# Digital Framework for Bridge Parametric Design: The BIM Library approach

Auckland University of Technology

Auckland, New Zealand 2024

Iurii Surovtsev

# Contents

1 Introduction 5

2 Benefits and Barriers to BIM Implementation 6

3 Governmental BIM Mandate 6

4 BIM Libraries 7

5 Analysis of Existing Practices 7
- 5.1 NBS 7
- 5.2 INNOVance Project 9
- 5.3 KiwiRail's Digital Engineering 10
- 5.4 Deutsche Bahn Digital Library 12

6 Open-Source Model 13

7 Novelty 14

8 Methodology 14

9 The Bridge Decomposition 16

10 Driven Pile 16

11 Driven Pile Reinforcement 20

12 Pile Group 23

13 3D Space Adjustments 25

14 Pile Cap 26

15 Pile Cap Reinforcement 27

16 Column 30

17 Column Reinforcement 33

18 Pier Cap 35

19 Pier Cap Reinforcement 36

20 Bridge Pier 38

21 Results 39

22 Further Research 40

23 Discussions 40

24 Conclusion 41

**Abstract**

This article explores the impact of Building Information Modelling (BIM) in bridge construction. It emphasises BIM's capacity to enhance design, construction, and management efficiency in infrastructure projects, notably bridges. Despite the evident advantages of BIM, the construction industry exhibits a general reluctance towards its widespread adoption. A significant gap is identified in developing data schemas for civil infrastructure, including bridges, which necessitates a more detailed data schema and a dedicated framework for bridge engineering. The article reviews the benefits and barriers to BIM implementation, highlighting the role of government regulations as a potential catalyst for adoption. It also discusses the concept of a National BIM Object Library as a solution to promote collaboration and efficiency in building projects. The paper proposes an approach for establishing a BIM object library for bridge design, focusing on geometrical data and element decomposition. This includes an examination of existing practices, as well as NBS requirements, to develop guidelines for BIM object templates. The proposed framework aims to standardise geometrical and semantic data sets for Bridge design, fostering a more sustainable, efficient, and innovative future in civil engineering.

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person (except where explicitly defined in the acknowledgements), nor material which to a substantial extent has been submitted for the award of any other degree or diploma of a university or other institution of higher learning.

Iurii Surovtsev

26/03/2024

# 1 Introduction

Building Information Modelling (BIM) has emerged as a revolutionary technology in the evolving landscape of civil engineering and construction, offering unprecedented capabilities in infrastructure projects' design, construction, and management. Among these, bridge construction stands out as a field greatly benefiting from the advanced features of BIM [1]. BIM integration streamlines the design and building processes for infrastructure projects and bridge design and significantly enhances project efficiency, cost-effectiveness, and stakeholder collaboration [2].

However, the general construction industry must still be more willing to make this change. For instance, according to the Deloitte Global Leadership Study [3], the Construction industry ranks last in terms of percentage of technological budgets, with numbers twice as low as the average.

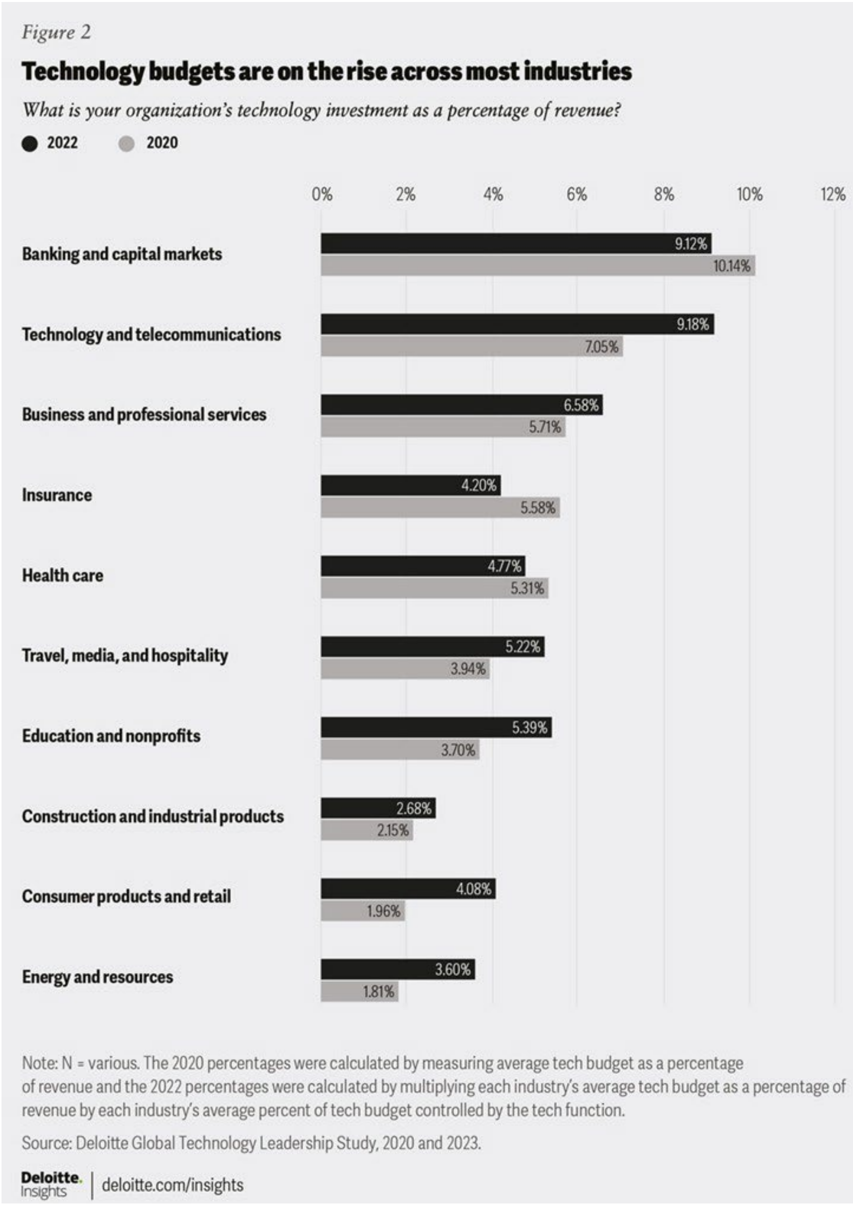



Figure 1: Technology investment as a percentage of revenue [3]

Several factors contribute to that hesitation to invest in new technologies, such as limited short-term financial gain [4], an overwhelming number of different solutions available on the market making it hard to approve a final decision [5] and waiting for the tech world to catch up and offer a ready-to-use solution simply. This lack of technological progress is especially notable in infrastructure and bridge design. Compared to the building industry, there are many gaps in data schema development for civil infrastructure, including bridges, roads, and tunnels. The current schemas need more geometric and semantic information, so a more detailed data schema model for bridges is required. [6] [7] It is also clear that a dedicated framework tailored for bridge engineering will also have to be developed to increase collaboration and productivity throughout the bridge lifecycle.

Overall, infrastructure, specifically bridge design, presents unique challenges [8], making progress in that sector slightly slower. Recent discussions at the IABSE Symposium Prague 2022 mentioned two obstacles to fully implementing BIM in Bridge design: availability of 3D models [9] and BIM decomposability. [10]

Parametric design, a facet of BIM, offers a promising solution to these challenges as well as introducing a new dimension of flexibility and efficiency in bridge construction. Engineers and designers can easily manipulate and modify design parameters by leveraging parametric modelling and adapting to diverse construction challenges and requirements [11]. This approach allows for the creation of more complex and aesthetically pleasing bridge designs and ensures that these structures are both functional and sustainable. Adopting BIM in bridge construction is a technical upgrade and a paradigm shift in how infrastructure projects are conceptualised, executed, and maintained. BIM's impact on bridge construction is multifaceted, from facilitating improved collaboration among multidisciplinary teams to enabling a more efficient utilisation of resources. Moreover, the technology's ability to integrate with other cutting-edge technologies like Augmented Reality (AR) and Digital Twins opens up new avenues for innovation in bridge design and maintenance. [12] [13]

This paper delves into BIM's transformational role in bridge construction, highlighting its benefits, applications, and future trajectory. It also suggests a possible approach to establish a foundation for bridge parametric design in the form of a BIM object library.

# 2 Benefits and Barriers to BIM Implementation

Building Information Modeling (BIM) offers several benefits in its implementation. It is considered a business change program rather than just an IT project, providing strategic outcomes and value propositions for asset owners. [14] BIM enhances cost estimation and control, efficient planning and management, and improves design and project quality. [15] It serves as a collaboration tool opening possibility for simultaneous work by multiple teams, a better element alignment, visualization and seamless data extraction and exchange, all of which are vital for the success of complex projects.
[16] [17] It can offer an extensive overview of the environmental data during the design phase, allowing for a robust analysis of different factors in search of an optimal engineering solution. [18] As the technology is based on the creation of multi-dimensional models, it facilitates a better understanding of the design of the construction a better understanding of project conditions. It thus increases decision-making power and simultaneously improves feedback from different departments involved in the design process. [19]

During the construction stage, its collaborative nature allows it to work much more efficiently, avoiding errors leading to potential demolition and repair operations. [18] It improves safety, enabling risk detection implementation and resource management. [19]

At the same time, significant barriers to BIM implementation include stakeholders' resistance to change, inadequate organisational support, and the absence of industry standards. [15] Issues related to the high software cost, training expenses, and technical complexities often hinder the full implementation of BIM. [20] Smaller companies also note that they don't feel BIM is applicable for smaller domestic projects and their lack of in-house expertise as obstacles. Another problem arises with smaller clients also not being interested in BIM for their projects and the smaller the company, the less likely they are to push for BIM usage. [21]

Countries establishing a BIM mandate somewhat negate this, as companies do not have to develop their own implementation plans, thus not facing high costs and unique challenges.

# 3 Governmental BIM Mandate

All public sector construction projects in the UK must comply with BIM Level 2 requirements. [22] In Ireland, in 2017, 85% of the industry sample, an increase from 66% in 2015, believed it is necessary to follow the UK example by establishing a governmental BIM usage mandate. [23] A more thorough review of BIM mandates in different parts of the world could be found in [24], while this paper will focus a little bit more on New Zealand.

In 2015, a study about obstacles to BIM adoption in New Zealand found that a lack of industry standards and government reluctance to intervene was a major problem. [25] A more recent study identified the absence of a government mandate as a significant barrier to BIM adoption. [26]

There is substantial evidence that government regulations can drive BIM adoption and influence its diffusion in the construction industry. [27]

At the same time, the establishment of a governmental mandate in New Zealand appears imminent. The 2022 Action Plan for the Digital Strategy for Aotearoa outlined a vision of a world with a paper-based approach becoming obsolete. [28] In addition, an Infrastructure Action Plan published in May 2023 lists BIM adoption as one of the core actions to be undertaken soon. [29]

To effectively prepare for the upcoming BIM mandate, addressing existing challenges and establishing robust frameworks is crucial. The NZ BIM Handbook [30] emphasizes the need for Good Modelling Practice and the creation of Model Objects, suggesting the establishment of a National BIM Object library similar to the one in the UK. This initiative is expected to meet the demand for a BIM object library in both the UK and New Zealand.

A 2018 study [31] identified several critical success factors for implementing BIM-like technologies. These include collaboration, early and accurate 3D visualization, improved coordination, and information exchange. Likewise, a UNECE 2021 report [32] on European BIM implementation strongly suggested further improvements in data interoperability, automatic code validation, and cost estimation. These aspects need to be developed, sometimes independently, for the technology to start taking root.

# 4 BIM Libraries

A concept like the National BIM Object Library is potentially one of the best solutions to work on these aspects, as the evidence suggests that BIM libraries promote collaboration among stakeholders and improve efficiency in building projects by providing easily accessible, high-quality BIM objects. [33] A study in 2017 in Italy [34] has shown evidence that a standardised and software-independent method for managing construction information (INNOVance BIM library, which takes much inspiration from NBSE UK) leads to efficient information transmission and collection.

The National Building Information Model Standard–United States (NBIMSUS) has a similar concept. [35] Likewise, Swiss Federal Railways is working intensively to exploit BIM's potential. [36] Their goal is lifecycle-based data management for all assets and systems in the railway infrastructure by 2025. To do this, interfaces will be created to exchange standardised data across national borders. At the same time, existing processes would have to be checked and renewed if necessary.

This plan is not straightforward. One of the major challenges in developing BIM libraries [37] is establishing operational standards to ensure the sharing of quality information across the construction process.

Overall, the infrastructure industry significantly demands schemas, workflows, and library templates for implementing parametric design. This paper will offer an approach based on NBSE requirements for creating BIM library elements for bridge design, mainly focusing on geometrical data and element decomposition.

# 5 Analysis of Existing Practices

## 5.1 NBS

Initiated in 2011, the NBS National BIM Library serves as the free-to-access, premier resource of BIM content for the construction sector. Esteemed as one of the leading BIM libraries, its acclaim stems from the meticulous adherence of all its objects to the globally recognised NBS BIM Object Standard. This adherence ensures that users can confidently select and employ BIM objects, be assured of their compatibility across various platforms, be correct in format, and complete necessary information. [38]

The NBS BIM Library adheres to rigorous standards for defining BIM objects, ensuring quality, coherence, and consistency across the construction process. This standardisation facilitates information sharing among all professionals involved in a building project, which is crucial for seamlessly integrating various phases of the construction lifecycle. The experience of the British National BIM Library, under the umbrella of the NBS, is particularly noted for its contribution to the development of operational standards that enhance the utility and interoperability of BIM libraries. [37]

The NBS Library features detailed descriptions, geometrical and technological attributes, and, most importantly, a section for BIM models with full attributes and 3D geometry. Elements added to the library have visual information such as photos or sometimes videos enclosed to the page,

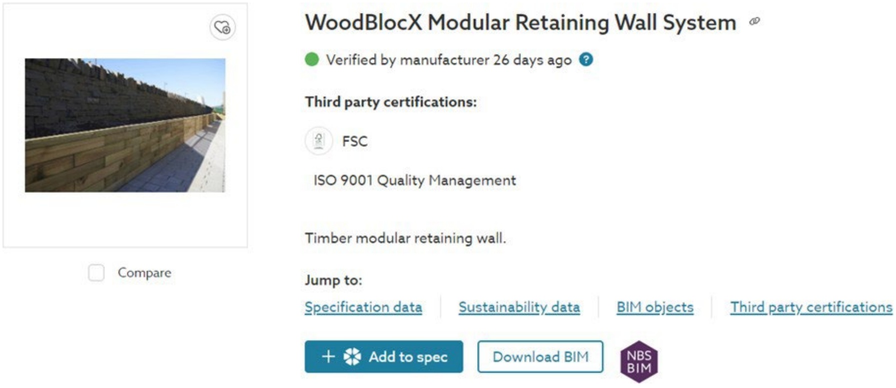



Figure 2: NBS BIM library screenshot, main screen of the element

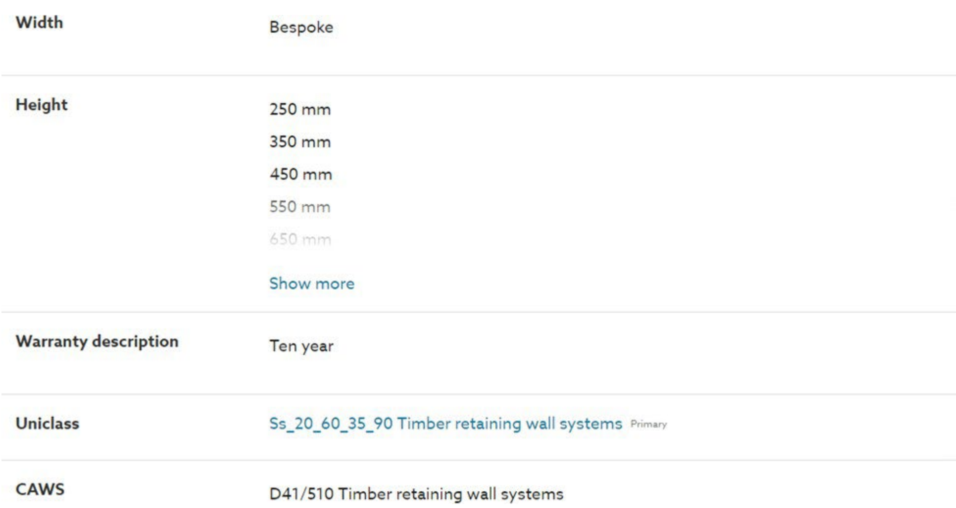
standardised specification data, physical and engineering attributes applicable to the element, and sustainability data covering the origin of materials and if and how they can be recycled. A part of the page is dedicated to various certifications, so they are always available and at hand when browsing the library. While there are no object schemes or dimensions immediately on the page available, a full 3D BIM model both in IFC and Revit format is included for download, which allows the generation of all necessary schemes and sections, calculation of material quantities or pricing estimation. At the end of the page, in the literature section catalogues and technical data sheets can be found along with case studies about this particular element. Being one of the oldest programs and introducing rigorous and very detailed standards for BIM objects, NBS must always be considered one of the strongest examples of BIM libraries and one of the very few accessible ones.

