between BIM and building energy performance simulation. *Autom. Constr.* **2018**, *90*, 91–103. [CrossRef]
20. Preidel, C.; Borrmann, A. Automated code compliance checking based on a visual language and building information modeling. In Proceedings of the 32nd International Symposium on Automation and Robotics in Construction (ISARC), Oulu, Finland, 15–18 June 2015; IAARC Publications: Oulu, Finland, 2015.
21. Motamedi, A.; Hammad, A.; Asen, Y. Knowledge-assisted BIM-based visual analytics for failure root cause detection in facilities management. *Autom. Constr.* **2014**, *43*, 73–83. [CrossRef]
22. Motawa, I.; Almarshad, A. A knowledge-based BIM system for building maintenance. *Autom. Constr.* **2013**, *29*, 173–182. [CrossRef]
23. Luo, H.; Gong, P. A BIM-based Code Compliance Checking Process of Deep Foundation Construction Plans. *J. Intell. Robot. Syst.* **2014**, *79*, 549–576. [CrossRef]
24. Martins, J.P.; Monteiro, A. LicA: A BIM based automated code-checking application for water distribution systems. *Autom. Constr.* **2013**, *29*, 12–23. [CrossRef]
25. Patlakas, P.; Livingstone, A.; Hairstans, R.; Neighbour, G. Automatic code compliance with multi-dimensional data fitting in a BIM context. *Adv. Eng. Inform.* **2018**, *38*, 216–231. [CrossRef]
26. Sulankivi, K.; Zhang, S.; Teizer, J.; Eastman, C.M.; Kiviniemi, M.; Romo, I.; Granholm, L. Utilization of BIM-based automated safety checking in construction planning. In Proceedings of the 19th International CIB World Building Congress, Brisbane, Australia, 5–9 May 2013; pp. 5–9.
27. Cooke, T.; Lingard, H.; Blismas, N.; Stranieri, A. ToolSHeDTM: The development and evaluation of a decision support tool for health and safety in construction design. *Eng. Constr. Archit. Manag.* **2008**, *15*, 336–351. [CrossRef]
28. Choi, J.; Choi, J.; Kim, I. Development of BIM-based evacuation regulation checking system for high-rise and complex buildings. *Autom. Constr.* **2014**, *46*, 38–49. [CrossRef]
29. Cheng, J.C.; Das, M. A BIM-based web service framework for green building energy simulation and code checking. *J. Inf. Technol. Constr.* **2014**, *19*, 150–168.
30. Yurchyshyna, A.; Zarli, A. An ontology-based approach for formalisation and semantic organisation of conformance requirements in construction. *Autom. Constr.* **2009**, *18*, 1084–1098. [CrossRef]
31. Beach, T.H.; Rezgui, Y.; Li, H.; Kasim, T. A rule-based semantic approach for automated regulatory compliance in the construction sector. *Expert Syst. Appl.* **2015**, *42*, 5219–5231. [CrossRef]
32. Beetz, J.; Van Leeuwen, J.; De Vries, B. IfcOWL: A case of transforming EXPRESS schemas into ontologies. *Ai Edam* **2009**, *23*, 89. [CrossRef]
33. Zhang, J.; El-Gohary, N.M. Automated information transformation for automated regulatory compliance checking in construction. *J. Comput. Civ. Eng.* **2015**, *29*, B4015001. [CrossRef]
34. Zhang, J.; El-Gohary, N.M. Semantic NLP-Based Information Extraction from Construction Regulatory Documents for Automated Compliance Checking. *J. Comput. Civ. Eng.* **2016**, *30*, 04015014. [CrossRef]

35. Zhang, J.; El-Gohary, N.M. Integrating semantic NLP and logic reasoning into a unified system for fully-automated code checking. *Autom. Constr.* **2017**, *73*, 45–57. [CrossRef]
36. Dimyadi, J.; Amor, R. Automated building code compliance checking—Where is it at? In Proceedings of the 2013 World Building Congress (CIB WBC), Brisbane, Australia, 5–9 May 2013.
