Development and analysis of an automated performance code checking workflow

Desenvolvimento e análise de verificação automática de código de desempenho

Eduardo Arantes
Paulo Roberto Pereira Andery
Flávio Andrade
Daniel Paes
Javier Irizarry

Abstract

This study aimed to evaluate the applicability, accuracy, and challenges involved in the automated checking of design requirements specified by a Performance Code (PC) using a BIM (Building Information Modeling)-enabled model checker software application. Specific goals include: (a) definition of a workflow for the parameterization of a set of computational rules addressing part of the PC design requirements, (b) testing and evaluation of the proposed parameterization using a BIM model of a real large-scale housing project, and (c) comparison of accuracy and checking times between manual and automated methods. The research team developed and tested the automated checking of approximately one-third of the requirements. Findings indicate a substantial checking time decrease and an increase in the number of nonconformities detected through the automated method in comparison to the traditional technique of compliance assessment.

Keywords: Automated rule checking. Building code. Building Information Modeling. Performance-based building design. Design technology.

[1]**Eduardo Arantes**
[1]Universidade Federal de Minas Gerais
Belo Horizonte – MG – Brasil

[2]**Paulo Roberto Pereira Andery**
[2]Universidade Federal de Minas Gerais
Belo Horizonte – MG – Brasil

[3]**Flávio Andrade**
[3]MRV Engenharia e Participações S/A
Belo Horizonte – MG – Brasil

[4]**Daniel Paes**
[4]Massey University
Auckland, New Zealand

[5]**Javier Irizarry**
[5]Georgia Institute of Technology
Atlanta, Georgia, USA

Recebido em 23/01/21
Aceito em 23/07/21

Resumo

Este estudo visou avaliar a aplicabilidade, precisão e desafios envolvidos na verificação automática dos requisitos de projeto especificados pela Norma de Desempenho utilizando um software de checagem de modelos BIM (Building Information Modeling). Os objetivos específicos incluem: (a) definição de um processo de parametrização de um conjunto de regras computacionais que incluem parte dos requisitos de projeto definidos pela Norma, (b) teste e avaliação da parametrização proposta utilizando o modelo BIM de um projeto habitacional real e de grande escala, e (c) comparação da precisão e dos tempos de checagem entre os métodos manual e automatizado. Os pesquisadores desenvolveram e testaram a checagem automática de aproximadamente um terço dos requisitos de projeto da Norma. Os resultados indicam uma diminuição substancial do tempo de checagem e um aumento do número de não-conformidades detectadas através do método automatizado em comparação com a técnica tradicional de avaliação de conformidade.

***Palavras-chave**: Verificação automática de requisitos. Norma de Desempenho. Modelagem da Informação da Construção. Projeto de construção baseado em desempenho. Tecnologia de projeto.*

Introduction

Due to an increasingly competitive environment, the construction sector has been fostering and promoting both technological and methodological innovations to improve quality and productivity. Ensuring building performance has an impact on the entire production chain and has been deemed a strategic factor to increase the competitiveness of construction companies and to promote innovation (BARRET; LEE, 2005; SZIGETI; DAVIS, 2005).

In many countries, the debates regarding construction quality assurance have developed into performance-based building codes. Besides having to respond to aesthetic, constructability, and budget demands, a performance-based design must also integrate solutions that address various performance requirements set by those codes at different building systems, which raises the complexity of the design task (OTTER; EMMIT, 2008). Furthermore, the increasing number, complexity, and interactions among the performance requirements of building codes hinder the development and management of performance-based projects, which must comply with that intricate network of requirements. This context has turned performance assurance through compliance checking (the assessment of a project's design in light of the building code regulations) into a quite challenging, expensive, time-consuming, and specialized task, particularly in large-scale projects.

The Building Information Modeling (BIM) approach and tools can help practitioners to deal with such complexity (ISMAIL; ALI; IAHAD, 2017). A BIM file contains a parametric three-dimensional construction model built from intelligent and semantically rich components with different types of information besides geometry (EASTMAN *et al*., 2008; ANDRADE; RUSCHEL, 2011). As a technology, BIM can support a collaborative and interactive environment for people, processes, and further technologies (SUCCAR, 2009) by consolidating useful information for a variety of purposes such as construction cost estimation, planning, inspection, building operation and maintenance (NAWARI, 2012). It can also increase collaboration and communication, contributing to the integration of processes, decrease of inefficiencies, redundancies, and rework, and increase of productivity (CAMPBELL, 2007). However, uncoordinated or unregulated BIM implementation and usage can eventually suppress its benefits (ARAYICI *et al*., 2011; GUREVICH; SACKS, 2017).

Given the increasing complexity of construction projects and building codes, there has been a growing interest in the automation of compliance checking of BIM models by practitioners and researchers (YALCINKAYA; SINGH, 2015). The implementation of automated compliance checking involves setting up rules within computational systems (e.g., model checkers) that are able to scan BIM models and autonomously detect any existing irregularities or nonconformities based on those pre-established rules. Automated compliance checking contributes to ensure design conformity in light of the code in place and to cut down the time spent by reviewers (designers and contractors) to conduct the compliance assessment, adding value to the development process and final product (TAN; HAMMAD; FAZIO, 2010; MARTINS; ABRANTES, 2010; KASIM *et al*., 2013; RODRIGUES, 2015).

Problem statement and research questions

This study aimed to evaluate the applicability, accuracy, and challenges involved in the automated checking of design requirements specified by a Performance Code (PC) using a model checker software application.

Specific goals include:

(a) definition of a workflow for the parameterization of a set of rules addressing part of the PC design requirements using a BIM-enabled model checker software application;

(b) testing and evaluation of the proposed parameterization using a BIM model of a real large-scale housing project; and

(c) comparison of accuracy and checking times between manual and automated methods.

These objectives were designed to answer the following research questions:

(a) Is the automated checking method suitable for a performance-based building code?

(b) What are the steps and requisites in the implementation of an automated checking process?

(c) Is a typical model checker software application able to embody and accurately express the language of a performance code?

(d) Is the automated checking process more effective than the manual method?

Background

This section provides theoretical background and context for the research questions presented in the previous one. The principles and applications of performance-based design, performance regulations, and automated rule checking are discussed.

Performance-based design

Since the beginning of the last decade, the discussions on performance-based buildings have led to the development of new design management workflows and methods towards increasing the integration of design and construction phases. The performance-based building approach demanded a more flexible and non-prescriptive set of guidelines for building design and construction – codes, standards, and regulations could not be limited to specific design solutions or construction systems (LEE; BARRET, 2003). The main characteristics of performance-based design and construction include:

(a) focus on end-user demands rather than on contractor's and owner's;

(b) use of functional terms to define how a building should operate, regardless of its materials, design and construction solutions;

(c) flexibility to select the most appropriate level of performance to a given requirement; and

(d) consideration of the entire project life-cycle, with an emphasis on the operation and maintenance phase.