**Description**

Timber modular retaining wall.

WoodBlocX is a Modular Timber system. All components are manufactured in the Highlands of Scotland and distributed across the UK & EU. WoodBlocX components can be used to create a number of landscaping structures such as raised planters, retaining walls and seating. The system offers complete design flexibility and our company offers a free design service for any scale of project.

Figure 3: NBS BIM library screenshot, description



| | |
|---|---|
| Width | Bespoke |
| Height | 250 mm<br>350 mm<br>450 mm<br>550 mm<br>650 mm<br>Show more |
| Warranty description | Ten year |
| Uniclass | Ss_20_60_35_90 Timber retaining wall systems Primary |
| CAWS | D41/510 Timber retaining wall systems |

Figure 4: NBS BIM library screenshot, attributes

## 5.2 INNOVance Project

The INNOVance project aims to offer comprehensive solutions for construction work information management. This initiative included developing a classification system, an information structure, a database, and BIM objects for construction products. The designed platform aims to function as a BIM library and a Common Data Environment (CDE) for sharing project information and pro- moting standardised data exchange and information management within the construction sector. [37]

In INNOVance, one should be able to locate a wide range of information ranging from the object model to installation videos, dimensional tolerances, specifics on control activities, preparation requirements for support structures, details regarding the typical team involved, operator licensing prerequisites including mandatory safety gear, packaging specifications, waste generation quantities, and disposal protocols. Additionally, it aims to provide insights into installation timelines, scheduled maintenance intervals, plant locations, product compatibility, room dimensions adhering to industry standards, cost breakdowns from production to project accounting, volumetric data, floor space requirements, urbanisation charges, crane load capacities and reaches, excavator power specifications, and compatible accessories such as buckets, compactors, and callipers, along with their respective attributes.

INNOVance is based on Italian standards of UNI 11337 "Building and engineering work—Coding criteria of works, activities and resources—Identification, description and interoperability [39], which aims to set up a very robust and detailed system for LOD classifications, geometrical and non-geometrical attributes, different attribute classes, and reviewing and borrowing from other international standards. [37]

A prototype of the library [40] promises that every element will have a vast set of data attached to it, including but not limited to geometry and shape of the element, dimensions, main components, safety information, synonyms, uses and labour and many more. At the time of the publication, the most recent update about the project was from 31st December 2019, and no public access to the library or to the prototype could be found. Its potential is huge, and if the project delivers on its promises, it will definitely serve as a fantastic example for the foundation of similar libraries.

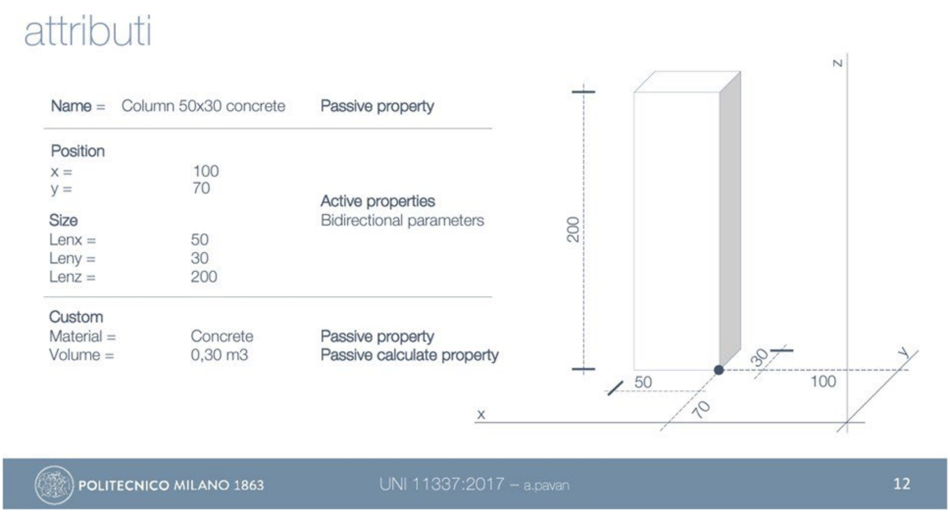



Figure 5: UNI 11337:2017, attributes

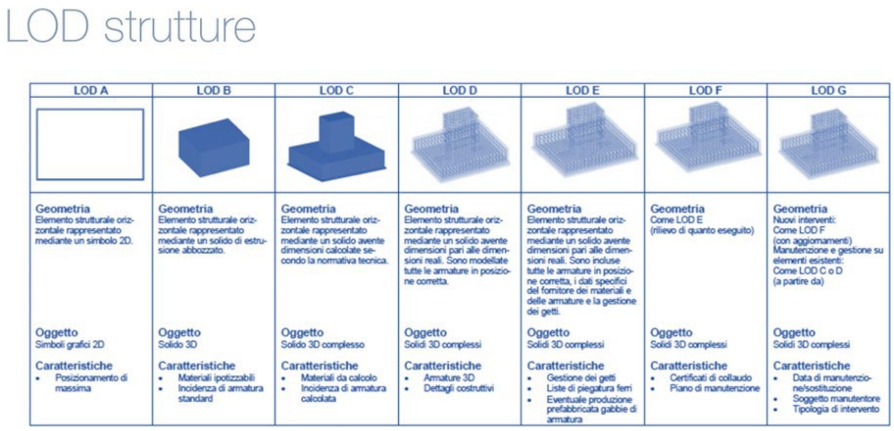



Figure 6: UNI 11337:2017, LOD levels

## 5.3 KiwiRail's Digital Engineering

KiwiRail's transformational journey into digital engineering marks a significant leap in modernising New Zealand's rail network. This initiative pivoted around adopting Building Information Modelling (BIM) and integrating collaborative digital tools. For instance, as KiwiRail's first BIM- enabled venture, the Trentham to Upper Hutt project tackled the complexities of live construction sites, emphasising partnerships and advanced technology infrastructure. [41] Similarly, the PACE project, part of the Wellington Metro Upgrade, showcased the shift from traditional 2D to advanced 3D design processes, underlining the role of collaborative environments in enhancing digital capabilities. [42]

The Wiri to Quay Park project further exemplified this shift by integrating Revizto, a pivotal coordination tool, to streamline complex rail infrastructure projects. These initiatives collectively led to substantial improvements across various facets of construction and design. Notably, the Trentham to Upper Hutt project achieved an 88% reduction in individual requests for information and a 55% reduction in forecasted contract costs, demonstrating the efficacy of digital tools in reducing construction risks and enhancing productivity. [41] Beyond physical infrastructure, these projects have created rich digital assets, providing a foundational platform for future enhancements in project management, maintenance, and upgrades. [43] The broader impact of KiwiRail's digital engineering approach has set new industry benchmarks, positioning New Zealand's rail network at the forefront of technological innovation in infrastructure development.

Expanding upon KiwiRail's Digital Engineering initiative, the KiwiRail Asset Data Dictionary [44] is a comprehensive guide detailing essential information for managing vertical and horizontal assets. This document meticulously categorises various asset types alongside their specific attribution requirements, aligning with KiwiRail's objectives in asset management. It acts as a crucial tool in standardising and streamlining the process of asset data collection and utilisation, which is essential for the adequate upkeep and development of KiwiRail's infrastructure.

It should be mentioned that KiwiRail also has its own Github repository [45], but no public projects are available at the time of publication.

To serve as a BIM Object library foundation, a KiwiRail Asset Definition Viewer [46] seems a strong candidate. Asset viewer provides a simple tree-like structure connecting related elements logically. Assets contain several technical attributes, allowing seamless integration into future projects and convenient buttons for file export. For most assets, it references the corresponding code in Uniclass classification (if a corresponding element can be found). This is crucial for a collaborative environment, allowing software and machines to find relevant data in BIM. [47] It is necessary to clarify that Assed Definition Viewer is not a BIM library and wasn't designed as such but rather serves as a visualisation of what the industry already gravitates towards, even without an established BIM mandate and without an ability to rely on an established national BIM library. However, given its collaborative and free-access nature, KiwiRail Digital Engineering Initiative and Asset Viewer could serve as a strong foundation for a New Zealand BIM object library. If expanded upon with geometric and additional semantic data, they could be a valuable addition to the BIM object library.

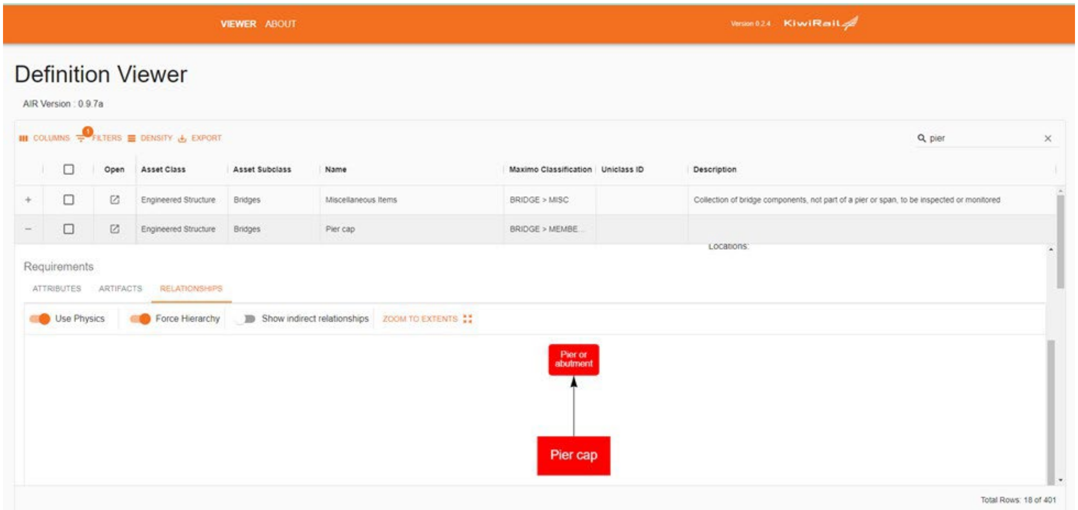



Figure 7: KiwiRail Asset Definition Viewer screenshot, main screen

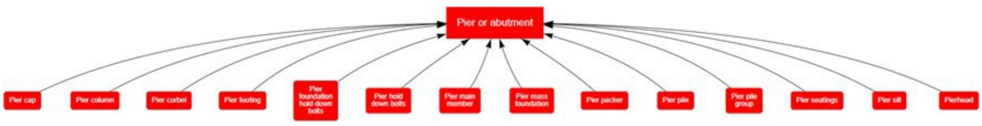



Figure 8: KiwiRail Asset Definition Viewer screenshot, structure

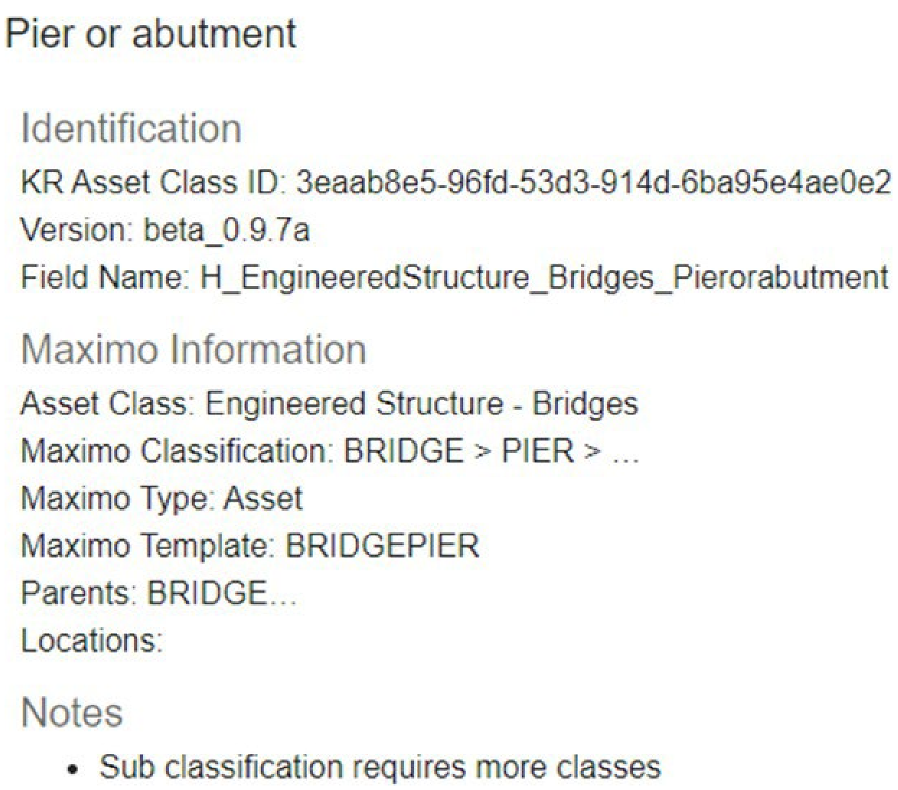



Figure 9: KiwiRail Asset Definition Viewer screenshot, attributes

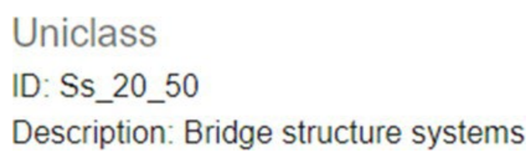



Figure 10: KiwiRail Asset Definition Viewer screenshot, UNICLASS code

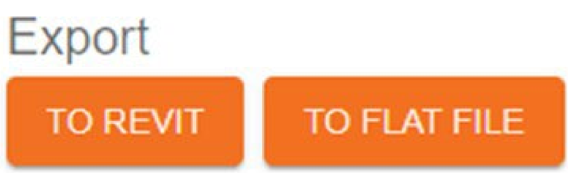



Figure 11: KiwiRail Asset Definition Viewer screenshot, export buttons

## 5.4 Deutsche Bahn Digital Library

Deutsche Bahn in Germany is another rail giant that should be considered a strong example of parametric design implementation. Deutsche Bahn's integration of Building Information Modelling (BIM) methodology, highlighted by the implementation of a digital component library and construction standards, represents a significant development in construction project management. These standards are constantly improved upon and updated, with the latest version (3.0) submitted on November 15, 2023.