37. Lin, T.A. Building Smart–A Strategy for Implementing BIM Solution in Singapore. *Synth. J.* **2006**, *5*, 117–124.
38. Khemlani, L. CORENET e-PlanCheck: Singapore's Automated Code Checking System, AECbytes (Building the Future). 2006. Available online: https://www.aecbytes.com/feature/2005/CORENETePlanCheck.html (accessed on 29 September 2021).
39. novaCITYNETS Pte. Ltd. Available online: http://www.novacitynets.com/company_aboutus.htm (accessed on 30 October 2020).
40. Drogemuller, R.; Woodbury, R.; Crawford, J. Extracting representation from structured text: Initial steps. In Proceedings of the 36th CIB W78 Conference on Information and Communication Technologies, Reykjavik, Iceland, 28–30 June 2000; pp. 302–307.
41. Ding, L.; Drogemuller, R.; Rosenman, M.; Marchant, D.; Gero, J. *Automating Code Checking for Building Designs-DesignCheck*; University of Wollongong: Wollongong, Australia, 2006.
42. Lee, J.M. Automated Checking of Building Requirements on Circulation Over a Range of Design Phases. Ph.D. Thesis, Georgia Institute of Technology: Atlanta, GA, USA, 2010.
43. Han, C.S.; Kunz, J.C.; Law, K.H. Compliance analysis for disabled access. In *Advances in Digital Government*; Springer: New York, NY, USA, 2002; pp. 149–162.
44. Eberg, E.; Heieraas, T.; Olsen, J.; Eidissen, S.H.; Eriksen, S.; Kristensen, K.H.; Christoffersen, O.; Lê, M.A.T.; Mohus, F. *Experiences in Development and Use of a Digital Building Information Model (BIM) According to IFC Standards from the Building Project of Tromsø University College (HITOS) after Completed Full Conceptual Design Phase*; Statsbygg: Oslo, Norway, 2006.
45. Solihin, W.; Eastman, C.; Lee, Y.-C.; Yang, D.-H. A simplified relational database schema for transformation of BIM data into a query-efficient and spatially enabled database. *Autom. Constr.* **2017**, *84*, 367–383. [CrossRef]
46. Dimyadi, J.; Pauwels, P.; Amor, R. Modelling and accessing regulatory knowledge for computer-assisted compliance audit. *J. Inf. Technol. Constr.* **2016**, *21*, 317–336.
47. Nawari, N.O. A Generalized Adaptive Framework (GAF) for Automating Code Compliance Checking. *Buildings* **2019**, *9*, 86. [CrossRef]
48. buildingSMART, Framing the Business Case for Automated Rule Checking. Available online: https://www.buildingsmart.org/wp-content/uploads/2020/06/buildingSMART-RR-TR1012-Framing-the-Business-Case-for-Automated-Rule-Checking-v1.1-Final-Dec-2019.pdf (accessed on 30 October 2020).
49. Krystallis, I.; Vernikos, V.; El-Jouzi, S.; Burchill, P. Future-proofing governance and BIM for owner operators in the UK. *Infrastruct. Asset Manag.* **2016**, *3*, 12–20. [CrossRef]
50. Erfani, A.; Tavakolan, M.; Mashhadi, A.H.; Mohammadi, P. Heterogeneous or homogeneous? A modified decision-making approach in renewable energy investment projects. *AIMS Energy* **2021**, *9*, 558–580. [CrossRef]
51. Zhong, R.Y.; Peng, Y.; Xue, F.; Fang, J.; Zou, W.; Luo, H.; Ng, S.T.; Lu, W.; Shen, Q.; Huang, G.Q. Prefabricated construction enabled by the Internet-of-Things. *Autom. Constr.* **2017**, *76*, 59–70. [CrossRef]
52. Sobhkhiz, S.; Taghaddos, H.; Rezvani, M.; Ramezanianpour, A.M. Utilization of semantic web technologies to improve BIM-LCA applications. *Autom. Constr.* **2021**, *130*, 103842. [CrossRef]
53. Park, J.W.; Chen, J.; Cho, Y.K. Self-corrective knowledge-based hybrid tracking system using BIM and multimodal sensors. *Adv. Eng. Informatics* **2017**, *32*, 126–138. [CrossRef]