Code compliance checking of railway designs by integrating BIM, BPMN and DMN

Authors: Marco Häußler, Sebastian Esser, André Borrmann
Chair of Computational Modeling and Simulation, Arcisstraße 21, Technical University of Munich, Germany
E-mail: marco.haeussler@tum.de, sebastian.esser@tum.de, andre.borrmann@tum.de
Corresponding Author: Marco Häußler

Declarations of interest: none

Abstract

Code compliance checking has been the subject of scientific research for more than four decades and has been put into practice in numerous projects. To date, however, no universally valid, sustainable approach to the rule-based compliance checking of models has been established. Visual programming languages are easier to understand and thus more transparent than textual formats. The study presented here analyzes the requirements specified in the guidelines of the Deutsche Bahn AG regarding the technical design of structures in railway construction and examines the feasibility of implementing these rules using BPMN and DMN. The rules analyzed are categorized into 12 different classes. Depending on the guideline subset, the BPMN/DMN approach was found to be useable in 37%-75 % of the 943 rules examined. Considering only those rules that are relevant for the digital railway model, 68 % of the rules can be represented and automated using BPMN and DMN.

Keywords: BIM, Infrastructure, Quality, Railway, Design, Code Compliance Checking, BPMN, DMN

1 Introduction

Digital methods like building information modeling (BIM) offer considerable advantages in the construction industry over the conventional methods that still largely prevail in current practice. “BIM makes a new level of optimization possible. Most notably, it can afford greater planning, scheduling and cost certainty through transparency over the entire life cycle of a built asset. It simplifies risk management and provides better control of planning quality and industrial manufacturing processes. For the client, the main advantage of BIM is the creation of comprehensive, openly accessible building information that can be used by many different parties” [1]. The key motives for introducing BIM at a national level are therefore planning quality, cost minimization, risk management and increased efficiency (see also [2]).

The quality assurance concept for infrastructure planning presented in [3] identifies five domains to be considered over the course of model validation. While the domains ‘Clashes’, ‘Semantics’, ‘Construction Sequence’ and ‘Quantities and Costs’ have been examined in detail, the ‘Construction Design’ domain has not yet been addressed. This aim of this study is to fill this gap.

On the basis of [3], the ‘Construction Design’ domain is defined as the phase of creating a 3D digital building model taking into account the regulations set by the authorities. The domain is relevant for designers in the preliminary, conceptual and detailed design stages.

DIN EN ISO 9000-2015-11 defines ‘quality’ as the “degree to which a set of inherent characteristics of an object fulfils requirements” and “can be used with adjectives such as poor, good or excellent.” The term ‘requirement’ is defined as a “need or expectation that is stated, generally implied or obligatory” while “a specified requirement is one that is stated, for example in documented information” [4]. If the specified requirements are not met, EN ISO 9000-2015-11 speaks of non-compliance or errors. In relation to the design of building structures, compliance with the documented requirements for a building type at the design stage means that the design can be described as error-free and of excellent quality. “In the Architecture, Engineering, and Construction (AEC) industry, building projects must be checked against numerous building codes for compliance. They are allowed to be executed only when compliance with all applicable rules of the building code has been guaranteed. Failure to correctly assess projects for compliance can also have negative effects on building performance and allow errors that are expensive to correct.” [5]. The aim must therefore be to employ quality assurance measures that minimize both errors in design and, in turn, additional costs that may arise through deviations from the defined requirements. According to [4], ‘efficiency’ is defined as the relationship “between the result achieved and the resources used”. In the context of digitalization, this means that increasing automation of building model checks will result in a more efficient planning checking process.

The field of rail design, in particular, is subject to a large number of rules that must be complied with to ensure the functional and safe operation of railway infrastructure. Although, in contrast to many other fields, these rules are generally already elaborated in a highly formalized manner, they mostly exist as human-readable text and not in a machine-processable form. As such, the introduction of code compliance checking procedures is highly desirable, especially in railway engineering, so as to improve planning efficiency and ensure the necessary high level of planning quality. An essential prerequisite for code compliance checking is that the planning data is available in a digital form, preferably as a digital building model, also known as a ‘building information model’ (BIM). As this is now increasingly the case, this paper focuses on the digital description of rule sets and checking processes.

Code compliance checking has been the subject of scientific research for more than four decades [6] and has been put into practice in numerous projects. However, most studies have focused on building design and not on infrastructure projects. To date, no universally valid, sustainable approach to the rule-based compliance checking of models has been established [6]. Black-box solutions with hard-coded hidden, implementations of rules in vendor-specific solutions hamper the broader adoption of checking mechanisms [7,8]. To overcome this limitation, recent research has focused on the development of open and transparent methods of rule encoding, based on general-purpose programming languages or domain-specific solutions [9–12]. However, while textual programming languages tend to be challenging for AEC practitioners, visual programming languages enjoy increasing acceptance and widespread use by domain experts, as "information systems which are described by a visual language can be interpreted much faster and easier by humans" [13–15].

In contrast to previous studies, which are either software-specific or describe the development of a proprietary visual programming language, this study investigates an approach based on known and standardized elements of business-process modeling and makes use of Business Process Model and Notation (ISO/IEC 19510:2013). Alongside the visual, process-based representation of guidelines, a workflow engine was used to execute the processes developed and check existing models. In addition, the study investigates the extent to which existing definitions of linear reference systems already exist in the Industry Foundation Classes (IFC) data exchange standard that can be employed for checking purposes. The paper focuses on the automated checking of building information models representing the design of a railway project, in accordance with the quality assurance concept described in [3].

Thus, the research questions that this paper aims to answer are: (1) Are the standardized *Business Process Model and Notation* (BPMN) and *Decision Model and Notation (DMN)* sufficiently expressive for encoding the regulations of railway engineering in a manner that enables automated code compliance checking? (2) Do the railway BIM models presented in the IFC format provide all the required information?

The paper is organized as follows: Section 2 provides an overview of the state of the art on the subject of code compliance checking, before going on to describe and discuss the BPMN and DMN standards in more detail. While Section 3 introduces the relevant guidelines issued by German Railways (Deutsche Bahn AG), Section 4 discusses the representation of these rules by means of BPMN and DMN to achieve the desired automation of the code checking process. Section 5 investigates in detail to what extent the DB guidelines can be implemented using the developed approach. In Section 6, three representative case studies are examined, before Section 7 briefly presents the front end of the developed system. Section 8 concludes the paper and discusses its main findings.

2 Related work

2.1 Code compliance checking

Overview

Automated checking of standards and guidelines has been a focus of scientific research for many years, which is no surprise given the central role of regulations in the building industry: "As part of the design process, building designers ensure that every aspect of their design adheres to various regulatory requirements. The design is then subject to formal audit by the consent processing authority as part of the approval process" [16]. "The building industry uses numerous engineering standards, building codes, specifications, and regulations […] and a diverse set of industry vocabularies to describe, assess, and deliver constructed facilities. These building regulations are available as hardcopy and searchable digital documents. Some building design software applications (e.g., building-energy analysis and fire-egress assessment) are available that include computer-interpretable representations of the logic and rules from relevant building regulations" [17]. "Legal knowledge, in particular, is conveyed in voluminous paper-based documents in natural language text written for human interpretation" [18].

Until now, "the conventional practice of code compliance checking in the industry has largely been a manual process, which is laborious, costly, and error prone" [16]. In most countries throughout the world, conventional practice is based on drawings rather than semantically rich building models: "Nowadays the checking process is performed to a large extent manually based on two-dimensional technical drawings and textual documents by the responsible planning consultant as well as the building permission authorities" [19].

According to the McGraw-Hill Construction SmartMarket Report, compliance checking took 25 hours or more in 39% of the cases investigated, and in 11% of the cases more than 100 hours. On average, between 49 and 60 hours were spent on compliance checking [20].

The automated checking of models against guidelines, standards etc. is called 'automated code checking' or 'code compliance checking' and can result in "*speedier, dematerialized and more transparent review processes*" [21].

A large number of studies have been conducted on the subject, the majority of which focus on applications in building designs. Building models can be examined according to a variety of aspects, and therefore different quality criteria can be defined. Solihin and Eastman classify the possible quality criteria as follows [22]:

- *Checks for the well-formedness of a building model, i.e. the syntactic properties of the digital model*
- *Building regulatory code checking*
- *Specific client requirements*
- *Constructability and other contractor requirements*
- *Safety and other rules with possible programmed corrective actions*
- *Warrantee approvals*
- *BIM data completeness for handover to facilities management*

Phases of automated compliance checking

Eastman et al [23] identify four phases of the review process, which have also been adopted for this study:

1. *Interpretation and logical representation of rules*
2. *Building model preparation*
3. *Rule execution*

(4) Rule check reporting

Garrett et al. [17] define the translation and implementation process of review procedures as follows:

(1) Development of a simple and easily understandable representation syntax for building-regulation writers and software developers
(2) Provision of computerized support to enable regulatory organizations to easily develop, check, and maintain these regulation representations
(3) Checking the sufficiency and implementability of the digital representations

The three steps described by Garrett et al. focus on the design of digital representations. It is a more detailed view of the first phase of Eastman et al., who describes the necessary steps for the whole checking process.

"One of the key criteria [...] is to be independent of any specific model-checker software used to check the regulation compliance of building information models" [17].

Digital representation of regulatory rules

An important requirement of automated code checking is that it must represent the rules that a regulation or guideline contains in a form that lends itself well to computer processing.

Various software products exist for digitally checking building models, but as yet, none of them offer comprehensive functionalities for user-driven rule definition: "With the exception of SMC [Solibri Model Checker], none of the existing tools deals with the geometry and spatial operations that frequently adorn the BIM-based rules, especially in building codes" [22,24]. "While some model-checking software systems exist, they either require that their users possess good software-programming knowledge to configure them with rules of interest, or they are black boxes, and not configurable at all" [7]. "This 'hard-coding' of design standards into design programs is a major barrier to the general acceptance and evolution of computer-aided engineering, as it does not provide designers having to make professional judgments with the ability to view and understand the representations of the design standard on which the computations are based" [8].

When developing automated checking methods for building models, it is important to consider the user and thus the practicability of the method: "Countries such as Singapore, USA, and Australia have begun a new era that utilizes BIM for an automated, flawless administrative building permit process. However, most studies are focused on a script language-based result. This type of result has a high threshold for the user, who requires a building rule-checking process but has little understanding of the computer-based process and the programming language. This causes the rule-checking process to require professional human resources. Therefore, the above users always require a rule-checking process with computer-related experts" [25]. "Most of the existing approaches [are lacking,] because of the insufficient transparency and visibility of the processing steps for the user. Many methods focus too much on the automation of the checking process and do not consider the incorporation of the user and therefore the practical applicability" [19].

The unavailability of digital regulations is also an obstacle to the sustainable use of automated checking methods. "The encoding of norms into rules is currently a manual process. It is expected that the development of […] representations of legal documents is to be undertaken by the same government agencies responsible for authoring the legal documents in the first place" [18]. A common approach for translating plain text into digital representations is to use RASE technology, which appends regulation texts with tag markups (requirement, applies, select, exception) [26,27]. "Other attempts at automated model checking have taken the Natural Language Processing (NLP) approach and aim to automatically transform rules from human-readable specifications into programmatic executable code. While these methods have many benefits in terms of ease of use, there is usually far too much leniency in the written language, which makes it impossible to process automatically and accurately; as a result, these methods are fundamentally limited in their capacity to capture the requirements around compliance checking" [7]. To tackle these limitations, *Zhang & El-Gohary* used machine learning methods to examine different NLP approaches [28]. Despite the progress achieved, significant research still remains, in order to make the approach suitable for use with a large set of different standards.

Regardless of the method used to translate natural language into machine-interpretable language, various types of rules must be considered when checking building models. Solihin and Eastman [22] categorize the rule types as follows:

1. *Rules that require a single piece or small number of explicit data*
2. *Rules that require simple derived attribute values*
3. *Rules that require extended data structures*
4. *Rules that require a "proof of solution"*

This study focuses on rules of type 3.

Rule encoding approaches

The numerous approaches in the field of code compliance checking are summarized in [16,23]. *Charles* presents an approach to performing compliance checks using RDF [29] while *Xu & Cai* investigate how RDF, ontologies and SPARQL GIS-based data on utilities networks can be checked for compliance [30].

Bus et al describe the applications of Semantic Web methods for BIM checking [31,32] and *Zhang & El-Gohary* use Semantic Web as the basis for implementing their NLP approach [33,34]

Various investigations have focused on checking models with regard to safety aspects, for example safety on construction sites [35], in the context of fire protection regulations [19,36] or in connection with the structural design of buildings [37].

Both the interaction of the different actors in a construction project and the building model check itself is a gradual and thus process-based procedure. “The conventional compliance audit process is procedural in nature, which lend itself to automation. However, there are still roles in the process that are best played by human experts such as specifying what information to retrieve from which sources and how to process them. Machines excel in executing instructions efficiently and accurately and so should be given such a role to play in the process.” [18]

Visual programming languages

Myers investigated visual programming languages (VPL) as early as 1990 and declared them to be an interesting subject area, concluding that they “show promise for improving the programming process, especially for non-programmers” [38]. “The construction of programs is probably easier in VPLs than in textual languages” [39]. “A VPL […] describes a system of signs and rules on the syntactic and semantic level with the help of visual elements. Through the visual presentation of the elements, the language may be interpreted more quickly and easily” [40]. Green et al. “believe that in many respects VPLs offer substantial gains over conventional textual languages […]. Improvements in secondary notation, in editing and in searching will greatly raise their overall usability” [39]. *Catarci and Santucci* also conclude from their investigations that visual programming languages (in this case QBD) have advantages over classical query languages (in this case SQL), since users find the visual language more accessible and are therefore less bogged down by programming [41].

Preidel et al. are developing a query language called QL4BIM, which is available in both text-based and visual-language versions [40,42]. In [19,43,44], Preidel et al. are also developing a ‘visual code checking language’, which “is intended to perform compliance checks automatically or semi-automatically [and] increases the efficiency and quality of the overall process significantly.”

Ghannad et al. investigate the possibility of mapping the contents of regulations and standards on an xml basis as LegalRuleML notation using NLP methods and checking them using VPL. Currently, translation from LegalRuleML to VPL is done manually [45].

Ritter et al. examine the state of the art of visual programming languages in civil engineering and sum up the fields of application of visual programming languages as follows [46]:

- Inquiry languages
- Geometric modeling
- Knowledge-based design
- Design decision support
- Code checking
- Modeling of systems

The areas ‘Design Decision Support’ and ‘Code Checking’ are especially relevant for this study. “Information systems which are described by a visual language can be interpreted much faster and more easily by humans” [13]. “However, the state-of-the-art ACC systems cannot achieve full automation because they rely on the use of hard-coded, proprietary rules for representing regulatory requirements, which requires major manual effort in extracting regulatory information from textual regulatory documents and coding this information into a rule format” [33].

Numerous studies have been undertaken in the field of code compliance checking, and a recurring challenge is how to deal with the many regulations, norms and standards available only in human-readable form. ‘Natural language processing’ provides methods for translating these into a machine-interpretable form. The studies undertaken up to now have been limited to text-based regulations, which represent only a part of the content. However, the effort required to prepare texts for translation is currently still high. Instead of first writing regulations in their present form and then translating them, they should be designed in a machine-interpretable form from the outset.

The majority of the published approaches for code compliance checking focus on building designs and not on infrastructure projects. Many of the tools available on the market are ‘black box’ solutions, which makes it difficult for users to ascertain how checks are conducted and therefore how correct they are, in turn hampering their adoption. ‘Visual programming languages’ (VPLs), on the other hand, have the advantage that the checking process can be displayed graphically, which significantly increases readability. They also enable users to design inspection routines without any previous programming knowledge.

Both text-based and visual programming have advantages and disadvantages. This study aims to combine the advantages of both variants: comprehensibility, traceability, simplicity, automation, and adaptability. To this end, it examines the potential of a pre-existing notation taken from business process modeling to ascertain its applicability in the area of code compliance checking.

Building information models and the IFC standard

A prerequisite of automated code compliance checking is the availability of design information in a semantically rich representation, i.e. a building information model. Most of the existing checking systems make use of information provided in the Industry Foundation Classes (IFC) data format, which is a standardized vendor-neutral format for representing and exchanging geometric-semantic building information models.

The SEEBIM project, for example, investigated whether and how IFC models can be automatically enriched with additional information so that more extensive compliance checks can be performed. Predefined operators are available for rule creation, which in turn reduce programming complexity to the benefit of the user [47]. The suitability of the IFC data model for code compliance checking in the railway domain has not yet been investigated. Only recently has the IFC-Rail project published a first extension of the IFC data model to cover the railway domain [48].

Solihin et al [49] examine the quality criteria of IFC exchanges and conclude that there is "*an urgent need to define robust and rigorous test criteria, processes and tools.*" In this study, this is extended to include the technical checking of infrastructure models. *Zhang et al.* present a method for checking the conformance of IFC models using mvdXML [50].

