

Article

Level of Naming Compliance: Syntactic, Referential, and LLM-Supported Semantic Validation of BIM/CDE File Names

Jędrzej Pasalski ^{1,*} , Tomasz Owerko ¹ , Karolina Tomaszekiewicz ¹  and Mateusz Kasznia ²

¹ Faculty of Geo-Data Science, Geodesy and Environmental Engineering, AGH University of Krakow, al. Mickiewicza 30, 30-059 Krakow, Poland; owerko@agh.edu.pl (T.O.); tomaszki@agh.edu.pl (K.T.)

² Independent Researcher, 30-059 Krakow, Poland; mkasznia95@gmail.com

* Correspondence: jpasalski@agh.edu.pl; Tel.: +48-694-164-576

Abstract

Inconsistent and erroneous file naming is a recurring information management problem in BIM and Common Data Environment (CDE) workflows, affecting document identification, retrieval, traceability, and control. This problem was observed during work on a real-world modular railway station project, where received documentation included files prepared under heterogeneous naming practices. This paper proposes the Level of Naming Compliance (LoNC), a multi-level framework for assessing file name compliance in BIM/CDE environments. The framework evaluates file names as structured information containers and combines three validation stages: syntactic validation using regular expressions, referential validation against project-specific lookup tables, and LLM-supported semantic validation comparing selected decoded file name fields with descriptive document titles. In the current implementation, semantic validation was limited to discipline and document type. The proposed approach was evaluated using a controlled synthetic validation dataset of 500 file-title pairs, developed from inconsistency types observed in the motivating project. The dataset contained predefined examples of syntactic errors, invalid reference values, semantic inconsistencies, and fully compliant cases. Within this controlled evaluation set, the validator showed full agreement with the predefined expected LoNC levels. Because the dataset contained an unequal distribution of compliance classes, the evaluation was supplemented with balanced accuracy and class-specific performance measures. The results indicate that the implemented workflow correctly operationalises the cumulative LoNC framework under the tested conditions. The findings demonstrate that LoNC provides a more diagnostic assessment than binary file name checking by identifying the stage and type of non-compliance. The approach can support CDE managers and document controllers in quality assurance processes, while future work should validate it on larger real-world datasets and extend semantic validation beyond the selected fields.



Academic Editors: Mohamed Yamany, Mohamed T. Elnabwy, Ahmed Abdelaty and Hosam Hegazy

Received: 26 April 2026

Revised: 4 July 2026

Accepted: 7 July 2026

Published: 20 July 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

Keywords: building information modelling; common data environment; file naming convention; level of naming compliance; document control; metadata validation; large language models; information management

1. Introduction

Compared with other sectors of the economy, particularly industrial production, the construction industry continues to face persistent efficiency challenges [1]. Building Information Modelling (BIM) has been promoted for many years as one of the main approaches for improving productivity, coordination, and information exchange in the architecture,

engineering, and construction (AEC) sector [2]. Although BIM is most commonly understood as Building Information Modelling, it is also increasingly associated with Building Information Management, emphasising the structured handling of information throughout the lifecycle of a built asset [3].

This management-oriented understanding of BIM is reflected in the growing number of standards, mandates, manuals, handbooks, and national guidelines developed to support information exchange and collaboration between project stakeholders [4]. Among these, ISO 19650 is one of the most widely used frameworks for information management in BIM-based projects [5,6]. It defines general principles for managing information over the lifecycle of a built asset, but it does not provide detailed project-specific instructions for every aspect of information production, classification, naming, and validation. As a result, practical implementations of ISO 19650 may differ significantly between projects, organisations, and Common Data Environment (CDE) platforms.

A key aspect of information management in BIM-based workflows is the consistent identification of information containers. According to ISO 19650, an information container may be understood as a named set of information that can be retrieved, exchanged, reviewed, revised, and approved within a defined information management process [5,6]. In practice, information containers are often represented by models, drawings, schedules, specifications, reports, spreadsheets, images, videos, and other project documents exchanged through a CDE. Their file names support information retrieval, document control, revision management, traceability, and the correct interpretation of project documentation. Previous industry observations indicate that information sharing, retrieval, and interpretation remain among the major challenges faced by engineers and project stakeholders [7].

CDE platforms are intended to reduce these problems by providing controlled environments for storing, exchanging, approving, and archiving project information [8,9]. Many commercial solutions are available, each offering different approaches to document management, metadata handling, workflow control, and information exchange [7,10]. The use of such platforms to support standardisation in a measurable way can contribute to efficiency gains in BIM-based projects [11]. However, the configuration and maintenance of naming rules, lookup tables, metadata structures, and approval workflows remain strongly project-specific. They often depend on the experience of CDE managers, BIM managers, document controllers, and representatives of the appointing and appointed parties. This limits the reuse of automation across projects and makes systematic file name quality control difficult to scale.

In this context, file naming should be treated not only as an administrative convention but also as a metadata quality problem. A BIM/CDE file name often encodes several pieces of project information, such as project number, location, volume, discipline, author, document type, project phase, status, revision, or unique document identifier. An incorrect file name may therefore result not only from a typographical error, but also from invalid metadata, outdated project codes, incorrect discipline classification, inconsistent document type assignment, or semantic inconsistency between the coded file name and the descriptive document title. Previous studies on BIM information quality and data quality assurance indicate that incomplete, inaccurate, inconsistent, or unintelligible information can significantly reduce the usefulness of BIM-based information, and that quality assurance should not be limited to geometric data only [12,13].

Existing research on BIM information management has addressed several related aspects of data standardisation, information requirements, model validation, and automated compliance checking. Automated compliance checking has been widely investigated in relation to BIM models, regulatory requirements, geometric constraints, property sets, classification systems, and information delivery requirements [14,15]. In openBIM environments,

classification resources such as Uniclass support information structuring and classification [16], while Industry Foundation Classes (IFC), Information Delivery Specification (IDS), the buildingSMART Data Dictionary (bSDD), and BIM Collaboration Format (BCF) provide mechanisms for exchanging model data, defining machine-readable information requirements, standardising terminology, and communicating coordination issues [17,18]. These developments show a clear trend towards machine-readable requirements and automated validation in BIM workflows.

However, these approaches primarily address model data, object properties, classifications, information requirements, or issue exchange. They do not directly provide a project-level metric for assessing whether file names themselves comply with project-specific naming rules, whether encoded values are valid according to lookup tables, and whether coded file name information is consistent with human-readable document titles. This distinction is important because many information containers exchanged in CDEs are not IFC models. They are often PDFs, drawings, reports, spreadsheets, schedules, images, or other documents whose primary machine-readable identifier remains the file name.

Recent studies have also explored the use of natural language processing (NLP) and large language models (LLMs) in BIM and AEC workflows. These studies on generative AI in AEC also indicate that text-based models are increasingly being explored for documentation workflows, information retrieval, design support, and decision assistance in construction project environments [19]. General reviews describe the potential, architectures, applications, and limitations of LLMs [20–22]. In the BIM domain, LLM-based approaches have been investigated for regulatory compliance checking, interpretation of textual requirements, generation of checking rules, integration with BIM authoring environments, and support for semi-automated code compliance checking [23,24], including approaches that combine large language models with deep learning for automated BIM compliance assessment [25]. These studies indicate that language-based AI methods can support the interpretation of semi-structured and natural-language information in construction project environments. Their application to CDE document naming, however, remains less explored, particularly in relation to semantic consistency between coded file names and descriptive titles, as existing LLM-oriented BIM studies focus mainly on model checking, code compliance, IFC-based workflows, or broader AEC applications [25,26].

The present study adopts the Optimal Naming Convention, hereafter referred to as the Optimal Standard (OS), as the reference naming convention for testing the proposed framework. The OS was previously proposed as an extension and refinement of naming recommendations associated with ISO 19650 [27]. It was developed on the basis of analyses of national and organisational naming standards, industry surveys, and considerations related to the cognitive aspects of encoding and decoding information in file names. Its purpose was to define a compact but informative structure for naming information containers in BIM/CDE environments. In the present paper, the OS is used as a reference structure, while project-specific lookup tables define the valid values permitted in individual fields. The broader need for consistent naming is also reflected in previous research concerning naming objects in BIM and nomenclature for offsite construction, where systematic naming is treated as a condition for reliable information exchange and automation [28,29].

Creating or adopting a naming convention is only the first step towards improving information exchange in CDE-based workflows. The next step is to verify whether the convention is correctly implemented in real projects. Such verification requires a formalised naming structure, project-specific reference values, and a grading mechanism capable of expressing partial compliance. Similar grading concepts have been introduced in other areas of BIM standardisation and information management. Model development levels describe the geometric and informational maturity of model elements depending on the project stage [30]. Other examples include

methods for assessing georeferencing accuracy, such as LoGeoRef [31], and approaches related to the assessment of information completeness and classification consistency in BIM models, including COBie data consistency [32]. These examples show that measurable levels, scales, and checking procedures can support communication between stakeholders and reduce ambiguity in information management processes.

Comparable approaches are less developed for file naming compliance. In many projects, file names are checked manually, only partially checked by CDE configuration rules, or validated only at the level of basic syntax. This limits the ability to evaluate naming quality in a measurable way and makes it difficult to distinguish between different types of naming errors. A file name may be structurally incorrect, structurally correct but referentially invalid, or both structurally and referentially correct while still being semantically misleading. A more diagnostic validation approach should therefore distinguish between syntactic compliance, referential compliance, and semantic consistency.

The development of the proposed framework was motivated by a real-world documentation management problem observed during work on the Czeremcha modular railway station project. The documentation received by the authors contained files prepared under heterogeneous naming practices, revealing inconsistencies in naming structure, coding logic, use of discipline and document type identifiers, and descriptive titles. This case indicated the practical difficulty of controlling file name quality in CDE-based workflows and informed the development of the controlled validation dataset used in this study.

The gap addressed in this paper is therefore not the absence of CDE platforms, BIM standards, or automated model-checking methods. Rather, it is the lack of a dedicated framework for assessing the degree of compliance of file names as structured information containers. In particular, existing approaches rarely combine: (1) rule-based validation of file name structure, (2) validation of individual code fields against project-specific lookup tables, and (3) semantic comparison between coded file name information and descriptive document titles. This gap is especially relevant in real-world CDE workflows, where naming conventions are implemented by multiple stakeholders, code tables may differ between projects, and document titles are often written in natural language.