As per Pham, Boxhall and Spekkink (2006), a performance-based design management process is based on a workflow that starts with the definition of functional demands by end-users, followed by the translation of these needs into performance requirements which are further processed into performance specifications and, finally, into technical specifications. These technical specifications are precisely the object of manual or automated compliance checking processes. An essential component in the translation of end-users' needs into performance requirements is the identification of physical factors that can be used as performance indicators, which could eventually be checked. Ideally, these measurable performance indicators should be suitable for computational analysis to allow for performance prediction (BECKER, 2005) and simulations at the conceptual design phase hence facilitating the study and implementation of performance-based design solutions (FREI; SAGERSCHNIG; GYALISTRAS, 2017).

Performance regulations

Regulatory building codes based on the principles of performance-based design and construction approaches have been gradually implemented worldwide (KERN; SILVA; KAZMIERCZAK, 2015). In Brazil, this occurred with the release of a single comprehensive Performance Code (PC): the NBR 15575 standard named Housing Facilities: Performance (ABNT, 2013), which guidelines provide the performance requirements for housing projects. It translates end-users' needs into minimum performance requirements designed to satisfy their demands in the use of a housing facility and its systems under certain conditions. Thus, it is centered on a residential facility's operation and maintenance performance, regardless of its construction systems and materials. As per Meacham *et al*. (2005), any performance-based building code should specify the fundamental properties of the different building elements regardless of their materials.

The PC is a non-prescriptive code and specifies various requirements concerning structural safety, fire safety, habitability, functionality, durability, and sustainability. It encompasses six main components: general requirements, structural requirements, floor requirements, internal and external vertical envelope requirements, roofing requirements, and hydrosanitary requirements. Within each of these components, there are further safety-related requirements (structural safety, fire safety, operation and maintenance safety), habitability requirements (watertightness, thermal, acoustic, and lighting comfort, health, hygiene and air quality, functionality and accessibility, tactile comfort), and sustainability requirements (durability, maintainability, and environmental suitability) (ABNT, 2013). Ultimately, the PC aims to promote and ensure minimum technical and functional quality to housing developments, establishing their fitness according to performance parameters.

Despite being more comprehensive by encompassing different construction systems and requirements concerning safety and habitability, the PC is analogous to other international performance-based codes –

particularly in the areas of structural and fire safety – such as the non-prescriptive Eurocodes that have been published since 1990, the ISO 24679 (INTERNATIONAL..., 2019) series (Fire Safety Engineering – Performance of structures on fire), and the Building Code of Australia 1996 (BCA), which became an international reference in performance-based building codes (PILZER, 2005; ARMSTRONG *et al.*, 2017).

Many scholars and industry practitioners exploring the benefits of automated checking have been focusing on design compliance with thermal and acoustic performance requirements using different model checker software applications, which can combine performance simulations with rule checking capabilities (TAN *et al*., 2010) or provide ontologies and semantic structures (ZHANG; EL-GOHARY, 2017a).

Automated rule checking

Because performance-oriented design solutions are usually more complex (requiring simultaneous consideration of requirements that interact with each other) and given that these solutions need to be validated by testing and simulations, companies can minimize costs associated with such a task by promoting standardization of design solutions. Studies indicate a trend to design standardization as performance regulations are gradually implemented (ABAZA, 2012). Regardless of the level of standardization of design solutions, design compliance with performance requirements should be submitted to verification through either a manual or automated checking method during the design phase as well as after the project completion. Requirement checking (aka "rule checking") consists of an assessment of design solution compliance with requirements set by the regulation in place (BENNING *et al*., 2010). The main goals of this process are to ensure quality, reduce costs, and increase process reliability (NAWARI, 2019).

Automated compliance checking appears to benefit BIM-based design processes (ISMAIL; ALI; IAHAD, 2017), given the increasing complexity of design solutions. The adoption of automated BIM-based checking methods can streamline the design process by reducing time and costs associated with the compliance assessment task, allowing for the determination of a project's level of maturity (NAWARI, 2012), and for designers to focus on the ideation and integration of solutions (SOLIHIN; EASTMAN, 2015).

The translation of regulations into computational rules towards the automation of the compliance checking process is not a simple task mainly because these codes are typically designed and written for human experts, not computers (BENNING *et al*., 2010). Solihin and Eastman (2015) state that the main challenges involved in the implementation of automated rule checking are the inherent complexity of regulatory codes and the variety of situations to which specified requirements apply. Such challenges require the support of a structuring framework for the process of translation of codes and regulations into computational rules. A framework would comprise four stages:

(a) semantic interpretation and translation of codes and regulations;

(b) development of a BIM model;

(c) set up of computational rules and execution of automated checking; and

(d) automated checking report.

In a recent study, Zhang and El-Gohary (2017b) proposed a method that integrates semantic natural language processing techniques and design information with semantic logic-based representations to improve automated code checking.

Special attention should be paid to the content and modeling methods in the development of the BIM model. These can impact the set-up of computational rules in the software application that runs the automated rule checking (i.e., model checker), as well as the overall efficiency of the checking process and quality of outcomes. Model view definitions should be used to specify the necessary content to a given checking system (HAN; KUNZ; LAW, 1998; BORRMANN; RANK, 2009). Moreover, it is crucial to ensure that a BIM model's Level of Development (LOD) matches the model checker application parameters. As per the BIMForum (2013), the LOD is an attribute of BIM models that indicates the level of reliability and certainty about the information embedded in a model, allowing practitioners to specify and manage the amount and accuracy of information over the project delivery process. For instance, an element in a LOD 300 BIM model is graphically represented as "[…] a specific system, object, or assembly' with reliable information on 'quantity, size, shape, location, and orientation.". Non-graphical information may also be linked to that element (AMERICAN..., 2013).

During the set-up of computational rules and execution of automated checking, specific conditions and restrictions are created and tested for each rule. At the end of the checking process, the model checker application generates a report including the nonconformities detected (i.e., elements in the model that do not comply with a code or regulation) and processing times (MAINARDI NETO, 2016).

Another critical procedure in the automation process is to determine the class of rules according to their level of complexity. The classification of rules involves a qualitative analysis to determine the code requirements to be parameterized (i.e., translated into computational rules), and the level of complexity of rules. The classification helps in the planning of the parameterization process, providing a picture of how much of a regulatory code could be automatically checked, the potential issues that could arise in the translation process, and the suitability of rules to the code requirements (MAINARDI NETO, 2016). Solihin and Eastman (2015) proposed four major classes of complexity, ranging from class 1 (the simplest rules) to class 4 (the most complex rules), to determine the level of complexity of computational rules. This classification method was adopted in this study, as summarized in Table 1.