2.2 BPMN: Business Process Model and Notation

VPLs are of particular interest when it comes to the interdisciplinary development (software development, civil engineering, and quality inspection) of automated inspection methods, due to their better comprehensibility. "In many enterprises, there is a need to facilitate smooth communication between business experts, software engineers and other people with technical knowledge. There are several methods of business knowledge representation, such as business rules or business process models. Both these representations can describe how the company works. Some issues, like constraints or detailed regulations, are better represented as rules, while others like procedures or workflows are better represented as process models" [51]. "The sequence of steps in a typical compliant design procedure can be represented as a series of activities, events, and sequence flows in a process model such as the open standard Business Process Model and Notation (BPMN)" [18,52].

BPMN is an international standard (ISO/IEC 19510) that is maintained by the Object Management Group [52]. *Dimyadi et al.* use BPMN to develop processes for the automated checking of building models [18,53,54]. In contrast to the work of Dimyadi et al. the study presented here focuses on railway projects and dedicated regulations, which are significantly more formalized than in the building domain. The BPMN-approach is additionally supplemented by the Decision Model and Notation (DMN), which will be in introduced in the next section.

The BPM notation provides a means of graphically representing processes in a formal manner. Various standardized node and edge elements are available, as illustrated in Figure 1.
Recker et al. describe BPMN as a "[..] structured, coherent and consistent way of understanding, documenting, modeling, analyzing, simulating, executing, and continuously changing end-to-end business processes and all involved resources in light of their contribution to business performance" [55]. And Janssens et al. state that "Business process management (BPM) and decision management (DM) are being used to improve the efficiency and effectiveness of organizations. Companies are interested in running effective and competitive processes, and use BPM to describe and improve these processes" [56].

In the AEC context, BPMN is often used to visualize the processes of model creation, collaboration and data exchange during planning (Information Delivery Manual or IDM) and larger development tasks, as shown in [57,58]. ISO 29481-1:2016 recommends the use of BPMN for the creation of IDMs [59], as implemented for example in [60,61]. The elements of BPMN used most frequently for creating IDMs are described in [62], which goes on to elaborate a proposal, based on the investigations, for which

BPMN elements should be used for the creation of IDMs (see Figure 1). “The most frequently used BPMN elements for representing business processes in the AEC industry were sequence flow, pool, lane, task/activity (exclusive), gateway, and message flow, which are all basic BPMN modeling elements” [62]. “One important type of activity in a BPMN-compliant process model is the script task, which allows the embedding of computer scripts that convey user-specified instructions such as where to retrieve which information and how to use the collected information to perform specific calculations” [18]. Many “modeling software systems also assist the user in developing processes using graphical specifications. In particular, the option to use BPMN in the context of workflow management systems as an implementation language [...] is an important criterion for use in construction projects” [63].

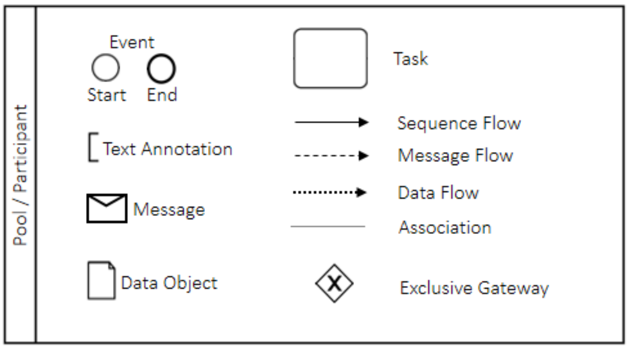


Figure 1: Proposal for the ‘Essential Subset’ of BPMN for IDM Development [62]**.**

How the process itself can be subjected to a compliance check is explained in more detail in [64–66]. *Awad* presents a method of querying process diagrams using query language (BPMN-Q) to identify similar patterns in different process diagrams [67].

Recker states that “‘Classical’ process management applications such as documentation, redesign, continuous improvement and knowledge management dominate the application areas of BPMN, while more technical application areas such as software development, workflow management or process simulation are not (yet) widespread” [68]. This a significant gap, which this study aims to help fill.

In addition to visualizing a process, it is possible to execute it automatically once generated, using a workflow engine that performs the process step by step and in accordance with the modeled logic and can react to events during runtime as well as trigger events itself. Such an event might be a data input by the user, the calculation of mathematical formulae, a decision resulting from an if-then condition, or the execution of text-based source code. By virtue of its ability to automate the developed processes by means of a workflow engine, BPMN can be classified as belonging to the category of visual programming languages. BPMN has already found its way into the construction industry, in particular for IDM development, but there are now also initial approaches to using the method in the area of code compliance checking. This is promising because, besides graphical notation, it also enables the development of individual extensions using script tasks to address aspects not covered by the notation.

A script task makes it possible to incorporate individual instructions, such as formulae, in the process, using script languages like Javascript or Groovy.

The utilization of BPMN for automating code compliance checking is something that is rarely addressed. Among the few published approaches, *Kog et al.* use Petri nets to check BPMN-based construction processes [69] and *Zolfagharian and Irizarry* use BPMN to visualize the process for the automated checking of construction site equipment models [70].

2.3 DMN: Decision Model and Notation

As discussed in the previous section, it is possible to represent processes visually by means of BPMN and execute them with the help of workflow engines. BPM notation’s gateway elements can be used to map simple decisions (in if-then-else relationships), but if several criteria within these relationships need to be evaluated, the process quickly becomes confusing. To simplify this situation and to present decision options more clearly, the ‘Decision Model and Notation’ (DMN) was developed, version 1.3 of which was published in December 2019. “The primary goal of DMN is to provide a common notation that is readily understandable by all business users, from the business analysts needing to create initial decision requirements and then more detailed decision models, to the technical developers responsible for automating the decisions in processes, and finally, to the business people who will manage and monitor those decisions. DMN creates a standardized bridge for the gap between the business decision design and decision implementation. DMN notation is designed to be usable alongside the standard BPMN business process notation” [71]. “The separation of Decision Modeling from Business Process Modeling is a good principle, whatever the models, notations and languages are. It will enhance agility when changes are required, in reducing their impact, among them the risk of failure, and in increasing the resilience of the Information System” [72].

As early as 1969, *Fenves et al.* described a method of automating decision paths with decision tables [73]. *Huysmans et al.* compared several methods of presenting decision paths in a machine-readable form and validated them against their comprehensibility for users. “The results showed that, on the aspect of comprehensibility, decision tables provide significant advantages. For each part of the experiment, the respondents were able to answer the questions faster, more accurately and more confidently using decision tables than using any of the other representation formats” [74]. “The use of DMN for modeling the requirements for automated decision-making is similar to its use in modeling human decision-making, except that it is entirely prescriptive, rather than descriptive, and there is more emphasis on the detailed decision logic. For full automation of decisions, the decision logic must be complete, i.e., capable of providing a decision result for any possible set of values of the input data” [71].

The DMN is therefore a useful supplement to the BPMN that makes it possible to keep process representations clear, understandable and easily comprehensible. With the help of DMN, process representations can be reduced to their essential parts. Like workflow engines, so-called decision engines enable automated decisions to be made. By integrating DMN into BPMN, the decision engine can be triggered by the workflow engine. This study shows how DMN and BPMN are integrated to achieve a high level of automation of code compliance checking in railway design.

2.4 BIM in railway design and construction

BIM is being increasingly adopted in the railway domain worldwide. A number of national railway organizations have declared BIM as the obligatory way forward in improving quality in design and construction while reducing project costs and delays. To reach these goals, such organizations publish master-plans and/or define mandatory standards. These include China Railway [75,76], Korean railways [77], French railways [78], Swiss railways [79], Italian railways [80], Swedish, Danish, Norwegian and Finnish railways [81,82] and German railways, among others.

BIM has been mandatory for DB Station & Service AG projects since 2017 [83]. It is the first company within the Deutsche Bahn AG Group to issue standardized *"Guidelines for the Application of the BIM Methodology"* for the provision of model-based planning and construction services. While the guidelines also mention checking methods for the quality assurance of models, including collision detection, visual inspection and compliance with regulations, they do not describe any concrete implementation strategies. Most of the quality assurance criteria are organizational criteria, for instance, concerning adherence to deadlines. The quality assurance concept developed by the authors, on the other hand, defines strategies for implementing and deriving model-based quality assurance criteria [3]. To check the digital building model, it is necessary to scrutinize both the model creation process and the transfer of model data.

2.4.1 Geometric modeling

This paper examines methods for checking the compliance of 3D infrastructure models and the underlying route mapping with the respective guidelines. To begin with, it is necessary to differentiate between "two fundamentally different approaches to modeling the geometry of three-dimensional bodies: Explicit modeling, which describes a volume in terms of its surface […]. Implicit modeling by contrast employs a sequence of construction steps to describe a volumetric body, and is therefore commonly termed a procedural approach" [84].

Borrmann et al. [84] outline the following procedures in detail:

- Explicit procedures
    - Boundary representation method
    - Triangulated surface description
- Implicit procedures
    - Constructive solid geometry
    - Extrusion and rotation processes
    - Parametric modeling
    - Freeform curves and surfaces

“Most of the software products used for infrastructure planning offer a drawing-oriented view – split into site plan, cross-section and elevation […]. This type of model is referred to as an implicit geometry description. […] With implicit models, the governing design parameters become significantly more accessible than with explicit models. An example is the objects and parameters that define alignment. Consequently, implicit models offer a better basis for checking these parameters against codes and guidelines” [3].

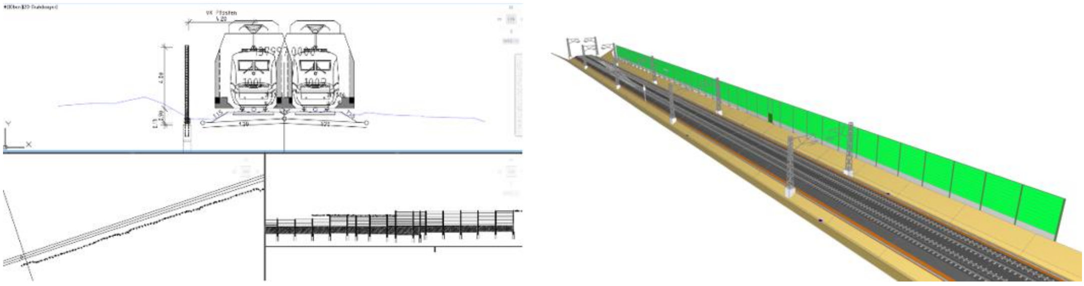


Figure 2: Comparison of implicit and volumetric 3D models: while implicit models (drawing-oriented view) are used during the design process, explicit models are used in the context of BIM-based analysis [3]**.**

In addition to the pure parametric description of digital building models, some software systems also keep a history log of the individual modeling steps and enable the modification of individual steps, which is an improvement over the drawing-based method. “The […] important concept provided by parametric CAD systems is the explicitly available construction history. The system records each single construction operation […]. All operations are parameterized, for example, the height of an extrusion is an explicitly available parameter. The maintenance of the construction history stands in strong contrast to conventional systems which typically only store the result of the construction operations as an explicit boundary representation. The procedural approach provides the user of the system with the possibility to easily modify an existing model by going back in the construction history and adapting the corresponding parameter […]” [85]. “Explicit parameters of building elements are one of the merits of a building model and resolve difficulties in interpreting code checking” [23]. “[…] It is equally important to have efficient access to the building data as it is having good, transparent, and maintainable computable rules. Efficient access to the building data is important since rule checking potentially has to iteratively go through almost the entire data set with a large number of rules. […] Another critical issue that has been absent in the discussion so far is the support for geometry and spatial operations on the BIM data. Studies of building codes show that the majority of rules involve geometry and spatial operations” [24]. “A parametric modeling system will require careful engineering judgment and responsibilities in setting up the input and reviewing the output and a method to specify the requirements in an unambiguous way” [86].

While planning and modeling tools used for railway design employ implicit data models, they support the generation and export of explicit geometries using the boundary representation method or triangulated surface descriptions. This information can then be transferred using data exchange formats such as Industry Foundation Classes.

2.4.2 Industry Foundation Classes

The vendor-neutral data model Industry Foundation Classes (IFC) allows the high-quality exchange of geometric-semantic models. Over the past few years, it has been continually extended for use in the model-based data exchange of infrastructure facilities. The ‘Overall Architecture’ project and the IfcAlignment extension (also known as IFC4x1) first introduced a set of classes for the exchange of alignment axes and digital terrain models [87]. These initiatives aimed to provide basic concepts that are now supplemented by domain and subject-specific requirements. In the context of railway design, *Reifenhäuser et al.* have expressed the criticism that although horizontal and vertical parts of an alignment geometry can be mapped with IFC-Alignment classes, comprehensively representing a railway line also requires that cant is taken into account [88]. This shortcoming has since been recognized and has recently been corrected in the extension project *IfcRail.* Similarly to the completed IfcBridge extension project [89,90] in the context of bridge design, the IfcRail project deals with the domain-specific requirements of vendor-neutral data exchange for railway design.

Even though IfcRail development is still in progress, the IFC4x1 version of IFC instance models (the most recent final schema version) can already be used for numerous automated rule checks in railway design. For example, an alignment is described in the IFC data model by its horizontal segments in the xy-plane and the corresponding vertical segments in the projected coordinate system (s,z) [91]. In addition, it is possible to place *IfcReferent* objects at specified locations along an alignment axis and to attach user-defined property sets to them. The parameters in these property sets are not defined in the data schema and provide the modeler with a very flexible method for transferring project- or organization-specific attributes. The downside is that any flexibility in a data schema also requires specific processing of user-defined, non-standard properties must also be specifically addressed for import processes, which makes automatic interpretation considerably more difficult. Nevertheless, this ability to dynamically extend the schema with additional semantics is a useful feature of IFC, making additional rules testable.

In many parts of the IFC data model, it is possible to choose between different representations and dynamic semantic extensions. This flexibility, however, inevitably means that not every instance model can be used for every check routine, since the data cannot be extracted in a consistent form. To counteract this, the model to be checked must comply with additional specifications [22] that can be defined through the mechanism of a model view definition (MVD). An MVD defines a subset of the entire IFC data model and thus restricts, for example, the geometric presentations that may be used [92]. At the time of writing, there are no internationally defined and published MVDs that reflect the correct use of *IfcAlignment* instances or related concepts such as *IfcLinearPlacement* for the description of positions along an alignment element. Nevertheless, the schema definition contains so-called *concept templates*, which represent specific rules of a complete MVD and can be assembled in a modular fashion.

An MVD for specifying the required input data can be elaborated using the *IfcAlignment Concept Template* and defining any additional necessary parameters.

The latest candidate standard extension is called IFC4x3 and includes the extension proposals of IfcBride, IfcRoad, IfcPortsAndWaterways and IfcRail projects [93].

3 The DB Netz AG guidelines

Due to the very stringent safety requirements that apply to rail facilities, numerous guidelines are in place that strictly regulate design as it relates to rail infrastructure facilities. While aesthetic aspects of the structure are important for the individual expression to the respective building designs, infrastructure facilities are on the whole characterized by a high degree of standardization both with components and geometric dimensions.

The study henceforth takes the guidelines of Deutsche Bahn AG as an example, but the principle can be extended to other countries with similar guidelines of their own. Comparable regulations also exist for road infrastructures.

The guidelines are presented in a clearly defined structure. In the context of code compliance checking with building information models, it is the construction engineering guidelines, in which most of the rules for design and construction are defined, that are the most interesting. The main groups are divided by trade into sub-groups. The guidelines examined in this study are as follows:

Table 1: Assignment of trades or sub-groups to guideline numbers.

Trade/sub-group	**Guideline number**
Network infrastructure technology design	800
Basics of superstructure	820
Earthworks and other geotechnical structures	836

Guideline Group 800 defines the parameters for track routing and sets out the basis for the entire track system. It includes definitions for switches and crossings as well as for track cross-sections and regulates the dependencies between individual objects. It also sets boundary conditions such as the design speed. Guideline Group 820 defines the design of the superstructure, such as the rail form and sleeper type, while Guideline Group 836 specifies the conditions relating to the required civil engineering measures, for example, earthworks, drainage, etc. These three groups set out the most relevant guidelines for the planning, construction and operation of a railway infrastructure and are therefore examined in detail in this study. Further guidelines exist that deal with technical equipment (e.g. control and safety equipment and overhead catenary lines) and structural engineering (e.g. bridges and tunnels) but are not considered here.

Particular attention must be paid to the modal verb formulations used in the guidelines, as they are used to indicate specifically whether the item in question is a requirement, permission, recommendation or possibility and capability. Table 2 shows the terms used and their implications based on the guidelines and DIN 820-2 [94–96]. As these are relevant for both planning and manual quality control, it is essential that they are used accordingly in model-based quality assurance.