Deutsche Bahn's approach illustrates a shift towards digitalisation in construction practices. Providing contractors with digital component libraries, project templates, and example models facilitates standardised project execution. The critical documentation offers detailed guidelines on the BIM methodology's objectives, processes, and expected outcomes. This indicates a strategic move to ensure consistency and accuracy in construction projects, aligning with digital transformation goals. Deutsche Bahn AG's guidelines for technical design [48] in railways have been analysed and recommended as a baseline for code validation in BIM models.

The company has made available a document titled "Anlage A - Digitale Bauteilbibliothek und Baustandards" under the section "Vorgaben zur Anwendung der BIM-Methodik" (Guidelines for the Application of the BIM Methodology). This document contains detailed instructions for project processing using the BIM methodology, outlining objectives, use cases, processes, and expected outcomes. Such documentation aids in maintaining consistency and quality in project execution. [49] [50] [51]

The document titled "Anlage A - Digitale Bauteilbibliothek und Baustandards" [52] contains comprehensive guidelines for project processing using the Building Information Modeling (BIM) methodology. It is a crucial document for all construction activities at DB passenger stations and is essential for large projects handled by DB Station and Service AG. The document outlines BIM project objectives, use cases, processes, and expected outcomes. It also documents any changes made within the components, including modifying, adding, or deleting attributes relevant to planning and execution. Notably, the document has been updated to include new or revised components such as coordination bodies, various types of benches, elevators including shafts, and DSA+ poles with attachments. This document also references the relevant standard drawings and performance texts for each component or system type.

The document in detail contains a large list of different elements, their technical attributes, descriptions, and links to corresponding IFC and Revit files (which seem to be inaccessible to the general public but can most probably be found in the DB internal database).

It is a very solid document demonstrating principles of potential data organisation, its illustration with an isometric view of a referenced object and data storage in the scope of the largest railway company in the world.

This paper will combine NBSE guidelines and requirements with some of these examples to suggest how a future BIM Object Library could be established.


Figure 12: Digitale Bauteilbibliothek und Baustandards, sample element

# 6 Open-Source Model

While specialised companies are developing their personal libraries, it also seems sensible to con- sider the success of open-source development approach, as exemplified by platforms like GitHub. Open-source development projects are characterised by their collaborative nature, transparency, and community-driven approach to creating software. Platforms like GitHub have revolutionised software development, enabling developers worldwide to share, contribute to, and fork projects, thus fostering innovation and rapid problem-solving.

Similar to open-source software projects, a BIM library developed on open-source principles can benefit from the collective expertise and innovation of a global community of architects, engineers, and construction professionals. This collaborative approach accelerates the development and refinement of BIM objects, ensuring they meet the industry's evolving needs. [53] [54]

Adopting an open-source approach for BIM libraries can significantly reduce costs for users and contributors. By making BIM objects freely available, the model enables professionals and firms of all sizes to access high-quality resources without the financial burden associated with proprietary solutions. [55] [56]

The transparent nature of open-source projects means that a wide community continuously reviews and improves BIM objects in the library. This not only enhances the quality of the models but also ensures that security and performance issues are promptly addressed, leading to more reliable and accurate BIM objects. [53] [56]

Open-source BIM libraries offer unparalleled flexibility, allowing users to modify and adapt BIM objects to fit the specific requirements of their projects. This level of customisation is crucial for addressing the unique challenges and constraints of different construction projects. [55] [56]

Building a BIM library on open-source principles encourages the development of a supportive community and ecosystem. This ecosystem contributes to the library, offers support, shares knowledge, and provides feedback, fostering a culture of continuous learning and improvement.

The open-source model ensures BIM libraries' long-term availability and sustainability. By avoiding vendor lock-in and ensuring that anyone can access and use the library, the model supports the enduring growth and accessibility of resources.

The open-source model promotes a collaborative, innovative, cost-effective approach to developing BIM libraries. It aligns with the needs of the architecture, engineering, and construction

(AEC) industries for flexible, high-quality, and accessible BIM resources, fostering a culture of sharing and continuous improvement.

Following these ideas, this paper will propose a simple foundation for an open-source style BIM library that could be gradually improved and expanded by a community of engineers.

# 7 Novelty

Despite BIM's acknowledged benefits in enhancing design, construction, and management efficiency across various infrastructure projects, a significant gap has been identified in the availability of a comprehensive data schema tailored specifically for civil infrastructure, particularly bridges. This paper proposes a foundation for a schema that addresses this gap, facilitating improved collaboration, efficiency, and productivity throughout the lifecycle of bridge projects. By leveraging parametric modelling, this paper aims to introduce a new dimension of flexibility and efficiency in bridge construction. This approach allows for the easy manipulation and modification of design parameters, adapting to diverse construction challenges and requirements specific to bridge design.

# 8 Methodology

To develop guidelines for creating BIM object templates, it is necessary to analyse current best modelling practices in the Industry. The NZ BIM Handbook in its Appendix A: Modelling and Documentation Practice [30] suggests establishing an internal standard for modelling in the scope of the project, referring to AIQS and NZIQS Australia and New Zealand BIM Best Practice Guidelines (2018). [57] For the purposes of BIM object library, it would be possible to focus on geometry and location recommendations generally; these are the following: Ensuring the model is adequate to its Level of Detail (LOD); Objects/elements should be modelled as they will be constructed; Naming should be consistent; Reference exact same coordinates for X, Y, Z, base point;

A comprehensive guide to object naming identification designed specifically for Bridges can be found published in September 2023 by Queensland Department of Transport and Main Roads [58]. It is worth noting that this mostly applies to the actual project naming scheme, and the paper will not focus on following these guidelines, as the decomposition of elements for Library purposes doesn't follow the same structure. This document also suggests a number of attributes that should be present in the information section about the element, such as its width, length, depth, minimum cover, etc. Another solid set of recommendations about the element attributes could be found in the Guidelines for Using Building Information Modelling in Statuary Plan Submissions (other than General Building Plan) by The Government of the Hong Kong Special Administrative Region Building Department and Construction Industry Council. [59] This document features a set of 2D annotation requirements, and this paper will rely on these suggestions to provide il- lustrations for each element. Following these recommendations, it's clear that a BIM library is expected to establish the element's correct geometry, including the details about the object's reinforcement. Material quantity calculations are also generally expected to be present in the model, so recommendations on obtaining these are to be included.

| Item | Geometry Settings | Geometrical information | Non-geometrical information |
|---|---|---|---|
| Precast concrete plank for structural slab | • Parametric object indexed/categorised as 'Structural Slab'<br>• Top of slab should be modelled to the top level of the precast plank<br>• Thickness of parametric object should be the thickness of the precast plank<br>• Rebar should be modelled with all required details for fabrication[4] | • Thickness<br>• Rebar size/ shape/spacing/ concrete cover | • Object mark<br>• Concrete grade<br>• Concrete density<br>• Rebar material grade<br>• Material specification |
| Structural beam (concrete) | • Parametric object indexed/categorised as 'Structural Framing'<br>• Structural beam should be modelled to the full structural size of its width and depth<br>• Rebar should be modelled with all required details for statutory plan submission[4] | • Width<br>• Depth<br>• Additional information should be provided to define the geometry (e.g. distance to change of depth)<br>• Rebar size/ shape/spacing/ concrete cover/ shear link<br>• Tendon number/ profile/type, if applicable | • Object mark<br>• Concrete grade<br>• Concrete density<br>• Rebar material grade<br>• Material specification<br>• Tendon number/ profile/type, specification, prestress force if applicable |
| Structural beam (steel) | • Parametric object indexed/categorised as 'Structural Framing'<br>• Structural beam should be modelled to the full structural size of the width, depth and thickness of flange/web | • Width<br>• Depth<br>• Additional information should be provided to define the geometry (e.g. thickness of flange/web and opening size on web if any) | • Object mark<br>• Type mark<br>• Steel grade<br>• Steel density<br>• Section Physical Properties (e.g. second moment of area, radius of gyration etc.)<br>• Material specification |

Figure 13: Data Driven Object Requirements [59]

The most comprehensive object modelling guidelines are still represented in NBS requirements. In the next sections this paper will develop guidelines for creating BIM object templates generally following and referencing these guidelines. Additionally, each element's attributes and geometrical data were gathered from real world construction examples as well as from current Bridge Design standards with a focus on the current AUS and NZ codes. Where applicable, a comparison with other international standards is given, as well as references to more generalist Bridge Design literature, including books and structural engineering dictionaries. A suggestion to a graphical representation of each element will be provided, following the examples in Analysis of Existing Practices. The use of Python classes supplemented with Python code snippets should provide additional opportunities to use that approach, serving as another illustration of an approach.

Overall, the paper aims to provide a foundation for future collaborative work on establishing a more extensive library of elements.

# 9 The Bridge Decomposition

While bridge design usually starts from the top to bottom, [60]: analysing the road or railway requirements on this road or railway section, with one of the significant steps defining the general span scheme of the bridge, for classification purposes, it would be easier to stand from the bottom, gradually increasing the complexity of each element.

The general bridge structure in most technical textbooks can be an excellent example of element decomposition. The typical bridge is generally split into superstructure and substructure; sometimes, the foundation is considered another major part. [61] The substructure comprises several piers and abutments, and a pier could be further divided into a pier body and a pier foundation. The foundation consists of a pile group and a pile cap, and the pile group combines several singular piles.

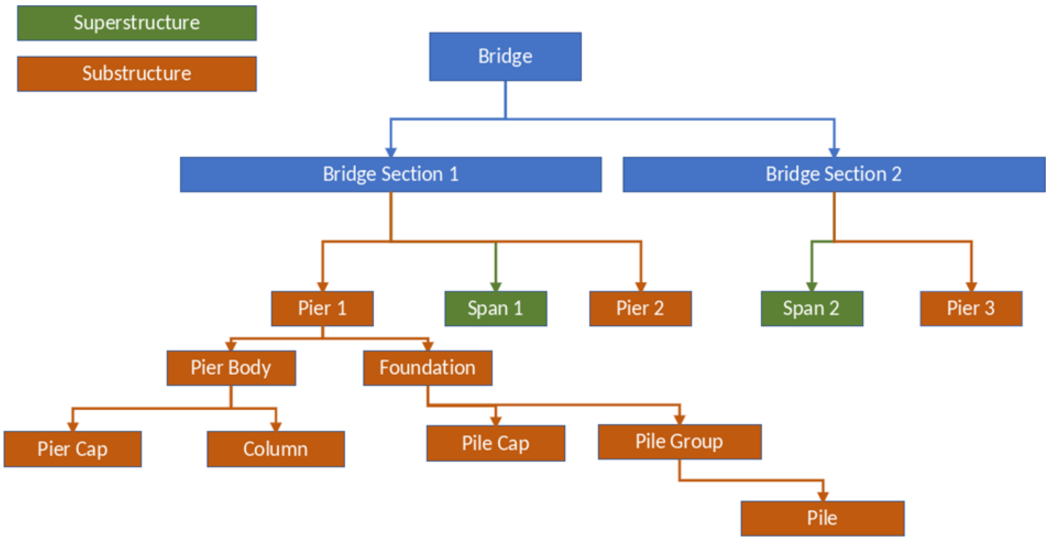



Figure 14: Typical Bridge Structure

With the general bridge structure outlined above, for this experiment, the first element would be a driven pile, an element of a pier foundation, and a part of a pile system.

# 10 Driven Pile

Focusing on the NBS BIM Object standard requirements, the first definitions will be of Geometrical data. Exerts from the requirements will be *italicised* to better differentiate from the core text of the paper.

*3.1.1 Modelling scale*

*The BIM object shall have geometry produced at the scale 1:1.*

We will comply with this requirement by ensuring an object's geometry is correct and all geometrical data is in mm. Depending on the project goals and requirements, a certain level of abstraction should be implemented according to the LOD and LOI definitions. [62] However, this will not be explored in the scope of this paper.

*3.1.2 Insertion point*

*The BIM object shall include an insertion point that is suitable for its intended use.*

This requirement is similar to the one presented in the Open BIM standard. In Open BIM, the point should logically characterise the placement of the element in the model. [63] Generally, for the driven pile 2 insertion points could be considered: the tip of the pile (or pile toe) [64], which is convenient to use with geotechnical calculations, or the intersection of centrelines on the top of the pile, which can be used for geometrical connection to the pile cap. While this is primarily technical, the top of the pile seems to be a better choice, as it can be used as a visual reference for the real object on the construction site after installation and should be a reference point for as-built documentation. [65] [66] The final definition would be: "A centre of symmetry on the top face of a driven pile."

*3.1.3 Parametric function*

*The BIM object may, where supported by the BIM platform and where appropriate:*

1. *Have parametric geometry that is locked and aligned to appropriate reference elements such as planes, lines, levels and points,*

2. *Include dimensions and labels that are constrained to reference planes.*

Driven piles usually should be aligned with the vertical Z axis. In addition, in some cases, they could be driven at an angle or, as it is more widely adopted for technical drawings, with an incline to the Z axis. [65] This incline or inclination should be added as a parametric function, but not in the Driven Pile classification; more appropriately, it would be added to the layer above, to the Pile System.

Another parametric function is a cut-off level, but with a Pile being largely a manufactured element, it would be a better solution to analyse the cut-off level in the Pile System. [65]

*3.1.4 Modelling units*

*The BIM object shall use metric geometry with units of millimetres, unless the local construction industry has (without dispute) retained an alternative unit of measurement*

While the standard unit of measurement for the Si system is 1 meter (1m) [67], the classification system will use mm, which is used far more often in the construction industry and technical documentation.

*3.1.5 Actual thickness*

*Layered BIM objects shall represent the actual thickness of a layer unless unsupported by the BIM platform, in which case the minimum thickness supported by the BIM platform shall be used.*

The driven pile will have a full 3D model with the correct dimensions. It is a solid object with reinforcement bars embedded into it. Suggestions for modelling reinforcement will be given separately.

*3.2.1 Geometrical extent*

*The geometrical detail of an object shall be appropriate for its intended use and informational purpose. The geometrical detail of an object shall represent the extent of the object and its connectivity. Objects can be represented by a 3D bounding box to show BIM Object Standard V2.1/March 2019 23 location, size and spatial relationship in the model, but preferably be geometrically sufficient to recognize the object and allocated space without containing excessive geometrical detail.*

This paper will define a full 3D model with complete and correct dimensions. Again, in the future development of BIM object libraries, having different geometrical representations of an element is generally a good idea to limit the hardware load. [68]

*3.2.2 Dimensions*

*Generic objects shall include nominal or expected dimensions where actual dimensions are unknown. Manufacturer objects shall include accurate overall dimensions and any further dimensions necessary for the object to fulfil its intended purpose.*

An important step is to define what object dimensions are and what geometrical data they represent. This data, combined with the insertion point, will be used for future clash detection and code validation in complex queries for model analysis. [69]

According to [65], the longitudinal or lateral dimension, or the general length of a driven pile, is usually called Pile Length. [64] Refers to the same dimension as Pile Height. While an inevitable confusion might occur in the case of rectangular section piles [70], it is worth noting that these piles are relatively rare. For general purposes, Pile Length should serve as the primary name, mainly because this dimension is referenced by this definition in the standard [71]. Pile Height could be added as an alias.

As driven piles are usually square or round in section, the other dimension, medial or transversal, should be Pile Width or Pile Diameter for circular-shaped sections. Other common names for that dimension include Pile Side or Pile Size. Generally, there is little confusion with the names of dimensions, but it is worth establishing proper terminology.

The geometrical configuration of the driven pile's tip (or toe) is required for correct 3D geometry. It is not an unimportant element, as its shape and dimensions may heavily influence the penetration resistance. [72] But it's important to remember that these serve as a parametric function of the pile dimensions, as it is usually the manufacturer who determines them.