To address this gap, this paper proposes the Level of Naming Compliance (LoNC), a multi-level framework for assessing file name compliance in BIM/CDE environments. LoNC evaluates file names not only as character strings but also as structured information containers that encode project-specific metadata. The proposed framework combines syntactic validation, referential validation, and LLM-supported semantic validation. In the present implementation, semantic validation is tested for selected fields, namely discipline and document type.

The main contributions of this paper are as follows:

- (1) the formalisation of file name compliance as a multi-level assessment problem in BIM/CDE workflows;
- (2) the development of the LoNC framework for evaluating compliance across syntactic, referential, and semantic dimensions;
- (3) the implementation of a validator supporting automated file name checking against an adopted naming standard and project-specific lookup tables; and
- (4) the evaluation of the proposed approach using a controlled synthetic validation dataset inspired by inconsistencies observed in a real-world CDE project, with performance assessed across predefined compliance and non-compliance classes.

The study addresses the following research questions:

RQ1: How can file name compliance in BIM/CDE environments be formalised as a measurable, multi-level assessment framework?

RQ2: To what extent can syntactic and referential validation detect inconsistencies between project-specific naming conventions and actual file names?

RQ3: Can LLM-supported semantic validation identify inconsistencies between coded file names and descriptive document titles within the tested scope of discipline and document type?

RQ4: How accurately and consistently does the proposed LoNC framework classify predefined syntactic, referential, and semantic file name inconsistencies across different LoNC classes in a controlled validation dataset?

2. Materials and Methods

2.1. Methodological Objective and Scope

The objective of the proposed method is to assess the compliance of file names used in BIM and Common Data Environment (CDE) workflows with an adopted naming convention in a measurable and reproducible manner. The method treats a file name as a structured information container that encodes project-specific metadata and classifies it using the Level of Naming Compliance (LoNC).

The proposed assessment covers three complementary dimensions of file name compliance: syntactic compliance, referential compliance, and selected semantic consistency. Syntactic validation verifies whether the file name follows the required formal structure of the adopted naming convention. Referential validation checks whether the parsed code fields correspond to valid project-specific values defined in lookup tables. Semantic validation assesses whether selected decoded file name fields are consistent with the descriptive document title. In the present implementation, the semantic stage is limited to discipline and document type.

The method was evaluated using a controlled synthetic validation dataset developed from inconsistency types observed in a real-world CDE project. This design made it possible to assign an expected LoNC level to each case before validation and to compare the validator output with a predefined reference classification.

2.2. Motivating Case and Controlled Validation Dataset

The development of the LoNC framework was motivated by documentation management problems observed during work on the Czeremcha modular railway station project in Poland, carried out under Europe's Rail Flagship Project 4—Sustainable and Green Rail initiative. The documentation received by the authors consisted of 519 files prepared by different stakeholders under heterogeneous naming practices. The files revealed inconsistencies in file name structure, coding logic, discipline and document type identifiers, and descriptive titles.

The original Czeremcha documentation was not used as a direct quantitative validation dataset because it did not represent a consistent ground-truth dataset. The files were prepared under different naming rules; not all of them followed a single reference naming convention, and some metadata required for full validation were incomplete, ambiguous, or project-specific. Therefore, using these files directly as a benchmark would not allow unambiguous assignment of expected LoNC levels.

For this reason, the Czeremcha documentation was used as a motivating case and as a qualitative source for identifying recurring inconsistency types. Based on these observations, a controlled synthetic validation dataset of 500 file-title pairs was developed. Each synthetic case was assigned an expected validation outcome before testing.

The dataset was designed according to the sequential logic of the LoNC framework. Out of 500 cases, 450 were designed to pass syntactic validation and therefore reach at least LoNC 1. Of these, 250 were designed to pass referential validation and therefore reach at

least LoNC 2. Of these, 100 were designed to pass LLM-supported semantic validation within the tested scope and therefore reach LoNC 3. Consequently, the expected final LoNC distribution was 50 files at LoNC 0, 200 files at LoNC 1, 150 files at LoNC 2, and 100 files at LoNC 3. Because this distribution was intentionally unequal, the evaluation procedure included both overall and class-balanced performance measures.

The sequential design of the controlled validation dataset and the resulting expected LoNC distribution are summarised in Table 1.

Table 1. Controlled validation dataset design.

Validation Stage/Expected Final Level	Number of Files	Percentage	Meaning
Total synthetic file-title pairs	500	100%	Complete controlled validation dataset
Passed syntactic validation	450	90%	Files reaching at least LoNC 1
Passed syntactic and referential validation	250	50%	Files reaching at least LoNC 2
Passed syntactic, referential, and selected semantic validation	100	20%	Files reaching LoNC 3
Final LoNC 0	50	10%	Syntactic validation failed
Final LoNC 1	200	40%	Syntactic validation passed, referential validation failed
Final LoNC 2	150	30%	Syntactic and referential validation passed, selected semantic validation failed or was not confirmed
Final LoNC 3	100	20%	Syntactic, referential, and selected semantic validation passed

The controlled dataset intentionally included both correct and incorrect examples. The incorrect cases were assigned to predefined error categories corresponding to the three validation stages of the LoNC framework. This allowed syntactic errors, referential errors, semantic inconsistencies, and fully compliant cases to be tested separately. The main types of intentionally introduced errors are listed in Table 2.

Table 2. Error types intentionally included in the controlled dataset.

Expected LoNC Level	Number of Files	Designed Condition	Example Error Type
LoNC 0	50	File name does not pass syntactic validation	Incorrect number of fields, incorrect field length, missing separator, invalid character type
LoNC 1	200	File name passes syntactic validation but fails lookup-table validation	Invalid discipline code, invalid author code, invalid document type code, invalid phase code
LoNC 2	150	File name passes syntactic and referential validation but fails selected semantic validation	Discipline-title contradiction or document type-title contradiction
LoNC 3	100	File name passes all tested validation stages	Correct structure, valid lookup values, and semantic consistency for discipline and document type
Total	500	Not applicable	Not applicable

2.3. Input Data and Reference Resources

Each record in the controlled validation dataset consisted of a file name, a descriptive document title, an expected LoNC level, and an expected error category. The file name was used in all validation stages. The descriptive document title was used only in the LLM-supported semantic validation stage. The expected LoNC level and expected error

category were not used by the validator during classification; they were used only after validation to compare the returned result with the predefined reference classification. The fields included in each record of the controlled validation dataset are presented in Table 3.

Table 3. Input data and reference resources used in the LoNC validation procedure.

Dataset Field	Description	Used by Validator	Used for Evaluation
File name	Synthetic name of the information container	Yes	Yes
Document title	Human-readable title corresponding or not corresponding to selected file name fields	Yes, semantic stage	Yes
Expected LoNC	Predefined reference classification	No	Yes
Expected error category	Predefined type of inconsistency	No	Yes
Discipline code	Extracted from the file name	Yes	Yes
Document type code	Extracted from the file name	Yes	Yes
Notes/error description	Explanation of the designed inconsistency	No	Yes

The validation method requires four categories of input data: the file name to be validated, the descriptive document title, the reference naming convention, and project-specific lookup tables. The naming convention defines the structure of the file name, including the number, order, length, and meaning of individual fields. The lookup tables define the valid values that may appear in selected fields. This distinction is important because the same naming structure can be reused across projects, while project codes may differ depending on client requirements, CDE configuration, project scope, and document control procedures.

2.4. Reference Naming Convention

The reference naming convention used in this study is the Optimal Standard (OS) proposed in previous work [27]. The OS was developed as an extension and refinement of the naming recommendations associated with ISO 19650 [5,6]. Its purpose is to provide a compact but informative structure for naming information containers exchanged in BIM/CDE environments.

The OS is based on an eight-field structure separated by dashes. The fields are arranged from general project-level information to more specific document-level information and include codes related to the project, volume, part or location, discipline, author, document type, phase, and unique document identifier. The separator is a dash “-”, and the field lengths are predefined in order to balance conciseness, readability, and information capacity. The structure of the Optimal Standard used as the reference naming convention is shown in Table 4, where “D” stands for digits and “L” stands for letters.

Table 4. Structure of the Optimal Standard used as the reference naming convention [27].

WHERE			WHO		WHAT		
Project Number	Volume	Part	Discipline	Author	Document Type	Phase	Number
PPPPP	VVV	RRR	DD	OOO	FF	SS	NN
5	3	3	2	3	2	2	6
D(+L)	D+L	D+L	L	L	L	D+L	D(+L)

The OS defines the structure of the file name, but it does not prescribe all possible values for each code field. These values must be defined for a specific project and made available to stakeholders, usually in the form of naming tables or lookup tables. In BIM projects, such tables are typically part of, or an appendix to, the BIM Execution Plan and should be aligned with information management responsibilities defined under ISO 19650 [5,6].

For the controlled validation dataset, the OS structure was instantiated using a fixed project number and volume code in order to isolate the validation of file name structure, reference values, and selected semantic consistency. The project number was set to 24680 and the volume code was set to CRM. The part field used two values: EXT for exterior-related documentation and INT for interior-related documentation. The discipline field used five predefined discipline codes: AR, EL, PL, ME, and ST. The document type field used four predefined document type codes: RE, DR, SH, and VI. The reference codes used in the controlled validation dataset are listed in Table 5.

Table 5. Reference codes used in the controlled validation dataset.

Field	Code	Meaning
Project number	24680	Project identifier
Volume	CRM	Czeremcha modular railway station
Part	EXT	Exterior
Part	INT	Interior
Discipline	AR	Architecture
Discipline	EL	Electrical
Discipline	PL	Plumbing
Discipline	ME	Mechanical
Discipline	ST	Structural
Document type	RE	Report
Document type	DR	Drawing
Document type	SH	Schedule
Document type	VI	Visualisation

Synthetic author and phase codes were also used in the controlled dataset, but they served only as project-specific reference values for lookup-table validation.

2.5. Definition of the Level of Naming Compliance

The Level of Naming Compliance (LoNC) is defined as a discrete compliance level assigned to a file name after a sequence of validation operations. It expresses the degree to which the file name satisfies the adopted naming convention, project-specific reference data, and selected semantic consistency criteria.

LoNC is cumulative. A file name can reach a higher level only if it satisfies the requirements of all preceding levels. A file name that fails syntactic validation is classified as LoNC 0. A file name that passes syntactic validation but fails referential validation is classified as LoNC 1. A file name that passes syntactic and referential validation but does not receive semantic confirmation is classified as LoNC 2. A file name that passes all three validation stages reaches LoNC 3.