Table 2: Meaning of auxiliary modal verbs in DB Netz AG guidelines according to [94–96]**.**

Auxiliary modal verb	Meaning
Shall	Requirement
Shall not	
May	Permission
Need not	
Should	Recommendation
Should not	
Can	Possibility and capability
Cannot	

4 Rule representation using BPMN and DMN

Building information modeling is used increasingly in practice for designing buildings. The quality assurance concept presented in [3] has therefore been taken as the basis for the concept presented in this study. Our concept additionally considers the domain of ‘Construction Design’, which is concerned with checking digital building models during the design stages. An essential aspect here is the ability to perform a digital and machine-evaluable representation of the existing guidelines for automated model checking. “Case examples and requirements in guidelines [...] can be presented in many different ways, ranging from simple and clearly structured tables with limiting values to graphical representations or written descriptions” [19]. This also applies to the guidelines of Deutsche Bahn AG. For the purpose of the concept, the process of checking models against guidelines involves performing calculations and comparisons and making decisions. The process-based description of the guideline content is represented by BPMN and DMN. The following section begins by formulating the general requirements for automation before going on to describe the logic of translation into a machine-interpretable form. The translation is done manually, but supported by the RASE methodology [26], which is a semi-formal process for analyzing regulatory text in natural language and marking it up for further processing.

4.1 General requirements for automation

BPMN, as a node-edge model comparable to visual programming languages, provides a set of process elements to enable the representation of policy content or rules. Node elements include events (e.g. start, end), tasks (e.g. user tasks, script tasks, or DMN tasks), and gateways (e.g. exclusive, parallel) as shown in **Figure** 1. Edge elements (e.g. sequence flow) connect the respective nodes into a continuous process. In addition, it is possible to aggregate sub-processes into a higher-level process to enable the hierarchical organization of processes.

To facilitate automation, the following conditions apply when creating processes:

(1) Every process has a start and an end event.

(2) Events, tasks, and gateways are connected by flow elements.
(3) All tasks are addressed and initiated by a start event.
(4) All tasks are integrated into the process in such a way that they are connected to an end event.
(5) Gateways have at least two output edges, but there may be a default path.

4.2 Principles of process modeling

As discussed by *Preidel & Borrmann in* [19], regulations can be presented in various ways, for example as continuous text, tables, graphics, or formulae. This section describes the principles of process modeling and explains how guidelines are represented in this concept using BPMN and DMN.

Textual descriptions place a rule in an overall context and describe decision paths, individual parameters and entire parameter sets. They also contain embedded formulae as well as references to graphics, tables, attachments, etc. Typically, the superordinate process description is textual.

Graphics are used to clarify the content of a textual description and illustrate component interdependencies as well as parameters and boundary conditions that need to be considered. Because they can be part of an aggregated representation, graphics can describe both individual as well as multiple rules.

4.2.1 Translation of regulations

RASE-syntax [26] is used to support the translation of regulatory texts into workflow diagrams. The regulations are structured with the tags *requirement <r> or <R>*, *applicability <a>*, *select <s>* and *exception <e>*.

The following section illustrates the process of translating a rule using the example of the routing element 'arc' and a chosen radius. The resulting process and its individual steps are shown in Figure 4, along with the textual descriptions.

Guideline 800.0110 § 6 No. 3 defines the following requirement:

The arc radius (r) of a track curve is calculated by the formula [Equation 1] taking into account the speed (v), superelevation (u) and superelevation deficit (u_f) and shall not be greater than 25.000 m.

$$r = 11{,}8 \times \frac{v^2}{u + u_f}$$

Equation 1: Determination of the arc radius of a track curve according to Guideline 800.0110.

The marked rule following the RASE-syntax looks like this:

<R>The <a>arc radius (r) of a track curve is calculated by the formula <r>[Equation 1]</r> taking into account the speed (v), superelevation (u) and superelevation deficit (u_f) and shall <e>not be greater than 25.000 m</e>.</R>

The rule does not contain any selection.

4.2.2 Representation with BPMN and DMN

In principle, all notational elements described by ISO/IEC 19510 may be used, but in this study, the elements shown in Figure 3 (script task, gateway, DMN task) were found to be particularly suitable and therefore play a focal role.

Guideline specifications in the form of calculation formulae (requirements) are incorporated in the workflow by means of a script task (see Figure 3, number 1). The calculation formula is written in Javascript.

Both text and tables can describe decisions (select-tag in RASE) that can usually be resolved into simple if-then-else relationships. Such decisions can be represented in the process by a gateway or as a DMN task. If the result of the decision is described with formulae, representation by means of a gateway (see Figure 3, number 2) and the following script task is chosen, since DMN tasks do not accommodate formula-based evaluations. The if-condition is formulated using the flow elements assigned to the gateway. A gateway can also be used to compare individual parameters with the model independently of other boundary conditions.

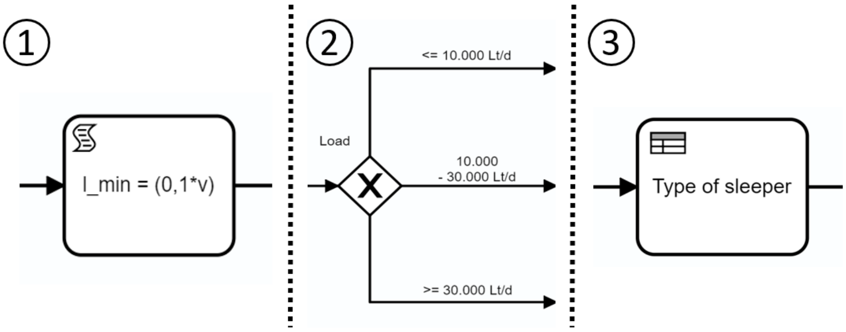


Figure 3: BPMN elements: 1 – Script task used to represent formulae; 2 – Exclusive gateway used to represent simple decisions/branches; 3 – DMN task used to represent complex decisions.

A DMN task (see Figure 3, number 3) is used if the decision result depends on several input variables (if-conditions) and is expressed as a static value (no formula). When creating a decision table, the necessary input variables and results must be defined. Table 3 shows a schematic decision table. Compared with a succession of numerous gateways, the tabular representation of several decision paths is both clear and easily understandable. There is no limit to the number of input or output variables or rules to be considered.

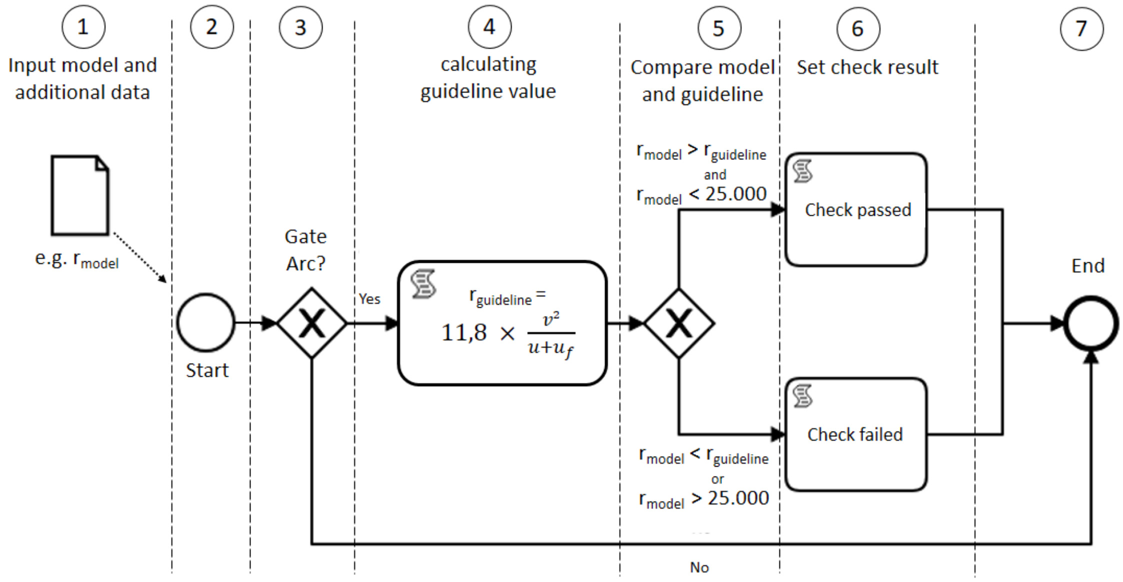
Table 3: Schema of a DMN decision table, in which any number of rules can be created. To generate a decision, the necessary input and output variables, including associated values, need to be defined.

Rule number	**Input**			**Output**		
	Input variable i_1	***Input variable*** i_2	***Input variable*** i_m	***Output variable*** o_1	***Output variable*** o_2	***Output variable*** o_z
1	*value* $i_{1,1}$	*value* $i_{1,2}$	*value* $i_{1,m}$	*value* $o_{1,1}$	*value* $o_{1,2}$	*value* $o_{1,z}$
2	*value* $i_{2,1}$	*value* $i_{2,2}$	*value* $i_{2,m}$	*value* $o_{2,1}$	*value* $o_{2,2}$	*value* $o_{2,z}$
3	*value* $i_{3,1}$	*value* $i_{3,2}$	*value* $i_{3,m}$	*value* $o_{3,1}$	*value* $o_{3,2}$	*value* $o_{3,z}$
n	*value* $i_{n,1}$	*value* $i_{n,2}$	*value* $i_{n,m}$	*value* $o_{n,1}$	*value* $o_{n,2}$	*value* $o_{n,z}$

In this way, rules are broken down into the smallest constituent units so that the process representation is comprehensible and can be easily followed. Consequently, complex calculations or observations are divided into multiple tasks. To ensure that the method is accessible and comprehensible to users without programming knowledge, only simple formula and variable definitions are used in the process. Textual descriptions of individual elements and additional text annotations can be used to provide explanations of the process modules.

Mapping into BPMN is described in this section on the basis of a marked paragraph, following the RASE-syntax described in Section 4.2.1 (compare also Figure 4). The paragraph of the guideline begins with a description of the input variables (velocity, superelevation, superelevation deficit) necessary for calculating the arc radius and then describes their interdependency in terms of a formula. Therefore, to enable automated checking of the digital building model, it is essential that the necessary input variables are made available for processing (Step 1). Alongside the quantities declared in the guideline, the modeled radius must also be passed as a parameter. The process can then be started (Step 2). It is necessary to confirm that the model data will be checked by the appropriate rules, e.g. that an arc will be checked by rules dealing with arc information. This is taken care of by a gateway in Step 3 which represents the RASE-tag applicability. The radius calculation formula according to the guideline is represented by a script task (Step 4), and the result of the calculation is stored as an independent variable. The model’s compliance with the guideline is checked by comparing the modeled radius with the radius calculated according to the guidelines. This takes the form of a simple if-then condition in the form of a gateway (Step 5), whereby the two sequence flows contain the corresponding decision logic for the subsequent process path. The flows also contain the exception to the rule. Depending on which path is automatically selected, the check result is set (Step 6) to either ‘passed’ or ‘failed’. At the end of the process, the result is returned to the user (Step 7).

Figure 4: Process definition of a simple rule: Guideline 800.0110 – calculation of the arc radius and validation of model data.



From the example described, it is possible to formulate generally valid steps for the creation of a process:

(1) The model data to be checked is transferred to the workflow engine. Information that is not defined in the model but is essential for correct execution of the process is defined by the user and passed to the workflow engine together with the model data to be checked for compliance.
(2) In a pre-processing stage, the data is prepared for the workflow (transfer to variables, adaptation of dot-comma notation, etc.) and passed to the start of the checking process.
(3) The workflow engine selects objects and parameters from the submitted data set to be checked for compliance. If the workflow is not usable for checking submitted objects or parameters, the process is skipped.
(4) The workflow engine determines the target parameters based on the guidelines for the defined boundary conditions of the model and other information.
(5) The workflow engine compares the target parameters with the transferred model data (actual parameters).
(6) The workflow engine defines whether the check has passed or failed.
(7) The checking process is terminated, and the results are made available to the user in the form of a report.

5 Evaluating guidelines for implementation with BPMN

Studies of norms and standards in England and Wales have shown that 20% of the rule sets examined are declarative and thus directly computer-interpretable, while 47% need additional human intervention to make them computer-readable. In 33% of the cases examined, automation is not possible [97].

The extent to which models can be checked for compliance with the guidelines of Deutsche Bahn AG using BPMN and DMN was checked using Guideline Groups 800, 820 and 836 and analyzed for a total of 943 rule sets. Of all the rules analyzed, 486 (52 %) can be generally classified as automatable. A rule is classified as automatable if all of the following criteria apply (necessary conditions):

1. All necessary input variables have been described.
2. Boundary conditions/decision paths to be considered have been described.
3. All required output variables have been described.

To be able to automate the content of guidelines, the rules must have a deterministic description. Policy contents are classified as non-automatable when at least one of the three criteria is not met or the contents aim to describe the readability and structure of the policy.

As described in Section 2.4, BIM models in infrastructure planning are created with the aid of parameters (e.g. length, height, width of objects) using an implicit data model. Analysis of the guidelines includes examining the extent to which guidelines describe the parameters that are required for modeling. A total of 460 rules (49%) were classified as 'parameter-oriented'.

A total of 437 rules (46%) were classified as 'BPMN implementable', while 3% of the rules (32 in absolute terms) were evaluated as only implementable by means of higher-level quality tests, such as simulations.

A comparison of the evaluation results at policy level (see **Table 4**) reveals that Policy Group 800 in particular is predominantly parameter-oriented (82%), so here, models can be checked to a great extent using BPMN (75%). The rules contained in Guideline Groups 820 (38%) and 836 (39%) are far less parameter-oriented, and consequently have significantly lower potential for implementation with BPMN (37% and 39%, respectively).

Overall, the potential for implementing automated rule checking by means of BPMN was deemed sufficiently high (46% of the analyzed rules), which is a valuable basis on which to continue with the method, despite the unequal distribution across the individual guidelines.

Table 4: Evaluation of feasibility of Guideline Groups 800, 820 and 836 for checking models for guideline compliance.

Guideline group	Number of paragraphs	Automatable?	Parameter-oriented?	Realizable with BPMN?	Simulation necessary?
800	223	187 (84%)	182 (82%)	168 (75%)	19 (9%)
820	203	85 (42%)	80 (39%)	80 (39%)	1 (0%)
836	517	214 (41%)	198 (38%)	189 (37%)	12 (2%)

Over the course of the detailed analysis, the examined rules were divided into 12 different classes. These rule classes are defined in **Table 5**. Figure 5 shows a graphical representation of the classes 2, 4 and 7.

Rule classes are mainly defined by their typology. In the case of distance definitions, the rule classes are subdivided, depending on the influence they have on direction in a 3D-coordinate system.

Table 5: Definition of rule classes and their respective descriptions.

No.	Rule class	Description
1	Not relevant for modeling	Specifications that have no influence on the modeling of buildings, e.g. procedural processes in communications with the client or specifications for document structures such as explanatory reports.
2	Distance definitions: transverse direction	Definitions (horizontal distances) that can be derived from the cross-section of the building model (see Figure 5).
3	Distance definitions: position	Site plan view in the sense of a 3-panel projection: distances between elements.
4	Distance definitions: height	Definitions (vertical distances) that can be derived from the cross-section of the building model (see Figure 5).
5	Component definitions	Specifications determining the component or component type to be used.
6	Definitions for alignment elements	Definitions that define parameters for vertical and horizontal alignment elements (straight line, arc, transition arc).
7	Slope definitions	Definitions (slope distances) that can be derived from the cross-section of the building model (see Figure 5).
8	Directional definitions: Longitudinal direction	Orientations of components in the layout plan.
9	Clearance tests	Rules for ensuring clearance along railway lines.
10	Other definitions	Rules that do not correspond to the other rule classes but still have an impact on the building model.
11	Construction process	Rules that define temporal components of a construction site.
12	Dimensioning specification	Rules for dimensioning components, such as hydraulic calculations.

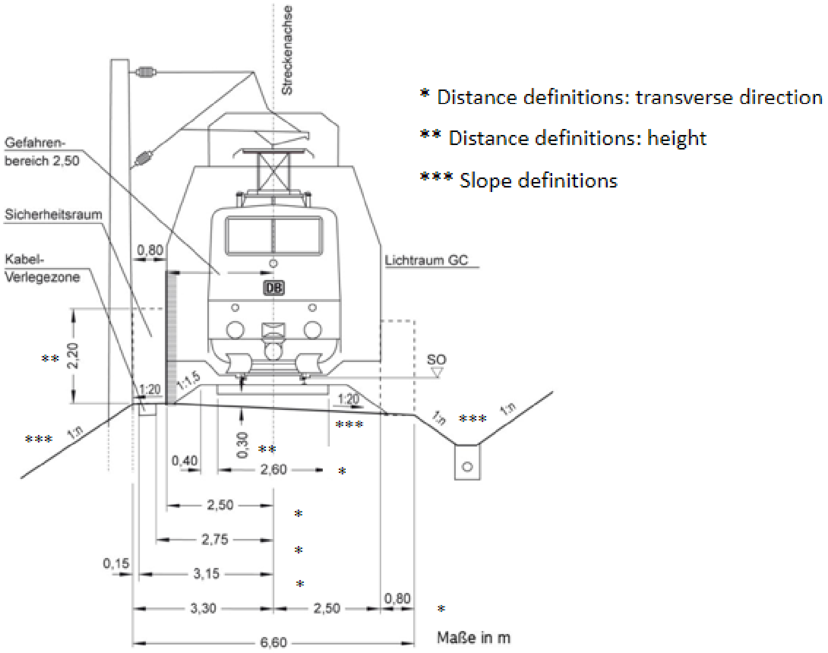


Figure 5: Standard cross section of a single-line railway alignment incl. corresponding distances (drawing adapted from [98]**).**

Table 6 shows the detailed evaluation of the analysis. No distinction is made with respect to the underlying guideline; rather, the results shown are cumulated for the examined guidelines. Column 2 of Table 6 shows the number of evaluated rules. The large proportion of rules classified as ‘not relevant for modeling’ (33%) is striking, and consequently column 3 shows the distribution with rule class 1 factored out. Column 3 shows the number of rules per rule class that can be implemented using BPMN/DMN. When the rules not relevant to model creation are factored out, the proportion of rules that can be implemented using BPMN increases from 46% of all those reviewed to 68% of those in rule classes 2 to 12. The three most frequently occurring classes that can be implemented with BPMN are 2, 4 and 5.