*3.3.1 Essential geometry The BIM object shall include:*

1. *Geometric representation of the space defined by the construction product's external boundary.*

2. *Geometry with a defined purpose.*

3. *Essential openings and geometric details, from which meaningful information can be gained.*

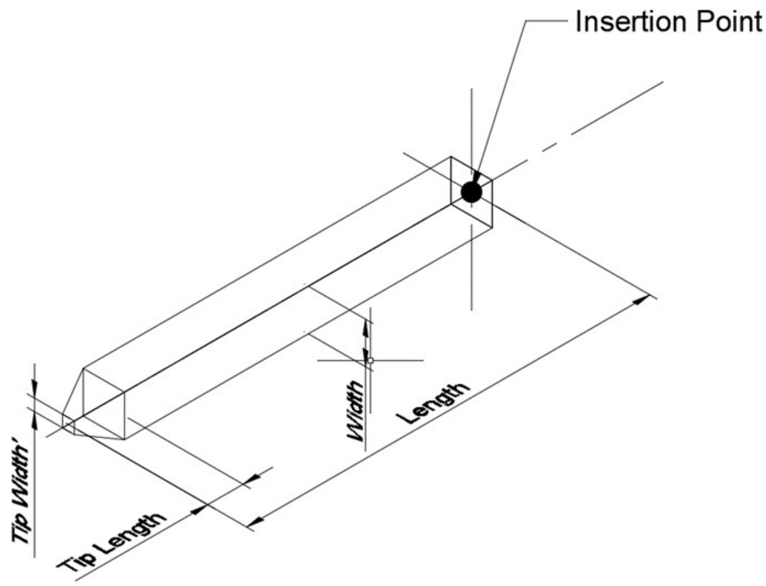



Figure 15: Driven Pile Scheme

The whole geometry of the driven pile is its external boundary. Reinforcement will be discussed later in this section.

In the scope of this paper, it will be impossible to analyse the full set of requirements in the NBS standard and apply these to each element. The paper will focus only on parts of the NBS standard, setting up a strong foundation for future polish.

*1.2 Object Type 1.2.1 Identification The BIM object type shall be identifiable within the associated BIM authoring system and assigned using the appropriate IfcElementType and PredefinedType from the BuildingSMART International IFC4 (Add2) schema (ISO 16739). If an appropriate type does not exist, the following shall be used: a) IfcBuildingElementProxyType for the IfcElementType.*

*b) 'USERDEFINED', in upper case, for the PredefinedType.*

While most of the bridge elements will be missing from the IFC4 schema, the driven pile can be identified as 7.8.3.3 IfcPile [73].

*2.2.2 Product variants The BIM object can represent product variants using a property with a value comprising an alphanumeric or numerical single value, list value, range value, enumerated value and reference value or bounded value. a) The BIM object property can be assigned a single value where a value has a single selection. The value shall be predetermined and completed where the value is available and known. b) The BIM object property can be assigned a list value where several unique values of the same type are given in an ordered list, the order of which is significant, e.g. 200, 400, 600, 800. c) The BIM object property can be assigned a range value where a value has an upper and lower limit (bound). The lowest bound shall be presented first, followed by the highest bound. Where the range uses positive and negative signs, the numbers are separated using 'to'; for all other situations, use a hyphen. If the value is not given, it indicates an open bound, e.g. OverallWidth 0.9–1.25 m.*

To set up an example, we will focus on the parametrisation of 350x350 and 450x450 piles, with OverallLength set as 3-72 m. The NBS requirements do not specify a requirement to list an increment for values, but it would be logical to include the OverallLengthIncrement at 1000 mm. [74] Considering that we provide all dimensions in mm, values could be presented in a table.

Or as an algorithmic choice:

It is also good practice to provide formulas to calculate values for economic analysis, though it could be expected that the manufacturer would provide most values for factory-made driven piles. A possible approach to these calculations could be found later in this section.

*4.1.1 Object behaviour The BIM object shall behave in an appropriate manner that reflects its relationship with associated objects within the BIM platform.*

A Driven Pile is a part of the Pile Group element. One possible way to demonstrate how this classification will work is to set up a Python class and use visual programming in Dynamo to produce 3D geometry. The following snippet of code will demonstrate the Python class approach.

| Width (Size) | Length (Height) | Tip Length | Tip Width |
|---|---|---|---|
| 350 | 3000 | 300 | 100 |
| 350 | 4000 | 300 | 100 |
| 350 | ... (1000 increment) | 300 | 100 |
| 350 | 72000 | 300 | 100 |
| 450 | 3000 | 300 | 100 |
| 450 | 4000 | 300 | 100 |
| 450 | ... (1000 increment) | 300 | 100 |
| 450 | 72000 | 300 | 100 |

Table 1: Driven Pile attributes

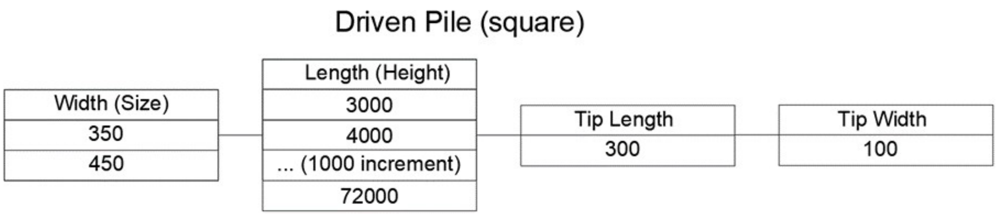



Figure 16: Driven Pile Atrributes

```
class DrivenPile:
    def __init__(self, length, width, tip_length, tip_width, ifc):
        self.width = width
        self.length = length
        self.tip_width = tip_width
        self.tip_length = tip_length
        self.concrete_volume = length * width * width + (1 / 3) *
(tip_width**2+width**2+tip_width*width**0.5)
        self.ifc=ifc
```

Certain functions could also be added to this class, such as calculating point coordinates and assigning vertices.

```
def coordinates(self):
        self.coordinates = []
        a = 1
        b = 1
        for i in range(12):
            if i < 4:
                pt = [a * self.width, b * self.width, 0]
                self.coordinates.append(pt)
            elif i < 8:
                pt =[a * self.width, b * self.width, self.length-self.tip_length]
                self.coordinates.append(pt)
            else:
                pt = [a * self.tip_width, b * self.tip_width, self.length]
                self.coordinates.append(pt)
            if a==1 and b==1:
                b = b*(-1)
            elif a==1 and b==-1:
                a = a*(-1)
            elif a==-1 and b==-1:
                b = b*(-1)
            elif a==-1 and b == 1:
                a = 1
                b = 1
```

```
        return self.coordinates
    def vertices(self):
        self.vertices = []
        layers=deque([[0, 1, 2, 3],[4, 5, 6, 7],[8, 9, 10, 11]])
        self.vertices.append(layers[0])
        for j in range(2):
            for i in range(4):
                self.vertices.append([layers[j][i]]+[layers[j][i-1]]+[layers[j+1][i-
1]]+[layers[j+1][i]])
        self.vertices.append(layers[2])
        return self.vertices
```

It should be noted that it is unnecessary to improvise complicated Python algorithms for these tasks. Calculating the coordinates in graphical software like AutoCAD might be easier. In that case, a much simpler approach could work with snippets of code replaced as:

```
def coordinates(self)
self.coordinates = [[self.width, self.width, 0], [self.width, -self.width, 0], [-
self.width, -self.width, 0], [-self.width, self.width, 0], [self.width, self.width,
self.length-self.tip_length], [self.width, -self.width, self.length-self.tip_length],
[-self.width, -self.width, self.length-self.tip_length], [-self.width, self.width,
self.length-self.tip_length], [self.tip_width, self.tip_width, self.length],
[self.tip_width, -self.tip_width, self.length], [-self.tip_width, -self.tip_width,
self.length], [-self.tip_width, self.tip_width, self.length]]
return self.coordinates

def vertices(self)
self.vertices = [[0, 1, 2, 3], [0, 3, 7, 4], [1, 0, 4, 5], [2, 1, 5, 6], [3, 2,
6, 7], [4, 7, 11, 8], [5, 4, 8, 9], [6, 5, 9, 10], [7, 6, 10, 11], [8, 9, 10,
11]]
return self.vertices
```

For the pile 12000 mm in length with a 350x350 mm section, coordinates can be entered as numbers. This approach would create every pile type as a separate class.

```
def coordinates(self)
self.coordinates = [[350, 350, 0], [350, -350, 0], [-350, -350, 0], [-350, 350, 0],
[350, 350, 11700], [350, -350, 11700], [-350, -350, 11700], [-350, 350, 11700], [100,
100, 12000], [100, -100, 12000], [-100, -100, 12000], [-100, 100, 12000]]
return self.coordinatesz
```

# 11 Driven Pile Reinforcement

Generally driven piles are precast fabricated elements, and reinforcement is rarely an engineering input. [75] [76] For illustration purposes, we will use one of the most common forms of reinforcement: several lateral bars of primary reinforcement and continuous ties of lateral reinforcement.
[76] [71]

The primary reinforcement is usually formed by 4 or 8 rods (or strands). They are positioned behind the continuous tie (or link throughout), so to correctly define their geometric position, both lateral and main reinforcement parameters need to be specified as input. We can add a new class that will rely on our DrivenPile class:

```
class DrivenPileReinforcement():
    def __init__(self, driven_pile, main_bar_numb, main_bar_diam, cont_tie_inc,
cont_tie_diam):
        self.driven_pile = driven_pile
        self.main_bar_numb = main_bar_numb
        self.main_bar_diam = main_bar_diam
        self.cont_tie_inc = cont_tie_inc
        self.cont_tie_diam = cont_tie_diam
```

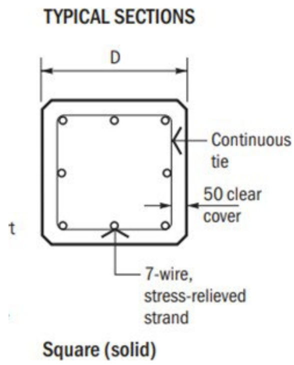



Figure 17: Driven Pile reinforcement [76]

SECTION PROPERTIES

| Section | Size, D (mm) | Area, A ($mm^2$) | Moment of inertia, I ($mm^4$) | Radius of gyration, r (mm) | Section modulus, Z ($mm^3$) | Mass (kN/m) | Perimeter (mm) | Strand No./size | Allowable concentric imposed action (kN)* for concrete strength (MPa) 32 | 40 | 50 | 60 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Square (solid) | 350 | 122 500 | $1250 \times 10^6$ | 101 | $7146 \times 10^3$ | 3.00 | 1400 | 8/12.7 | 1055 | 1375 | 1780 | 2185 |
| | 400 | 160 000 | $2133 \times 10^6$ | 116 | $10\,666 \times 10^3$ | 3.92 | 1600 | 8/12.7 | 1450 | 1870 | 2400 | 2930 |
| | 450 | 202 500 | $3417 \times 10^6$ | 130 | $15\,187 \times 10^3$ | 4.97 | 1800 | 10/12.7 | 1840 | 2375 | 3040 | 3710 |
| | 600 | 360 000 | $10\,800 \times 10^6$ | 173 | $36\,000 \times 10^3$ | 8.83 | 2400 | 16/12.7 | 3220 | 4275 | 5460 | 6650 |
| Octagonal (solid) | 400 | 132 622 | $1403 \times 10^6$ | 103 | $7015 \times 10^3$ | 3.25 | 1328 | 8/12.7 | 1160 | 1510 | 1950 | 2385 |
| | 450 | 167 652 | $2242 \times 10^6$ | 116 | $9964 \times 10^3$ | 4.11 | 1488 | 8/12.7 | 1530 | 1970 | 2525 | 3080 |
| | 550 | 250 660 | $5011 \times 10^6$ | 141 | $18\,223 \times 10^3$ | 6.15 | 1824 | 12/12.7 | 2290 | 2950 | 3780 | 4605 |

* Allowable actions based on: $N = A_{pc}(0.33f'_c - 0.27f_{pe})$ where: $f_{pe} = 0.6P_u/A_{pc}$ (See **Chapter 6** for explanation of notations)

Figure 18: Driven Pile specifications [76]

Another function could be added to calculate a set of coordinates of primary reinforcement. Generally, the biggest problem for this task would be understanding the geometrical position of each point to define where in 3D space the inside of the pile would be and where it is outside so we can shift the coordinates of reinforcement accordingly. However, thanks to the fact that the pile is in local coordinates with [0, 0, 0] placed precisely in the centre of symmetry, shifting the coordinates for reinforcement becomes an effortless task, as the only required check is if the current coordinates are positive numbers or not.

```
def coordinates(self):
    if self.main_bar_numb == 4:
        a = 50 #clear cover in mm
        b = self.cont_tie_diam
        c = self.main_bar_diam/2
        pt = []
        coordinates = []
        for i in range(12):
            point = []
            for j in range(3):
                corner = self.driven_pile.coordinates[i]
                displ = a + b + c
                if j < 2:
                    if corner[j] <0:
                        coordinate = corner[j] + displ
                    else:
                        coordinate = corner[j] - displ
                    point.append(coordinate)
                else:
                    point.append(corner[j])
            pt.append(point)
        for i in range(4):
            coordinates.append([pt[i], pt[i+4], pt[i+8]])
    return(coordinates)
```

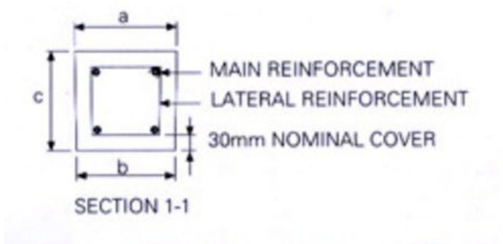


Figure 19: Driven Pile reinforcement [75]

**HUME CLASS 'A' REINFORCED CONCRETE SQUARE PILES - GRADE 60**

**Pile Dimensions and Details**

| Nominal Size | Dimensions | | | Main Reinforcement | | Lateral Reinforcement | | | | | | | Joint Plate Thickness |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | a | b | c | | | Wire Size | P1 | X | P2 | Y | P3 | Z[1] | |
| (mm) | (mm) | (mm) | (mm) | no. | Dia. (mm) | (mm) | (mm) | (mm) | (mm) | (mm) | (mm) | (mm) | (mm) |
| 150 x 150 | 153 | 147 | 150 | 4 | T 10 | 4.0 | 32 | 450 | 32-75 | 450 | 75 | 4200 | 6 |
| 175 x 175 | 178 | 172 | 175 | 4 | T 10 | 4.5 | 35 | 525 | 35-85 | 525 | 85 | 3900 | 6 |
| 200 x 200 | 204 | 196 | 200 | 4 | T 12 | 4.5 | 35 | 600 | 35-100 | 600 | 100 | 3600 | 8 |
| 235 x 235 | 239 | 231 | 235 | 8 | T 10 | 5.0 | 40 | 705 | 40-115 | 705 | 115 | 6180 | 9 |
| 250 x 250 | 254 | 246 | 250 | 4 | T 16 | 5.0 | 40 | 750 | 40-110 | 750 | 110 | 9000 | 9 |
| 275 x 275 | 279 | 271 | 275 | 4 | T 16 | 6.0 | 50 | 825 | 50-135 | 825 | 135 | 8700 | 9 |
| 300 x 300 | 305 | 295 | 300 | 8 | T 12 | 6.0 | 50 | 900 | 50-145 | 900 | 145 | 8400 | 12 |
| 320 x 320 | 325 | 315 | 320 | 4 | T 20 | 6.0 | 45 | 960 | 45-140 | 960 | 140 | 8160 | 12 |
| 350 x 350 | 355 | 345 | 350 | 4 | T 20 | 6.0 | 45 | 1050 | 45-130 | 1050 | 130 | 7800 | 12 |
| 381 x 381 | 386 | 376 | 381 | 4 | T 22 | 6.0 | 40 | 1145 | 40-120 | 1145 | 120 | 7420 | 12 |
| 400 x 400 | 405 | 395 | 400 | 8 | T 16 | 6.0 | 40 | 1200 | 40-115 | 1200 | 115 | 7200 | 12 |