Let x denote a single naming instance, such as a file name. The three binary validation functions used to determine the LoNC level are defined in Equation (1):

$$f_s(x), f_r(x), f_m(x) \in \{0, 1\} \quad (1)$$

In Equation (1), $f_s(x)$ denotes syntactic validation, $f_r(x)$ denotes referential validation, and $f_m(x)$ denotes semantic validation. A value of 1 indicates that the respective validation stage has been passed, whereas a value of 0 indicates that it has not been passed. The cumulative LoNC classification rule is defined in Equation (2):

$$LoNC(x) = \begin{cases} 0, & \text{if } f_s(x) = 0 \\ 1, & \text{if } f_s(x) = 1 \text{ and } f_r(x) = 0 \\ 2, & \text{if } f_s(x) = 1, f_r(x) = 1 \text{ and } f_m(x) = 0 \\ 3, & \text{if } f_s(x) = 1, f_r(x) = 1 \text{ and } f_m(x) = 1 \end{cases} \quad (2)$$

As shown in Equation (2), a higher LoNC level can be reached only when all preceding validation stages have been successfully completed.

This definition makes it possible to identify the highest level of compliance reached by each file name and the stage at which non-compliance occurs. The four LoNC levels and their interpretation are summarised in Table 6.

Table 6. Definition and interpretation of LoNC levels.

LoNC Level	Validation Stage Passed	Interpretation
LoNC 0	None; syntactic validation failed	The file name does not comply with the required OS structure and cannot be reliably parsed.
LoNC 1	Syntactic validation passed	The file name has the correct formal structure, but the validity of project-specific code values is not confirmed.
LoNC 2	Syntactic and referential validation passed	The file name has the correct structure and uses valid project-specific codes, but semantic consistency with the title is not confirmed.
LoNC 3	Syntactic, referential, and semantic validation passed	The file name is structurally and referentially compliant, and the discipline and document type encoded in the file name are semantically consistent with the descriptive document title.

2.6. Sequential Validation Procedure

The validation procedure follows the cumulative logic of the LoNC definition. First, the file name is checked against the formal structure of the reference naming convention. If this check fails, the process stops and LoNC 0 is assigned. If the file name passes syntactic validation, the parsed fields are compared with project-specific lookup tables. If at least one required code is invalid or missing, the process stops and LoNC 1 is assigned. If referential validation is passed, selected decoded fields are compared with the descriptive document title using LLM-supported semantic validation. If semantic consistency is confirmed, the file name reaches LoNC 3; otherwise, it remains at LoNC 2 or is marked for manual review. The sequential validation workflow and the expected transition of cases between LoNC levels are illustrated in Figure 1.

In the controlled validation dataset, this sequential logic was reflected directly in the expected distribution of cases across LoNC levels. This made it possible to evaluate the validator at each stage and to verify whether the final classification corresponded to the predefined expected level.

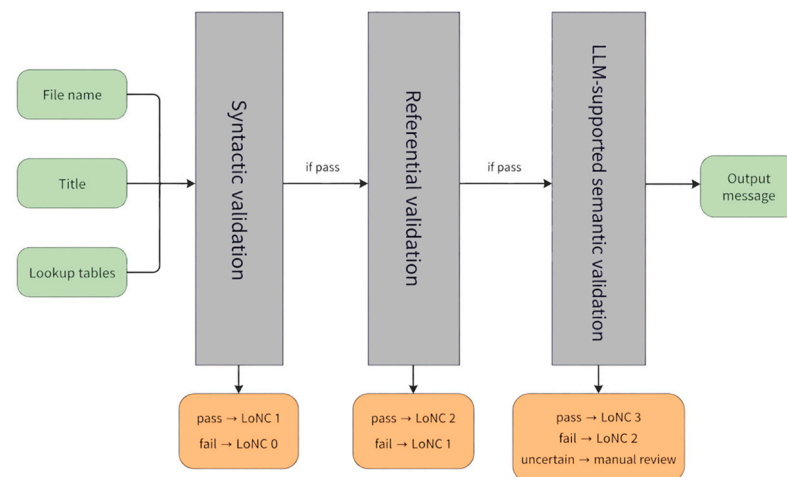


Figure 1. Sequential LoNC validation workflow. Green boxes represent input data and the final user-facing output. Grey boxes indicate the consecutive validation modules. Orange boxes show decision outcomes, including LoNC classification, failure states, and cases requiring manual review.

2.7. Syntactic Validation

Syntactic validation verifies whether the file name follows the formal structure required by the reference naming convention. This stage does not assess whether the codes are meaningful in the project context. It only checks whether the file name can be parsed into the expected fields.

Regular expressions were used because they provide a deterministic and reproducible mechanism for validating string patterns, including field order, field length, separators, and permitted character classes [33–35]. In the proposed method, syntactic validation checks the number and order of fields, the use of the required separator, the length of each field, the permitted character types in each field, and the overall compatibility of the file name with the reference pattern.

For the OS structure used in this study, the general regular expression can be written as follows:

$$\wedge([A-Z0-9]{5})-([A-Z0-9]{3})-([A-Z0-9]{3})-([A-Z]{2})-([A-Z]{3})-([A-Z]{2})-([A-Z0-9]{2})-([A-Z0-9]{6})\$$$

This expression checks whether the file name consists of eight fields separated by dashes and whether each field satisfies the required length and character-type constraints. A file name that passes this stage is assigned at least LoNC 1. A file name that fails this stage is classified as LoNC 0.

In the controlled dataset, syntactic errors were introduced by modifying the structure of selected file names. These modifications included incorrect field length, missing separators, missing fields, additional fields, and invalid character types. The logic of the syntactic validation stage based on regular expressions is shown in Figure 2.

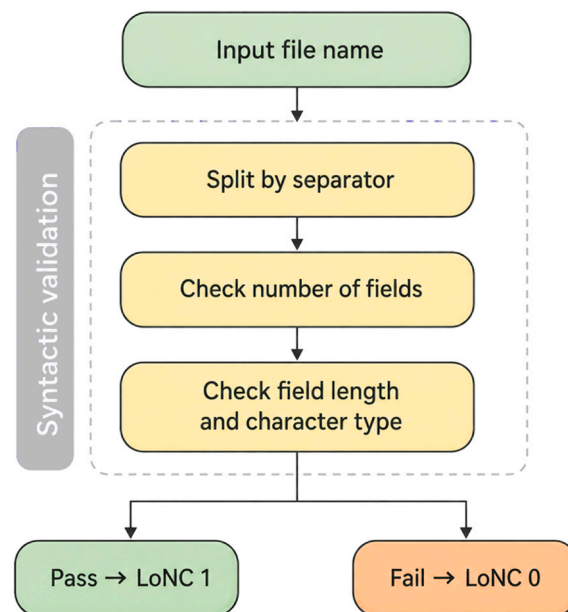


Figure 2. Syntactic validation using regular expressions. Green indicates input data and successful validation outcomes, yellow represents individual validation operations, grey identifies the validation module, and orange denotes failed validation outcomes.

2.8. Referential Validation Using Lookup Tables

Referential validation verifies whether the individual code fields extracted from the file name correspond to valid values defined in project-specific lookup tables. In contrast to syntactic validation, this stage assesses whether the parsed fields are meaningful within the project context. For example, a discipline code may be formally correct as a two-character string, but invalid if it does not occur in the project discipline table.

Each relevant field is mapped to a corresponding lookup table. A field passes referential validation if its value is present in the appropriate table and assigned to the expected category. If all required fields pass this check, the file name is considered referentially compliant and reaches LoNC 2. If at least one required code is missing from the relevant table or appears in an incorrect category, the file name remains at LoNC 1. The mapping between OS fields, lookup tables, and validation questions is presented in Table 7.

Table 7. Example mapping between OS fields and lookup tables.

OS Field	Lookup Table	Validation Question
Project number	Project identifiers	Is the project code valid for the analysed project?
Volume	Volume/building/zone codes	Is the volume code defined in the project?
Part	Location/level/part codes	Is the part or location code defined in the project?
Discipline	Discipline codes	Is the discipline code valid?
Creator/author	Organisation or originator codes	Is the author code registered?
Document type	Document type codes	Is the document type valid?
Phase	Project phase codes	Is the phase code valid?
Unique number	Numbering rules or document register	Is the identifier consistent with the adopted numbering logic?

In the controlled dataset, referential errors were introduced by replacing selected valid codes with syntactically correct but non-existing codes. As a result, these file names could pass regular-expression validation but fail lookup-table validation. This allowed LoNC 1 cases to be generated in a controlled manner.

Lookup tables may be prepared specifically for a project or derived partly from existing classification systems. In actual projects, tables of code field values are often included in BIM-related documentation, for example, as appendices to the BIM Execution Plan or as a dedicated File Naming Protocol. The appointing party may define the required parameter values at the beginning of a project, but in practice these values are frequently agreed upon by multiple stakeholders. Therefore, lookup tables should be considered project-specific reference resources rather than universal code lists. The referential validation workflow using project-specific lookup tables is illustrated in Figure 3.

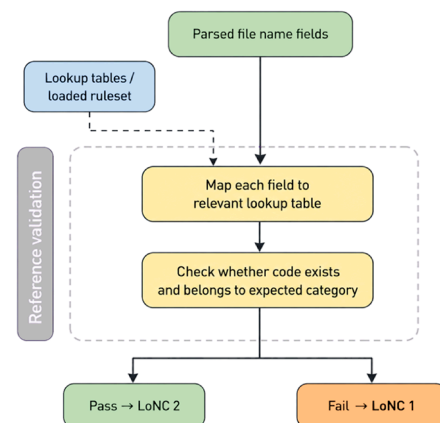


Figure 3. Referential validation using project-specific lookup tables. Green denotes validated input data and successful outcomes, blue identifies external reference sources such as lookup tables and project-specific rulesets, yellow represents validation operations, grey indicates the validation module, and orange denotes failure at the current validation stage.

An example layout of a project-specific lookup table used by the validator is shown in Table 8.

Table 8. Example layout of a project-specific lookup table.

Project Number	Volume		Part	Discipline	Discipline Description	Originator	Form
00001	B01	building or block 01	B01	AR	Architect	ABC	VI
00002	B02	building or block 02	B02	SE	Structural Engineer	NPQ	TX
00003	B03	building or block 03	B03	CE	Civil Engineer	FPD	CL
00004	B04	building or block 04	B04	ME	Mechanical Engineer	BXR	M3

2.9. LLM-Supported Semantic Validation

Semantic validation assesses whether selected information encoded in the file name is consistent with the descriptive document title. This stage is required because a file name may be syntactically correct and may use valid project codes while still being semantically inconsistent with the actual document description. For example, a file name may include a code corresponding to an architectural drawing, whereas the title may describe a structural report.