Table 6: Detailed evaluation of the guideline analysis according to the rule classes defined in Table 5. The top three rule classes are highlighted.

1	2	3	4
Rule class	Number of rule sets examined	Ratio convertible with BPMN/ DMN, without rule class 1	Ratio not convertible with BPMN/ DMN, without rule class 1
Total	**943 (100%)**	**432 (100%)**	**202 (100%)**
1 Not relevant for modeling	309 (33%)	- -	- -

2	Distance definitions: transverse direction	77	(8%)	74	(17%)	3	(1.5%)
3	Distance definitions: position	46	(5%)	43	(10%)	3	(1.5%)
4	Distance definitions: height	69	(7%)	69	(16%)	0	(0%)
5	Component definitions	166	(18%)	141	(33%)	25	(12.5%)
6	Definitions for alignment elements	10	(1%)	10	(2%)	0	(0%)
7	Slope definitions	33	(3%)	32	(7%)	1	(0.5%)
8	Directional definitions: Longitudinal direction	9	(1%)	8	(2%)	1	(0.5%)
9	Clearance tests	24	(3%)	3	(1%)	21	(10.5%)
10	Other definitions	72	(8%)	25	(6%)	47	(23%)
11	Construction process	1	(0%)	1	(0%)	0	(0%)
12	Dimensioning specification	127	(13%)	26	(6%)	101	(50%)

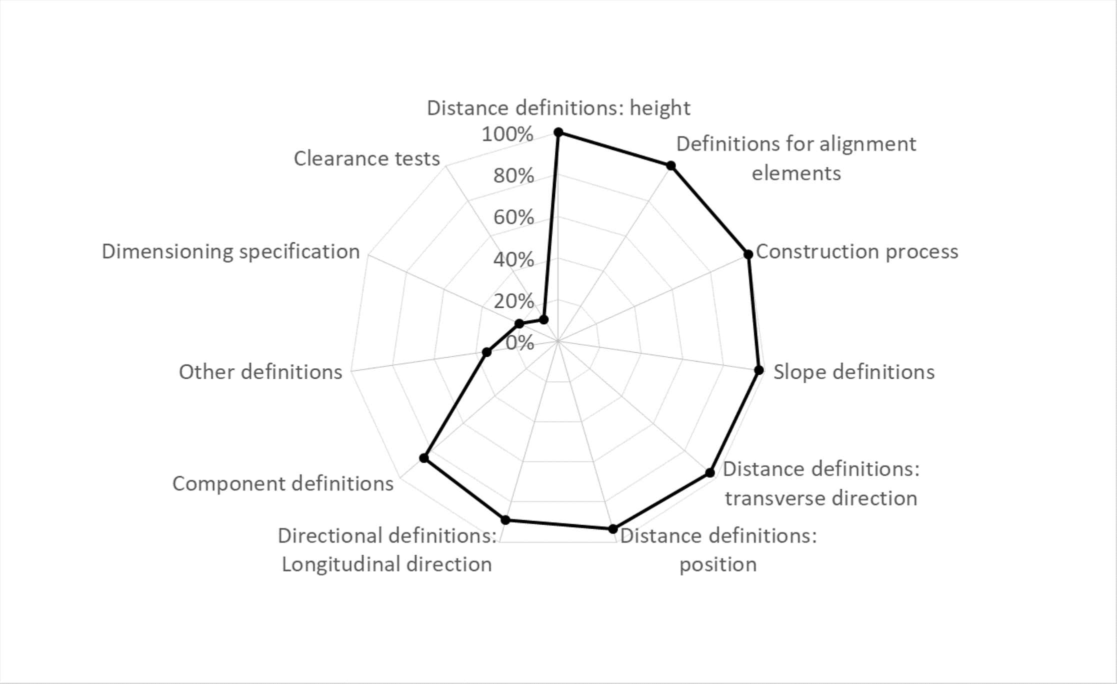
Column 4 shows the ratio of rules that *cannot* be implemented with BPMN and DMN. The rule classes ‘Dimensioning specification’, ‘Other definitions’, ‘Component definitions’ and ‘Clearance tests’ are the four most common ones whose rules cannot be implemented with BPMN and DMN. These rules or rule classes account for 30.6% of all rules relevant to the model. Reasons for being classified as ‘not implementable with BPMN and DMN’ are:

- Rules and their descriptions do not fulfill the necessary conditions for automation
- Sophisticated software for structural analysis or simulation is necessary
- Clearance tests are better performed as clash detection

The remaining rule classes thus account for 1.3 %. The rule class ‘Component definitions’ is an interesting case, as it is also the class whose rules can be automated most often with BPMN and DMN. This is due to the comparatively large number of rules that this class contains.

Diagram 1 shows the relationship between the total number of rules checked and the rules that can be implemented with BPMN and DMN. With a rate of between 12% and 35%, the rule classes ‘Dimensioning specification’, ‘Other definitions’, and ‘Clearance tests’ are much less frequently automatically verifiable with BPMN and DMN than the other classes, whose rates range between 85% and 100%.

Diagram 1: Ratio of ‘total’ rules to ‘rules implementable with BPMN/DMN’.



The following section considers the three rule classes ‘Component definitions’, ‘Distance definitions: transverse direction’, and ‘Distance definitions: height’ in more detail, as they contain the most frequently occurring rules. These are discussed in case studies. A further example highlights the importance of vertical and horizontal alignment in infrastructure planning.

6 Case studies

6.1 Software configuration

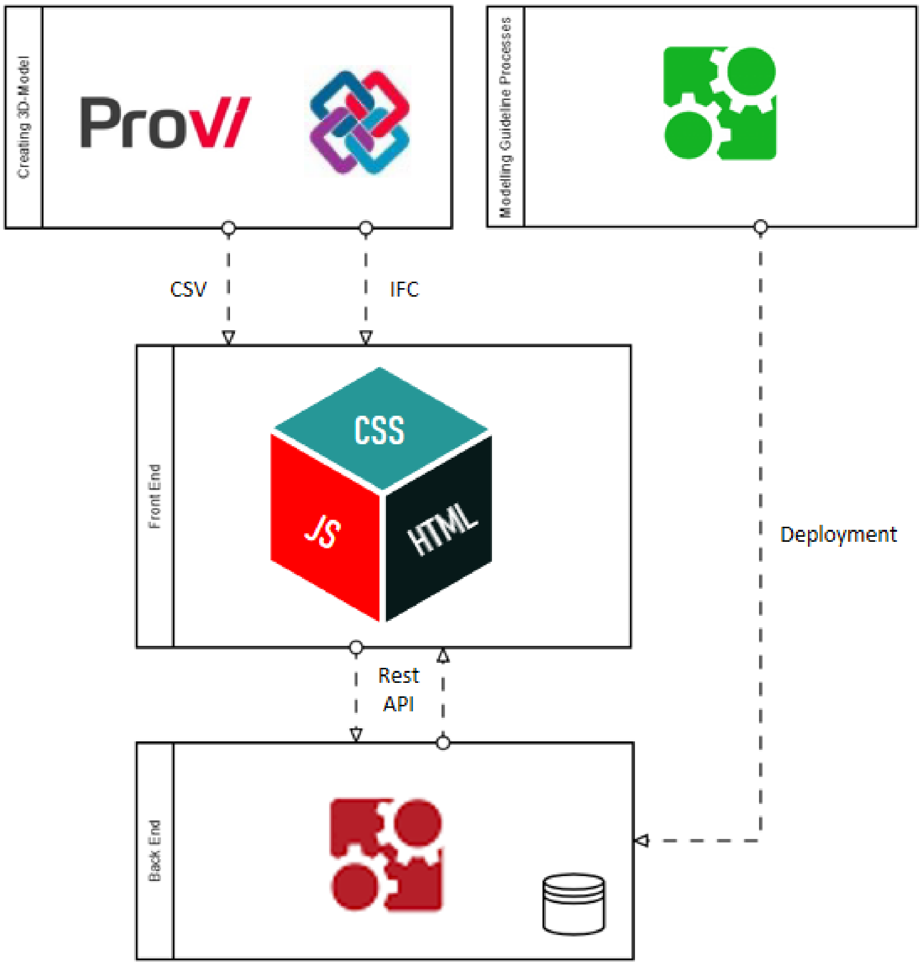
Various software products were used to develop and validate the case studies detailed in according to BPM notation (see also Figure 6).

To automate the process of checking the compliance of model data with guidelines, both models and guidelines need to be available in digital form. ProVI [99] software was used to generate the models for checking. For the purpose of alignment, the model data was transferred using IFC. Model data that cannot be represented in IFC schema version 4x1 is transferred in Comma-Separated Value (CSV) format, implementing a table structure listing stations (rows) and the corresponding parameters of the model (columns). For example, with a railway superstructure, the exported CSV will include a list of rail types, sleeper types and distances between sleepers per station. The advantage of the CSV format is that it is the most straightforward way of exporting the parameters of the implicit data model from ProVI. However, as this representation is not vendor-neutral, it is strongly recommended that the IFC schema is used as soon as it is extended to cover railway designs.

As discussed in the previous Sections, BPMN and DMN were used to implement automation. Camunda Modeler [100] was used to model the guideline contents in these notations. The translation and representation of the guidelines as BPMN and DMN were performed using the rules described in Section 4.2. In addition to Camunda Modeler, Camunda Community Platform was also used as a workflow engine. Together with the supplied database, they represent the back end.

The IFC and CSV data were evaluated using programs written in Javascript. A front end was developed to transfer the model data to the workflow engine using web technology (web page: HTML, CSS, Javascript) and display the results of the check accordingly. A Rest API was used to facilitate data exchange with the back end.

Figure 6: Software configuration (representation according to BPMN).



For the purpose of model preparation, in addition to the technical mapping of the rules and regulations in BPMNs, an information extractor was used to extract the required input data from an IFC model and make it available to the downstream checking process. Only IFC 4x1 models were used in the prototype extractor, since this version – unlike its successors – had already attained 'final standard' status. The

IfcInfra proposals collected in IFC4x3, by comparison, are tested in a validation phase before granted as the next “Final Standard” version of IFC. As a result, the example models used do not contain all the required parameters available from the proprietary database of the BIM route modeling tool ProVI.

6.2 Case study: Alignment

Given the importance of alignment in the planning of infrastructure facilities, the following section describes the process of checking alignment elements for their compliance with the guidelines. The checks relate to Rule Class 6 ‘Definitions for alignment elements’. Figure 7 shows the elements that occur along an alignment: straight lines (black), transition arcs (red), and circular arcs (green).

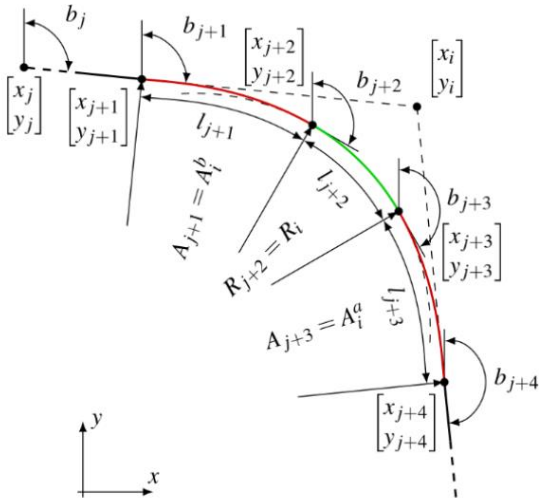


Figure 7: Different types of segments along a horizontal alignment: straight elements in black, transition curves in red, circular arc in green [91]**.**

The description of a track or route alignment (three-dimensional space curve) with IFC 4x1 and used for verifying the here presented approach. Table 7 provides an overview of the parameters that are either available or can be derived in the authoring software or in the IFC model. It shows only those parameters that are relevant for the subsequent analysis. Up to version 4x1, no specifications for superelevation and velocity exist in the IFC schema. The importance of this information is apparent in the checking routines described below.

Table 7: Comparison of available information in the ProVI authoring software and the neutral data format IFC 4x1. While all information in ProVI is explicitly available, some information in IFC can only be derived through additional calculation. No information on superelevation or velocity is defined in the data schema.

Parameter / characteristic value	ProVI	IFC 4x1 – explicitly defined	IFC 4x1 – implicitly derivable
Entry station of an element	✓		✓

Element type, with exact subtype for transition curves	✓	✓	
Element length	✓	✓	
Radius for circular arcs	✓	✓	
Superelevation	✓	✗	✗
Superelevation deficit	✓	✗	✗
Velocity	✓	✗	only if velocity is the same for all segments

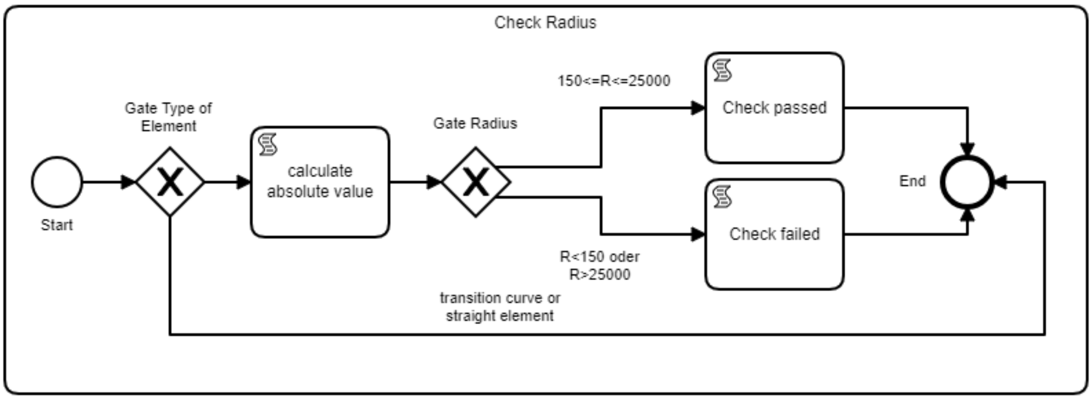
Guideline 800.0110 specifies that the radius of a circular arc should be in the range 150 m to 25,000 m [101]. **Figure** 8 presents the logic employed for checking arc radii according to Guideline 800.0110. Since an alignment can consist of different element types (see Figure 7), the first step (1) is to verify that the element to be checked is an arc. A BPMN exclusive gateway is used for this purpose. If the element to be checked is a straight line or a transition arc, the process is terminated immediately. The directional description of the radii is specified in the alignment modeling software as positive (right-curved) or negative (left-curved). For this reason, the second step (2) calculates the absolute value of the radius to be checked, using a script task containing the following code:

Code 1: Calculating the absolute value of an arc radius in Groovy notation [102]**. The code is integrated into the process as a script task.**

$$Radius = Math.abs(Radius)$$

Now it is possible to check whether the radius is within the range 150 m to 25,000 m, as defined in the guideline. Here, too, an exclusive gateway is used, returning ‘check passed’ when true, or ‘check failed’ when false.

Figure 8: Process flow: Check the radius of an arc element of an alignment, represented as a BPMN process.



According to Guideline 800.0110 § 6 No. 2, the regular and minimum length of track curves and straight lines must be determined according to the details given in Table 8. Different formulae for determining the minimum length of straight lines and circular arcs apply, depending on the velocity. For the regular length, a single formula not subdivided into different velocity groups applies.