[1] For maximum pile length only

Figure 20: Driven Pile specifications [75]

Calling this method will return a set of pairs of coordinates, which can be used to draw main reinforcement bars in specialised graphical software. For instance, for the same Pile that was used as an example before, a 12m long 350x350 precast concrete pile, the reinforcement coordinates will be the following:

```
[[[280.0, 280.0, 0], [280.0, 280.0, 11700]], [[280.0, -280.0, 0],
 [280.0, -280.0, 11700]], [[-280.0, -280.0, 0],
[-280.0, -280.0, 11700]], [[-280.0, 280.0, 0], [-280.0, 280.0, 11700]]]
```

Making a method for eight reinforcement strands will require finding coordinates for additional strands between the ones positioned in the corner. From the if self.main bar numb == 4: We will make a new snippet starting with elif self.main bar numb == 8: operator:

```
elif self.main_bar_numb ==8:
    a = 50  # clear cover in mm
    b = self.cont_tie_diam
    c = self.main_bar_diam / 2
    pt = []
    coordinates = []
    for i in range(24):
        point = []
        for j in range(3):
            displ = a + b + c
            if i % 2 == 0:
                corner = self.driven_pile.coordinates[i // 2]
                if j < 2:
                    if corner[j] < 0:
                        coordinate = corner[j] + displ
                    else:
                        coordinate = corner[j] - displ
```

```
                        point.append(coordinate)
                    else:
                        point.append(corner[j])
                else:
                    corner = self.driven_pile.coordinates[i // 2]
                    if (i+1) % 8 == 0:
                        next_corner = self.driven_pile.coordinates[(i - 7) // 2]
                    else:
                        next_corner = self.driven_pile.coordinates[(i + 2) // 2]
                    if j < 2:
                        coordinate = (corner[j]+next_corner[j])/2
                        if coordinate == 0:
                            pass
                        elif coordinate <0:
                            coordinate += displ
                        else:
                            coordinate -= displ
                        point.append(coordinate)
                    else:
                        point.append(corner[j])

        pt.append(point)
    for i in range(8):
        coordinates.append([pt[i], pt[i + 8], pt[i + 16]])
return(coordinates)
```

When calculating the arithmetic mean, it's important to keep in mind that points on numbers 7, 15, and 23 need to take the first point in a layer as the next corner. This could be done either by setting up different deques with layers or by checking if the next point is a multiple of the total number of points in the section (( i + 1 ) % 8 == 0 in this particular case). Moreover, the new

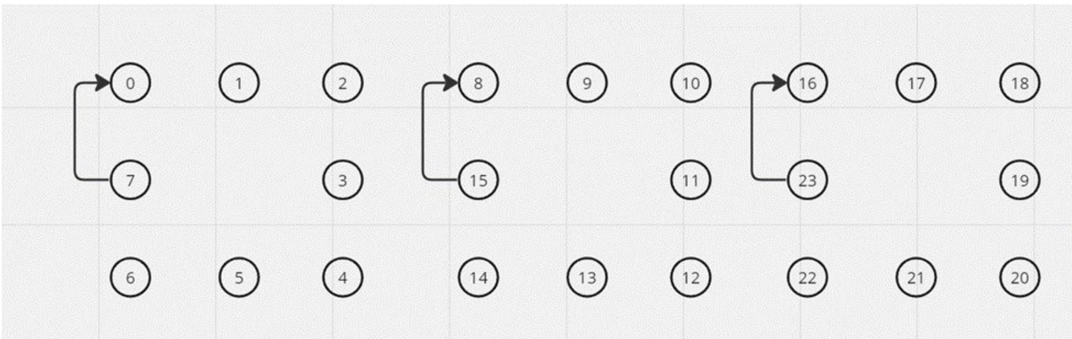



Figure 21: Points numeration

coordinates calculated this way will be collected in the following list:

```
[[[280.0, 280.0, 0], [280.0, 280.0, 11700], [30.0, 30.0, 12000]],
[[280.0, 0.0, 0], [280.0, 0.0, 11700], [30.0, 0.0, 12000]],
[[280.0, -280.0, 0], [280.0, -280.0, 11700], [30.0, -30.0, 12000]],
[[0.0, -280.0, 0], [0.0, -280.0, 11700], [0.0, -30.0, 12000]],
[[-280.0, -280.0, 0], [-280.0, -280.0, 11700], [-30.0, -30.0, 12000]],
[[-280.0, 0.0, 0], [-280.0, 0.0, 11700], [-30.0, 0.0, 12000]],
[[-280.0, 280.0, 0], [-280.0, 280.0, 11700], [-30.0, 30.0, 12000]],
[[0.0, 280.0, 0], [0.0, 280.0, 11700], [0.0, 30.0, 12000]]]
```

# 12 Pile Group

According to AS2159 [77], a Pile Group is simply a group of piles under a shared pile cap. From an engineering perspective, a Pile Group can be defined by four main parameters [78]: the Type of Pile used, the Number of Rows of piles in a Lateral direction along the centre line of the future or existing road, the Number of Rows of piles in Transversal direction, perpendicular to the centre line, and spacing between the lateral and transversal rows. Again, it is also essential to define which rows of piles are lateral and which are transversal, and the best reference line for that is the road centre line, as the bridge is always aligned to it.

In some cases, this approach cannot be used, and the coordinates of every Pile in the Pile Group should be considered separately. For instance, in the case of computer optimisation of asymmetric pile groups. [79] However, this is a very specific case, and for general purposes, a set of coordinates usually works poorly, as it is unwieldy as an engineering input.

Two additional parameters we discussed earlier can be applied to the Pile Group. [77] Pile Incline or Pile Row Incline should be introduced to the Pile Group. Technically, a cut-off level can also be applied to the Pile Group, but it should be noted that it is used only when the pier design is aligned with the actual terrain, so it might be better applied to the Bridge Pier. A Pile Group is also an element that can be used to demonstrate the addition of a Code Validation to the classification.

According to AS5100.3 [71], the spacing between piles centre-to-centre shall be not less than 2.5 times the diameter or nominal size of the pile or not less than two times the diameter or nominal size of the pile for end-bearing piles. While the second case is not very common, primarily encountered when piles transfer the load directly to a layer of rock, the first one will be implemented as an example of default code validation. [77] Additionally, this spacing should also be a default between the piles, as unless significant horizontal loads are expected, increasing the spacing beyond the requirement will increase the cost of the pier. [80]

Pile Group

| Rows Lateral | Rows Transversal | Spacing Lateral | Spacing Transversal | Pile Row Incline (for any row) |
|---|---|---|---|---|
| 1 | 1 | 100* | 100* | 2 : 1 |
| 2 | 2 | 110* | 110* | 3 : 1 |
| ... | ... | ... (increment 10) | ... (increment 10) | ... |
| N | N | N | N | 17 : 1 |

* - No less than 2.5 Pile Width or Pile Diameter (AS5100.3 - 6.5.2)

Figure 22: Pile Group attributes

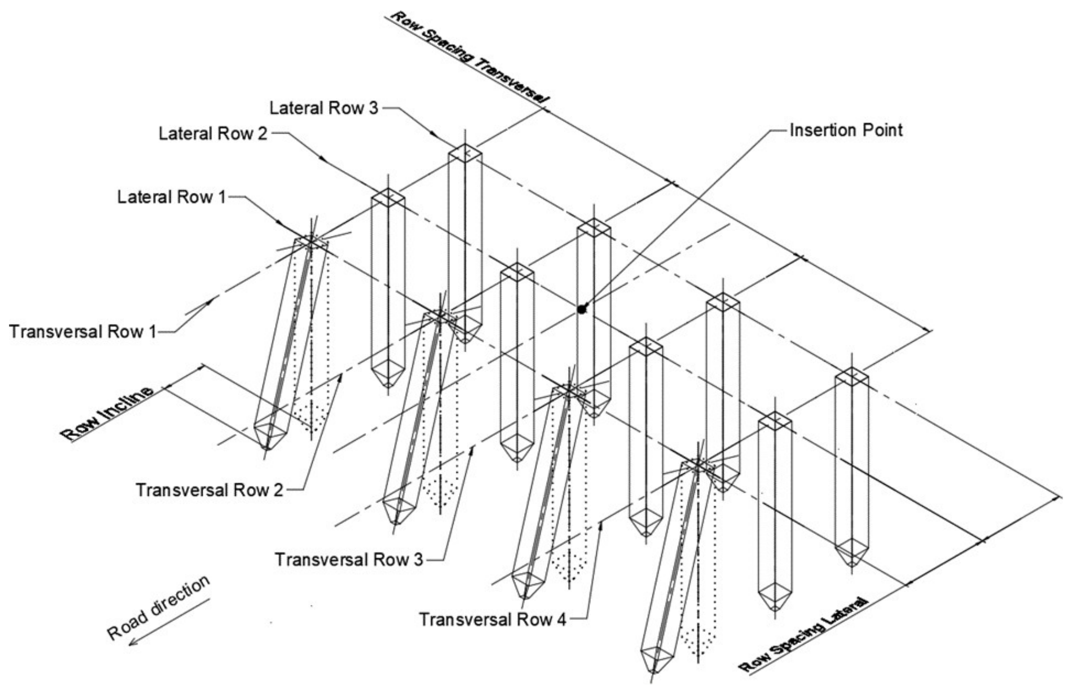



Figure 23: Pile Group scheme

Two more parametric functions can be calculated from the Pile Group definition: Pile Group Length and Pile Group Width. This will be measured between the centre lines of two lateral and transversal rows with the most significant distance between them. The Pile Group Length will be defined as the distance between the two lateral rows of piles located furthest from each other. The Pile Group Width is the measurement between the two transverse rows of piles located furthest from each other. These dimensions will become handy for the Pile Cap design.

```
class PileGroup:
    def __init__(self, driven_pile, rows_lateral, rows_transversal):
        self.rows_lateral = rows_lateral
        self.rows_transversal = rows_transversal
        self.driven_pile = driven_pile
        self.spacing_lateral = 2.5 * driven_pile.width
        self.spacing_transversal = 2.5 * driven_pile.width
        self.length = (rows_lateral-1) * self.spacing_lateral
        self.width = (rows_transversal-1) * self.spacing_transversal
```

Implementing a function to find coordinates also seems like a solid addition to this snippet.

```
def coordinates(self):
    self.coordinates = []
    a = (self.rows_lateral-1)/2
    b = (self.rows_transversal-1)/2
    for i in range(self.rows_lateral):
        for j in range(self.rows_transversal):
            pt = [-a * self.spacing_lateral+i*self.spacing_lateral, -b *
self.spacing_transversal+j*self.spacing_transversal, 0]
            self.coordinates.append(pt)
    return self.coordinates
```

## 13 3D Space Adjustments

It should be noted that working with geometry in pure Python is ill-advised. Let us demonstrate that taking Driven Pile and Pile Group as an example. Firstly, we may apply the base point as a separate parameter to our Python classes and then adjust the coordinates of every point. The following code snippet demonstrates the difference between two new classes: DrivenPileBP and PileGroupBP (as with added BasePoint). For a singular pile, there will be very little difference; we need to establish a base point variable in the class definition.

```
class DrivenPile_BP:
    def __init__(self, length, width, tip_length, tip_width, base_point, ifc):
        self.base_point = base_point
```

Before the last line of the coordinates function, insert a short loop to adjust the final coordinates.

```
for i in range(len(self.coordinates)):
    for j in range(3):
        self.coordinates[i][j] =  self.coordinates[i][j] + self.base_point[j]
return self.coordinates
```

Thus, if base point differs from [0, 0, 0], the final function will adjust the coordinates of every point of the chosen Pile, translating the geometry to another part of 3D space. While this seems simple, actual implementation would require additional code inputs. For a Pile Group, we first need to establish a DrivenPile or DrivenPileBP with [0, 0, 0] base point coordinates to calculate the spacing between the piles. That way, DrivenPile will be just a template we use in this project, and DrivenPileBP will represent a physical object.

```
Pile_1 = DrivenPile(12000, 350, 300, 100, 'IFC')
Pile_Group_1 = PileGroup(Pile_1, 2, 2)
```

Then, using the PileGroup that was established, it is possible to gradually add the new instances of DrivenPileBP to the new list, PilesPileGroup1, with their coordinates adjusted to the list of coordinates of centre points in the new PileGroup1:

```
Piles_PileGroup_1 = []
for i in range(len(Pile_Group_1.coordinates)):
    Basepoint = Pile_Group_1.coordinates[i]
    Pile_BP = DrivenPile_BP(1200, 350, 300, 100, Basepoint, 'IFC')
    Piles_PileGroup_1.append(Pile_BP)
```

This will put a list of elements of a class DrivenPileBP in a unified list. Elements in that list are still accessible by the same functions. So, for example, if we would like to print adjusted coordinates of every single pile in PilesPileGroup1, a simple loop going through all the elements of that list and printing the result of a function coordinates being called will get the whole set on screen.

```
for i in range(len(Pile_Group_1.coordinates)):
    print(Piles_PileGroup_1[i].coordinates())
```

This is a possible approach, but it's becoming increasingly unwieldy with every operation. For example, we will be required to run the procedure once again for a Pile Group base point that could be different from [0, 0, 0], once again for the Pier insertion base point, and if the bridge is in local coordinates, one more time for the Bridge insertion point. It's important to remember that every element can have a rotation as well, and sometimes, the point of rotation can be different from the insertion point (or base point). A function can be defined to solve rotational operations, such as the snippet below. This function will return the new coordinates of a point rotated at a certain angle around the rotation centre. It can be called inside a loop if it is required for multiple points.

```
def rotate(rotation_centre, base_point, angle):
    new_point=[0, 0, base_point[2]]
    new_point[0] = rotation_centre[0] + math.cos(angle) * (base_point[0] -
rotation_centre[0]) - math.sin(angle) * (
                base_point[1] - rotation_centre[1])
    new_point[1] = rotation_centre[1] + math.sin(angle) * (base_point[0] -
rotation_centre[0]) + math.cos(angle) * (
                base_point[1] - rotation_centre[1])
    return new_point
```

It should be noted that this snippet ignores the possibility of rotation in 3D space, working only in the OXY plane. That is, again, not a foolproof solution; as mentioned above, pile row incline is relatively common in civil engineering and would require rotation in the OXZ plane. Thus, additional code snippets will be necessary either in the DrivenPileBP and PileGroup classes or in the core program. This seems out of the scope of object library establishment. There are numerous ways to apply geometrical adjustments in specialised software, which will, at the same time, provide the option to verify the correctness of the results visually. Examples of applications using this approach will be mentioned in the use cases. For this reason, the rest of the paper will continue to establish library elements only in local coordinates, relying on the fact that correct adjustments in 3D space about elements' position and angle can be made in specialised software.

# 14 Pile Cap

AS2159 [77], AS5100.3 [71], AS5100.5 [81], and NZS3101 [82] do not have a formal definition of Pile Cap. [83] describes the element as follows: a pile cap transfers the load from the structures to a pile group, which transfers the load to firm soil through friction end bearing or a combination of both.

Another definition may be found in [84], which states that a pile cap spreads the load on a group of piles so that the load is shared equally between the piles depending on their stiffness.