In the present implementation, LLM-supported semantic validation was tested for two OS fields: discipline and document type. These fields were selected because they are usually directly reflected in descriptive document titles and can be meaningfully compared with natural-language expressions. Other OS fields, such as project number, volume, part, phase, author, and unique number, were not included in semantic validation because their interpretation depends more strongly on project-specific context, internal numbering logic, or external document registers.

Semantic inconsistencies in the controlled dataset were introduced only after syntactic and referential correctness had been ensured. This means that LoNC 2 cases contained formally correct file names and valid lookup-table codes, but the descriptive title contradicted either the discipline code or the document type code. Such cases were designed to test whether the LLM-supported validation component could identify inconsistencies that cannot be detected by regular expressions or lookup tables.

The semantic validation was performed using a large language model. In this study, DeepSeek-R1 was accessed through an API and used for the LLM-supported semantic validation stage. The model was queried using deterministic settings in order to improve reproducibility. The temperature was set to 0.0, no explicit maximum output length was set, and the model was instructed to return a structured response containing a decision, a short justification, and the fields responsible for any detected inconsistency.

The method uses a dynamic-context approach rather than model fine-tuning or retrieval-augmented generation. Fine-tuning requires a dedicated training dataset, computational resources, and repeated retraining when project requirements change. Retrieval-augmented generation can support access to external knowledge bases, but it introduces an additional retrieval layer and requires knowledge-base maintenance [36–39]. In the proposed workflow, the LLM does not create new codes, modify lookup tables, or expand project vocabularies. Its role is limited to assessing whether the decoded meaning of the selected file name fields is consistent with the descriptive document title. Therefore, the LLM acts as a semantic comparison component rather than as an autonomous decision-making system [40,41]. This positioning is consistent with recent BIM-oriented LLM research, where language models are increasingly used to support interpretation and checking tasks, but require task-specific constraints, structured inputs, and controlled outputs [26].

The workflow of LLM-supported semantic validation for the selected OS fields is shown in Figure 4.

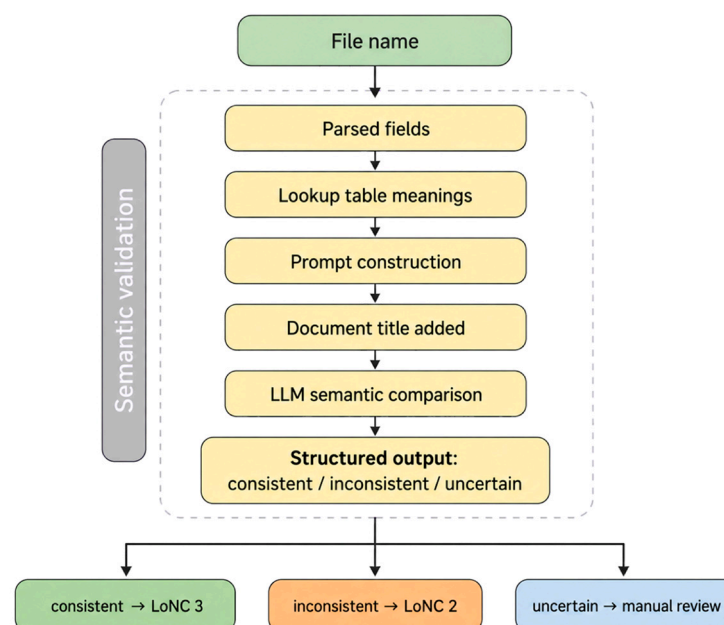


Figure 4. LLM-supported semantic validation workflow. Green indicates validated inputs and successful outcomes. Yellow represents internal processing and validation operations. Grey identifies the parent validation module. Orange denotes a failed validation outcome at the respective LoNC level. Blue marks external reference data or cases requiring manual review.

2.10. Prompt Design and LLM Output Structure

The prompt was designed to minimise open-ended interpretation and to force the model to compare clearly defined input fields. Each prompt contained the original file name, the parsed fields extracted from the file name, the corresponding code meanings obtained from lookup tables, the descriptive document title, and explicit decision criteria for semantic consistency.

The model was instructed not to assess the general quality of the title or the document, but only to determine whether the decoded file name information was consistent with the document title within the tested semantic scope. The expected output was structured in order to support automatic parsing by the validator.

The general prompt structure used in the validation procedure was as follows:

You are validating the semantic consistency of a BIM/CDE file name.

Input data:

File name: {file_name}

Decoded file name fields used for semantic validation:

- Discipline: {discipline_code} = {discipline_meaning}
- Document type: {document_type_code} = {document_type_meaning}

Additional file name fields provided only as context:

- Project number: {project}
- Volume: {volume}
- Part/location: {part}
- Author: {author}
- Phase: {phase}
- Unique number: {unique_number}

Document title:

{document_title}

Task:

Compare the decoded discipline and document type information with the document title.

Assess only whether these two fields are semantically consistent with the title.

Do not evaluate project number, volume, part/location, author, phase, or unique number.

Return the answer in the following JSON format:

```
{
  "semantic_consistency": "consistent" or "inconsistent" or "uncertain",
  "inconsistent_fields": ["field_name_1", "field_name_2"],
  "justification": "short explanation"
}
```

Decision rules:

- Return "consistent" if the decoded discipline and document type agree with the title.
- Return "inconsistent" if the title clearly contradicts either discipline or document type.
- Return "uncertain" if the title does not contain enough information to verify consistency.
- Do not infer information that is not present in the title.
- Do not create new codes.
- Do not change the meaning of existing codes.

The application parsed the LLM output as a structured JSON response. Cases classified as consistent were allowed to reach LoNC 3. Cases classified as inconsistent or uncertain were retained at LoNC 2 and treated as requiring correction or manual review.

2.11. Evaluation Criteria and Interpretation of Results

Because each synthetic case had a predefined expected LoNC level, the controlled dataset enabled direct comparison between the expected classification and the classification returned by the validator. Overall accuracy was calculated as the ratio of correctly classified cases to all cases in the dataset. Because the expected LoNC levels were not equally represented in the controlled dataset, overall accuracy was supplemented with balanced accuracy, defined as the arithmetic mean of the recall values calculated separately for LoNC 0, LoNC 1, LoNC 2, and LoNC 3. Accuracy was also calculated separately for each LoNC level to verify whether the validator performed consistently across all predefined compliance classes.

For the semantic validation stage, false-positive and false-negative cases were analysed separately. A false positive was defined as a case in which the LLM classified a semantically inconsistent file-title pair as consistent. A false negative was defined as a case in which the LLM classified a semantically consistent file-title pair as inconsistent or uncertain. Cases classified as uncertain were treated conservatively and were not counted as LoNC 3. The evaluation metrics used to assess the validator and the LLM-supported semantic validation stage are summarised in Table 9.

Table 9. Evaluation metrics used for validator and LLM-supported semantic validation.

Assessment	Sample Size	Purpose	Metric
Expected vs. returned LoNC	500	Overall validator correctness	Classification accuracy
Expected vs. returned LoNC	500	Class-balanced validator correctness	Balanced accuracy
LoNC-level classification	500	Correctness for each LoNC level	Accuracy per LoNC level
LLM semantic validation	250	Semantic-stage correctness	False positives/false negatives

The results of the validation procedure were interpreted at two levels. First, each file was assigned the highest LoNC level reached during the sequential validation process. This made it possible to calculate the distribution of files across LoNC levels in the controlled dataset. Second, each non-compliant case was assigned to an error category corresponding to the stage at which validation failed. This allowed the analysis to distinguish between structural errors, invalid project codes, and semantic inconsistencies. The error categories used to interpret the validation results are listed in Table 10.

Table 10. Error categories used for result interpretation.

Error Category	Validation Stage	Description	Resulting LoNC
Structural error	Syntactic	Incorrect separator, incorrect number of fields, incorrect field length, or invalid character type	LoNC 0
Invalid reference value	Referential	Code not found in the relevant project lookup table	LoNC 1
Incorrect reference category	Referential	Code exists but is assigned to an incorrect field or category	LoNC 1
Semantic contradiction	Semantic	Semantic contradiction in discipline or document type	LoNC 2
Insufficient semantic evidence	Semantic	Insufficient semantic evidence for discipline or document type	LoNC 2/manual review
Full compliance	All stages	Structure, reference values, and semantic consistency confirmed	LoNC 3

At dataset level, the distribution of files across LoNC levels can be used as an indicator of naming quality under controlled conditions. The number of files failing at each stage indicates whether the validator correctly detects the intended types of non-compliance. This supports the assessment of the LoNC framework as a diagnostic tool for file name validation.

2.12. Implementation of the Validation Procedure

The proposed method was implemented in a dedicated proof-of-concept file name validator. To support transparency, reproducibility, and further testing, the validator has been made publicly available in a GitHub repository, as indicated in the Data Availability Statement. The validator was developed as a proof-of-concept application to test whether the LoNC framework could be operationalised in a practical CDE-related workflow. The application receives a file name, a descriptive document title, and project-specific lookup tables as input. It then applies the three validation stages sequentially and returns the assigned LoNC level together with diagnostic information about detected errors.

The validator first performs syntactic validation using the regular expression defined for the OS structure. If the file name fails this check, the process stops and LoNC 0 is assigned. If syntactic validation is passed, the validator extracts the individual fields and compares them with lookup tables. If one or more values are not found in the relevant table, the process stops and LoNC 1 is assigned. If referential validation is passed, the validator generates a prompt for the LLM by combining the parsed fields, their decoded meanings, and the descriptive document title. The LLM response is then parsed and used to assign LoNC 2 or LoNC 3, depending on the semantic consistency result.

The validator also generates diagnostic information indicating the validation stage at which non-compliance occurred, the field responsible for the error, and the reason for the assigned LoNC level. This output is intended to support document controllers and CDE managers in correcting naming errors and improving information consistency. The pseudocode of the implemented validation procedure is provided in Appendix A to support reproducibility of the workflow.

2.13. Methodological Limitations

The proposed method has several methodological limitations. First, the quantitative validation was performed on a controlled synthetic dataset rather than directly on the original Czeremcha documentation. This decision was made because the original files were created under heterogeneous naming practices and did not constitute a consistent ground-truth dataset. The controlled dataset enabled predefined expected LoNC levels to be assigned to each case, but future work should validate the method on larger real-world datasets prepared according to a single reference naming convention.