Hjelseth (2016) outlines a conceptual categorization framework for BIM-based model checking (BMC) alternatives and discusses three approaches (among others):

(a) validation checking, to verify whether design solutions comply with rules of a rule set;

(b) model content checking, which focuses on checking the content of a BIM model for a specific use; and

(c) smart object checking, which allows "[…] the object itself observe its environment and automatically adapt to this by embedded pre-defined rules or algorithms […]" (HJELSETH, 2016).

A similar categorization is presented by Fan, Chi and Pan (2019), who provide different conceptual directions, including the predicate logic-based, object-oriented, and description logic approaches. Despite the existence of different approaches to BIM-based code checking, studies in the field emphasize one that involves encoded rules separated from the model features to be checked. For instance, Yang and Xu (2004) developed a conceptual workflow based on an object-based approach for representing both building systems and building codes checked by the developed application. An in-depth discussion on the different approaches to model checking is beyond the scope of this paper and can be found in the studies mentioned in this section.

Generally, BMC is often identified by the use of one particular type of software. This can occur in three different ways:

(a) developing a plug-in software application for a given BIM platform;

(b) using a stand-alone software application integrated with a BIM platform; or

(c) employing a web-based application.

The most commonly adopted technique involves the second approach, that is, the use of integrated software applications (ISMAIL; ALI; IAHAD, 2017). The level of integration with BIM software, however, varies largely across stand-alone applications, which can be very loosely coupled or not coupled at all when open standard representations are used.

Table 1 - **Classes of complexity of rules**

Class	Description
1	Computational rules that require little data. The software makes use of component data in the checking process, such as parameters.
2	Computational rules that require a simple derived value such as geometry, utilized by the software in the checking process.
3	Computational rules that require complex data structure. The software makes use of geometric and topological data, properties, and possible algorithms in the checking process.
4	Computational rules that generate solutions to eventual problems.

Eastman *et al*. (2009), Nawari (2018), and Narayanaswamy, Liu and Al-Hussein (2019) provide an overview of software applications (including stand-alone ones) for BMC. In summary, different applications have been used to support the implementation of automated checking systems based on IFC (Industry Foundation Classes) data models, such as:

(a) the Solibri Model Checker (SMC) – a Java-based application;

(b) the SMARTcodesu, which uses object-based technology for code representation;

(c) the Jotne EDModelChecker or Express Data Manager (EDM), which provides an objects database to facilitate the development of checking systems utilizing EXPRESS language (EASTMAN *et al*., 2009);

(d) the CORENET, adopted by Singapore's Building and Construction Authority, and in some cases coupled with the FORNAX platform; and

(e) the Norwegian Systems, an adaption of de CORENET using the SMC Ruleset Manager (NAWARI, 2018).

There is a relatively limited number of commercial applications, such as the SMC, Autodesk Navisworks, and Tekla BIMinsight (HJELSETH, 2015).

The SMC is a widely used stand-alone software application for BIM-based automated code checking. It has the capability of detecting elements in a BIM model that conflict with or violate user-defined rules (nonconformities). Furthermore, it appears to be the only commercial software available for clash detection and space validation of building designs (NARAYANASWAMY; LIU; AL-HUSSEIN, 2019). The SMC has been adopted for automated code checking in public projects in the US (ISMAIL; ALI; IAHAD, 2017), and automated design assessment considering construction safety in Italy (GETULI *et al*., 2017) and accessibility in Portugal (RODRIGUES, 2015). The SMC utilizes the IFC data scheme to read BIM models and comprises a set of rule templates that can be customized by the user. One can also set different combinations of rule templates, which increases the application's power and usefulness in rule checking. Its user interface is relatively simple and does not require programming skills, except for the creation of new rule templates or modifications to existing ones. In addition, the application generates a report on the rules that have been met and those that have been violated, highlighting the nonconformities in the BIM model, which facilitates to locate the existing violations and their relationships.

Method

The research method is formally defined as Constructive Research, a well-known method for the development of innovations backed by theoretical knowledge aiming at solving real-life problems (LUKKA, 2003). Oyegoke (2011) describes the six steps in Constructive Research:

(a) define a relevant practical problem;

(b) evaluate the research potential;

(c) obtain theoretical and practical knowledge in the field;

(d) propose a solution;

(e) implement and test the solution; and

(f) analyze the applicability of the solution proposed.

This study is backed by the research team's practical experience and comprehensive literature review. The resulting innovative artefact consists of an interface or link between the internal and external environments of a computational system, as defined by Lukka (2003) and Dresch, Lacerda and Antunes (2015). As per the literature, automated BIM-based code checking techniques are still incipient and little disseminated. There are still many technical barriers to overcome for such technology to become widely adopted.

The selection of the PC as the building code for this study was guided by its comprehensiveness and large impact on a quite dynamic and globalized construction sector. The 255 regulations within the PC provide the requirements for various design specialties, which makes it one of the main national building codes. In turn, the use of a real large-scale housing project – a residential complex of 128 apartments – as the testbed for the parameterization ensures that findings have a more practical and relevant impact on industry practices. The BIM models (architectural, structural, and building systems) were developed using the Autodesk Revit

software application, at LOD 300. They were provided by a partner construction company with expectations to benefit from the results of this investigation (improving its current compliance checking practice).

Developing and testing the automated performance code checking workflow

The SMC version used in this study comes with a set of built-in rule templates (pre-defined rules provided by the software manufacturer). Despite the model checker limitation regarding the fact that it provides only a few default rules (NAWARI, 2012), the ability to edit and combine them allowed for the automated checking of some of the PC's requirements in a large-scale housing project, as described below.

The first phase in the development of the computational checking system consists of a qualitative categorization of the PC's requirements to identify which could be verified using the SMC application. The requirements were categorized as follows:

(a) non-propositional requirements;

(b) requirements that are automatically verifiable;

(c) requirements that are partially verifiable; and

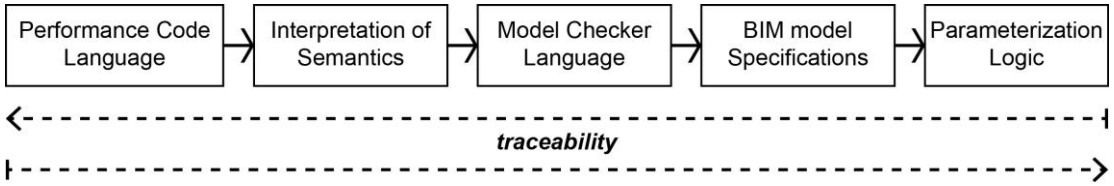
(d) requirements that are not automatically verifiable.