Table 8: Planning values for the lengths of track curves and straight lines according to [101]**, grouped by velocity classes, showing formulae for calculating minimum length and regular length (with no differentiation by velocity).**

Minimum length	
$v \leq 70$ km/h	$l_{min} \geq 0.10 \times v$ [m]
$70 \leq v \leq 100$ km/h	$l_{min} \geq 0.15 \times v$ [m]
$v > 100$ km/h	$l_{min} \geq 0.20 \times v$ [m]
Regular length	
$l_{reg} \geq 0.40 \times v$ [m]	

The checking process is shown in Figure 9. As in the process described above, the first step is to determine whether the element is a straight line or an arc (Gateway 1). If the element to be checked is a transition arc, the checking process is terminated without any further steps. If the element is a straight line or an arc, the next process module – a parallel gateway – is executed. This enables the process to be split into two or more paths, so that both the regular length and the minimum length can be determined within a process run according to the formulae specified in Table 8. An exclusive gateway is used to apply the respective formula to the corresponding velocity. The calculation of the minimum or control length is carried out using different script tasks. The calculated target value is then compared against the model value using an exclusive gateway in which the status of the 'regular length' is set to 'check passed' if the condition

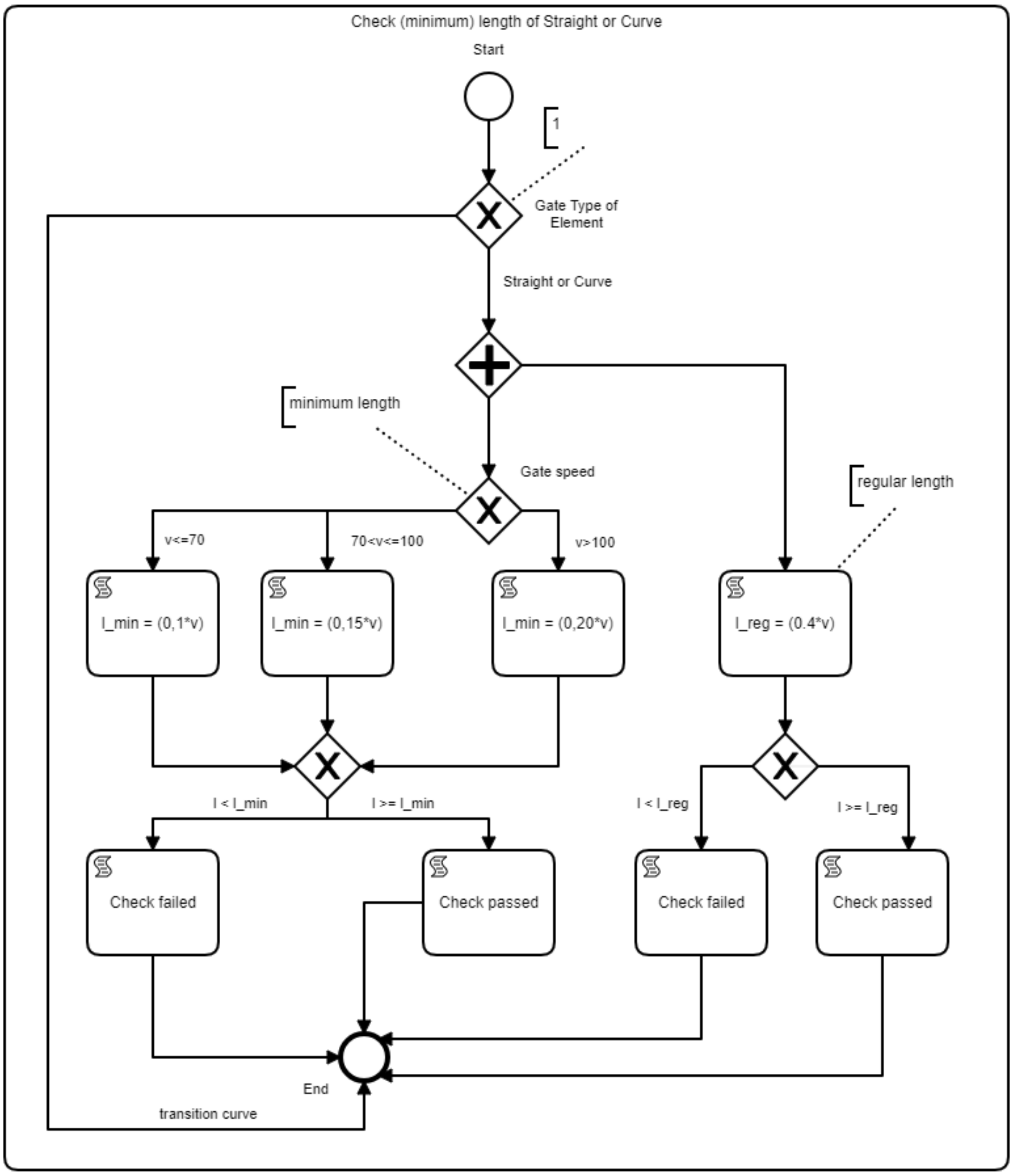
$$l_{Model} \geq l_{regular}$$

is fulfilled. The status of the 'minimum length' check is set to 'check passed' if the following condition applies:

$$l_{Model} \geq l_{min}$$

Otherwise, the status of the checks is set to 'check failed'.

Figure 9: Process: Check the regular and minimum lengths of straight and curved arc elements of an alignment represented as a BPMN process.



In addition to the minimum lengths of straight lines and curved arcs, the guideline also describes the minimum length of transition arcs. The guideline describes three different types of transition curves: clothoid, Bloss-type, and s-shaped. The minimum length again depends on the velocity (v) and the difference of the superelevation deficits of two successive elements (Δu_f). In our case here, it is assumed that

$$\Delta u_f = u_f$$

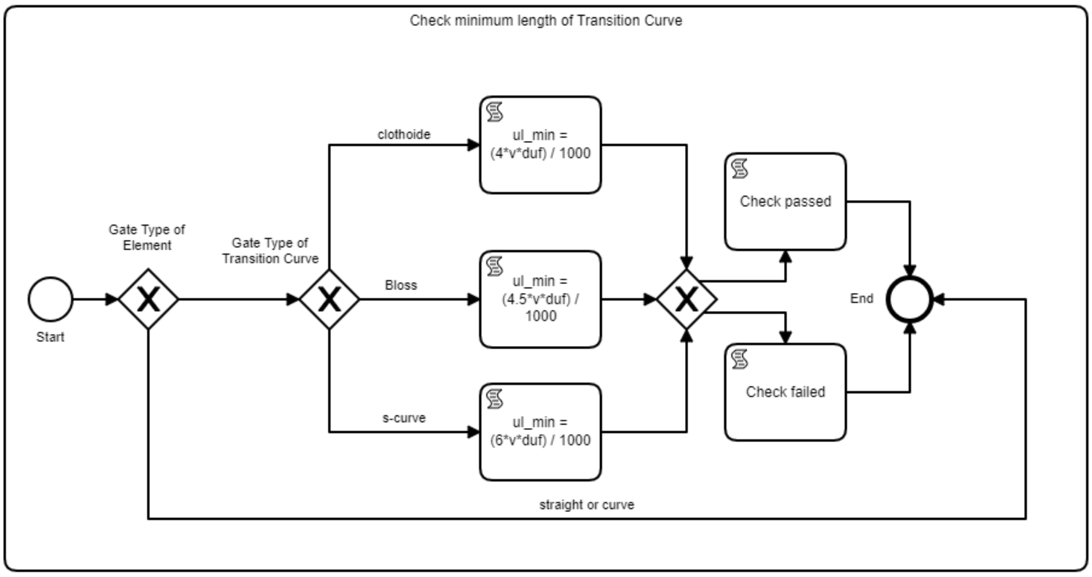
Table 9 shows the formulae needed to determine the minimum length of the transition curves according to [101].

Table 9: Minimum length of transition curves according to [101]**.**

	Transition curve	
Clothoid	Bloss-type	s-shaped
$\min l_U = \frac{4 \times v \times \Delta u_f}{1000}$	$\min l_{UB} = \frac{4{,}5 \times v \times \Delta u_f}{1000}$	$\min l_{US} = \frac{6 \times v \times \Delta u_f}{1000}$

The respective BPMN process is shown in Figure 10. Here, too, the first step is to determine whether the element type to be checked is a transition curve. If the check is negative, the process is terminated without any further steps. The second gateway splits the process based on the transition curve type (clothoid, Bloss, s-shaped) and passes the process to the respective script task, which then calculates the required minimum length using the relevant formula according to [101]. The third gateway then checks the result against the model and returns either 'check passed' or 'check failed'.

Figure 10: Process: Check the minimum length of transition curves represented as a BPMN process.



In line with the four model checking phases of Eastman et al [23], the time needed for checking data manually is compared to that of automated checking. The first phase describes the steps needed to interpret and represent the rules given by the regulations. For the manual working process, the auditor needs due training to be able to check designs correctly. Since it is a subjective process, it is not possible to measure the training time needed. The comparison therefore does not take this phase into consideration. The time required for the manual process in phases two to four is approximated. The time needed to create a process definition for the automated checking process depends on the level of difficulty. A simple rule, as shown in Figure 8, can be created in two days, including the time needed for researching the guideline, creating the BPMN/ DMN, and writing the model analyzer for extracting the parameters. An axis of a total length of 9.5 km is used as test data. In total, the axis contains 45 elements (arc, straight, transition curve) and 259 parameters that describe them. Table 10 presents the results of the measurements. The automated checking process takes 20 seconds (measured) to perform model preparation and rule execution. It is assumed that doing the same manually will take 20 min. Because the tool creates the report automatically, the results have to be validated and the report has to be sent to the designer. It is assumed that this phase will take 5 to 10 min. Following the manual method, the auditor also has to write a report, which takes approximately 15 min. In total, the ratio between manual and automated checking is approximately 24 %.

Table 10: Comparison of manual and automated checking and the time needed to check the axis test data

Phase	Description	Time needed for manual checking (approximation)	Time needed for automated checking	Ratio
1	Interpretation and logical representation of rules	-	-	
2	Building model preparation	5 min	10 sec	
3	Rule execution	10 min	10 sec	
4	Rule checking report	15 min	7 min	
	Total	**30 min**	**7 min 20 sec**	**24%**

In addition to the checking routines described above, further processes can be derived from the specifications of Guideline 800.0110. These include:

- Determining and checking the exact radius according to §6 No. 3
- Determining and checking necessary track extensions according to §6 No. 5
- Checking track curve radii in platform areas according to §6 No. 7
- Determining and checking design values for superelevation and superelevation deficit according to §7
- Determining and checking design values for superelevation ramps according to §8
- Determining and checking longitudinal inclination and changes in inclination in accordance with §10, including the design values for fillet radii and intermediate straight lines

The list shows that there is much greater potential for automating the checking of alignment elements than just the processes shown above. There are many more possibilities for modeling checking processes in the context of alignment elements.

6.3 Case study: Superstructure

Guideline 820 (“Basics of Superstructure”) summarizes the stipulations of general and overarching importance that govern the design, construction and maintenance of the railway superstructure [103]. This guideline is for use “within the area of DB Netz AG” for all railway superstructures up to a permissible velocity of v = 300 km/h” [103]. The following examples refer specifically to Guideline 820.2010 “Standard specifications for ballasted track for tracks and switches”. These serve as a basis for describing and developing checking processes for the rule classes ‘Component definitions’ and ‘Distance definitions: Height’.

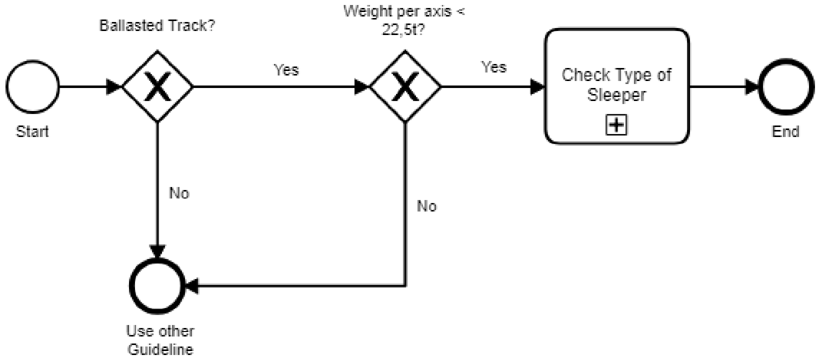
Guideline 820.2010 “governs the application areas of superstructure components in tracks and switches as well as the dimensions of the ballast bed cross-section according to technical/economic aspects. In the following, the term ‘superstructure components’ is understood to comprise rails, sleepers, rail fastenings, sub-ballast mats, insulated joints, ballast and switch components” [103]. Sub-Clause 1 Sentence 3 stipulates that the guideline applies “in the area of DB Netz AG in tracks and switches travelled on with wheelset loads of up to 22.5 t” [103].

As such, the area of application of the guideline is clearly defined and the applicability of the guideline can be summarized as follows:

(1) Only valid for ballasted track
(2) Only valid for wheelset loads up to 22.5 ton
(3) Only valid for the superstructure components defined above

The corresponding process logic checks conditions 1 and 2 at the beginning of each run (see Figure 11) using two successive exclusive gateways: the first queries whether the model is a “ballasted track” and the second whether the wheelset load is less than 22.5 t. In the current process, the user enters this data, but if the data is stored in the model, it can also be used for the decision. If the conditions are met, the system continues with the sub-process of the respective superstructure components. In Figure 11, the sub-process is shown in a ‘collapsed’ state but will be explained in more detail shortly.

Figure 11: Basic data check to ascertain applicability of the guideline: Does the model data meet the requirements of ballasted track and axle load < 22.5 t (currently inputted manually by the user)?



The following parameters and characteristic values are available in the modeling software for station-wise evaluation:

- Station
- Rail form
- Sleeper type
- Sleeper spacing
- Bedding thickness

- Ballast shoulder

The definition of the superstructure can be refined in the modeling software using additional parameters and definitions, opening up further possibilities for automated checking (that are not described here in detail).

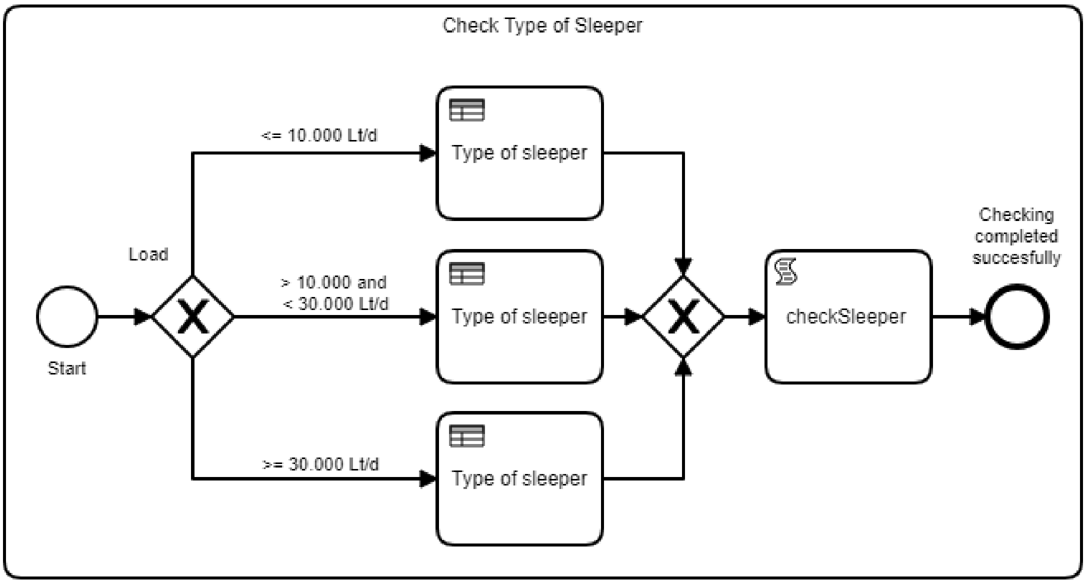
The guideline defines permissible superstructure components according to different parameters. “The superstructure components are determined according to the track load per day (Lt/d) and the maximum speed according to the List of Permissible Speeds (Hg VzG) or the local permissible speeds […]” [103].

The guideline is supplemented by various appendices that define the superstructure components and the respective dependencies. The process checks the sleeper type according to the logic of the guideline (see Figure 12) using an exclusive gateway that selects the respective appendix of the guideline depending on the track load. There are three categories for track loading:

(1) ≤ 10.000 Lt/d
(2) > 10,000 and < 30,000 Lt/d
(3) ≥ 30.000 Lt/d

The respective guideline appendix is integrated in the review process as a decision table in the “Decision Model and Notation” (DMN) standard. Table 11 shows the content of this decision table in the original format.

Figure 12: Process of checking the sleeper type represented as a BPMN process.



The example shown in Table 11 is an extract of the contents of the decision table for the selection of sleepers at a daily track load of >= 30,000 Lt/d. A relevant input variable for the choice of permissible sleeper type is speed. According to the table, three sleeper types, B70W, B70W-2,4 and B90W, are available for speeds below 160 km/h. For speeds in excess of 230 km/h, only sleepers of type B07W are permissible. Each sleeper type has a corresponding rail fastening and a permissible condition. In the example, only new material is permissible. Finally, the ‘annotation’ column contains notes on the use of the materials and any restrictions in use.

As the table shows, there are situations where several outcomes are possible, i.e. several sleeper types are permissible for the same speed. If there are no further restrictions, or none can be applied, all the applicable results are equally valid. In the context of the model check described here, a model already exists in which a corresponding configuration of superstructure components has been defined. Before the process is completed, a script task compares the sleeper type of the model with the possible options in the decision table, and the sleeper type in the model is checked to see if it matches one of the permissible sleeper types in the decision table. If the check is positive, the check status is set to ‘check passed’, and if it is negative, to ‘check failed’ (see Code 2).