Second, referential validation depends on the completeness and correctness of project-specific lookup tables. If a lookup table is incomplete or outdated, valid file names may be incorrectly classified as non-compliant. Third, semantic validation depends on the quality and informativeness of document titles. If a title is too generic, ambiguous, abbreviated, or inconsistent with the actual document content, semantic agreement may be difficult to determine.

Fourth, LLM-supported validation may be affected by model version, prompt formulation, parameter settings, context length, and non-deterministic output behaviour. For this reason, the method uses structured prompts, deterministic settings where possible, conservative acceptance criteria, and a separate consistency assessment. Finally, the present implementation validates semantic consistency only for discipline and document type. It does not semantically validate all eight OS fields and does not analyse the full content of the file. This limitation should be addressed in future research by extending semantic validation to additional fields and, where technically and legally possible, to document content.

3. Results

3.1. Controlled Validation Dataset

The controlled validation dataset consisted of 500 synthetic file-title pairs. Each case had a predefined expected LoNC level assigned before validation. The dataset was

structured according to the sequential logic of the LoNC framework: 450 cases were designed to pass syntactic validation, 250 to pass both syntactic and referential validation, and 100 to pass syntactic, referential, and LLM-supported semantic validation within the tested scope. This resulted in the expected LoNC distribution shown in Table 11.

Table 11. Expected LoNC distribution in the controlled validation dataset.

Expected LoNC Level	Number of Files	Percentage of Dataset	Designed Validation Outcome
LoNC 0	50	10%	Syntactic validation failed
LoNC 1	200	40%	Syntactic validation passed, referential validation failed
LoNC 2	150	30%	Syntactic and referential validation passed, selected semantic validation failed or was not confirmed
LoNC 3	100	20%	Syntactic, referential, and selected semantic validation passed
Total	500	100%	

The dataset therefore included all four LoNC levels and enabled separate testing of structural errors, invalid reference values, selected semantic inconsistencies, and fully compliant cases.

3.2. Sequential Validation Results

The validation results followed the cumulative LoNC logic. Of the 500 file-title pairs, 450 passed syntactic validation and 50 failed at this first stage, resulting in LoNC 0 classifications. Among the 450 syntactically valid cases, 250 also passed referential validation against the lookup tables, while 200 failed because at least one code was not found in the corresponding reference table and were classified as LoNC 1.

The 250 cases that passed both syntactic and referential validation were submitted to LLM-supported semantic validation. Of these, 100 were assessed as semantically consistent with the descriptive title and reached LoNC 3. The remaining 150 cases did not receive semantic confirmation and were classified as LoNC 2. The sequential flow of cases through the validation stages is shown in Figure 5.

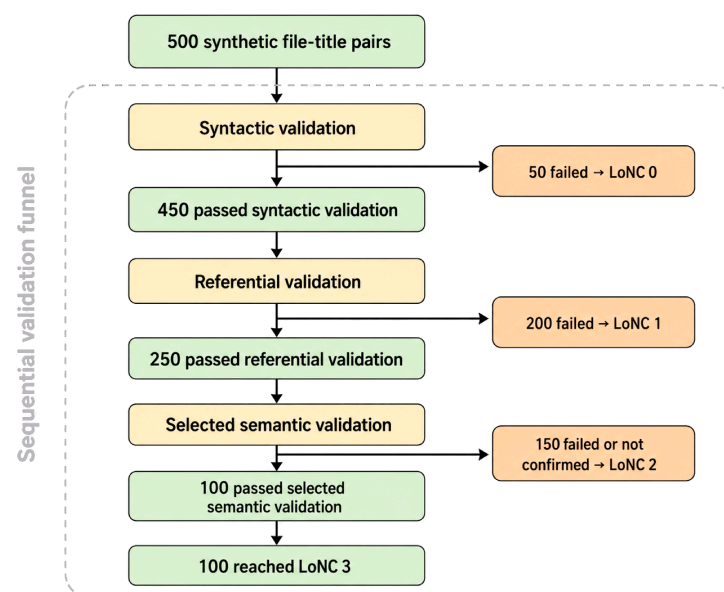


Figure 5. Sequential validation funnel for the controlled dataset. Green boxes indicate the subset of records retained after successful completion of each validation stage. Yellow boxes represent the validation operations. Orange boxes show records excluded at a given stage and assigned to the corresponding lower LoNC level.

3.3. Final LoNC Classification

The final LoNC classification returned by the validator is presented in Table 12 and Figure 6. The largest group consisted of LoNC 1 cases, representing 40% of the dataset. LoNC 2 cases accounted for 30%, LoNC 3 cases for 20%, and LoNC 0 cases for 10%.

Table 12. Final LoNC classification returned by the validator.

LoNC Level	Number of Files	Percentage of Dataset	Interpretation
LoNC 0	50	10%	Structural validation failed
LoNC 1	200	40%	Structure correct, referential validation failed
LoNC 2	150	30%	Structure and reference values correct, selected semantic consistency not confirmed
LoNC 3	100	20%	Structure, reference values, and selected semantic consistency confirmed
Total	500	100%	—

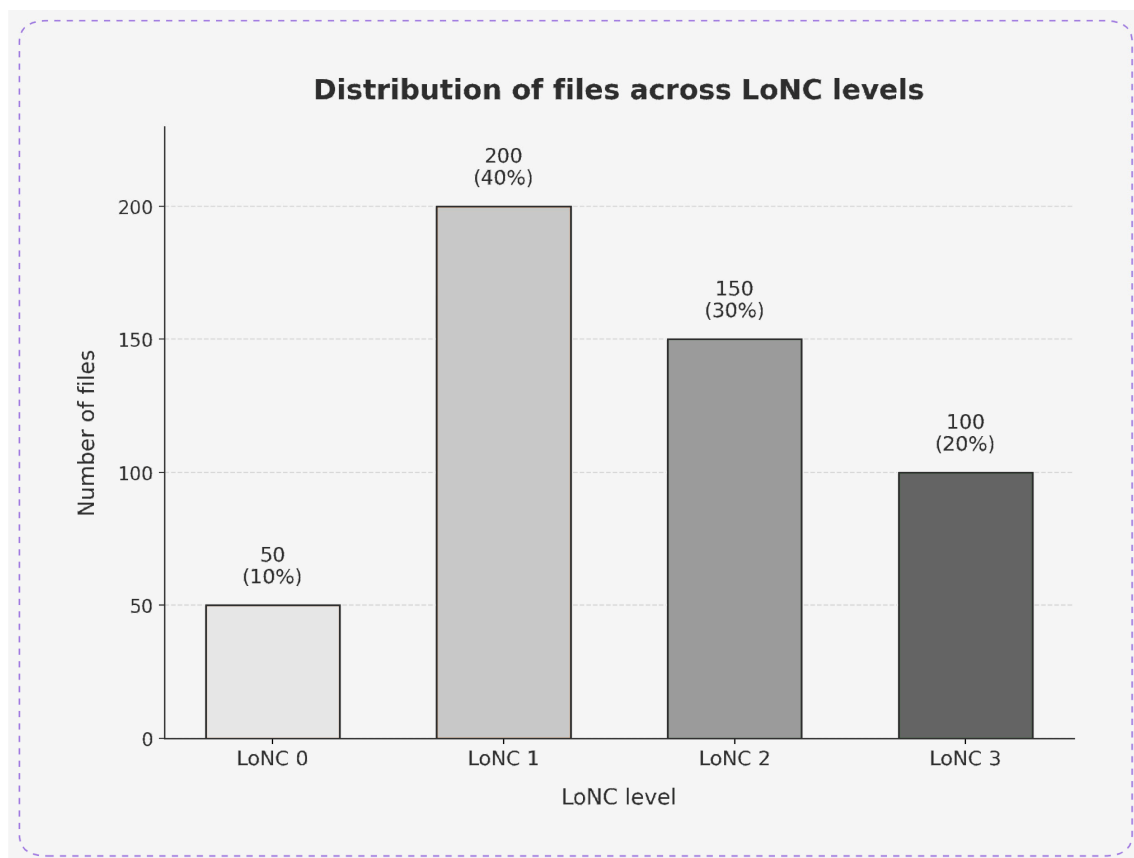


Figure 6. Distribution of files across LoNC levels.

The distribution indicates that the controlled dataset covered all four LoNC levels. The higher proportion of LoNC 1 and LoNC 2 cases was intentional and reflected the purpose of the dataset: to test whether the validator could distinguish between different stages of non-compliance rather than reproduce a random project sample. LoNC 3 should be interpreted within the tested semantic scope, which covered discipline and document type only.

3.4. Classification Accuracy

The validator output was compared with the predefined expected LoNC level for each file-title pair. The returned classifications matched the expected classifications for

all 500 controlled cases, corresponding to full agreement between expected and returned LoNC levels in the evaluated dataset. Overall accuracy was therefore 100%.

Because the expected LoNC levels were not equally represented in the controlled dataset, balanced accuracy was also calculated. Since all cases were correctly classified within each LoNC level, the class-specific recall values for LoNC 0, LoNC 1, LoNC 2, and LoNC 3 were all 100%, resulting in a balanced accuracy of 100%.

This result indicates that the implemented validation logic correctly operationalised the cumulative LoNC framework under controlled conditions. It should be interpreted as evidence of internal consistency and correct implementation of the validation workflow, not as a general estimate of expected performance in uncontrolled real-world CDE datasets.

3.5. Error Categories Detected by the Validator

The controlled dataset included predefined error categories corresponding to the three validation stages. This made it possible to verify whether the validator not only assigned the correct LoNC level but also identified the type of inconsistency responsible for the classification. The detected error categories are summarised in Table 13.

Table 13. Error categories identified in the controlled validation dataset.