Non-propositional requirements are those that cannot be translated into objective propositions based on which a given design solution could be determined adequate or not. As non-propositional requirements are not based on objective criteria, compliance verification is currently performed by subjective assessment of an analyst, leading to different results depending on who is checking such a requirement. Automatically verifiable and not automatically verifiable requirements refer to those that succeeded and failed in the developed parameterization, respectively. Finally, partially verifiable requirements are those that needed some kind of human intervention – either to check the results, make decisions over the process, or manually complete the checking process of a given requirement when, due to software limitations, the computer could only perform part of it. These procedures normally consist of confirmation prompts shown on the software interface (pop-up windows), which the user must manually accept or not by clicking on the corresponding field. From this prior screening of all requirements, it resulted that 33% were found to be either automatically verifiable (14%) or partially verifiable (19%), whereas 70% were identified as either non-propositional (45%) or not automatically verifiable (25%). In total, 24 requirements categorized as either automatically or partially verifiable were parameterized.

The second phase in the system development was the actual parameterization of the PC's requirements using the SMC application, which provides rule templates that can be combined and customized depending on a requirement's complexity. In general, the parameterization process consists of specifying the parameters of a rule (or rules) addressing a single code requirement. This process is dynamic; each rule adjusted in the model checker application requires several tests to correct eventual issues. The user should trust – and be able to double-check – that the developed rules are relevant and correct (BELL; BJØRKHAUG; HJELSETH, 2009). Therefore, this intricate process must be transparent and properly documented for traceability. Due to the number of requirements and related rules, spreadsheets were created to record and facilitate the parameterization. The process documentation contributed to ensuring reliability, coherence, and understanding of how each requirement was parameterized.

The proposed parameterization workflow (Figure 1) is similar to others found in the literature (e.g., GETULI *et al*., 2017), where an initial semantic analysis is performed to translate the code requirements into objective parameters that can be entered into the computational rules of the model checker application.

Figure 1 – **Parameterization workflow**



The spreadsheets help to structure and document the translation process. They include:

(a) requirement description;

(b) requirement location in the PC document;

(c) requirement category (automatically verifiable or partially verifiable);

(d) requirement details and relationship with other regulations;

(e) BIM model specifications and components, including LOD; and

(f) identification of correspondent rules in the model checker (preliminary and main rules), with a description of what is being verified.

The documentation of procedures to translate the PC's requirements into computational rules is exemplified in Table 2, for the "Operation and Maintenance Safety" requirement. The categorization of this requirement indicates a case of partial verification, with clear semantic information and no subjective items, except for the term "devices", which were assumed as the anchoring hooks of ropes and safety belts. In the translation of this requirement into the model checker language, the system was set to check for roof slopes over 30% and anchoring hooks in the BIM model, which elements were previously classified to enable the automated checking. The preliminary checking phase detects the existence of objects classified as "roof" (rule 5.1), whereas the main checking phase verifies whether the roof slope is over 30% (rule 5.2) and the presence of anchoring hooks (rule 5.3).

Table 2 - **Requirement parameterization spreadsheet**

Requirement Description		Performance: Operation and Maintenance Safety Requirement: Operation and Maintenance Design guidelines – Roof: a) provide safety devices anchored on the main structure to allow for the attachment of ropes, safety belts and other PPE for slopes greater than 30%; b) provide access for maintenance procedures		
Requirement Location		Brochure 5, Page 20, Item 9.2.3.2		
Requirement Category		Partially verifiable		
Requirement Details	**Q**	Hidden information?	Ambiguous information?	Ambiguous nomenclature?
	A	No	No	Yes: "devices". Anchoring hooks for tying ropes and belts.
Translation into SMC language		Partially automated checking should detect roof slopes over 30%, and any anchoring hooks in the BIM model. User should complete checking task manually.		
BIM model specs	**LOD**	300		
	Components	Model should include roof and anchoring hooks. Items should be under the *components* category.		
Preliminary Rules	**SMC template / Rule # / Class #**	SOL 11 / Rule 5.1 / Class 1		
	Construction Description	Rules: 1 – checks if model has roof; 2 – checks if roof is *components* Construction: In the *required classification* field, select *components* category. Select *all components must be classified*, and add *roof* to *required components*.		
Main Rules	**SMC template / Rule # / Class #**	SOL 203 / Rule 5.2 / Class 2		SOL 11 / Rule 5.3 / Class 1
	Construction Description	Rule: checks if there are roof slopes over 30% Construction: In the *check components* field, add *roof* with *operator matches* encompassing all items classified as *roof*. In the *property sets* field, select *slope* property. In the *operation* field, add *less than* condition.		Rules: 1 – checks if model has anchoring hooks; 2 – checks if anchoring hooks are *components* Construction: In the *required classification* field, select *components* category. Select *all components must be classified*, and add *hook* to *required components*.

Figure 2 shows the configuration of the distance rule template, which checks for the maximum distance between a slab's top surface and a window's top edge.

There are, however, more complex requirements that demand the use of a combination of rules. In this study, this situation occurred in the parameterization of the "Functionality and Accessibility" code requirement, which establishes the minimum size of spaces for use and operation of the housing facility. Figure 3 shows the different rules that were combined to check this requirement.

Moreover, one can specify several parameters within a rule. Figure 4 provides an example of parameter editing for the rule "Free Floor Space" (SOL 209) – one of the several used in the parameterization of the "Functionality and Accessibility" code requirement.

In order to run the automated checking of those 24 automatically or partially verifiable requirements, 94 rules were necessary in total – 3.5 times the number of requirements. This ratio increases up to 4.5 rules per requirement if considering automatically verifiable requirements only.

At the end of the automated checking process, the model checker application generates reports with detailed information on every nonconformity detected, which are also depicted graphically. Moreover, nonconformities are ranked in three different levels of severity, which helps in the decision-making process of design analysts. Figure 5 shows the graphic displaying of a nonconformity concerning the window-sill height (medium-severity) in the BIM model.

Qualitative comparison to manual code checking method

Data from both the manual and automated methods were collected to perform the comparison. These include checking times and number of nonconformities detected (accuracy); both deemed as efficiency indicators. Also, two premises were adopted for the comparison. First, the comparison considered automatically verifiable requirements only due to the complexity and possible inconsistency in the determination of checking times for partially verifiable requirements, as their processing was not fully automated, hence demanding human intervention. In the automated method, the total checking time encompasses generating the IFC model, running the IFC model in the model checker application, and running the actual checking task (i.e., executing the preliminary and main rules for each requirement in the model checker environment).