Table 11: Decision table for sleeper types at a track load of ≥ 30,000 Lt/ d according to [103] **(extract).**

Input	**Output**			
Pace	**Type of sleeper**	**Type of rail fastening**	**Condition**	**Annotation**
< 160	B 70W	W14K 686a/687a	new	
< 160	B 70W-2,4	W14K 686a/687a	new	Only in crowded areas (e.g. slim subgrade)
< 160	B 90W	W14K 686a/687a	new	Only at railroad crossings and connection areas
[160…230]	B 70W	W14K 900	new	If radius > 800 m
> 230	B 07W	W21K 1000	new	
≥ 160	B 07W	W21K 1000	new	Only at connection areas

Code 2: Pseudo-code for comparing the superstructure components with the results of the decision table.

```
For (var index=0; index < decisionTypeOfSleeper.length; index++)
{
        if (decisionTypeOfSleeper == modelTypeOfSleeper) {
                setVariable("statusCheckSleeper", "check passed");
                exit for-loop;
        } else {
                setVariable("statusCheckSleeper ","check failed");
        }
```

}

The process configuration shown in Figure 12 can be used for all the track components mentioned above. Alongside text values, a decision table can also hold numerical values for automatic evaluation. The decision table shown in Table 12 describes the guideline specification for the ballast thickness of a track with a load ≤ 10,000 Lt/d. Depending on speed and sleeper type, the specified ballast bed thickness is 0.20 m or 0.30 m for steel sleepers. At track loads ≤ 10,000 Lt/d, speeds > 120 km/h are not regulated. In addition, several input variables can be supplied to the decision table. The table also serves as an example of how rules in the class ‘Distance Definitions: height’ can be defined with the help of BPMN and DMN.

Table 12: Decision table for ballast bed thickness at a track load of ≤ 10,000 Lt/d according to [103]**. The main decision criterion is sleeper type. Speeds greater than 120 km/h are not regulated.**

Input		**Output**	
Pace [km/h]	**Type of sleeper**	**Thickness of ballast** [m]	**Annotation**
≤ 120	not("steel sleeper")	0.20	
≤ 120	"steel sleeper"	0.30	

It takes 1-2 days to create a process as shown in Figure 12. A configuration of a superstructure with a total length of 15.1 km is used as test data. In total, the dataset contains 49 stations with 250 parameters that describe them. Table 13 presents the results of the measurements. The automated checking process takes 40 seconds (measured) for model preparation and rule execution. It is assumed that doing the same manually will take 25 min. The manual process takes approximately 40 min in total, compared with 7 min 40 sec for the automated checking process, which equals a ratio of 19 %.

Table 13: Comparison of manual and automated checking and the time needed to check superstructure test data

Phase	**Description**	**Time needed for manual checking (approximation)**	**Time needed for automated checking**	**Ratio**
1	Interpretation and logical representation of rules	-	-	
2	Building model preparation	10 min	20 sec	
3	Rule execution	15 min	20 sec	
4	Rule checking report	15 min	7 min	
	Total	**40 min**	**7 min 40 sec**	**19%**

In addition to the checking routines described here, other processes based on Guideline 820.2010 can be developed to check the compliance of track components, for example:

- Check rail type (rule class ‘Component definitions’)
- Distance check for ballast shoulder (rule class ‘Distance definitions: transverse direction’)
- Distance check for sleeper spacing (rule class ‘Distance definitions: position’)

6.4 Case study: Distance between tracks

The second most common rule class is ‘Distance definitions: transverse direction’. The case study presented here concerns a check of the distance between the track centers of double-track railway lines according to Guideline 800.0130, Appendix 02.

The following parameters and characteristic values are available in the modeling software for station-wise evaluation:

- Station per track
- Speed per track
- Radius per track
- Vertical distance between track centers

The process can be modeled using the same basic BPMN nodes as in the case study discussed above. Decision tables are likewise integrated into the process.

The guideline distinguishes between different areas and different structural and operational conditions when defining track spacing. Consequently, several decisions have to be taken to ascertain the respective situation and allow the model to be checked correctly. These are as follows:

(1) Is the radius of the alignment element greater than 250 m?
(2) Is the model of an open section of track or a station area?
(3) Is the open section equipped for bi-directional line operation?
(4) Are catenary masts needed between the tracks?
(5) Is an inspection walkway needed between the tracks?
(6) Is the track a new line or an upgrade of an existing line, and are the routes regional or long-distance?
(7) Are these main, secondary or passing tracks?

Irrespective of conditions 2 to 7, the question of whether the radius of the underlying elements at the station is greater than 250 m needs to be clarified. According to the guideline, this is the first exclusive gateway in the process (see Figure 13).

If the radius of the line element is greater than 250 m, the next step (Gateway 2) is to check whether catenary masts needs to be provided between the two tracks. If no catenary masts are required, then an inspection walkway is needed for the safe passage of railway personnel in the track area (Gateway 3). If the answer to this question is ‘No’, the required track spacing (without catenary masts or inspection walkway) shall be determined by means of a decision table. At Gateway 5, the track spacing in the model is checked against the result of the checking process and, if met, the status is set to ‘check passed’, otherwise to ‘check failed’.

If the radius of the alignment element is less than 250 m, the checking process is diverted to the ‘No’ branch at Gateway 1. In such cases, Guideline 800.0130 stipulates that the track spacing must be increased as specified in **Table 14**. The required increase in track spacing for different radii is given in the table, and the guidelines specify that intermediate values should be interpolated linearly. In the checking process, this table is modeled as a sequence of exclusive gateways, in combination with script tasks. As the degree of spacing enlargement is not described by a uniform formula for all radius ranges in Table 14, the interpolation formulae for each radius range are anchored in the process. The calculation only concerns the necessary delta of enlargement. The decision path for the different radius ranges cannot be defined as a decision table, because the DMN engine used does not permit the use of formulae as values. Radii of less than 100 m are not covered by the guideline, and in such cases, the checking process is aborted.

Alongside Gateway 3, Gateway 4 also represents a decision node that directs the process either to the decision tables “Track spacing with intermediate inspection walkway” or “Track spacing without intermediate inspection walkway”. The determined necessary enlargement is added to the result of the respective decision table by means of a script task. Gateway 5 then compares the model value with the criteria of the guideline and evaluates the result.

Figure 13: Process: Checking track spacing, represented as a BPMN process.

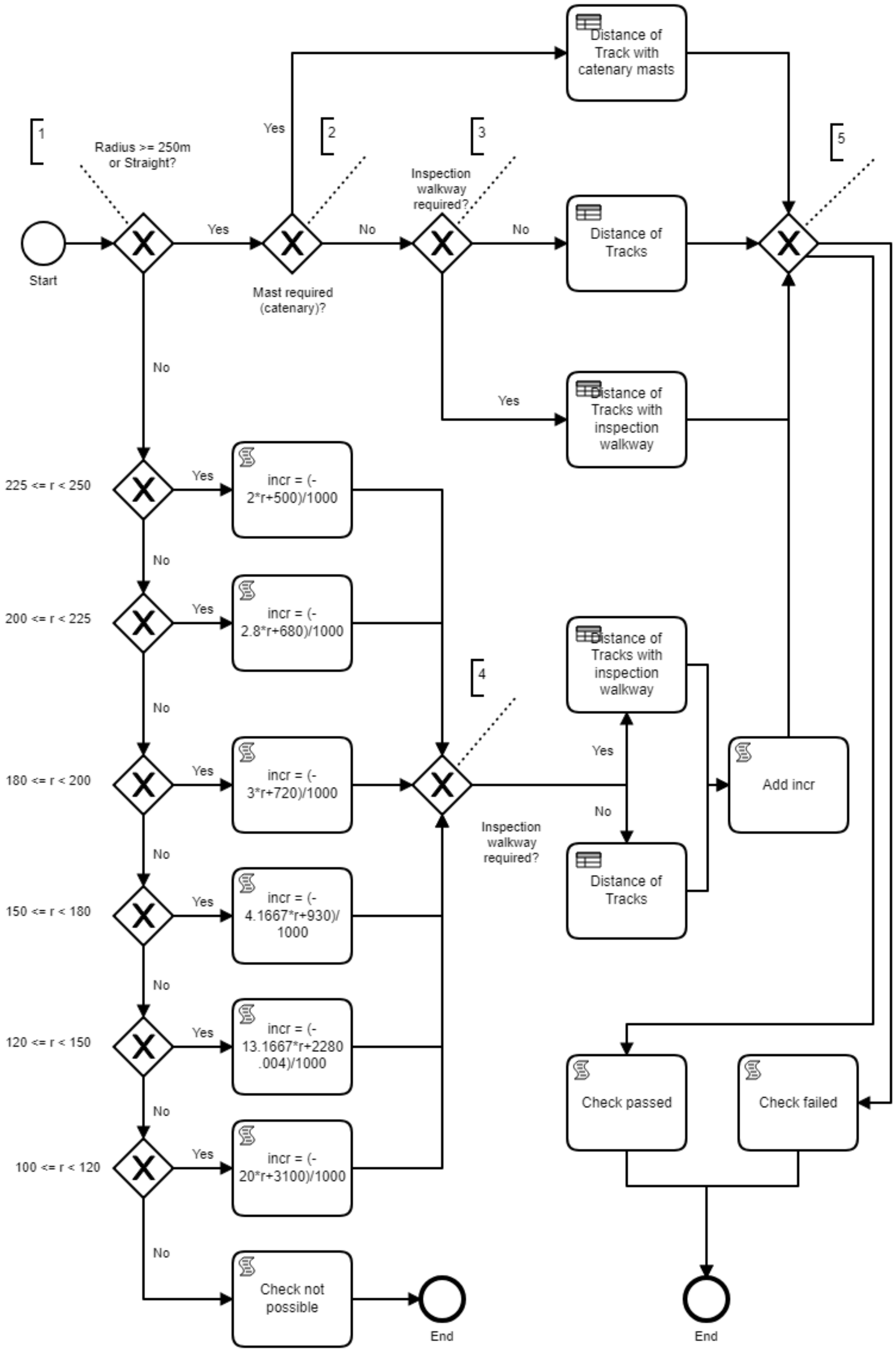


Table 14: Required increase in track distance according to the radius pursuant to [98]**. Intermediate values should be interpolated according to the guideline.**

Radius r [m]	Required increase [mm]
250	0

225	50
200	120
180	180
150	300
120	700
100	1000

Table 15 presents the decision table for the distance between tracks without an intermediate inspection walkway or catenary masts. Several input variables are needed, as determined by the questions above. Both the line itself (‘open section’ or ‘station area’) and the two tracks to be checked must be defined. Furthermore, the guideline defines speed specifications for the respective categories. For example, a track spacing of 4.5 m must always be maintained within station areas. Along upgraded tracks, at least 4.0 m is required in the speed range 0 to 200 km/h. “When upgrading existing tracks, a track spacing of 4.00 m can be maintained up to a design speed $_{\text{of ve}}$ = 230 km/h, if this faster speed is restricted to trains with favorable aerodynamic characteristics (such as the ICE)” [98]. Track spacings are also defined for new lines and local lines with suburban trains.

Table 15: Decision table for distances of tracks without intermediate walkway or catenary masts.

Input				**Output**	
Type of track	**Category Track 1**	**Category Track 2**	**Pace** [km/h]	**Distance of Tracks** [m]	**Annotation**
Outside stations	Upgraded track	Upgraded track	[0…200]	4.0	
Outside stations	New track	New track	[0…300]	4.5	
Outside stations	Short-distance track	Short-distance track	[0…120]	3.8	
Outside stations	Upgraded track	Upgraded track	[200…230]	4.0	Only for trains with aerodynamic characteristics
Inside stations				4.5	

In the case described, the track specifications are categorized from a construction point of view. Where definitions for track spacings for walkways are made, the categorization follows operational concerns. The terms main track, secondary track and passing track are used here. For this study, it has been assumed that this information is not contained in the model and must be supplied manually by the user. For the user, however, the problem arises that several descriptions are available, and it is not clear which categorization is used for which decision table. A user might assign the track data ‘correctly’ from his or her point of view – for example from a construction point of view – but the decision table requires operational categorization. In such a case, the check would not be carried out correctly. This is a limitation in the context of check and decision automation.

It takes 1-2 weeks to create a process such as the one shown in Figure 13. Two axes, each of a total length of 9.4 km, are used as test data. The dataset contains 946 distance values in total. Table 16 shows the results of the measurements. The automated checking process takes 40 seconds (measured) for model preparation and rule execution. It is estimated that performing the same checking process manually will take approximately 35 min. As a result, the manual process takes approximately 50 min in total, compared with 9 min 30 sec for the automated checking process, which equals a ratio of 19 %.

Table 16: Comparison of manual and automated checking and the time needed to check superstructure test data

Phase	**Description**	**Time needed for manual checking (approximation)**	**Time needed for automated checking**	**Ratio**
1	Interpretation and logical representation of rules	-	-	
2	Building model preparation	10 min	30 sec	
3	Rule execution	25 min	2 min	
4	Rule checking report	15 min	7 min	
	Total	**50 min**	**9 min 30 sec**	**19%**

7 The front end of the checking tool

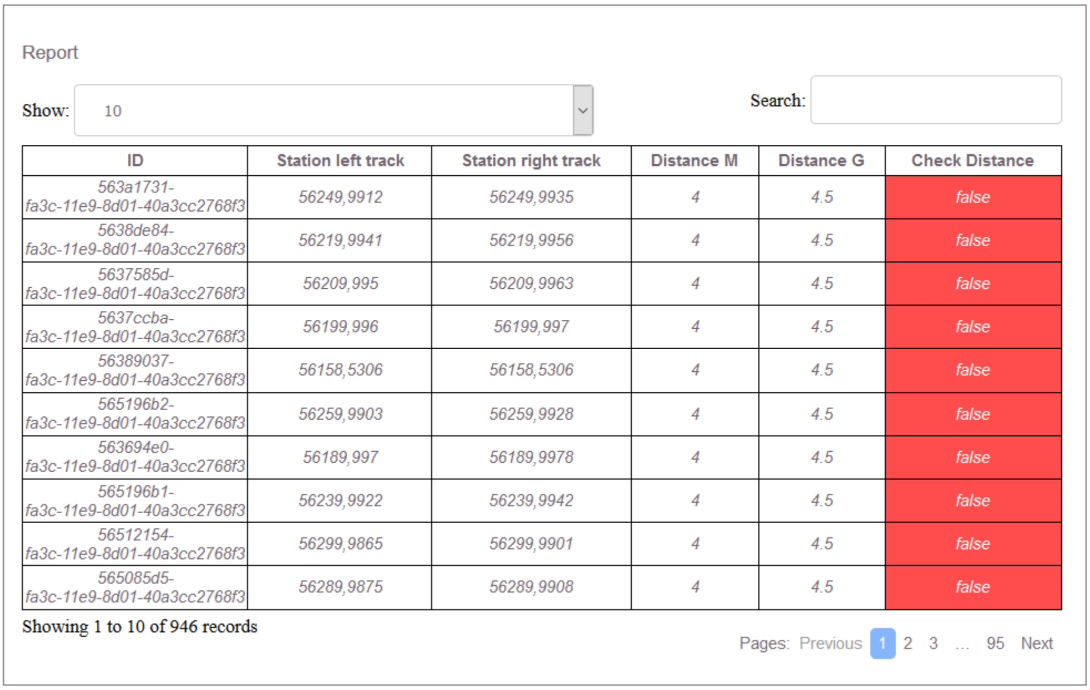
To make the checking system more user-friendly, a front end using current web technologies has been developed that echoes the four phases defined by Eastman in [23]:

(1) Interpretation and logical representation of rules
(2) Building model preparation
(3) Rule execution
(4) Rule check reporting

As explained in the preceding sections, the interpretation of the guidelines and their specifications and their translation into process diagrams were undertaken manually in DM and BPM Notation. The user can access the checking processes in the user interface using a BPMN viewer. The 3D model is transferred to the checking routine by the user. Additional information which is required but not available in the model can be supplied by the user via web input forms. The check is then performed automatically by the workflow engine and the results passed from the back end (workflow engine) to the front end via a Rest API.

The results are displayed in the form of a table showing each checked station and each checked parameter. Figure 14 shows an example report of the checking procedure for the distance between two tracks. The model contains a track distance of 4.0 m but the boundary conditions supplied by the user (as described in Section 6.4) meant that the guidelines specified a track distance of 4.5 m. The model check therefore returns a negative result throughout.

Figure 14: Example of a report showing the result of a compliance check for the required track distance. The distance in the model is 4.0m but the guidelines stipulate a distance of 4.5m. The report also shows the ID of the checking process and the station where the two tracks are located.