Error Category	Validation Stage	Expected LoNC	Number of Cases	Example Issue
Incorrect field length	Syntactic	LoNC 0	20	Project number shorter or longer than five characters
Missing or incorrect separator	Syntactic	LoNC 0	15	Dash missing between fields
Invalid character type	Syntactic	LoNC 0	15	Invalid character used in a coded field
Invalid discipline code	Referential	LoNC 1	50	Code not present in AR, EL, PL, ME, ST
Invalid author code	Referential	LoNC 1	50	Author code not present in the lookup table
Invalid document type code	Referential	LoNC 1	50	Code not present in RE, DR, SH, VI
Invalid phase code	Referential	LoNC 1	50	Phase code not present in the accepted phase table
Discipline-title contradiction	Semantic	LoNC 2	75	File name indicates one discipline, while the title describes another
Document type-title contradiction	Semantic	LoNC 2	75	File name indicates one document type, while the title describes another
Fully compliant case	All stages	LoNC 3	100	Structure, codes, and selected semantic consistency confirmed
Total			500	

The validator correctly distinguished between the intended error categories. Syntactic errors were stopped at the first validation stage and classified as LoNC 0. Referential errors were detected only after successful syntactic validation and were classified as LoNC 1. Semantic contradictions were detected only after syntactic and referential validation had been passed and were classified as LoNC 2. Fully compliant cases passed all tested validation stages and reached LoNC 3.

3.6. LLM-Supported Semantic Validation Results

LLM-supported semantic validation was applied only to cases that had already passed syntactic and referential validation. In the controlled dataset, this corresponded to 250 file-title pairs. Of these, 100 were expected to be semantically consistent, and 150 were expected to contain a contradiction between the file name and the descriptive title.

The LLM-supported component returned classifications consistent with the predefined expected semantic outcomes for all 250 cases. All 100 semantically consistent cases were

classified as consistent, while all 150 semantically inconsistent cases were classified as inconsistent or not confirmed. No false-positive or false-negative cases were observed in the controlled semantic validation set. Cases returned as uncertain were treated conservatively and were not assigned LoNC 3.

The 100% agreement at this stage reflects the controlled and unambiguous design of the semantic test cases. It indicates that the LLM-supported component correctly handled the designed cases under controlled conditions. It should not be interpreted as a general estimate of LLM performance for real-world datasets with ambiguous, incomplete, multilingual, or inconsistent document titles.

3.7. Representative Validation Cases

Representative validation cases are shown in Table 14. They illustrate how the LoNC framework distinguishes between structural errors, referential errors, semantic contradictions, and fully compliant cases.

Table 14. Representative validation cases from the controlled dataset.

Case	File Name	Document Title	Designed Issue	Expected LoNC	Validator Output
A	24680-CRM-INT-AR-AEC-DR-02-000001	Architectural interior drawing	None	3	3
B	2468-CRM-INT-AR-AEC-DR-02-000002	Architectural interior drawing	Incorrect project number length	0	0
C	24680-CRM-INT-XX-AEC-DR-02-000003	Architectural interior drawing	Invalid discipline code	1	1
D	24680-CRM-EXT-AR-AEC-DR-02-000004	Structural exterior report	Discipline and document type contradiction	2	2
E	24680-CRM-INT-ME-BMT-SH-02-000005	Mechanical equipment schedule	None	3	3
F	24680-CRM-EXT-EL-ENG-VI-02-000006	Electrical cable routing drawing	Document type contradiction	2	2

The examples show that the same file name structure may lead to different LoNC classifications depending on the stage at which validation fails. Case B fails at the syntactic stage because the project number does not have the required length. Case C has a formally correct structure, but the discipline code is not present in the lookup table. Cases D and F pass syntactic and referential validation but fail semantic validation because the descriptive title contradicts the discipline or document type encoded in the file name. Cases A and E pass all tested validation stages and reach LoNC 3.

3.8. Diagnostic Output of the Validator

The developed validator generated diagnostic feedback indicating the validation stage at which non-compliance occurred and the field responsible for the detected inconsistency. This output is relevant because LoNC is intended not only to classify file names but also to support corrective actions by document controllers and CDE managers.

The validator distinguished between structural errors, lookup-table inconsistencies, and semantic contradictions. Examples of diagnostic outputs generated by the validator for the four LoNC outcomes are presented in Figure 7.

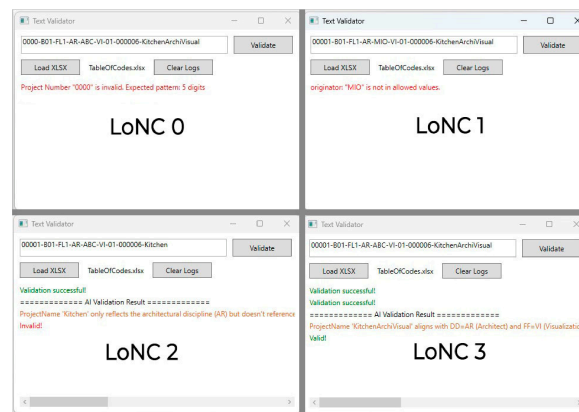


Figure 7. Examples of diagnostic outputs generated by the validator: syntactic error classified as LoNC 0, invalid lookup-table value classified as LoNC 1, semantic inconsistency classified as LoNC 2, and successful validation classified as LoNC 3.

The diagnostic output confirms that the validator can identify both the level and the location of non-compliance. This supports targeted correction of file names and reference data rather than requiring manual inspection of all project files.

3.9. Summary of Results

Overall, the results confirm that the validator correctly implemented the cumulative LoNC logic under controlled validation conditions. The framework distinguished between structural errors, invalid reference values, selected semantic inconsistencies, and fully compliant cases. Since the dataset was synthetically constructed and the semantic cases were intentionally unambiguous, the obtained full agreement should be interpreted as confirmation of internal consistency and correct implementation, rather than as a general performance estimate for uncontrolled CDE datasets.

4. Discussion

The results demonstrate that the proposed Level of Naming Compliance (LoNC) framework can be operationalised as a structured approach to assessing file naming compliance in BIM/CDE workflows. In the controlled validation dataset, the validator reproduced the predefined LoNC classification for all 500 file-title pairs. This indicates that the implemented workflow follows the cumulative logic of the framework and can distinguish between syntactic errors, invalid reference values, and selected semantic inconsistencies.

The obtained high accuracy should be interpreted within the methodological boundaries of the study. The validation dataset was synthetically constructed, and each case had a predefined expected LoNC level before testing. The result therefore indicates the internal consistency of the implemented validator and the controlled test design. It should not be interpreted as evidence that the same level of accuracy would necessarily be achieved in uncontrolled real-world CDE datasets, where file names, titles, metadata, and reference tables may be incomplete, ambiguous, inconsistent, or project-specific.

Because the controlled dataset contained an unequal distribution of LoNC classes, the result was also assessed using balanced accuracy and class-specific performance measures. These additional metrics showed that full agreement was observed across all predefined LoNC levels, rather than being driven only by the more frequent classes. Nevertheless, this does not remove the need for further validation on independently annotated real-world CDE datasets.

The main practical value of LoNC is that it provides more diagnostic information than binary file name checking. A binary approach can indicate whether a file name is

accepted or rejected, but it does not identify the stage at which non-compliance occurs. LoNC differentiates between structural errors, lookup-table inconsistencies, and semantic contradictions. This distinction is relevant for document controllers and CDE managers because each error type requires a different corrective action: structural correction of the file name, correction or extension of reference tables, or review of the relationship between coded metadata and the descriptive document title.

This staged interpretation is also important when LoNC is compared with syntax-only validation. Regular expressions are effective for checking field order, separators, field lengths, and permitted character classes [36–39]. However, they cannot determine whether a formally correct code is valid in a specific project or whether the decoded meaning of a code corresponds to the document title. Lookup-table validation extends the process by verifying project-specific reference values, while LLM-supported semantic validation addresses selected cases where the coded file name and the natural-language title must be compared. In the present implementation, this semantic layer was demonstrated for discipline and document type only.

The semantic validation component should therefore be interpreted conservatively. LoNC 3 means that the file passed syntactic validation, referential validation, and semantic validation within the tested scope. It does not mean that all eight fields of the Optimal Standard were semantically validated against the document title. Fields such as project number, volume, part, author, phase, and unique number require other validation mechanisms, including project registers, document numbering rules, CDE metadata, or additional contextual information. This distinction is important because the OS defines the file name structure, while project-specific lookup tables and project documentation define the permissible values and their interpretation [27].

The absence of false positives and false negatives in the controlled semantic validation set indicates that the LLM-supported component handled the intentionally clear test cases correctly. This supports the feasibility of using LLMs for selected semantic checks, but it does not remove the need for further testing on real-world data. Document titles in practice may be abbreviated, incomplete, multilingual, discipline-specific, or inconsistent with the actual document content. Under such conditions, LLM-supported validation may require additional context, conservative decision rules, and manual review for uncertain cases.

Several risks are associated with the use of LLMs in this workflow. Model outputs may vary with prompt formulation, model version, parameter settings, and deployment environment. Ambiguous titles may also lead to incorrect or uncertain classifications. In addition, practical implementation may involve token costs, latency, privacy constraints, and dependence on an external model provider or local deployment infrastructure. For this reason, the proposed method separates deterministic rule-based validation from LLM-supported semantic comparison and uses structured prompts, controlled outputs, and conservative acceptance criteria. These precautions are consistent with the broader need to manage the limitations of LLM-based methods in BIM and AEC workflows, including challenges related to reliability, explainability, domain specificity, data quality, and professional acceptance [20–23,42].

From the perspective of ISO 19650-based information management, LoNC can support the controlled identification of information containers [5,6]. File naming conventions are closely connected with document control, information retrieval, revision management, and the consistent use of project-specific metadata. CDE platforms already provide important functions for document storage, metadata handling, workflow control, permissions, versioning, and approval processes [7–10]. The contribution of LoNC is more specific: it provides a staged method for assessing the compliance of file names themselves. It

could therefore be implemented as a QA/QC module within a CDE platform or used as an external checking tool before documents are uploaded to a shared environment.

The proposed approach also creates opportunities for further development. One direction is the extension of semantic validation to additional fields of the naming convention. This would require richer contextual data, such as project registers, location breakdown structures, phase definitions, author registers, and document numbering rules. Another direction is the integration of controlled vocabularies and classification systems, such as Uniclass and bSDD, to improve the consistency and scalability of reference data across projects [16–18]. Such integration could support more robust validation across different project types, disciplines, and languages.

Future versions of the method could also use Retrieval-Augmented Generation (RAG) or fine-tuned models. RAG could allow the validation component to retrieve project-specific naming rules, lookup tables, BEP requirements, and CDE metadata at validation time [36–39]. Fine-tuning could be considered if a sufficiently large and stable dataset of classified BIM/CDE naming cases becomes available. However, both approaches introduce additional implementation requirements. RAG requires a maintained knowledge base and retrieval pipeline, while fine-tuning requires training data, computational resources, and model maintenance. The dynamic-context approach used in this study therefore remains a practical first step for testing LLM-supported file name validation [40,41].