Figure 2 - **Specification of parameters of the distance rule**

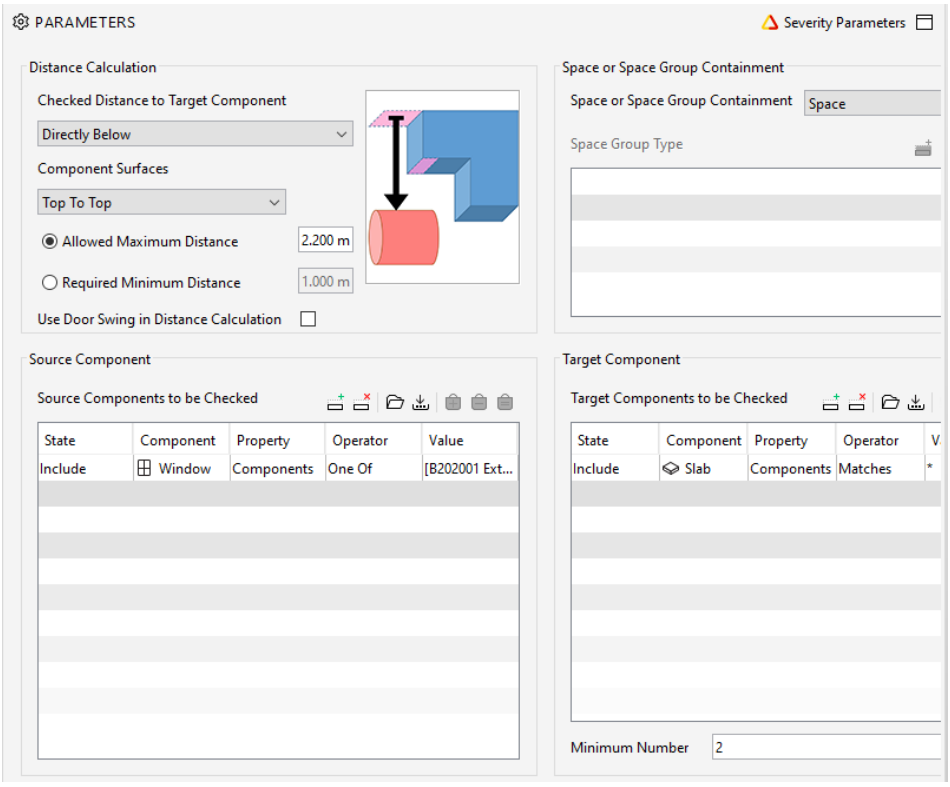


Figure 3 - **Set of rules used to check the Functionality and Accessibility requirement**