Report

Show: 10

Search:

ID	Station left track	Station right track	Distance M	Distance G	Check Distance
563a1731-fa3c-11e9-8d01-40a3cc2768f3	56249,9912	56249,9935	4	4.5	false
5638de84-fa3c-11e9-8d01-40a3cc2768f3	56219,9941	56219,9956	4	4.5	false
5637585d-fa3c-11e9-8d01-40a3cc2768f3	56209,995	56209,9963	4	4.5	false
5637ccba-fa3c-11e9-8d01-40a3cc2768f3	56199,996	56199,997	4	4.5	false
56389037-fa3c-11e9-8d01-40a3cc2768f3	56158,5306	56158,5306	4	4.5	false
565196b2-fa3c-11e9-8d01-40a3cc2768f3	56259,9903	56259,9928	4	4.5	false
563694e0-fa3c-11e9-8d01-40a3cc2768f3	56189,997	56189,9978	4	4.5	false
565196b1-fa3c-11e9-8d01-40a3cc2768f3	56239,9922	56239,9942	4	4.5	false
56512154-fa3c-11e9-8d01-40a3cc2768f3	56299,9865	56299,9901	4	4.5	false
565085d5-fa3c-11e9-8d01-40a3cc2768f3	56289,9875	56289,9908	4	4.5	false

Showing 1 to 10 of 946 records

The meaning of the modal auxiliary verbs (see Section 3) for the rules are simple to integrate into the workflow using a script task for the entire workflow that defines a corresponding variable. Where the weighting of a rule's importance of needs to be represented in the context of decision tables, a variable can be outputted. When displaying the results in the front end, the variable is then evaluated accordingly. For example, violations of rules and prohibitions can be marked in red, as these are non-permissible. Deviations from principles can be marked in yellow as these may still be permissible and do not necessarily impact negatively on the model quality. The quality of the model can therefore also be weighted in relation to these conditions.

8 Conclusion

In the construction industry, numerous standards such as norms and guidelines exist in written form that can be read by humans but not processed by machines. The checking of plans and models against such specifications is currently performed predominantly manually and is therefore a time-consuming and error-prone process. This particularly applies to the railway domain, where numerous regulations must be strictly followed to ensure the safety of rail traffic and passengers.

This study examined the extent to which the Business Process Model and Notation defined in ISO 19510 can be used to graphically represent guideline content and make them executable for model checking.

Although applying similar approaches to *Dimyadi et al.*, who focus on buildings [104], the focus of the study is on railway projects and their dedicated guidelines. In the railway domain, the BPMN approach particularly benefits from the precision and clarity of rules originally written in natural language. The BPMN approach was additionally supplemented by the Decision Model and Notation, which is also divergent from other research.

The study presents an analysis of selected Deutsche Bahn AG guidelines and the capability to represent and automate them with BPMN and DMN elements. The analysis showed that 52% of the rule sets examined can be automated, and 46 % of the examined rules were classified as being implementable using BPMN and DMN. By conducting a detailed classification, the rules were categorized into 12 different rule classes. By evaluating the frequency of occurrence of the individual classes, the top three rule classes were identified, and corresponding case studies designed and tested for feasibility. A further case study was also designed for elementary alignment planning within the infrastructure planning. To allow data to be imported from railway modeling software, reference was made to the current state of the IFC schema, and the use of IFC data validated.

Overall, the representation of guideline content using BPMN and DMN is a promising approach. Aside from fulfilling the designated goal, the graphical representation of the process makes the process transparent and enables the user to understand the checking process, which is a significant advantage over hard-coded ‘black box’ solutions. Due to the standardized representation of the process elements and the widespread use of the notation in the context of IDM, it is conceivable that this method could be widely implemented in the construction industry.

In this study, existing guidelines were manually translated into process diagrams to make them machine-readable. RASE-syntax was used to support the translation of regulation texts into workflow diagrams. Natural language processing methods offer a promising alternative. Future studies could aim to combine both methods, with a view to automatically translating guideline contents into process diagrams using NLP. It is also conceivable that authors could specify future regulations from the outset as BPMN processes, to facilitate the compliance checking of models. Currently there is no connection to a graphical 3D-representation such as a BIM-Viewer. The realized tool analyzes the given parametric of 3D-geometries, but does not calculate e.g. distances between 3D-objects (clearance test). These limitations could be interesting in future studies. Even if the visual representation of guidelines is more a ‘white-box’ than a ‘black-box’ solution, workflow-developers have to have experience in programming. BPMN is primarily a standard for modeling business processes. It is not meant to be a representation for engineering aspects and it is not specific for any engineering design software, which means, it is always necessary to develop import and analyzing functionalities. In this study, it is also shown that BPMN and DMN cannot be used if sophisticated software e.g. for structural analysis or simulation is necessary.

Although the current status of IFC in the field of infrastructure is not yet sufficient to carry out fully-fledged model checks in the railway domain, the recent supplementary studies in the context of IFC-Railway are promising and will likely provide all the information required to enable efficient checking of railway models against current regulations.

9 Reference literature

[1] ZukunftBAU, BIM-Leitfaden für Deutschland, (Guideline BIM for Germany), (2014) pp. 1–109. https://www.bbsr.bund.de/BBSR/DE/FP/ZB/Auftragsforschung/3Rahmenbedingungen/2013/BIMLeitfaden/Endbericht.pdf;jsessionid=12F63EC6FEA13FAC9753B52378D265D6.live11292?__blob=publicationFile&v=2 (accessed February 3, 2020).

[2] Bundesministerium für Verkehr und digitale Infrastruktur, Abschlussbericht der Reformkommission Großprojekte, (Final Report of the Reform Commission of Large-Scale Construction Projects). (2015) pp. 1–112. http://www.bmvi.de/SharedDocs/DE/Publikationen/G/reformkommission-bau-grossprojekte-endbericht.pdf?__blob=publicationFile (accessed June 20, 2019).

[3] M. Häußler, A. Borrmann, Model-based quality assurance in railway infrastructure planning, Automation in Construction. 109 (2020) pp. 1–15. doi:https://doi.org/10.1016/j.autcon.2019.102971.

[4] DIN Deutsches Institut für Normung e. V., DIN EN ISO 9000:2015-11, Quality management systems – Fundamentals and vocabulary, (2015) pp. 1–104.

[5] S. Macit İlal, H.M. Günaydın, Computer representation of building codes for automated compliance checking, Automation in Construction. 82 (2017) pp. 43–58. doi:10.1016/j.autcon.2017.06.018.

[6] J. Dimyadi, R. Amor, BIM-based Compliance Audit Requirements for Building Consent Processing, European Conferences on Product and Process Modeling in the Building Industry. (2018) pp. 1–6. https://www.researchgate.net/profile/Johannes_Dimyadi/publication/327384647_BIM-based_Compliance_Audit_Requirements_for_Building_Consent_Processing/links/5b99f2f392851c4ba81815c8/BIM-based-Compliance-Audit-Requirements-for-Building-Consent-Processing.pdf (accessed February 3, 2020).

[7] C. Sydora, E. Stroulia, Towards Rule-Based Model Checking of Building Information Models, 36th International Symposium on Automation and Robotics in Construction. (2019) pp. 1–7. https://pdfs.semanticscholar.org/c921/479a5341aac384b16ab3d2d5277f66acb2f8.pdf (accessed February 3, 2020).

[8] S.J. Fenves, J.H. Garrett, H. Kiliccote, K.H. Law, K.A. Reed, Computer Representations of Design Standards and Building Codes: U.S. Perspective, The International Journal of Construction Information Technology. 3 (1995) pp. 13–34. https://pdfs.semanticscholar.org/3b4e/d73499c6be2a8d07dfeacdc94bb381764c27.pdf (accessed February 3, 2020).

[9] L. Ding, R. Drogemuller, M. Rosenman, D. Marchant, Automating code checking for building designs - DesignCheck, Cooperative Research Centre (CRC) for Construction Innovation. (2006) pp. 1–16. doi:http://ro.uow.edu.au/engpapers/4842/.

[10] H. Kim, F. Grobler, Design Coordination in Building Information Modeling (BIM) Using Ontological Consistency Checking, in: Computing in Civil Engineering (2009), American Society of Civil Engineers, Reston, VA, (2009) pp. 410–420. doi:10.1061/41052(346)41.

[11] A. Yurchyshyna, A. Zarli, An ontology-based approach for formalisation and semantic organisation of conformance requirements in construction, Automation in Construction. 18 (2009) pp. 1084–1098. doi:10.1016/j.autcon.2009.07.008.

[12] J.K. Lee, Building environment rule and analysis (BERA) language and its application for evaluating building circulation and spatial program, (2011) pp. 1–217. https://smartech.gatech.edu/bitstream/handle/1853/39482/Lee_Jin-Kook_201105_PhD.pdf (accessed July 18, 2020).

[13] C. Preidel, A. Borrmann, Towards code compliance checking on the basis of a visual programming language, Journal of Information Technology in Construction. 21 (2016) pp. 402–421. http://www.itcon.org/2016/25 (accessed February 3, 2020).

[14] R. Navarro-Prieto, J.J. Cañas, Are visual programming languages better? The role of imagery in program comprehension, International Journal of Human Computer Studies. 54 (2001) pp. 799–829. doi:10.1006/ijhc.2000.0465.

[15] J.M. Rodriguez Corral, I. Ruiz-Rube, A. Civit Balcells, J.M. Mota-Macias, A. Morgado-Estevez, J.M. Dodero, A study on the suitability of visual languages for non-expert robot programmers, IEEE Access. 7 (2019) pp. 17535–17550. doi:10.1109/ACCESS.2019.2895913.

[16] J. Dimyadi, R. Amor, Automated Building Code Compliance Checking – Where is it at?, Proceedings of the 19th World Building Congress: Construction and Society. (2013) pp. 172–185. doi:10.13140/2.1.4920.4161.

[17] J.H. Garrett, M.E. Palmer, S. Demir, Delivering the Infrastructure for Digital Building Regulations, Journal of Computing in Civil Engineering. 28 (2014) pp. 167–169. doi:10.1061/(asce)cp.1943-5487.0000369.

[18] J. Dimyadi, R. Amor, Automating Conventional Compliance Audit Processes, J. Ríos et Al (Eds) Product Lifecycle Management and the Industry of the Future, Vol 517. Proceedings of the 14th IFIP WG 5.1 International Conference (PLM 2017), Seville, Spain, 10-12 July 2017. (2017) pp. 324–334. https://www.researchgate.net/profile/Johannes_Dimyadi/publication/321691957_Automating_Conventional_Compliance_Audit_Processes/links/5b263743aca272277fb5d260/Automating-Conventional-Compliance-Audit-Processes.pdf (accessed January 5, 2020).

[19] C. Preidel, A. Borrmann, Automated Code Compliance Checking Based on a Visual Language and Building Information Modeling, Proceedings of the 32nd International Symposium of Automation and Robotics in Construction, 15-18 June. (2015) pp. 256–263. doi:10.13140/RG.2.1.1542.2805.

[20] N.W. Young Jr., S. a Jones, H.M. Bernstein, Interoperability in the Construction Industry,

SmartMarket Report. (2007) pp. 1–36. https://vdcscorecard.stanford.edu/sites/g/files/sbiybj8856/f/mcgraw_hill_s_smartmarket_0.pdf (accessed December 10, 2019).

[21] J.P. Martins, B. Carvalho, V.A. Almeida, Automated rule-checking – a tool for design development, International Association for Housing Science - 41st World Congress on Housing - Sustainability and Innovation for the Future. (2016) pp. 1–8. https://core.ac.uk/download/pdf/143403202.pdf. (accessed January 25, 2019).

[22] W. Solihin, C. Eastman, Classification of rules for automated BIM rule checking development, Automation in Construction. 53 (2015) pp. 69–82. doi:10.1016/j.autcon.2015.03.003.

[23] C. Eastman, Jae-min Lee, Yeon-suk Jeong, Jin-kook Lee, Automatic rule-based checking of building designs, Automation in Construction. 18 (2009) pp. 1011–1033. doi:10.1016/j.autcon.2009.07.002.

[24] W. Solihin, J. Dimyadi, Y. Lee, C. Eastman, R. Amor, The Critical Role of Accessible Data for Bim- Based Automated Rule Checking Systems, LC3 2017 - Proceedings of the Joint Conference on Computing in Construction (JC3), July 4-7, 2017, Heraklion, Greece. I (2017) pp. 53–60. doi:https://doi.org/10.24928/JC3-2017/0161.

[25] H. Kim, J.K. Lee, J. Shin, J. Choi, Visual language approach to representing KBimCode-based Korea building code sentences for automated rule checking, Journal of Computational Design and Engineering. 6 (2019) pp. 143–148. doi:10.1016/j.jcde.2018.08.002.

[26] E. Hjelseth, N. Nisbet, Capturing Normative Constraints by Use of the Semantic Mark-up Rase Methodology, CIB W78-W102 2011: International Conference. (2011) pp. 26–28. https://itc.scix.net/pdfs/w78-2011-Paper-45.pdf (accessed July 15, 2020).

[27] T. Kasim, H. Li, Y. Rezgui, T. Beach, Automated Sustainability Compliance Checking Process: Proof of Concept, Proceedings of the 13th International Conference on Construction Applications of Virtual Reality (ConVR). (2013) pp. 11–21. https://itc.scix.net/pdfs/convr-2013-1.pdf (accessed July 15, 2020).

[28] N.M. El-Gohary, R. Zhang, A Machine Learning Approach for Compliance Checking-Specific Semantic Role Labeling of Building Code Sentences, in: T. Hartmann, I. Mutis (Eds.), Advances in Informatics and Computing in Civil and Construction Engineering, Springer Nature Switzerland AG, (2019) pp. 561–568. doi:https://doi.org/10.1007/978-3-030-00220-6.

[29] A. Charles, Automated rule checking for in-house BIM norms of building models, (2017) pp. 1–78. https://pure.tue.nl/ws/files/58778770/Ayyadurai_Charles_0923390.pdf (accessed February 17, 2019).

[30] X. Xu, H. Cai, Semantic approach to compliance checking of underground utilities, Automation in Construction. 109 (2020) pp. 1–21. doi:10.1016/j.autcon.2019.103006.

[31] N. Bus, A. Roxin, G. Picinbono, M. Fahad, Towards French smart building code: Compliance Checking Based on Semantic Rules, Proceedings of the 6th Linked Data in Architecture and Construction Workshop Towards. (2018) pp. 6–15. http://ceur-ws.org/Vol-2159/01paper.pdf (accessed January 10, 2020).

[32] M. Fahad, N. Bus, Conformance checking of IFC models via semantic BIM reasoner, EG-ICE

2019 - Workshop on Intelligent Computing in Engineering. (2019) pp. 1–10. https://pdfs.semanticscholar.org/28ec/e3b8dc96379edb20d1b4ce19604a1f26b142.pdf (accessed January 10, 2020).

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

[34] J. Zhang, N.M. El-Gohary, Semantic-Based Logic Representation and Reasoning for Automated Regulatory Compliance Checking, Journal of Computing in Civil Engineering. (2016) pp. 1–42. doi:10.1061/(ASCE)CP.1943-5487.0000583.

[35] V. Getuli, S.M. Ventura, P. Capone, A.L.C. Ciribini, BIM-based Code Checking for Construction Health and Safety, in: Creative Construction Conference 2017, CCC 2017, 19-22 June 2017, Primosten, Croatia, The Author(s), (2017) pp. 454–461. doi:10.1016/j.proeng.2017.07.224.

[36] K. Kincelova, C. Boton, P. Blanchet, C. Dagenais, BIM-based code compliance checking for fire safety in timber buildings: A comparison of existing tools, CSCE Annual Conference, 12-15 June 2019, Laval, Greater Montreal. (2019) pp. 1–10. https://www.semanticscholar.org/paper/BIM-BASED-CODE-COMPLIANCE-CHECKING-FOR-FIRE-SAFETY-Kincelova-Boton/6c610b35fdd4746999bfba5710b51fa044dc2ceb (accessed January 11, 2020).

[37] P. Patlakas, A. Livingstone, R. Hairstans, G. Neighbour, Automatic code compliance with multi-dimensional data fitting in a BIM context, Advanced Engineering Informatics. 38 (2018) pp. 216–231. doi:10.1016/j.aei.2018.07.002.

[38] B.A. Myers, Taxonomies of visual programming and program visualization, Journal of Visual Languages and Computing. 1 (1990) pp. 97–123. doi:10.1016/S1045-926X(05)80036-9.

[39] T.R.G. Green, M. Petre, Usability analysis of visual programming environments: A “cognitive dimensions” framework, Journal of Visual Languages and Computing. 7 (1996) pp. 131–174. doi:10.1006/jvlc.1996.0009.

[40] C. Preidel, S. Daum, A. Borrmann, Data retrieval from building information models based on visual programming, Visualization in Engineering. 5 (2017) pp. 1–14. doi:10.1186/s40327-017-0055-0.

[41] T. Catarci, G. Santucci, Are visual query languages easier to use than traditional ones?: an experimental proof, Proceedings of the HCI’95 Conference on People and Computers X. (1995) pp. 323–338. https://pdfs.semanticscholar.org/815c/4a0a2fc011c91e3351fa2ceef675a176cc8e.pdf (accessed October 29, 2019).

[42] S. Daum, A. Borrmann, Checking Spatio-Semantic Consistency of Building Information Models By Means of a Query Language, Proceedings of the International Conference on Construction Applications of Virtual Reality. (2013) pp. 1–9. https://publications.cms.bgu.tum.de/2013_daum_convr.pdf (accessed February 15, 2019).

[43] C. Preidel, A. Borrmann, Refinement of the visual code checking language for an automated

checking of building information models regarding applicable regulations, 2017 ASCE International Workshop on Computing in Civil Engineering, IWCCE 2017. (2017) pp. 157–165. doi:10.1061/9780784480823.