[85] defines: "A pile cap is a type of reinforced concrete footing typically used to provide structural support for columns, piers, or any other vertical load-bearing structures. They are typically designed as reinforced concrete slab structures that transfer the loads from the columns or piers to the piles, which extend deep into the ground below the foundation level.".

The AASHTO Bridge Design Specifications define a Pile Cap as "a flexural substructure element located above or below the finished ground line that receives loads from substructure columns and is supported by shafts or piles." [86]

The only reference to the Pile Cap design in AS5100.3 is covering the overhang or the distance between any pile's outside surface and the Pile Cap's edge. Additional information on this attribute can be found in [87].

It initially appears that a Pile Cap is a simple cuboid, and its geometry should be defined by three simple parameters: its length, width, and height. While true, two of these parameters usually are not an engineering input but a parametric function of Pile Group design. In most cases, the Code requirement for the distance between the outside of the pile and the edge of the Pile Cap defines the Length and width of the Pile Cap.

Nonetheless, it is essential to establish the correct definition of the geometric parameters, even if the Pile Group defines them.

The combination of transversal loads, such as wind pressure, uneven traffic load, and general width of live load force, is almost always much bigger than the combination of lateral loads, which usually would be the force generated by the braking of the live load. This ensures that the geometry of the substructure of the bridges is much more significant in the transversal direction than in the lateral (if we consider lateral measurement following along the centre line). At the same time, this is counter-intuitive to the usual names of cuboid measurements. In practice, the object's length is usually its largest measurement. In this paper, the geometric definitions will follow the ones established in the prior stage. The Pile Cap Length will be its measurement along the centre line of the road. The Pile Cap Width will be its measurement transverse to the centre line. The Pile Cap Height is its measurement in the direction perpendicular to the ground surface.

As mentioned above, the Pile Cap Length and the Pile Cap Width should not be an engineering input by default; they are a parametric function of the Pile Group Length and the Pile Group Width.

The most logical Insertion Point for a Pile Cap is the point on the intersection of its symmetry lines, on the face facing straight down. This insertion point is usually used as a geodesic reference before Pile Cap construction is initiated and generally pairs well with the chosen Insertion Point of Pile Group. It is essential to keep in mind that the Pile Group insertion point after all damaged material has been removed, should project a certain distance into the pile cap, or in other words, the piles should be embedded into the Pile Cap for at least 50 mm [71], so the insertion point of Pile Group cannot be used as an insertion point of Pile Cap for correct calculation and 3D geometry.

```
class PileCap:
    def __init__(self, driven_pile, pile_group, height):
        self.height = height
        self.length = pile_group.length + driven_pile.width + 100*2
        self.width = pile_group.width + driven_pile.width + 100*2
```

A set of coordinates can be calculated as the following:

```
def coordinates(self):
    a = (self.length)/2
    b = (self.width)/2
    c = self.height
    self.coordinates = [[-a, -b, 0], [-a, b, 0], [a, b, 0], [a, -b, 0],

 [-a, -b, c], [-a, b, c], [a, b, c], [a, -b, c]]
    return self.coordinates
```

We can also apply the same method we used in Pile to the Pile Cap to retrieve the vertices. This function will generally work well on most objects with horizontal sections that can be defined with 4 points. The only adjustment would be in the number of horizontal layers of the object.

```
def vertices(self):
    self.vertices = []
    layers = deque([[0, 1, 2, 3], [4, 5, 6, 7]])
    self.vertices.append(layers[0])
    for j in range(1):
        for i in range(4):
            self.vertices.append([layers[j][i]] + [layers[j][i - 1]] + [layers[j + 1][i
- 1]] + [layers[j + 1][i]])
    self.vertices.append(layers[1])
    return self.vertices
```

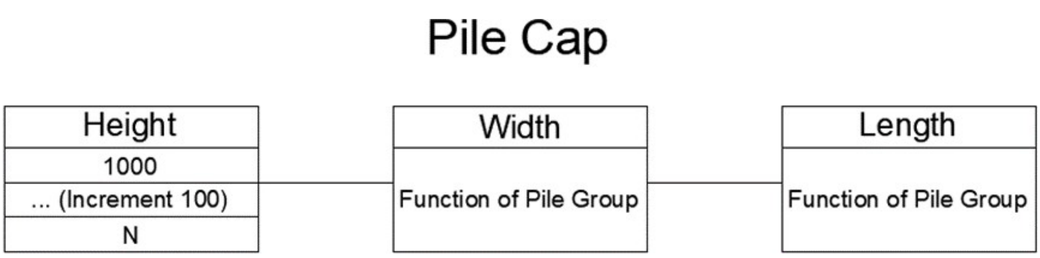


Figure 24: Pile Group attributes

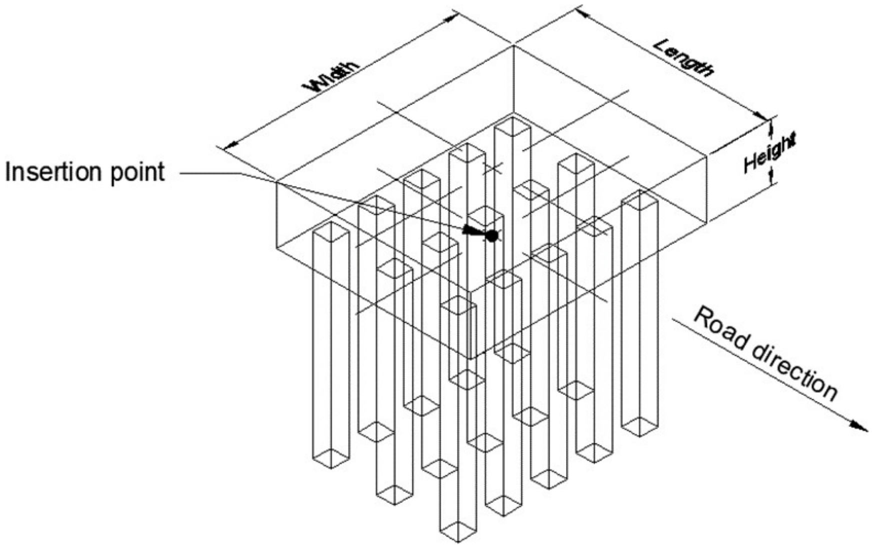



Figure 25: Pile Group scheme

# 15 Pile Cap Reinforcement

There are generally no specific rules about Pile Cap reinforcement. Both [71] AS 5100.3 and [88] AS 5100.6 do not mention any specific requirements, as does NZ 3101 [82]. All three standards suggest using flexural theory or a strut-and-tie approach for structural calculations. For comparison, SP 24.13330.2011 [89] does not have specific recommendations but suggests using webs rather than bars for Pile Cap reinforcement.

Generally, from Eurocode 2 [90] and [91], we may suggest a primary reinforcement in the zone around the top of the piles, leaving it to user input to set up reinforcement spacing and bar diameters.

Two reinforcement plans are usually implemented in pile cap reinforcement: reinforcement at the top of the piles and reinforcement at the bottom of the pile cap.

Following the Python code snippets approach, let us establish a new class for Pile Cap Reinforcement.

```
class PileCapReinforcement:
    def __init__(self, PileCap, PileGroup, lateral_diameter, lateral_number,
transverse_diameter, transverse_number, clear_cover):
        self.PileCap = PileCap
        self.PileGroup = PileGroup
        self.lateral_diameter = lateral_diameter
        self.lateral_number = lateral_number
        self.transverse_diameter = transverse_diameter
        self.transverse_number = transverse_number
        self.clear_cover = clear_cover
```

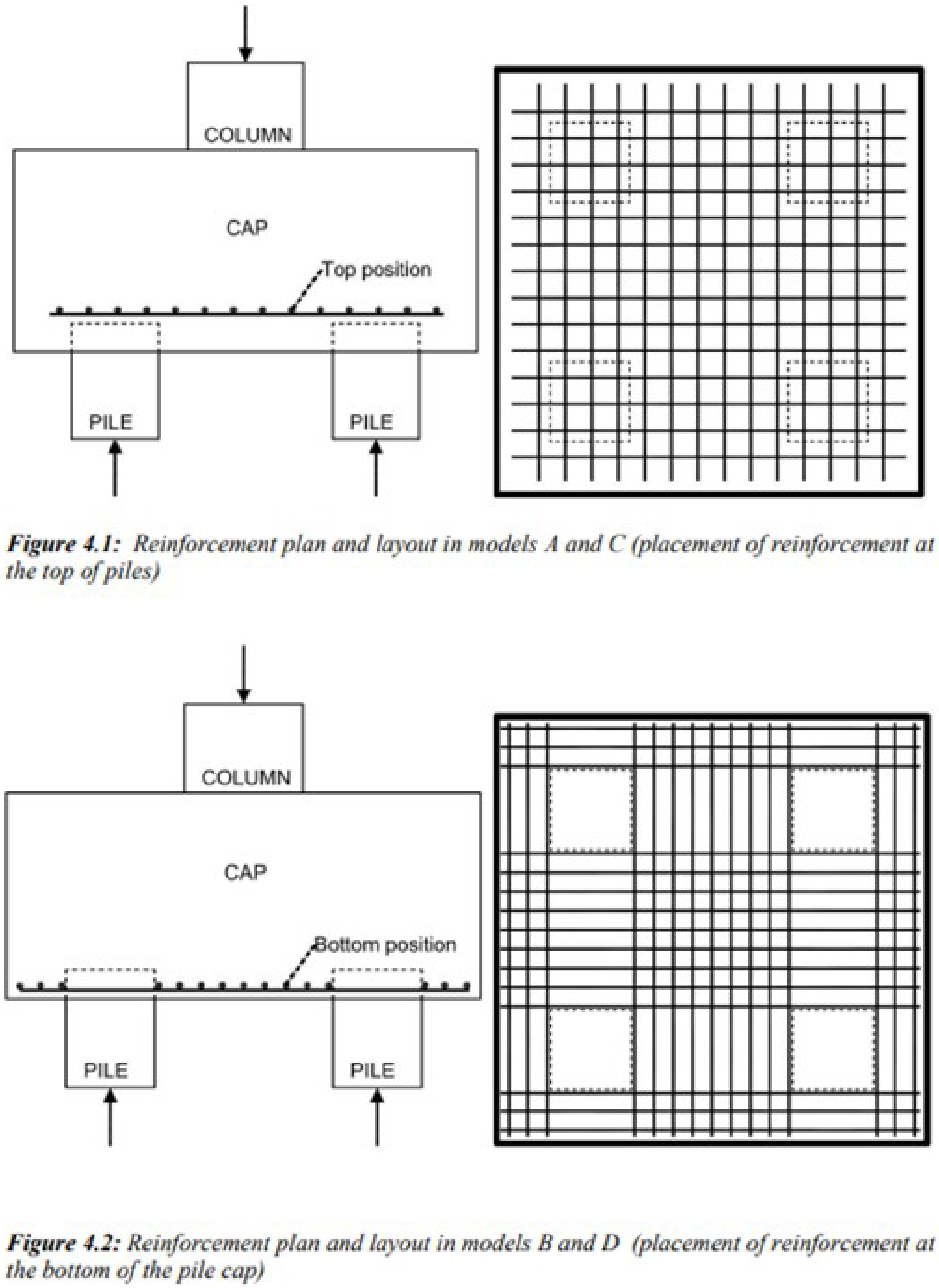



Figure 26: Pile Group attributes

As this is a simplified reinforcement example, it is possible to omit the spacing, as this parameter can be calculated. While dimensions for clear cover can be found in standards, AS2159 [77], AS3600 [92], AS3735 [93], AS4058 [94] and AS5100.5 [81] all give different recommendations depending on the type of construction, type of environment and design life.

```
self.lateral_spacing = (self.PileCap.width - self.clear_cover*2) / self.lateral_number
self.transverse_spacing = (self.PileCap.length - self.clear_cover*2) /
self.transverse_number
```

[95] gives a good overview of clear cover dimensions according to most AS standards and other international recommendations. For reinforcement at the top of the pile, we can make the following in-built functions:

```
def coordinates_top_lat(self):
    lateral = []
    z_base = 50 + self.clear_cover
    x_base = (-self.PileCap.length / 2) + self.clear_cover
    y_base = (-self.PileCap.width / 2) + self.clear_cover
    y = y_base
```

```
    for i in range(self.lateral_number):
        points = [[x_base, y, z_base], [-x_base, y, z_base]]
        lateral.append(points)
        y += self.lateral_spacing
    return(lateral)
def coordinates_top_transv(self):
    transversal = []
    z_base = 50 + self.clear_cover
    x_base = (-self.Pilecap.length / 2) + self.clear_cover
    y_base = (-self.Pilecap.width / 2) + self.clear_cover
    x = x_base
    for i in range(self.transverse_number):
        points = [[x, y_base, z_base], [x, -y_base, z_base]]
        transversal.append(points)
        x += self.transverse_spacing
    return(transversal)
```

Finding the reinforcement coordinates with reinforcement placement at the bottom of the **Pile Cap** may be more time-consuming, as it becomes essential to incorporate piles' locations in the calculation. As we have established in previous classes, the spacing between the edge of the pile and the pile cap edge is 100 mm. With a 50 mm clear cover, that would mean that in the most common example, it is doubtful that placing more than 1 rod in that zone will be possible. Thus, we can calculate the rest of the section unaffected by pile bodies as follows:

```
empty_space = self.PileCap.width - self.PileGroup.driven_pile.width *
self.PileGroup.rows_lateral - 100 * 2
```

And a new spacing as:

```
lat_spacing_bottom = empty_section / (self.lateral_number-2)
```

Depending on the number of rows of piles, we will split the remaining reinforcement bars evenly between the sections. We will not focus on the case when there is an uneven number of bars in each section, as that would require much more class input. For this case, it would be just easier to place the reinforcement in graphical software and export the coordinates.
Otherwise, the following code snippets could be used (these functions already incorporate the operators from the few snippets above).

```
def coordinates_bottom_lat(self):
    lateral = []
    z_base = self.clear_cover
    empty_space = self.PileCap.width - self.PileGroup.driven_pile.width *
self.PileGroup.rows_lateral - 100 * 2
    lat_spacing_bottom = empty_space / (self.lateral_number - 2)
    number_of_sections = self.PileGroup.rows_lateral - 1
    bars_in_section = (self.lateral_number-2) / number_of_sections
    if bars_in_section % 1 == 0: #Check for is there the same number of bars in each
section
        next_pile = 1
        x_base = (-self.PileCap.length / 2) + self.clear_cover
        piles_coordinates = self.PileGroup.coordinates
        points = [[x_base, self.clear_cover - self.PileCap.width/2, z_base], [-x_base,
self.clear_cover - self.PileCap.width/2, z_base]]
        lateral.append(points)
        if bars_in_section % 2 == 0: #Placement for even number of bars
            for i in range(self.PileGroup.rows_lateral):
                middle=piles_coordinates[i][1]-(piles_coordinates[i][1]-
piles_coordinates[i+next_pile][1])/2
                n = bars_in_section / 2
                base_bar = middle - n * lat_spacing_bottom
                y = base_bar
```

```
                for j in range(int(bars_in_section)):
                    points = [[x_base, y, z_base], [-x_base, y, z_base]]
                    lateral.append(points)
                    y += lat_spacing_bottom
        else: #Placement for odd number of bars
            for i in range(self.PileGroup.rows_lateral):
                middle=piles_coordinates[i][1]-(piles_coordinates[i][1]-
piles_coordinates[i+next_pile][1])/2
                n = bars_in_section / 2
                base_bar = middle - ( n-1 ) * lat_spacing_bottom - lat_spacing_bottom/2
                y = base_bar
                for j in range(int(bars_in_section)):
                    points = [[x_base, y, z_base], [-x_base, y, z_base]]
                    lateral.append(points)
                    y += lat_spacing_bottom
        points = [[x_base, self.PileCap.width/2 - self.clear_cover, z_base], [-x_base,
self.PileCap.width/2 - self.clear_cover, z_base]]
        lateral.append(points)
        return (lateral)
    else:
        print("A number of bars between the piles should be a whole number")
```