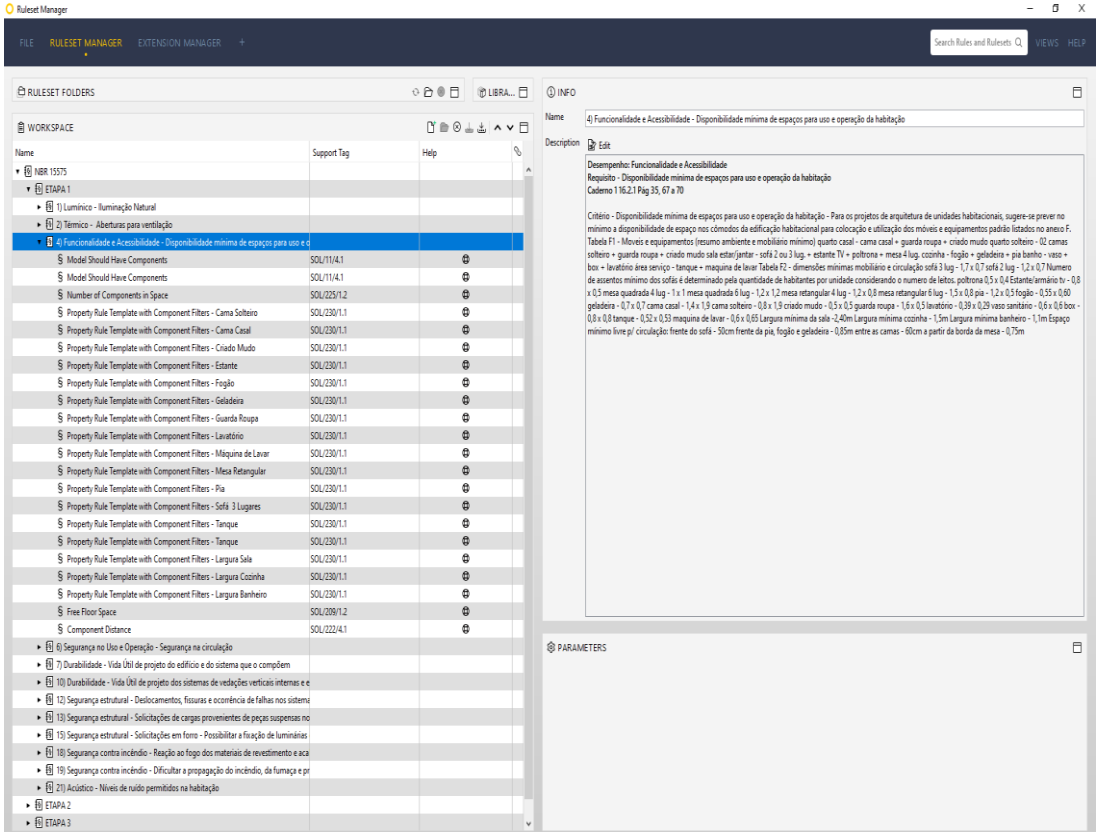


Figure 4 - **Set of parameters within a given rule**

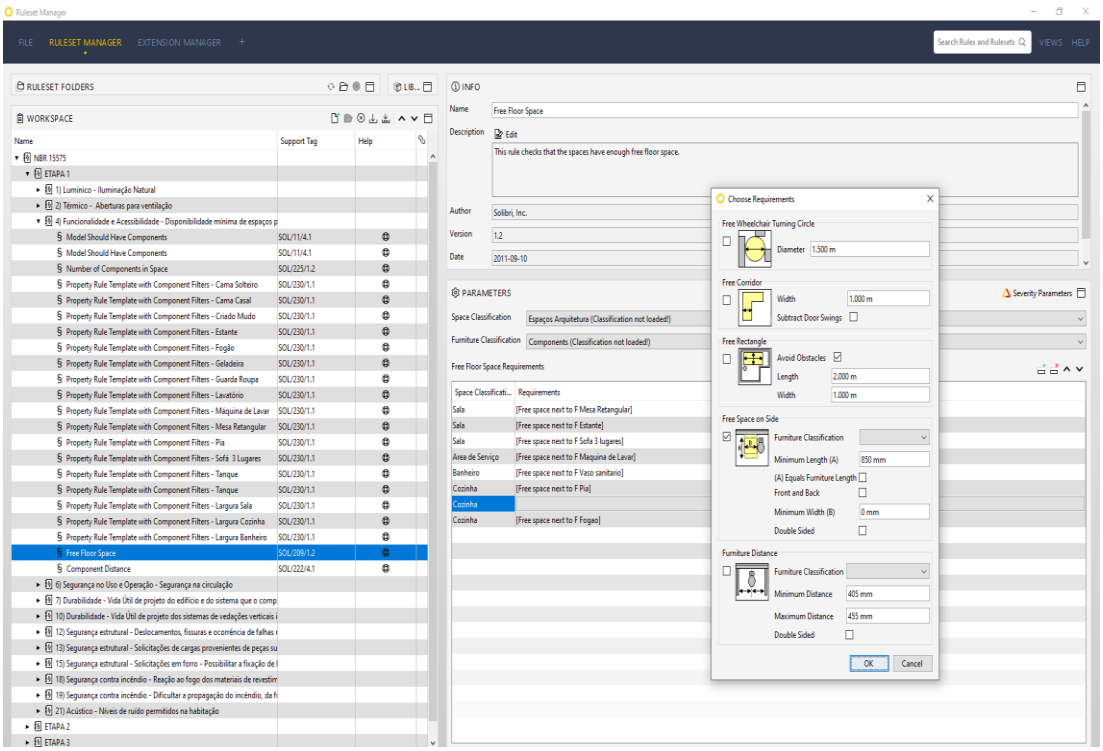
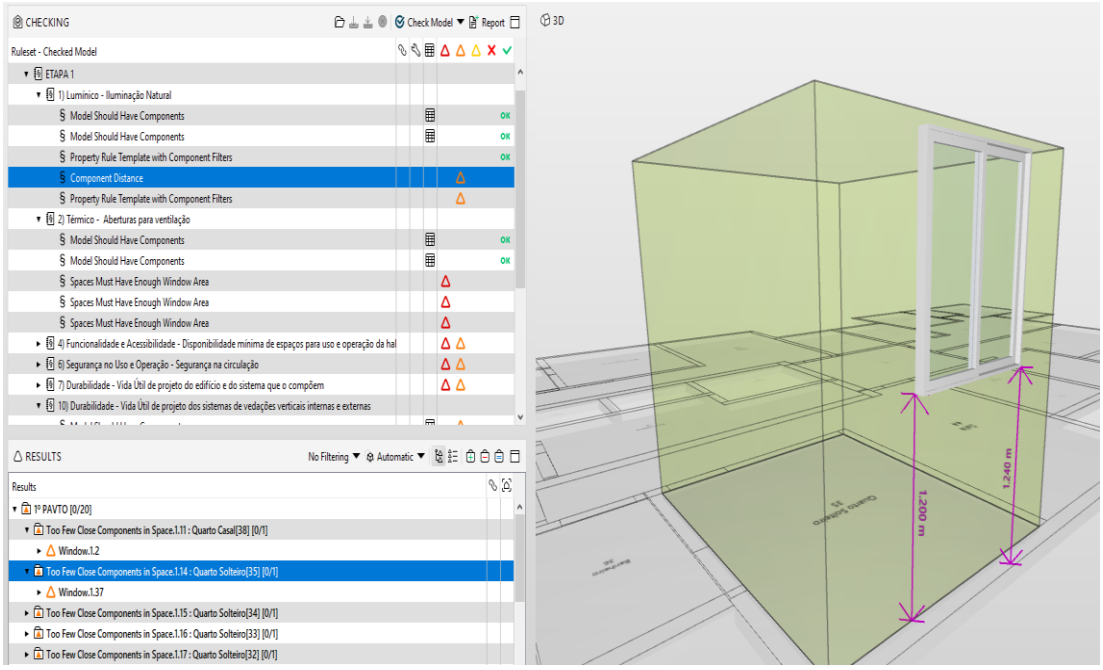


Figure 5 – **Nonconformity concerning the window-sill height in the BIM model**



Second, the manual checking method was simulated as close as possible to the common practice in the industry. Traditionally, code compliance checking is performed manually with the support of checklists and CAD (Computer-Aided Design) files, on which annotations about nonconformities are made. The detection of irregularities is done visually with the help of basic CAD software features, such as area and distance calculation, and drawings overlay. The nonconformities are then communicated to the designers for correction via e-mail or face-to-face meetings. In this study, the manual checking method was performed by two different professionals with an equivalent level of expertise in design compliance assessment. Both reviewers had full access to files providing the requirements descriptions and locations in the PC document, as well as a list of additional references. Reviewers were asked to complete the following tasks:

(a) detect any nonconformities;

(b) record the time spent to check each requirement;

(c) briefly describe how each requirement was checked; and

(d) provide general comments.

They were introduced to each requirement and oriented on how to complete the tasks checklist beforehand. Naturally, the total checking time for each requirement included tasks a and b only.

Results and discussions

Ninety-four (94) rules were created in the SMC to check 24 requirements. An average of 3.5 tests was performed for each rule, in a total of 329 tests. The automated checking method yielded positive results in terms of the number of nonconformities detected and time spent in the process. In this method, detected nonconformities had to be manually confirmed or rejected, which required some analysis and decision-making time. Also, it is worth noting that the time to execute preliminary rules was around half the total checking time of requirements.

The comparison indicates a relevant time difference between automated and manual checking techniques. More specifically, results show an average decrease of 60% in checking times with the computerized method, although this percentage appears to vary largely across the PC's requirements. Such differences may have occurred due to the varying number and level of complexity of rules for each requirement, file size, and resulting computational performance. Although the rule/requirement ratio varies depending on a requirement's complexity, this study has shown that the number of rules will usually be much higher than

the number of code requirements, meaning a one-time substantial time investment at the early stages of setting up the automated checking system.

Results also indicate that the automated method was more efficient than the manual one when it comes to the overall accuracy of the checking process (detection of nonconformities), as shown in Table 3 below. Again, only automatically verifiable requirements (a total of 9) were considered for comparison purposes.

Although the decrease in checking times provided by the automated method may not significantly impact a project's cost – especially for large-scale housing projects such as the one of this study –, this method certainly contributes to the increase of the reliability of the checking task by reducing the likelihood of overlooking nonconformities and false-positives (mistakenly detecting a nonconformity that does not really exist), as opposed to the manual method which is inevitably prone to human error.

Because the set of requirements specified in the PC is considerably broad, even if many of these are quantifiable or measurable requirements, one cannot expect that human reviewers carrying out manual checking have full control over the assessment criteria or could guarantee flawless analyses. Similarly, one should not expect a completely error-free automated checking, as the semantic interpretation of code requirements is a necessary manual step in the parameterization process, regardless of the computerized method utilized. Nonetheless, in a scenario where there are not enough experts on performance-based design and building regulations (such as in the Brazilian construction sector), the adoption of automated BIM-based code checking systems sounds particularly interesting. It could contribute to addressing the current lack of expert reviewers while improving the overall efficiency of the compliance checking task.

All materials and datasets can be made available to the reader upon request. These include the parameterization spreadsheets, checking logs/reports, spreadsheets of checking times and nonconformities detected for both manual and computerized methods, and the BIM models of the housing project. The complete PC documentation (NBR 15575) is available for download from the ABNT website (ABNT, 2013).

Conclusion

Automated compliance checking is an innovative yet little disseminated approach. The technology could streamline the checking task carried out by designers and contractors, cutting down associated costs while improving outcomes. In summary, the automated checking method allows for a fast and accurate evaluation of solutions based on performance regulations, contributing to the development of increasingly efficient buildings over the design phase. In this context, this study aimed to develop and test an automated checking workflow and analyse the challenges and benefits resulting from this new technique.