The proposed validation procedure may also be interpreted in the context of agentic AI systems for information management. Individual operations, such as parsing a file name, checking syntactic conformity, validating codes against lookup tables, comparing decoded metadata with a title, and generating a compliance report, can be formalised as task-specific skills. The validator developed in this study is not a fully autonomous agent, but it defines structured validation operations that could be orchestrated by an AI agent in future CDE-based QA/QC workflows. Such workflows could support automatic LoNC reporting, detection of naming changes between revisions, and identification of recurring error patterns across projects.

The main limitation of the study is that the quantitative validation was performed on a controlled synthetic dataset rather than on a fully standardised real-world CDE dataset. This choice was deliberate because the original Czeremcha documentation was prepared under heterogeneous naming practices and did not provide a consistent ground truth. The controlled dataset enabled predefined LoNC levels to be assigned to each case and allowed systematic testing of the validator. Nevertheless, further research should validate the method on larger real-world datasets prepared according to a single reference naming convention and containing complete metadata and document titles.

A second limitation is that semantic validation was limited to discipline and document type. This was sufficient to demonstrate the feasibility of the approach, but it does not cover the full semantic scope of the Optimal Standard. A third limitation concerns dependency on lookup-table quality: if reference tables are incomplete, outdated, or inconsistent, the validator may classify valid names as non-compliant or fail to identify actual naming errors. A fourth limitation is that the method validates file names and titles, but not the full internal content of documents. Therefore, a document could still contain content that is inconsistent with both its file name and its title.

Despite these limitations, the study shows that file name compliance can be formalised as a measurable, staged validation problem and that the proposed LoNC workflow can be consistently operationalised under controlled validation conditions. The proposed LoNC framework provides a basis for distinguishing between different levels of naming quality and for supporting more transparent document control in BIM/CDE environments.

5. Conclusions

This study proposed the Level of Naming Compliance (LoNC), a multi-level framework for assessing file naming compliance in BIM/CDE environments. The framework treats a file name as a structured information container and distinguishes between syntactic compliance, referential compliance, and selected semantic consistency.

The proposed method combines regular-expression-based syntactic validation, lookup-table-based referential validation, and LLM-supported semantic validation comparing selected decoded file name fields with descriptive document titles. In the present implementation, semantic validation was limited to discipline and document type.

The framework was motivated by naming inconsistencies observed in a real-world CDE project, while the quantitative validation was performed on a controlled synthetic dataset of 500 file-title pairs. The validator achieved full agreement with the predefined expected LoNC levels in the controlled dataset. Because the LoNC classes were unequally represented, the result was also assessed using balanced accuracy and class-specific measures, which confirmed that agreement was obtained across all predefined LoNC levels. This indicates the internal consistency of the implemented workflow under controlled conditions.

The main contribution of the study is the formalisation of file name compliance as a staged and measurable assessment problem. The proposed approach enables identification not only of whether a file name is non-compliant, but also of the stage at which non-compliance occurs. This can support CDE managers, BIM managers, and document controllers in diagnosing naming problems, correcting reference data, improving naming procedures, and generating compliance reports.

Future work should validate the method on larger real-world datasets, extend semantic validation beyond discipline and document type, integrate richer project-specific context, and investigate implementation within CDE platforms as an automated QA/QC module. In the longer term, the syntactic, referential, and semantic validation functions defined in LoNC could also support more automated QA/QC workflows, including agentic AI-based document control scenarios.

Supplementary Materials: The following supporting information can be downloaded at: <https://www.mdpi.com/article/10.3390/buildings16142886/s1>, The controlled synthetic validation dataset of 500 file-title pairs is available as Supplementary File S1. An example lookup table is available as Supplementary File S2.

Author Contributions: Conceptualisation, J.P. and K.T.; methodology, J.P.; software, M.K.; validation, T.O.; formal analysis, T.O.; investigation, J.P.; resources, J.P.; data curation, J.P.; writing—original draft preparation, J.P.; writing—review and editing, K.T., T.O. and M.K.; visualisation, M.K.; supervision, T.O.; project administration, K.T.; funding acquisition, T.O. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the Europe's Rail Flagship Project 4—Sustainable and Green Rail Systems, co-funded by the Horizon Europe Framework Programme under grant agreement No. 101101917 and by the Ministry of Education and Science in Poland under grant agreement No. MEiN 5337/HE/2023/2. The FP4—Rail4EARTH project under Grant Agreement No 101101917 is supported by the Europe's Rail Joint Undertaking and its members. Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or the Europe's Rail Joint Undertaking. Neither the European Union nor the granting authority can be held responsible for them. The work was also supported by AGH University of Krakow under research subsidy No. 16.16.150.545.

Data Availability Statement: The controlled synthetic validation dataset generated for this study is available in the Supplementary Materials. The original Czeremcha project documentation is not publicly available due to project confidentiality and was used only as a motivating case for identifying

naming inconsistency types. The source code of the proof-of-concept LoNC validator developed for this study is publicly available in a GitHub repository: [<https://github.com/Kaszniam/LoNC-Validator>] (accessed on 20 May 2026).

Acknowledgments: The authors would like to thank the project stakeholders whose documentation management challenges motivated the development of the proposed framework.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

AEC	Architecture, Engineering and Construction
AI	Artificial Intelligence
BEP	BIM Execution Plan
BIM	Building Information Modelling/Building Information Management
bSDD	buildingSMART Data Dictionary
BCF	BIM Collaboration Format
CDE	Common Data Environment
EIR	Exchange Information Requirements
IFC	Industry Foundation Classes
IDS	Information Delivery Specification
LLM	Large Language Model
LoNC	Level of Naming Compliance
NLP	Natural Language Processing
OS	Optimal Standard
QA/QC	Quality Assurance/Quality Control
RAG	Retrieval-Augmented Generation

Appendix A. Pseudocode of the LoNC Validation Procedure

INPUT:

REQUIRED:

FILE NAME // string, 8 parts separated by “-”

DOCUMENT_TITLE // descriptive title used for semantic validation

OPTIONAL:

RULES_XLSX // spreadsheet containing project-specific lookup tables:

// [project number, volume, part, discipline,

// author/originator, document type/form, phase/stage, number]

SETTINGS_JSON // contains LLM configuration and semantic validation context

OUTPUT:

LoNC_LEVEL // 0, 1, 2, or 3

VALIDATION_STATUS // “VALID” or “INVALID”

REGEX_ERRORS[] // syntactic validation errors

VALUE_ERRORS[] // lookup-table validation errors

SEMANTIC_RESULT // semantic consistency result for discipline and document type

DIAGNOSTIC_MESSAGE // explanation of detected non-compliance

PROCESS:

BEGIN

// 1. Load input data

File name = GET_USER_INPUT()

documentTitle = GET_DOCUMENT_TITLE()

rulesMap = LOAD_XLSX_IF_EXISTS("rules.xlsx")

settings = LOAD_JSON_IF_EXISTS("settings.json")

// 2. Split and parse the file name

parts = SPLIT(file name, "-")

IF LENGTH(parts) != 8:

LoNC_LEVEL = 0

VALIDATION_STATUS = "INVALID"

REGEX_ERRORS.ADD("File name must contain 8 fields")

RETURN OUTPUT

fieldMap = MAP(

"project_number" -> parts[0],

"volume" -> parts[1],

"part" -> parts[2],

"discipline" -> parts[3],

"author" -> parts[4],

"document_type" -> parts[5],

"phase" -> parts[6],

"number" -> parts[7]

)

// 3. Syntactic validation

FOR EACH (fieldName, fieldValue) IN fieldMap:

IF NOT MATCHES_REGEX(fieldValue, REGEX[fieldName]):

REGEX_ERRORS.ADD(fieldName)

IF REGEX_ERRORS IS NOT EMPTY:

LoNC_LEVEL = 0

VALIDATION_STATUS = "INVALID"

RETURN OUTPUT

ELSE:

LoNC_LEVEL = 1

// 4. Referential validation using lookup tables

FOR EACH (fieldName, fieldValue) IN fieldMap:

IF fieldName EXISTS IN rulesMap:

IF fieldValue NOT IN rulesMap[fieldName]:

VALUE_ERRORS.ADD(fieldName)

```

        IF VALUE_ERRORS IS NOT EMPTY:
LoNC_LEVEL = 1
VALIDATION_STATUS = "INVALID"
RETURN OUTPUT
ELSE:
LoNC_LEVEL = 2

// 5. LLM-supported semantic validation
prompt = BUILD_PROMPT(
file name,
fieldMap["discipline"],
DECODE(fieldMap["discipline"], rulesMap),
fieldMap["document_type"],
DECODE(fieldMap["document_type"], rulesMap),
documentTitle
)

semanticResponse = CALL_LLM(settings, prompt)
SEMANTIC_RESULT = PARSE_JSON_RESPONSE(semanticResponse)

IF SEMANTIC_RESULT.semantic_consistency == "consistent":
LoNC_LEVEL = 3
VALIDATION_STATUS = "VALID"
ELSE:
LoNC_LEVEL = 2
VALIDATION_STATUS = "INVALID"

RETURN OUTPUT

END

```

References

1. Mischke, J.; Stokvis, K.; Vermeltfoort, K.; Biemans, B. Delivering on Construction Is No Longer Optional. 2024. Available online: <https://www.mckinsey.com/capabilities/operations/our-insights/delivering-on-construction-productivity-is-no-longer-optional#/> (accessed on 10 February 2026).
2. Waqar, A.; Othman, I.; Hayat, S.; Radu, D.; Khan, M.B.; Galatanu, T.F.; Almujiabah, H.R.; Hadzima-Nyarko, M.; Benjeddou, O. Building Information Modeling—Empowering Construction Projects with End-to-End Life Cycle Management. *Buildings* **2023**, *13*, 2041. [CrossRef]
3. Souza Valadares, F.; Cristina Souza Moura, N.; Nakagomi Fernandes Pereira, T.; de Oliveira Arantes, M. Identification of the Main Traditional Project Management Methods Through a Systematic Literature Review. *Int. J. Adv. Comput. Sci. Appl.* **2024**, *15*, 952. [CrossRef]
4. Ganah, A.; Lea, G. A Global Analysis of Bim Standards across the Globe: A Critical Review. *J. Proj. Manag. Pract.* **2021**, *1*, 52–60. [CrossRef]
5. ISO 19650-2:2018; Organization and Digitization of Information about Buildings and Civil Engineering Works, Including Building Information Modelling (BIM)—Information Management Using Building Information Modelling—Part 2: Delivery Phase of the Asse. ISO: Geneva, Switzerland, 2018. Available online: <https://www.iso.org/standard/68080.html> (accessed on 3 November 2025).
6. ISO 19650-1:2018; Organization and Digitization of Information about Buildings and Civil Engineering Works, Including Building Information Modelling (BIM)—Information Management Using Building Information Modelling—Part 1: Concepts and Principles. ISO: Geneva, Switzerland, 2018. Available online: <https://www.iso.org/standard/68078.html> (accessed on 3 November 2025).
7. Jaskula, K.; Papadonikolaki, E.; Rovas, D. Comparison of Current Common Data Environment Tools in the Construction Industry. In Proceedings of the European Conference on Computing in Construction, Crete, Greece, 10–12 July 2023.