[44] C. Preidel, A. Borrmann, Integrating Relational Algebra into a Visual Code Checking Language for Information Retrieval from Building Information Models, in: 16th International Conference on Computing in Civil and Building Engineering, Osaka, Japan, (2015) pp. 1–8. doi:10.13140/RG.2.1.4618.5201.

[45] P. Ghannad, Y.C. Lee, J. Dimyadi, W. Solihin, Automated BIM data validation integrating open-standard schema with visual programming language, Advanced Engineering Informatics. 40 (2019) pp. 14–28. doi:10.1016/j.aei.2019.01.006.

[46] F. Ritter, C. Preidel, D. Singer, Visuelle Programmiersprachen im Bauwesen - Stand der Technik und aktuelle Entwicklungen, (Visual Programming Languages in Construction Domain - State of the Art and Current Evolution). (2015) pp. 1–9. https://publications.cms.bgu.tum.de/2015_Ritter_FBI.pdf (accessed October 29, 2019).

[47] T. Bloch, M. Katz, R. Yosef, R. Sacks, Automated model checking for topologically complex code requirements – security room case study, Proceedings of the 2019 European Conference on Computing in Construction. 1 (2019) pp. 48–55. doi:10.35490/ec3.2019.157.

[48] buildingSMART, IFC Rail, (2020) p. https://www.buildingsmart.org/standards/rooms/railway/ifc-rail-project/ (accessed July 19, 2020).

[49] W. Solihin, C. Eastman, Y.C. Lee, Toward robust and quantifiable automated IFC quality validation, Advanced Engineering Informatics. 29 (2015) pp. 739–756. doi:10.1016/j.aei.2015.07.006.

[50] C. Zhang, J. Beetz, M. Weise, Model View Checking: Automated validation for IFC building models, 1 (2014) pp. 1–15. doi:10.1201/b17396-24.

[51] K. Kluza, K. Honkisz, From SBVR to BPMN and DMN Models. Proposal of Translation from Rules to Process and Decision Models, in: L. Rutkowski, M. Korytkowski, R. Scherer, R. Tadeusiewicz, L. Zadeh, J. Zurada (Eds) Artificial Intelligence and Soft Computing. ICAISC 2016. Lecture Notes in Computer Science, Springer, Cham, (2016) pp. 453–462. doi:10.1007/978-3-319-39384-1_39.

[52] ISO/IEC 19510:2013 - Information technology — Object Management Group Business Process Model and Notation, 2013 (2013) pp. 1–534. https://www.omg.org/spec/BPMN/ISO/19510/PDF (accessed January 20, 2020).

[53] J. Dimyadi, C. Clifton, M. Spearpoint, R. Amor, Regulatory Knowledge Encoding Guidelines for Automated Compliance Audit of Building Engineering Design, in: 2014 International Conference on Computing in Civil and Building Engineering, (2014) pp. 1–8. doi:10.1061/9780784413616.067.

[54] J. Dimyadi, C. Clifton, M. Spearpoint, R. Amor, Computerizing Regulatory Knowledge for Building Engineering Design, Journal of Computing in Civil Engineering. 30 (2016) pp. 1–13. doi:10.1061/(asce)cp.1943-5487.0000572.

[55] J. Recker, M. Indulska, M. Rosemann, P. Green, How good is BPMN really? Insights from Theory and Practice, 14th European Conference on Information Systems. (2006) pp. 1–12. https://aisel.aisnet.org/ecis2006/135/?utm_source=aisel.aisnet.org%2Fecis2006%2F135&utm_medium=PDF&utm_campaign=PDFCoverPages (accessed December 30, 2019).

[56] L. Janssens, E. Bazhenova, J. De Smedt, J. Vanthienen, M. Denecker, Consistent integration of decision (DMN) and process (BPMN) models, in: S. Espana, M. Ivanovic, M. Savic (Eds.): Proceedings of the CAiSE'16 Forum at the 28th International Conference on Advanced Information Systems Engineering, Ljubljana, Slovenia, 13-17 June 2016, (2016) pp. 121–128. https://pdfs.semanticscholar.org/2e68/54e24050a62d58657aeaea97620e7caead3d.pdf (accessed October 10, 2019).

[57] S.V. Aram, C. Eastman, Introducing a new methodology to develop the Information Delivery Manual for AEC projects, Proceedings of the CIB W78 2010: 27th International Conference – Cairo, Egypt. (2010) pp. 1–10. http://dcom.arch.gatech.edu/pcibim/documents/w78-2010-49_IDM.pdf (accessed January 5, 2020).

[58] E. Alreshidi, M. Mourshed, Y. Rezgui, Cloud-based BIM governance platform requirements and specifications: Software engineering approach using BPMN and UML, Journal of Computing in Civil Engineering. 30 (2016) pp. 1–41. https://ascelibrary.org/doi/full/10.1061/%28ASCE%29CP.1943-5487.0000539 (accessed January 8, 2020).

[59] ISO 29481-1:2016 - Building information models — Information delivery manual — Part 1: Methodology and format, (2016) pp. 1–40.

[60] M. Weise, T. Liebich, J. Wix, Integrating use case definitions for IFC developments, (2008) pp. 1–9. doi:10.1201/9780203883327.ch71.

[61] M. Obergriesser, A. Borrmann, Infrastructural BIM Standards - Development of an Information Delivery Manual for the geotechnical infrastructural design and analysis process, (2012) pp. 1–7. doi:10.1201/b12516-93.

[62] Y.H. Park, C.Y. Cho, G. Lee, Identifying a subset of BPMN for IDM development, Proceedings of the CIB W78-W102, International Conference, Sophia Antipolis, France, 26-28 October. (2011) pp. 1–6. https://www.semanticscholar.org/paper/IDENTIFYING-A-SUBSET-OF-BPMN-FOR-IDM-DEVELOPMENT-Lee/c7e6ec2a662007ac80601475d6917a776f27ae62 (accessed September 9, 2019).

[63] A. Borrmann, M. König, C. Koch, J. Beetz, 4.3 - Process modeling, in: Building Information Modeling - Technology Foundation and Industry Practice, (2017) pp. 1–597. ISBN:978-3-319-92862-3.

[64] A. Awad, M. Weske, Visualization of Compliance Violation in Business Process Models, in: International Conference on Business Process Management, (2009) pp. 182–193. doi:10.1007/978-3-642-12186-9.

[65] A. Ghose, G. Koliadis, Auditing Business Process Compliance, in: Proceedings of the International Conference on Service-Oriented Computing (ICSOC-2007), Lecture Notes in Computing Sicence, 4749, (2007) pp. 169–180. doi:10.1007/978-3-540-74974-5_14.

[66] A. Awad, G. Decker, M. Weske, Efficient Compliance Checking Using BPMN-Q and Efficient Compliance Checking Using BPMN-Q, in: M. Dumas, M. Reichert, MC. Shan (Eds) Business Process Management. BPM 2008. Lecture Notes in Computer Science, 5240, (2008) pp. 326–341. doi:https://doi.org/10.1007/978-3-540-85758-7_24.

[67] A. Awad, BPMN-Q : A Language to Query Business Processes, M. Reichert, S. Strecker & K. Turowski, Enterprise Modelling and Information Systems Architectures – Concepts and Applications. (2007) pp. 115–128. https://dl.gi.de/handle/20.500.12116/22195 (accessed November 11, 2019).

[68] J. Recker, Opportunities and constraints : The current struggle with BPMN, Business Process Management Journal. 16 (2010) pp. 181–201. doi:10.1108/14637151011018001.

[69] F. Kog, A. Dikbas, R.J. Scherer, Verification and validation approach of BPMN represented construction processes, Creative Construction Conference 2014 Verification. (2014) pp. 265–271. https://pdfs.semanticscholar.org/e065/e86f19d160702f3d4b38e651472753ce8b6e.pdf (accessed July 5, 2019).

[70] S. Zolfagharian, J. Irizarry, Current Trends in Construction Site Layout Planning, in: Construction Research Congress 2014, (2014) pp. 1723–1732. doi:10.1061/9780784413517.176.

[71] Object Management Group, Decision Model and Notation Version 1.3, (2019) pp. 1–261. https://www.omg.org/spec/DMN/1.3/PDF (accessed August 14, 2020).

[72] T. Biard, A. Le Mauff, M. Bigand, J.-P. Bourey, Separation of Decision Modeling from Business Process Modeling Using New “Decision Model and Notation” (DMN) for Automating Operational Decision-Making, in: L. Camarinha-Matos, F. Bénaben, W. Picard (Eds) Risks and Resilience of Collaborative Networks. PRO-VE 2015. IFIP Advances in Information and Communication Technology, (2015) pp. 489–496. doi:10.1007/978-3-319-24141-8_45.

[73] S.J. Fenves, E.H. Gaylord, S.K. Goel, Decision Table Formulation of the AISC Specification, (1969) pp. 1–178. https://www.researchgate.net/publication/39067052_Decision_Table_Formulation_of_The_1969_Aisc_Specification (accessed September 5, 2019).

[74] J. Huysmans, K. Dejaeger, C. Mues, J. Vanthienen, B. Baesens, An empirical evaluation of the comprehensibility of decision table, tree and rule based predictive models, Decision Support Systems. 51 (2011) pp. 141–154. doi:10.1016/j.dss.2010.12.003.

[75] G. Ge, China Railway BIM and Standards, (2015) pp. 1–34. https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwjGu4zm1vTqAhUSM-wKHTEBD_UQFjABegQIARAB&url=https%3A%2F%2Fsyncandshare.lrz.de%2Fopen%2FMktrQlVFTm1CTHBLS3lNcW5MQVdz%2FIFC%2520Rail%252BRoad%2F2015-12-19%2F3a_IfcRail_CRBIM_Slides_201 (accessed July 30, 2020).

[76] China Railway BIM Alliance, Railway BIM Data Standard, (2015) pp. 1–233. https://www.buildingsmart.org/wp-content/uploads/2017/09/bSI-SPEC-Rail.pdf (accessed July 30, 2020).

[77] G. Lee, Railway BIM 2030 roadmap of South Korea, (2018) pp. 1–31. http://big.yonsei.ac.kr/railbim/reports/RailBIM2030Roadmap_Full_Eng_Final.pdf (accessed July 30, 2020).

[78] DIGITAL SNCF, La méthode BIM, ou comment bâtir dans l'ère numérique (The BIM method, or how to build in the digital age), (2017) p. https://www.digital.sncf.com/actualites/la-methode-bim-ou-comment-batir-dans-lere-numerique (accessed July 29, 2020).

[79] Bauen digital Schweiz, 3rd Suisse BIM Congress - BIM Roadmap, (2018) pp. 1–51. https://bauen-digital.ch/assets/Downloads/de/BIM-Roadmap.pdf (accessed July 30, 2020).

[80] ITALFERR - Ferrovie dello Stato, Innovating to design the future, (2019) pp. 1–248. http://www.italferr.it/content/dam/italferr/expertise/innovazione/Innovating to design the future.pdf (accessed July 29, 2020).

[81] Trafikverket, BIM – the work approach of the future, (2019) p. https://www.trafikverket.se/en/startpage/projects/Road-construction-projects/the-stockholm-bypass/bim--the-work-approach-of-the-future/ (accessed July 30, 2020).

[82] Banedanmark, Launching an agreement for a common Nordic approach to BIM, (2018) p. https://uk.bane.dk/en/Supplier/News-for-suppliers/BIM (accessed July 30, 2020).

[83] Deutsche Bahn AG, BIM Methodik bei der DB Station & Service AG, (BIM Methodology at DB Station & Service AG). (2020) p. https://www1.deutschebahn.com/sus-infoplattform/start/Vorgaben-zur-Anwendung-der-BIM-Methodik (accessed March 22, 2020).

[84] A. Borrmann, M. König, C. Koch, J. Beetz, 2.2 - Solid modeling, in: Building Information Modeling - Technology Foundation and Industry Practice, (2017) pp. 1–597. ISBN:978-3-319-92862-3.

[85] A. Borrmann, T.H. Kolbe, A. Donaubauer, H. Steuer, J.R. Jubierre, M. Flurl, Multi-Scale Geometric-Semantic Modeling of Shield Tunnels for GIS and BIM Applications, Computer-Aided Civil and Infrastructure Engineering. 30 (2015) pp. 263–281. doi:10.1111/mice.12090.

[86] G. Lee, R. Sacks, C.M. Eastman, Specifying parametric building object behavior (BOB) for a building information modeling system, Automation in Construction. 15 (2006) pp. 758–776. doi:10.1016/j.autcon.2005.09.009.

[87] A. Borrmann, J. Amann, T. Chipman, J. Hyvärinen, T. Liebich, S. Muhic, L. Mol, J. Plume, P. Scarponcini, IFC Infra Overall Architecture Project Documentation and Guidelines, BuildingSMART. (2017) pp. 1–53. https://www.buildingsmart.org/wp-content/uploads/2017/07/08_bSI_OverallArchitecure_Guidelines_final.pdf (accessed March 8, 2020).

[88] M. Reifenhäuser, R. Klebermass, R. Mautz, BIM IFC-Alignment: Fachliche Anforderungen Rail der DB, ÖBB und SBB, (Technical Requirements Rail of DB, ÖBB, SBB). (2018) pp. 1–12. https://mediendienste.extranet.deutschebahn.com/DIBS/.

[89] A. Borrmann, S. Muhic, J. Hyvärinen, T. Chipman, S. Jaud, C. Castaing, C. Dumoulin, T. Liebich, L. Mol, The IFC-Bridge project – Extending the IFC standard to enable high-quality exchange of bridge information models, Proceedings of the 2019 European Conference for Computing in Construction. 1 (2019) pp. 377–386. doi:10.35490/EC3.2019.193.

[90] S. Esser, K. Aicher, IfcBridge Model Generation using Visual Programming, Forum Bauinformatik 2019. (2019) pp. 1–8. https://publications.cms.bgu.tum.de/2019_Esser_FBI.pdf (accessed March 5, 2020).

[91] S. Markic, J. Schlenger, I. Bratoev, Tangible Alignment Design, 30. Forum Bauinformatik, Weimar, Germany. (2018) pp. 1–8. https://publications.cms.bgu.tum.de/2018_Markic_FBI.pdf (accessed October 14, 2019).

[92] T. Chipman, T. Liebich, M. Weise, mvdXML specification 1.1, Specification of a standardized format to define and exchange Model View Definitions with Exchange Requirements and Validation Rules. By Model Support Group (MSG) of buildingSMART, BuildingSMART Malaysia. 1 (2016) pp. 1–49. http://buildingsmart-tech.org/mvd/XML/1.1 (accessed March 12, 2020).

[93] buildingSMART International, The UML Model Report of IFC Road Project, (2020) pp. 1–124. https://www.buildingsmart.org/wp-content/uploads/2020/06/IR-CS-WP2-UML_Model_Report_Part-5_.pdf (accessed August 14, 2020).

[94] DB Netz AG, Richtlinie 836 - Erdbauwerke und sonstige geotechnische Bauwerke planen, bauen und instand halten, (Guideline 836 - Design, Built and Operate Earthworks and Other Geotechnical Constructions). (2018) p. https://mediendienste.extranet.deutschebahn.com/DIBS/.

[95] DIN Deutsches Institut für Normung e. V., DIN 820-2:2004-10 Presentation of documents (ISO/IEC-Directives – Part 2, modified) - withdrawn, (2004) pp. 1–139.

[96] DIN Deutsches Institut für Normung e. V., DIN 820-2:2020-03 Presentation of documents (ISO/IEC-Directives – Part 2:2018, modified), (2020) pp. 1–232.

[97] S. Malsane, J. Matthews, S. Lockley, P.E.D. Love, D. Greenwood, Development of an object model for automated compliance checking, Automation in Construction. 49 (2015) pp. 51–58. doi:10.1016/j.autcon.2014.10.004.

[98] DB Netz AG, Richtlinie 800.0130 - Streckenquerschnitte auf Erdkörpern, (Guideline 800.0130 - Cross Sections at Earthwork). (2018) p. https://mediendienste.extranet.deutschebahn.com/DIBS/.

[99] ProVI GmbH, ProVI 6.1, (2019) p. https://www.provi-cad.de/ (accessed March 15, 2019).

[100] Camunda Services GmbH, Camunda, (2018) p. https://camunda.com/ (accessed October 23, 2019).

[101] DB Netz AG, Richtlinie 800.0110 - Linienführung, (Guideline 800.0110 - Alignment). (2015) p. https://mediendienste.extranet.deutschebahn.com/DIBS/.

[102] Apache Groovy project, Groovy, (2020) p. https://groovy-lang.org/ (accessed July 28, 2020).

[103] DB Netz AG, Richtlinie 820 - Grundlagen des Oberbaues, (Guideline 820 - Basics of Superstructure). (2013) p. https://mediendienste.extranet.deutschebahn.com/DIBS/.

[104] J. Dimyadi, R. Amor, Regulatory Knowledge Representation for Automated Compliance Audit of BIM-Based Models, (2013) pp. 9–12. https://www.researchgate.net/publication/257955455_REGULATORY_KNOWLEDGE_REPRESENTATION_FOR_AUTOMATED_COMPLIANCE_AUDIT_OF_BIM-

BASED_MODELS (accessed July 25, 2020).