In this study, automated checking was faster and more accurate than the manual method, and, in general, the computerized system was partially suitable for the performance-based code selected. Nonetheless, it is important to highlight the need for additional tests involving different projects, regulations, and experts to confirm the checking times and accuracy levels found in this work to increase the reliability of the proposed parameterization workflow, restricted to the model checker utilized in this study. Furthermore, it should be noted that this study's scope did not include an in-depth analysis of the SMC tool, nor a comparison to other available model checker software applications.

Existing rule templates in the model checker application could be easily adjusted to check various requirements, particularly those associated with the geometry of elements in the BIM model (dimensions, slopes, areas, etc.). However, as the level of complexity of rules increased, the software limitations were evidenced. Adjusting highly complex rules to address a complex requirement often demanded the use of templates that were not readily available in the application hence limiting its applicability, as it does not give the option to create or modify existing templates.

Table 3 - **Accuracy of automated method *vs*. manual method**

Accuracy indicators	**Number of code requirements**	
	Automated method	**Manual method**
True-positive (nonconformities detected)	9	6
True-negative (nonconformities undetected)	0	2 (by at least one reviewer)
False-positive (detection of nonconformities that did not exist)	0	1 (by at least one reviewer)

In agreement with Benning et al. (2010), translating the code language into the model checker language was another limitation in the parameterization process. Because most requirements were hard to interpret and translate into computational rules – as they have been written for humans, not computers – only a small percentage of them could be automatically checked using the model checker application.

It should be noted that this study selected a particularly complex performance-based building code. Nonetheless, such code adopts the same conceptual structure of most international performance-based regulations, which are all based on the principles of Performance-Based Buildings (BECKER, 2008). Therefore, the proposed parameterization workflow should work for all other performance-based codes that specify objective design requirements and how to comply with them. Similar computerized methods are expected to benefit the process of verifying compliance with such codes. To that end, this research provides an initial and replicable workflow for the development and implementation of automated BIM-based compliance checking systems using a specific model check application (SMC), regardless of the performance-based regulation in place, project size, and typology.

References

ABAZA, M. High Performance buildings using whole building integrated design approach. **Strategic Planning for Energy and the Environment**, v. 31, n. 4, p. 19-34, 2012.

AMERICAN INSTITUTE OF ARCHITECTS. **Document E203-2013: Building Information Modeling and Data Exhibit**. 2013. Available: http://www.aia.org/aiaucmp/groups/aia/documents/pdf/aiab099084.pdf. Access: 10 aug. 2017.

ANDRADE, M. L. V. X.; RUSCHEL, R. C. Building Information Modeling (BIM). In: KOWALTOWSKI, D. C. C. K. *et al*. (org.). **O processo de projeto em arquitetura**: da teoria à tecnologia. São Paulo: Oficina de Textos, 2011.

ARAYICI, Y. *et al*. Technology adoption in the BIM implementation for lean architectural practice. **Automation in Construction**, v. 20, n. 2, p. 189-195, 2011.

ARMSTRONG, A. *et al*. Enabling Innovation in Building Sustainability: Australia's National Construction Code. **Procedia Engineering**, v. 180, p. 320-330, 2017.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR 15.575**: edificações habitacionais: desempenho. Rio de Janeiro, 2013.

BARRET, P.; LEE, A. **Revaluing construction, A CIB Priority Theme**. Rotterdam: CIB, 2005.

BECKER, R. Fundamentals of performance-based building design. **Building Simulation**, v. 1, n. 4, p. 356-371, 2008.

BECKER, R. State of the Art findings and results. In: PERFORMANCE-Based Building State of the Art: Second International Report. Rotterdam: CIB, 2005.

BELL, H.; BJØRKHAUG, L.; HJELSETH, E. **Standardized computable rules**. Oslo: Standards Norway, 2009.

BENNING, P. *et al*. Collaboration processes. In: FRAMEWORK for Collaboration. Brussels: InPro Consortium, 2010.

BIMForum. **Level of Development Specification, For Building Information Models, Version**: 2013. 2013. Available: https://bimforum.org/resources/Documents/BIMForum_LOD_2013_reprint.pdf. Access: 18 Jul. 2019.

BORRMANN, A.; RANK, E. Specification and implementation of directional operators in a 3D spatial query language for building information models. **Advanced Engineering Informatics**, v. 23, n. 1, p. 32-44, 2009.

CAMPBELL, D. A. Building Information Modeling: the Web3D application for AEC. In: INTERNATIONAL CONFERENCE ON 3D WEB TECHNOLOGY, 12., Perugia, 2007. **Proceedings [...]** New York: Association for Computing Machinery, 2007.

DRESCH, A.; LACERDA, D. P.; ANTUNES, J. **Design science research**: método de pesquisa para avanço da Ciência e Tecnologia. Porto Alegre: Bookman, 2015.

EASTMAN, C. *et al*. Automatic rule-based checking of building designs. **Automation in Construction**, v. 18, n. 8, p. 1011-1033, 2009.

EASTMAN, C. *et al*. **BIM Handbook**: a guide to Building Information Modeling for owners, managers, designers, engineers, and contractors. Hoboken: John Wiley & Sons, 2008.

FAN, S.; CHI, H.; PAN, P. Rule checking Interface development between building information model and end user. **Automation in Construction**, v. 105, 2019.

FREI, B.; SAGERSCHNIG, C.; GYALISTRAS, D. Performance gaps in Swiss buildings: an analysis of conflicting objectives and mitigation strategies. **Energy Procedia**, v. 122, p. 421-426, 2017.

GETULI, V. *et al*. BIM-based code checking for construction health and safety. **Procedia Engineering**, v. 196, p. 454-461, 2017.

GUREVICH, U.; SACKS, R. Development of a BIM Adoption Impact Map. In: LEAN AND COMPUTING IN CONSTRUCTION CONGRESS, Heraklion, 2017. **Proceedings [...]** Heraklion: ITC Digital Library, 2017.

HAN, C.; KUNZ, J.; LAW, K. Client/server framework for on-line building code checking. **Journal of Computing in Civil Engineering**, v. 12, n. 4, p. 181-194, 1998.

HJELSETH, E. Classification of BIM-based model checking concepts. **Journal of Information Technology in Construction**, v. 21, Special issue CIB W78 2015 Special track on Compliance Checking, p. 354-370, 2016.

HJELSETH, E. **Foundations for BIM-based model checking systems**: transforming regulations into computable rules in BIM-based model checking systems. Ås, 2015. PhD Thesis - Department of Mathematical Sciences and Technology, Norwegian University of Life Sciences, Ås, 2015.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. **ISO 24679-1**: fire safety engineering: performance of structures in fire: part 1: general. Geneve, 2019.