8. Dolla, T.; Venkatachalam, S.; Delhi, V.S.K. Institutional Shaping of Cde Implementation in Bim-Enabled Aec Projects. *J. Inf. Technol. Constr.* **2024**, *29*, 826–849. [[CrossRef](#)]
9. Preidel, C.; Borrmann, A.; Mattern, H.; König, M.; Schapke, S.E. Common Data Environment. In *Building Information Modeling: Technology Foundations and Industry Practice*; Springer International Publishing: Berlin/Heidelberg, Germany, 2018; pp. 279–291.
10. Preidel, C.; Borrmann, A.; Oberender, C.; Tretheway, M. Seamless Integration of Common Data Environment Access into BIM Authoring Applications: The BIM Integration Framework. In *Proceedings of the eWork and eBusiness in Architecture, Engineering and Construction—Proceedings of the 11th European Conference on Product and Process Modelling, ECPPM 2016*; CRC Press: Boca Raton, FL, USA, 2016.
11. Seyis, S.; Özkan, S. Analyzing the Added Value of Common Data Environments for Organizational and Project Performance of Bim-Based Projects. *J. Inf. Technol. Constr.* **2024**, *29*, 247–263. [[CrossRef](#)]
12. Tsay, G.S.; Staub-French, S.; Poirier, E.; Zadeh, P.; Pottinger, R. BIM for FM: Understanding Information Quality Issues in Terms of Compliance with Owner’s Building Information Modeling Requirements. *Front. Built Environ.* **2023**, *9*, 1117066. [[CrossRef](#)]
13. Cann, S.; Mahamadu, A.M.; Prabhakaran, A.; Dziekonski, K.; Joseph, R. An Approach for Semi-Automated Data Quality Assurance within BIM Models. *Eng. Manag. Prod. Serv.* **2022**, *14*, 114–125. [[CrossRef](#)]
14. Kovacs, A.T.; Micsik, A. BIM Quality Control Based on Requirement Linked Data. *Int. J. Archit. Comput.* **2021**, *19*, 431–448. [[CrossRef](#)]
15. Nuyts, E.; Bonduel, M.; Verstraeten, R. Comparative Analysis of Approaches for Automated Compliance Checking of Construction Data. *Adv. Eng. Inform.* **2024**, *60*, 102443. [[CrossRef](#)]
16. Uniclass, Delivered by NBS. Available online: <https://uniclass.thenbs.com/> (accessed on 3 November 2025).
17. Kładź, M.; Borkowski, A.S. IDS Standard and BSDD Service as Tools for Automating Information Exchange and Verification in Projects Implemented in the BIM Methodology. *Buildings* **2025**, *15*, 378. [[CrossRef](#)]
18. buildingSMART International. Information Delivery Specification (IDS). Available online: <https://www.buildingsmart.org/standards/bsi-standards/information-delivery-specification-ids/#documentation> (accessed on 20 May 2026).
19. Onatayo, D.; Onososen, A.; Oyediran, A.O.; Oyediran, H.; Arowoiya, V.; Onatayo, E. Generative AI Applications in Architecture, Engineering, and Construction: Trends, Implications for Practice, Education & Imperatives for Upskilling—A Review. *Architecture* **2024**, *4*, 877–902. [[CrossRef](#)]
20. Hadi, M.U.; Qureshi, R.; Shah, A.; Irfan, M.; Zafar, A.; Shaikh, M.B.; Akhtar, N.; Wu, J.; Mirjalili, S. A Survey on Large Language Models: Applications, Challenges, Limitations, and Practical Usage. *Authorea Prepr.* **2023**, *1*. Available online: <https://www.techrxiv.org/doi/pdf/10.36227/techrxiv.23589741.v1?download=true&redirectToLatest=false> (accessed on 10 May 2026).
21. Raiaan, M.A.K.; Mukta, M.S.H.; Fatema, K.; Fahad, N.M.; Sakib, S.; Mim, M.M.J.; Ahmad, J.; Ali, M.E.; Azam, S. A Review on Large Language Models: Architectures, Applications, Taxonomies, Open Issues and Challenges. *IEEE Access* **2024**, *12*, 26839–26874. [[CrossRef](#)]
22. Patil, R.; Gudivada, V. A Review of Current Trends, Techniques, and Challenges in Large Language Models (LLMs). *Appl. Sci.* **2024**, *14*, 2074. [[CrossRef](#)]
23. Madireddy, S.; Gao, L.; Din, Z.U.; Kim, K.; Senouci, A.; Han, Z.; Zhang, Y. Large Language Model-Driven Code Compliance Checking in Building Information Modeling. *Electronics* **2025**, *14*, 2146. [[CrossRef](#)]
24. Beach, T.; Yeung, J.; Nisbet, N.; Rezgui, Y. Digital Approaches to Construction Compliance Checking: Validating the Suitability of an Ecosystem Approach to Compliance Checking. *Adv. Eng. Inform.* **2024**, *59*, 102288. [[CrossRef](#)]
25. Chen, N.; Lin, X.; Jiang, H.; An, Y. Automated Building Information Modeling Compliance Check through a Large Language Model Combined with Deep Learning and Ontology. *Buildings* **2024**, *14*, 1983. [[CrossRef](#)]
26. Nithyanantham, B.K.; Sesterhenn, T.; Nedungadi, A.; Garijo, S.P.; Zenkner, J.; Bartelt, C.; Lüdtke, S. MCP4IFC: IFC-Based Building Design Using Large Language Models. *arXiv* **2025**, arXiv:2511.05533.
27. Pasalski, J.; Owerko, T.; Kasznia, D.; Wrzosek, M.; Rochel, M.; Czerw, K. Efficient File-Naming Convention for Data Exchange in Construction Industry. *Geomat. Environ. Eng.* **2025**, *19*, 99–129. [[CrossRef](#)]
28. Chen, K.; Lu, W.; Wang, H.; Niu, Y.; Huang, G.G. Naming Objects in BIM: A Convention and a Semiautomatic Approach. *J. Constr. Eng. Manag.* **2017**, *143*, 06017001. [[CrossRef](#)]
29. Lou, J.; Lu, W.; Xu, J.; Li, X.; Wang, J. Nomenclature for Offsite Construction. *Build. Res. Inf.* **2022**, *50*, 894–908. [[CrossRef](#)]
30. Abualdenien, J.; Borrmann, A. Levels of Detail, Development, Definition, and Information Need: A Critical Literature Review. *J. Inf. Technol. Constr.* **2022**, *27*, 363–392. [[CrossRef](#)]
31. Clemen, C.; Görne, H. Level of Georeferencing (LoGeoRef) Using IFC for BIM. *J. Geod. Cartogr. Cadastre* **2019**, *10*, 15–20.
32. Kumar, V.; Teo, A.L.E. Development of a Rule-Based System to Enhance the Data Consistency and Usability of COBie Datasheets. *J. Comput. Des. Eng.* **2021**, *8*, 343–361. [[CrossRef](#)]
33. Friedl, J.E.F. *Mastering Regular Expressions*, 3rd ed.; O’Reilly Media, Inc.: Sebastopol, CA, USA, 2006.
34. Yarberry, W. RegEx Summary. In *CRAN Recipes*; Apress: Berkeley, CA, USA, 2021.

35. Câmpeanu, C.; Salomaa, K.; Yu, S. A Formal Study of Practical Regular Expressions. *Int. J. Found. Comput. Sci.* **2003**, *14*, 1007–1018. [[CrossRef](#)]
36. Gao, Y.; Xiong, Y.; Gao, X.; Jia, K.; Pan, J.; Bi, Y.; Dai, Y.; Sun, J.; Wang, M.; Wang, H. Retrieval-Augmented Generation for Large Language Models: A Survey. *arXiv* **2024**, arXiv:2312.10997.
37. Cheng, M.; Luo, Y.; Ouyang, J.; Liu, Q.; Liu, H.; Li, L.; Yu, S.; Zhang, B.; Cao, J.; Ma, J.; et al. A Survey on Knowledge-Oriented Retrieval-Augmented Generation. *arXiv* **2025**, arXiv:2503.10677.
38. Fan, W.; Ding, Y.; Ning, L.; Wang, S.; Li, H.; Yin, D.; Chua, T.-S.; Li, Q. A Survey on RAG Meeting LLMs: Towards Retrieval-Augmented Large Language Models. *arXiv* **2024**, arXiv:2405.06211.
39. Miao, J.; Thongprayoon, C.; Suppadungsuk, S.; Garcia Valencia, O.A.; Cheungpasitporn, W. Integrating Retrieval-Augmented Generation with Large Language Models in Nephrology: Advancing Practical Applications. *Medicina* **2024**, *60*, 445. [[CrossRef](#)] [[PubMed](#)]
40. Rasool, Z.; Barnett, S.; Willie, D.; Kurniawan, S.; Balugo, S.; Thudumu, S.; Abdelrazek, M. LLMS for Test Input Generation for Semantic Applications. In *Proceedings of the Proceedings—2024 IEEE/ACM 3rd International Conference on AI Engineering—Software Engineering for AI, CAIN 2024*; Association for Computing Machinery, Inc.: New York, NY, USA, 2024; pp. 160–165.
41. Meguellati, E.; Zeghina, A.; Sadiq, S.; Demartini, G. LLM-Based Semantic Augmentation for Harmful Content Detection. *arXiv* **2025**, arXiv:2504.15548.
42. Ghimire, P.; Kim, K.; Acharya, M. Opportunities and Challenges of Generative AI in Construction Industry: Focusing on Adoption of Text-Based Models. *Buildings* **2024**, *14*, 220. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.