And pretty much the same snippet for transversal reinforcement:

```
def coordinates_bottom_trans(self):
    transversal = []
    z_base = self.clear_cover
    empty_space = self.PileCap.length - self.PileGroup.driven_pile.width *
self.PileGroup.rows_transversal - 100 * 2
    trans_spacing_bottom = empty_space / (self.transverse_number - 2)
    number_of_sections = self.PileGroup.rows_transversal - 1
    bars_in_section = (self.transverse_number-2) / number_of_sections
    if bars_in_section % 1 == 0: #Check for is there the same number of bars in each
section
        next_pile = self.PileGroup.rows_transversal
        y_base = (-self.PileCap.width / 2) + self.clear_cover
        piles_coordinates = self.PileGroup.coordinates
        points = [[self.clear_cover - self.PileCap.length / 2, y_base, z_base],
[self.clear_cover - self.PileCap.length / 2, -y_base, z_base]]
        transversal.append(points)
        if bars_in_section % 2 == 0: #Placement for even number of bars
            for i in range(self.PileGroup.rows_lateral):
                middle=piles_coordinates[i][1]-(piles_coordinates[i][1]-
piles_coordinates[i+next_pile][1])/2
                n = bars_in_section / 2
                base_bar = middle - n * trans_spacing_bottom
                x = base_bar
                for j in range(int(bars_in_section)):
                    points = [[x, y_base, z_base], [x, -y_base, z_base]]
                    transversal.append(points)
                    x += trans_spacing_bottom
        else: #Placement for odd number of bars
            for i in range(self.PileGroup.rows_transversal):
                middle=piles_coordinates[i][0]-(piles_coordinates[i][0]-
piles_coordinates[i+next_pile][0])/2
                n = bars_in_section / 2
                base_bar = middle - ( n-1 ) * trans_spacing_bottom -
```

```
trans_spacing_bottom/2
                x = base_bar
                for j in range(int(bars_in_section)):
                    points = [[x, y_base, z_base], [x, -y_base, z_base]]
                    transversal.append(points)
                    x += trans_spacing_bottom
        points = [[self.PileCap.length / 2 - self.clear_cover, y_base / 2, z_base],
[self.PileCap.length - self.clear_cover, -y_base, z_base]]
        transversal.append(points)
        return (transversal)
    else:
        print("A number of bars between the piles should be a whole number")
```

## 16 Column

The definition of a Column as a part of a Bridge Pier can be extracted from AASHTO Bridge Design Specifications: Bent Piers—Bent-type piers consist of two or more transversely spaced columns of various solid cross-sections, and these types are designed for frame action relative to forces acting about the strong axis of the pier. They are usually fixed at the dock's base and are either integral to the superstructure or with a pier cap at the top. The columns may be supported on a spread- or pile-supported footing or a solid wall shaft, or they may be extensions of the piles or shaft above the ground line.

Single-Column Piers—Single-column piers, often referred to as "T" or "Hammerhead" piers, are usually supported at the base by a spread-, drilled shaft- or pile-supported footing and may be either integral with or provide independent support for the superstructure. Their cross-section can be of various shapes, and the column can be colourful or flared to form the pier cap or blend with the sectional configuration of the superstructure cross-section. This type of pier can avoid the complexities of skewed supports if integrally framed into the superstructure, and their appearance reduces the massiveness often associated with superstructures.

Columns can be of many different sections [96], but generally, all have symmetrical shapes. Most sections can be found in the Figure 27.

Looking closely at these sections, we notice they are almost always symmetrical around both axes. Most shapes could be narrowed down to circle, rectangle, oval, square, hexagon or octagon variations.

While it is possible to establish classes for each of these sections, and eventually in the scope of an entire BIM object library, it should be done so, this paper will focus on the round section. For square and rectangular sections, we can apply the same approach we tried with the Driven Pile and Pile Cap (keeping in mind that for a column, all dimensions are engineering input). Columns with regular hexagons or octagons in sections generally follow the same geometry and principles as columns with round sections.

We can find references to the Column's attributes in the 1.5 Notation chapter of AS 5100.5 [81]

The defining geometrical parameters for a circular column will be its Diameter and length (or Height). Keeping in mind that sometimes more than 1 column supports the Pier Cap, we will need to remember to introduce the Number of Columns and Column Spacing (centre-to-centre) when designing a class for the Pier.

```
class CicrularPierColumn:
    def __init__(self, diameter, height):
        self.diameter = diameter
        self.height = height
```

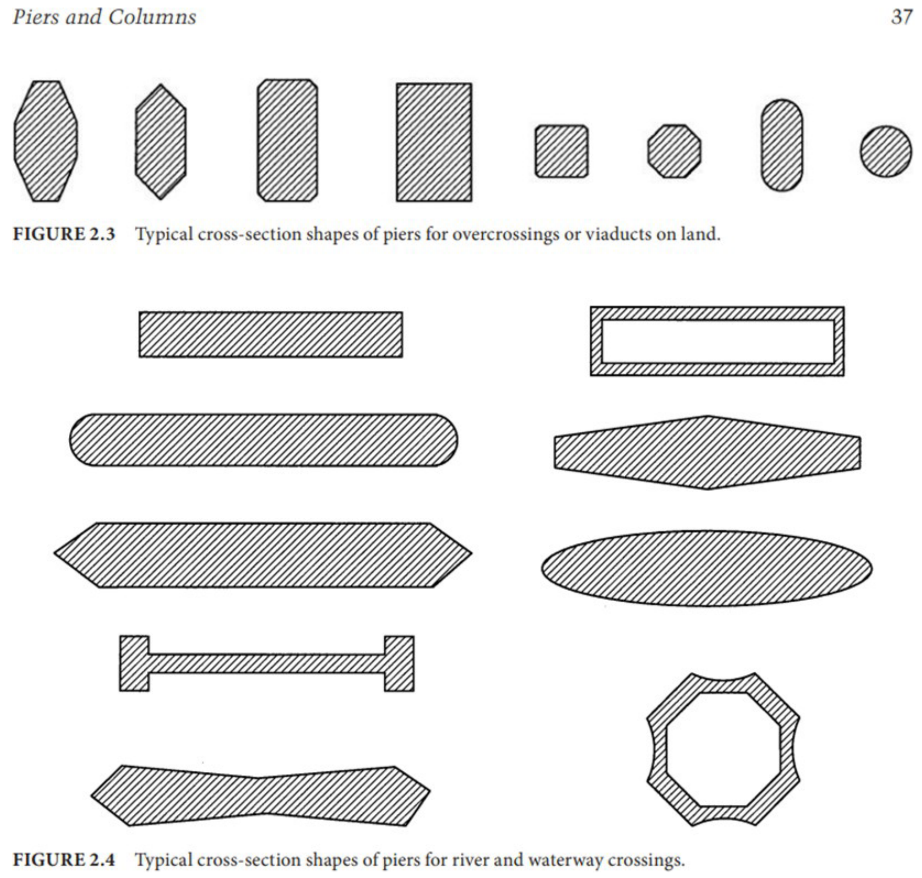



Figure 27: Columns and Piers cross-sections [96]

The geometry of columns of irregular polygonal sections and ovals usually cannot be described simply by a few-dimensional parameters, nor can their geometry be accurately reproduced without exact coordinates of all characteristic points. It would probably be a good idea to separate columns into entirely different classes, with specific inputs that allow carefully reproducing the geometry. For instance, the following snippet can be used for irregular hexagonal columns:

```
class IrregularHexPierColumn:
    def __init__(self, geometry):
        self.geometry = geometry #A list of coordinates
```

As put into the comments, geometry here requires a set of coordinates written as a list to be specified as an input. Returning to our circular column, we encountered another problem: not working with specialised graphical software. We cannot provide the set of coordinates for the column to accurately represent it in 3D space, as a circle has an infinite number of points. Based on simple geometry, the snippet below will provide a solution for extracting the coordinates of the circular column split into 360 parts. It also adds central coordinates of the top and bottom of the columns; these will be required to make vertices in the next step accurately.

```
def coordinates(self):
    z = 0
    radius = self.diameter / 2
    coordinates = []
    coordinates.append([0, 0, 0])
    coordinates.append([0, 0, self.height])
    while i<2:
        for degree in range(360):
            radians = math.radians(degree)
            x = radius * math.cos(radians)
            y = radius * math.sin(radians)
            coordinates.append([x, y, z])
```

```
        i +=1
        z = self.height
    return coordinates
```

However, next, we ran into a problem of generating a huge number of groups of vertices, as each point must be grouped properly with other points forming the outer plane of the columns. The following code groups the points into groups of 3.

```
def vertices(self):
    vertices = []
    third = 0
    base = 0
    i=0
    while i < 2:
        for degree in range(360):
            first = degree
            if degree+1 > 359:
                second = degree-360+1
            else:
                second = degree+1
            vertices.append([first+base+2, second+base+2, third])
        i += 1
        third += 1
        base = 360
    for degree in range(360):
        first = degree
        if degree + 1 > 359:
            second = degree - 360 + 1
        else:
            second = degree+1
        third = first
        vertices.append([first + 2, second + 2, third + 2 + 360])
    for degree in range(360):
        first = degree
        if degree + 1 > 359:
            second = degree - 360 + 1
        else:
            second = degree + 1
        third = first
        vertices.append([first + 2+ 360, second + 2+ 360, third + 2])
    return(vertices)
```

Next, we can check if our code works correctly by setting up a new element in Blender. The Blender works in meters so that we will adjust the sizes accordingly. The following snippet of code allows us to populate the scene with a new custom object:

```
import bpy
Column1 = CicrularPierColumn(1,5,2, 2)
verts=Column1.coordinates()
faces=Column1.vertices()
edges = []

mesh_data = bpy.data.meshes.new("column_data")
mesh_data.from_pydata(verts, edges, faces)
mesh_obj = bpy.data.objects.new("column_object", mesh_data)
bpy.context.collection.objects.link(mesh_obj)
```

The result on the screen is presented on Figure 28.

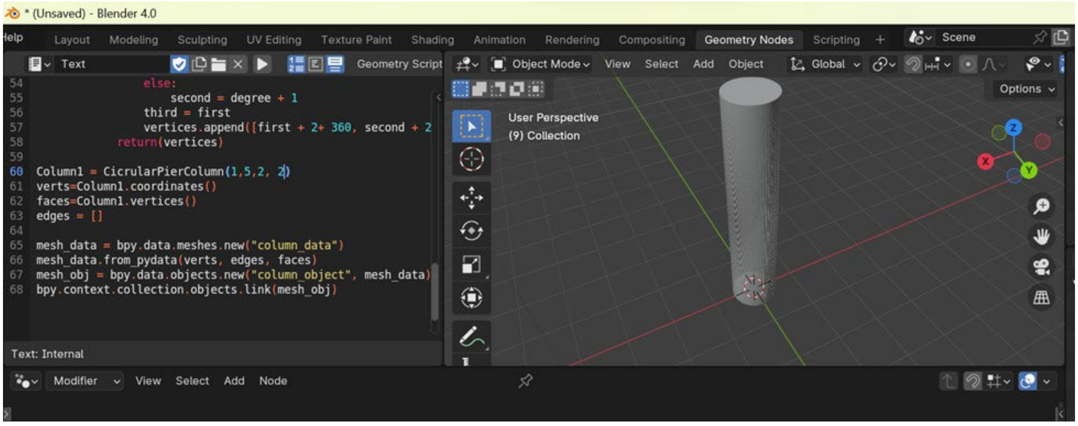



Figure 28: Blender Screenshot, Column coordinates code implementation

As a Column Insertion Point, we will use the centre of symmetry at its base, as this point directly connects to the Pile Cap.

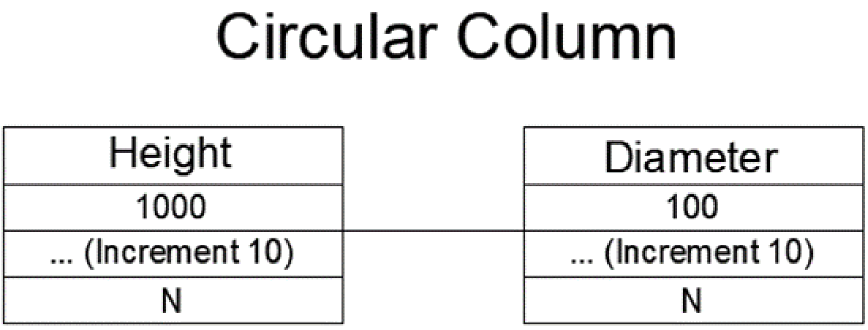
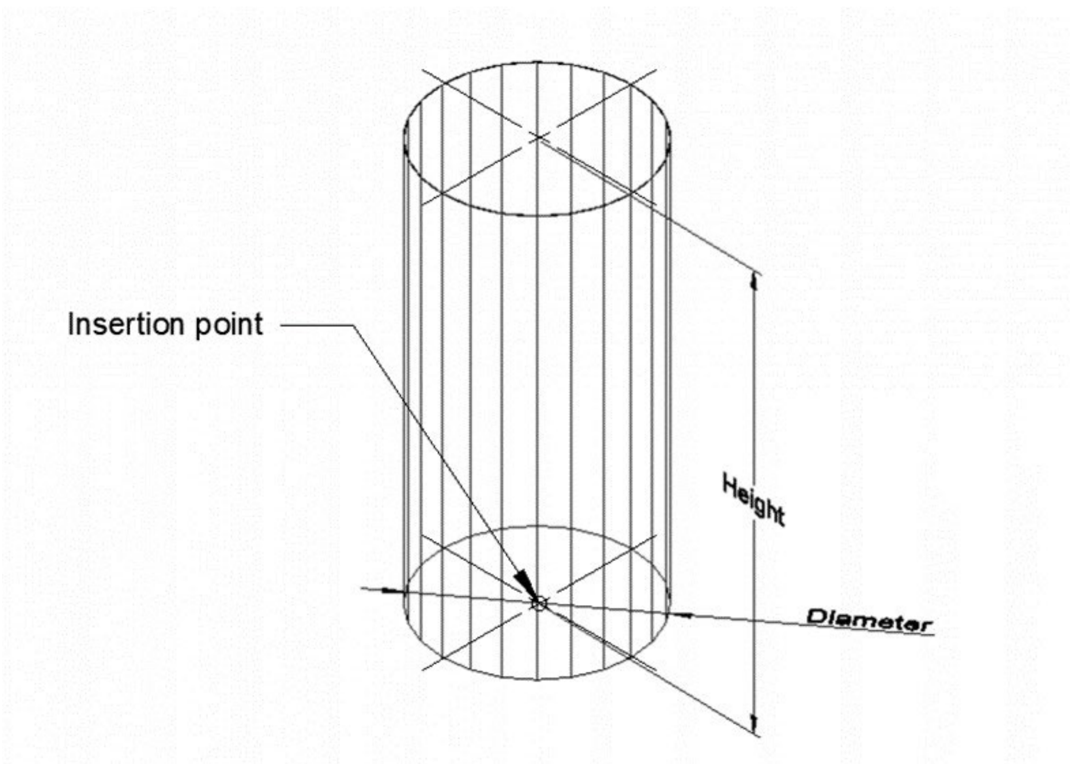