ISMAIL, A.; ALI, K.; IAHAD, N. A review on BIM-based automated code compliance checking system. In: INTERNATIONAL CONFERENCE ON RESEARCH AND INNOVATION IN INFORMATION SYSTEMS, Langkawi, 2017. **Proceedings [...]** Langkawi: Institute of Electrical and Electronics Engineers, 2017.

KASIM, T. *et al*. Automated sustainability compliance checking process: proof of concept. In: INTERNATIONAL CONFERENCE ON CONSTRUCTION APPLICATIONS OF VIRTUAL REALITY, 13., London, 2013. **Proceedings [...]** London: ITC Digital library, 2013.

KERN, A. P.; SILVA, A.; KAZMIERCZAK, C. S. O processo de implantação de normas de desempenho na construção: um comparativo entre a Espanha (CTE) e Brasil (NBR 15575/2013). **Gestão & Tecnologia de Projetos**, v. 9, n. 1, p. 89-102, 2015.

LEE, A.; BARRET, P. **Performance ased building**: first international State-of-the-Art report. Rotterdam: CIB, 2003.

LUKKA, K. The constructive research approach. In: CASE study research in logistics. Tyrku: Turku School of Economics and Business Administration, 2003. Series B1.

MAINARDI NETO, A. **Verificação de regras para aprovação de projetos de arquitetura em BIM para estações de metrô**. São Paulo, 2016. Dissertação (Mestrado em Engenharia Civil) - Escola Politécnica, Universidade de São Paulo, São Paulo, 2016.

MARTINS, J. P. P.; ABRANTES, V. Automated code-checking as a driver of BIM adoption. **International Journal for Housing Science and its Applications**, v. 34, n. 4, p. 287-295, 2010.

MEACHAM, B. *et al*. Performance-based building regulation: current situation and future needs. **Building Research & Information**, v. 33, n. 2, p. 91-106, 2005.

NARAYANASWAMY, H.; LIU, H.; AL-HUSSEIN, M. BIM-based automated design checking for building permit in the light-frame building industry. In: INTERNATIONAL SYMPOSIUM ON AUTOMATION AND ROBOTICS IN CONSTRUCTION, 36., Banff, 2019. **Proceedings [...]** Banff, 2019.

NAWARI, N. O. A Generalized Adaptive Framework (GAF) for automating code compliance checking. **Buildings**, v. 9, n. 4, 2019.

NAWARI, N. O. BIM Standard in off-site construction. **Journal of Architectural Engineering**, v. 18, n. 2, p. 107-113, 2012.

NAWARI, N. O. **Building Information Modeling automated code checking and compliance processes**. New York: Taylor and Francis, 2018.

OTTER, D.; EMMIT, S. Design team communication and design task complexity: the preference for dialogues. **Architectural Engineering and Design Management**, v. 4, n. 2, p. 121-129, 2008.

OYEGOKE, A. The constructive research approach in project management research. **International Journal of Managing Projects in Business**, v. 4, n. 4, p. 573-595, 2011.

PHAM, L.; BOXHALL, P. J.; SPEKKINK, D. Performance-Based Building Design Process – PeBBu – domain agenda and future development needs. In: CLIENTS driving innovation: moving ideas into practice. Brisbane: Cooperative Research Centre for Construction Innovation, 2006.

PILZER, D. Performance-based building regulations. In: PeBBu Domain 7 Final Report. Rotterdam: CIB, 2005.

RODRIGUES, J. **Utilização de modelos BIM para verificação automática de projetos**. Porto, 2015. Dissertação (Mestrado em Engenharia Civil) - Faculdade de Engenharia, Universidade do Porto, Porto, 2015.

SOLIHIN, W.; EASTMAN, C. Classification of rules for automated BIM rule checking development. **Automation in Construction**, v. 53, p. 69-82, 2015.

SUCCAR, B. Building Information Modeling framework: a research and delivery foundation for industry stakeholders. **Automation in Construction**, v. 18, n. 3, p. 357-375, 2009.

SZIGETI, F.; DAVIS, G. Performance-based building: conceptual framework. In: PeBBu Final Report. Rotterdam: CIB, 2005.

TAN, X.; HAMMAD, A.; FAZIO, P. Automated code compliance checking for building envelope design. **Journal of Computing in Civil Engineering**, v. 24, n. 2, p. 203-211, 2010.

YALCINKAYA, M.; SINGH, V. Patterns and trends in Building Information Modeling (BIM) research: a latent semantic analysis. **Automation in Construction**, v. 59, p. 68-80, 2015.

YANG, Q. Z.; XU, X. Design knowledge modelling and software implementation for building code compliance checking. **Building and Environment**, v. 39, n. 6, p. 689-698, 2004.

ZHANG, J.; EL-GOHARY, N. Integrating semantic NLP and logic reasoning into a unified system for fully-automated code checking. **Automation in Construction**, v. 73, p. 45-57, 2017a.

ZHANG, J.; EL-GOHARY, N. Semantic-based logic representation and reasoning for automated regulatory compliance checking. **Journal of Computing in Civil Engineering**, v. 31, n. 1, 2017b.

Eduardo Arantes
Departamento de Engenharia de Materiais e Construção, Escola de Engenahria | Universidade Federal de Minas Gerais | Av. Antonio Carlos, 6627, Pampulha | Belo Horizonte – MG – Brasil | CEP 31270-901 | Tel.: (31) 3409-1956 | E-mail: arantes@demc.ufmg.br

Paulo Roberto Pereira Andery
Departamento de Engenharia de Materiais e Construção, Escola de Engenahria | Universidade Federal de Minas Gerais | E-mail: paulo@demc.ufmg.br

Flávio Andrade
Departamento de Engenharia | MRV Engenharia e Participações S/A | Rua Lauro Ferreira, 101, 901-3, Buritis | Belo Horizonte – MG – Brasil | CEP 30575-080 | Tel.: (31) 99138-0326 | E-mail: flaviop@mrv.com.br

Daniel Paes
School of Built Environment | Massey University | East Precinct Albany Expressway, SH17 Quadrangle Building A, Level 3 | Albany | 0632 | Auckland, New Zealand | Tel.: +(64) 9212-7177 | E-mail: d.paes@massey.ac.nz

Javier Irizarry
School of Building Construction | Georgia Institute of Technology | 280 Ferst Drive | Atlanta, Georgia, USA | Tel.: +(1) 404385-7609 | E-mail: javier.irizarry@gatech.edu

Ambiente Construído
Revista da Associação Nacional de Tecnologia do Ambiente Construído
Av. Osvaldo Aranha, 99 - 3º andar, Centro
Porto Alegre – RS - Brasil
CEP 90035-190
Telefone: +55 (51) 3308-4084
www.seer.ufrgs.br/ambienteconstruido
www.scielo.br/ac
E-mail: ambienteconstruido@ufrgs.br