Figure 29: Circular Column attributes




igure 30: Circular Column scheme

# 17 Column Reinforcement

General principles of column reinforcement could be found in NZS 3101 [82], AS 3600 [92], and AS 5100.5 [81]. Again, for simplicity, this paper will work only with main reinforcement bars, omitting the spiral reinforcement. Three inputs will be required to model the primary reinforcement correctly: the number of bars, their diameter, and the radius of the circle in which they are placed. The last attribute could be calculated from clear cover and spiral diameter so that the final class will look like this:

```
class CircularColumnReinforcement:
    def __init__(self, circular_column, main_bar_number, main_bar_diameter,
spiral_diameter, clear_cover):
        self.circular_column = circular_column
        self.main_bar_number = main_bar_number
        self.main_bar_diameter = main_bar_diameter
        self.spiral_diameter = spiral_diameter
        self.clear_cover = clear_cover
```

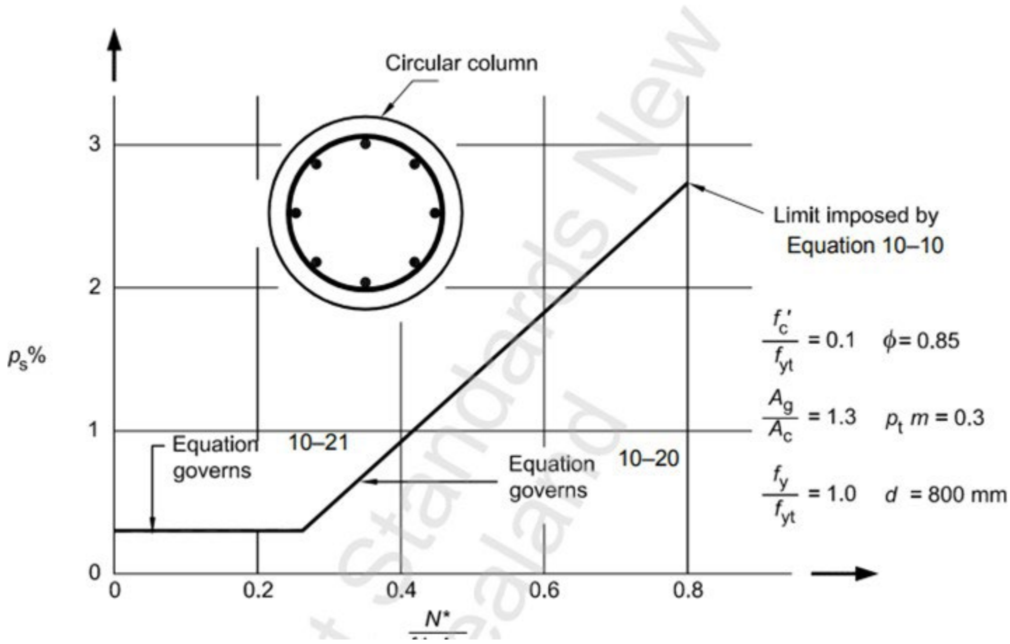



Figure 31: Circular column reinforcement [82]

The code for reinforcement coordinates will be similar to the one we used for the coordinates of the general dimensions of the column, as the main bar is placed in a circular pattern. It is important to remember that according to AS5100.5 [81], lateral reinforcement shall be extended into both the Pier Cap and the Pile Cap, so z coordinates should be calculated with this code requirement in mind.

```
def coordinates(self):
        z_top = self.circular_column.height + self.circular_column.diameter
        z_bot = - self.circular_column.diameter
        radius = self.circular_column.diameter / 2 - self.clear_cover -
self.spiral_diameter - self.main_bar_diameter/2
        coordinates = []
        for bar in range(self.main_bar_number):
            degree = 0 + bar*(360/self.main_bar_number)
            radians = math.radians(degree)
            x = radius * math.cos(radians)
            y = radius * math.sin(radians)
            coordinates.append([x, y, z_top])
```

```
            coordinates.append([x, y, z_bot])
        return(coordinates)
```

# 18 Pier Cap

A definition of a Pier Cap could not be found in AS 5100 [71], NZS 3101 [82], or AS 3600 [92]. AASHTO briefly mentions it in the definition of Bent piers as a structure conjoining the columns of the pier. Defines it as a horizontal structure that transfers the bridge deck load to the pier.

Bridge Engineering Handbook does not go into specific details but mentions that this element is sometimes called "A Cap Beam". [96]

A Dictionary of Civil Engineering [97] describes it as a concrete-bearing part of a bridge pier that distributes loads evenly over its area. In BRIDGE ENGINEERING by Tonias Zhao [98], we can also find many pier cap designs, which largely depend on the architectural design of the whole pier. In the same book, we can see the simplest form of Pier Cap design — a cuboid.

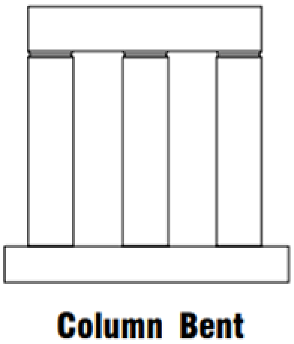



Figure 32: Typical Bent scheme

With that in mind, it is safe to say that Pier Caps will share many similarities with Pile Caps, but unlike Pile Caps, all Pier Cap dimensions will be engineering input. While counterintuitive, we will follow suit with other classes and definitions regarding geometrical dimensions so that the Pier Cap's Length will be its dimension along the span direction. Its Width will be the largest geometrical dimension, transverse to the span direction. Height will be the third dimension, perpendicular to the ground level. We can establish another class for Pier Cap, and most of the code can be copied from the Pile Cap with minor alterations:

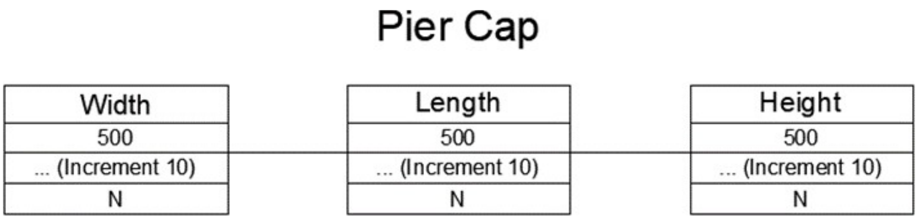



Figure 33: Pier Cap attributes

```
class PierCap:
    def __init__(self, length, width, height):
        self.length = length
        self.width = width
        self.height = height
    def coordinates(self):
        a = (self.length)/2
        b = (self.width)/2
        c = self.height
        self.coordinates = [[-a, -b, 0], [-a, b, 0], [a, b, 0], [a, -b, 0], [-a, -b,
c], [-a, b, c], [a, b, c], [a, -b, c]]
```

```
        return self.coordinates
    def vertices(self):
        self.vertices = []
        layers = deque([[0, 1, 2, 3], [4, 5, 6, 7]])
        self.vertices.append(layers[0])
        for j in range(1):
            for i in range(4):
                self.vertices.append([layers[j][i]] + [layers[j][i - 1]]

+ [layers[j + 1][i - 1]] + [layers[j + 1][i]])
        self.vertices.append(layers[1])
        return self.vertices
```

The code for coordinates() and vertices() functions are identical to the same functions in the Pile Cap class. The Pier Cap insertion point will be at its base, at the intersection of symmetry lines.

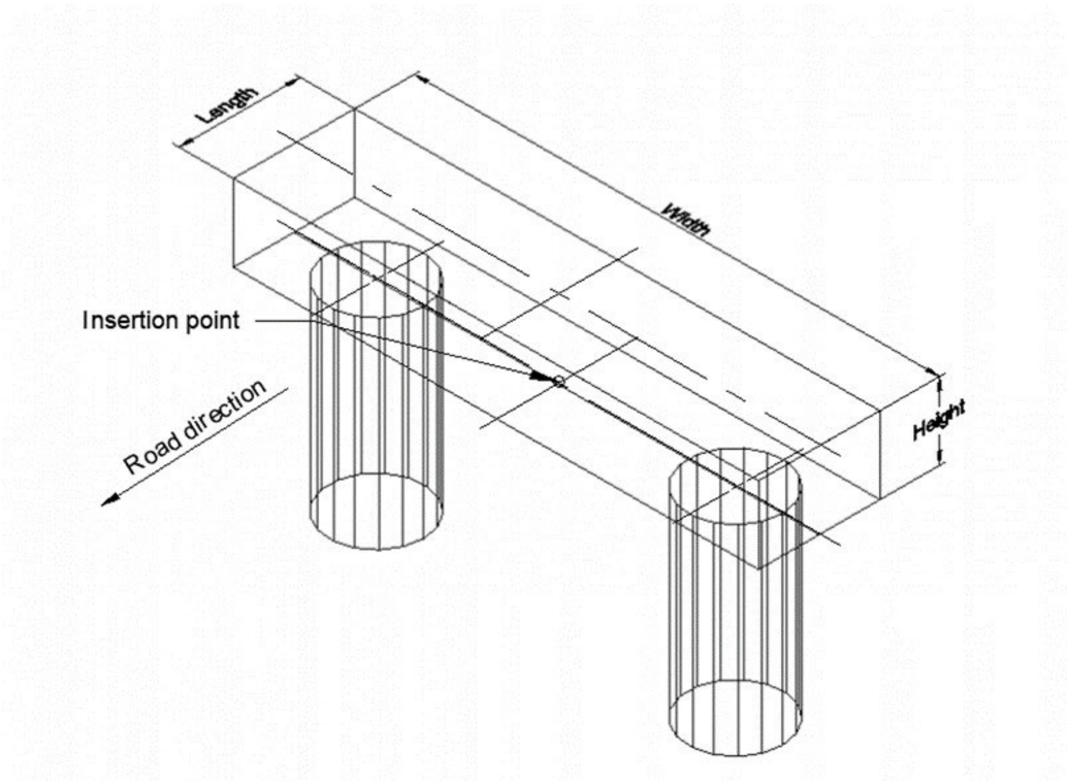



Figure 34: Pier Cap scheme

## 19 Pier Cap Reinforcement

We can also use the code we wrote for Pile Cap reinforcement to model Pier Cap reinforcement. Remember that its primary reinforcement is located in both the upper and lower sections of the element, and these could be different from each other.

```
class PierCapReinforcement:
    def __init__(self, PierCap, bot_lateral_diameter, bot_lateral_number,
bot_transverse_diameter, bot_transverse_number, top_lateral_diameter,
top_lateral_number, top_transverse_diameter, top_transverse_number,  clear_cover):
        self.PierCap = PierCap
        self.bot_lateral_diameter = bot_lateral_diameter
        self.bot_lateral_number = bot_lateral_number
        self.bot_transverse_diameter = bot_transverse_diameter
        self.bot_transverse_number = bot_transverse_number
        self.top_lateral_diameter = top_lateral_diameter
        self.top_lateral_number = top_lateral_number
        self.top_transverse_diameter = top_transverse_diameter
```

```python
        self.top_transverse_number = top_transverse_number
        self.clear_cover = clear_cover
        self.bot_lateral_spacing = (self.PierCap.width - self.clear_cover*2) / 
self.bot_lateral_number
        self.bot_transverse_spacing = (self.PierCap.length - self.clear_cover*2) / 
self.bot_transverse_number
        self.top_lateral_spacing = (self.PierCap.width - self.clear_cover * 2) / 
self.top_lateral_number
        self.top_transverse_spacing = (self.PierCap.length - self.clear_cover * 2) / 
self.top_transverse_number
    def coordinates_bot_lat(self):
        lateral = []
        z_base = self.clear_cover
        x_base = (-self.PierCap.length / 2) + self.clear_cover
        y_base = (-self.PierCap.width / 2) + self.clear_cover
        y = y_base
        for i in range(self.bot_lateral_number):
            points = [[x_base, y, z_base], [-x_base, y, z_base]]
            lateral.append(points)
            y += self.bot_lateral_spacing
        return(lateral)
    def coordinates_bot_transv(self):
        transversal = []
        z_base = self.clear_cover
        x_base = (-self.PierCap.length / 2) + self.clear_cover
        y_base = (-self.PierCap.width / 2) + self.clear_cover
        x = x_base
        for i in range(self.bot_transverse_number):
            points = [[x, y_base, z_base], [x, -y_base, z_base]]
            transversal.append(points)
            x += self.bot_transverse_spacing
        return(transversal)
    def coordinates_top_lat(self):
        lateral = []
        z_base = self.PierCap.height-self.clear_cover
        x_base = (-self.PierCap.length / 2) + self.clear_cover
        y_base = (-self.PierCap.width / 2) + self.clear_cover
        y = y_base
        for i in range(self.top_lateral_number):
            points = [[x_base, y, z_base], [-x_base, y, z_base]]
            lateral.append(points)
            y += self.top_lateral_spacing
        return(lateral)
    def coordinates_top_transv(self):
        transversal = []
        z_base = self.PierCap.height-self.clear_cover
        x_base = (-self.PierCap.length / 2) + self.clear_cover
        y_base = (-self.PierCap.width / 2) + self.clear_cover
        x = x_base
        for i in range(self.top_transverse_number):
            points = [[x, y_base, z_base], [x, -y_base, z_base]]
            transversal.append(points)
            x += self.top_transverse_spacing
        return(transversal)
```

# 20 Bridge Pier

One definition of the Bridge Pier can be found in the BRIDGE ENGINEERING by Tonias Zhao [98]. It defines it as a substructure element that supports spans between abutments, while the dictionary defines it as a structure that supports the bridge superstructure. In practical terms, a Pier would be just a sum of its components, with instructions about their positions. It could also be an actual representation of a physical object, a 3D model of one of the actual piers, with its coordinates defined in the world and its elements interacting with the ground surface or geology. Within the scope of this paper, we will establish a simple class that serves as a template and is not fully suitable for inclusion in a BIM model without some additional data.

```
class Pier:
    def __init__(self, PileGroup, PileCap, Column, PierCap, number_of_columns,
column_spacing):
        self.PileGroup = PileGroup
        self.PileCap = PileCap
        self.Column = Column
        self.PierCap = PierCap
        self.number_of_columns = number_of_columns
        self.column_spacing = column_spacing
```

This class takes other classes as input and merges them into one element. We can add functions that could use calls to the functions in these underlying elements. For example, we can calculate the maximum width of the Pier by comparing the width of the Pile Cap and the width of the Pier Cap.

```
def max_width(self):
    max_width = max(self.PileCap.width, self.PierCap.width)
    return max_width
```

To access that in the core part of the program, we will have to input the parameters of each class.

```
Pile_1 = DrivenPile(12000, 350, 300, 100, 'IFC')

Pile_Group_1 = PileGroup(Pile_1, 3, 3)

Pile_Cap_1 = PileCap(Pile_1, Pile_Group_1, 2000)

Column1 = CicrularPierColumn(1000,5000)

Pier_Cap_1 = PierCap(1000,8000,2000)
```

Alternatively, if working in PyCharm, we will have handy hints about each parameter that is being entered into each element:

Calling out the new max_width() function and printing the result, we will get the following:

```
Pile_1 = DrivenPile( length: 12000,  width: 350,  tip_length: 300,  tip_width: 100,  ifc: 'IFC')
Pile_Group_1 = PileGroup(Pile_1,  rows_lateral: 3,  rows_transversal: 3)
Pile_Cap_1 = PileCap(Pile_1, Pile_Group_1,  height: 2000)
Column1 = CicrularPierColumn( diameter: 1000, height: 5000)
Pier_Cap_1 = PierCap( length: 1000, width: 8000, height: 2000)
Pier1 = Pier(Pile_Group_1,Pile_Cap_1,Column1,Pier_Cap_1, number_of_columns: 2, column_spacing: 4000)
print(Pier1.max_width())
```

Figure 35: Pycharm input interface screenshot

```
8000

Process finished with exit code 0
```

Figure 36: Pycharm output screenshot

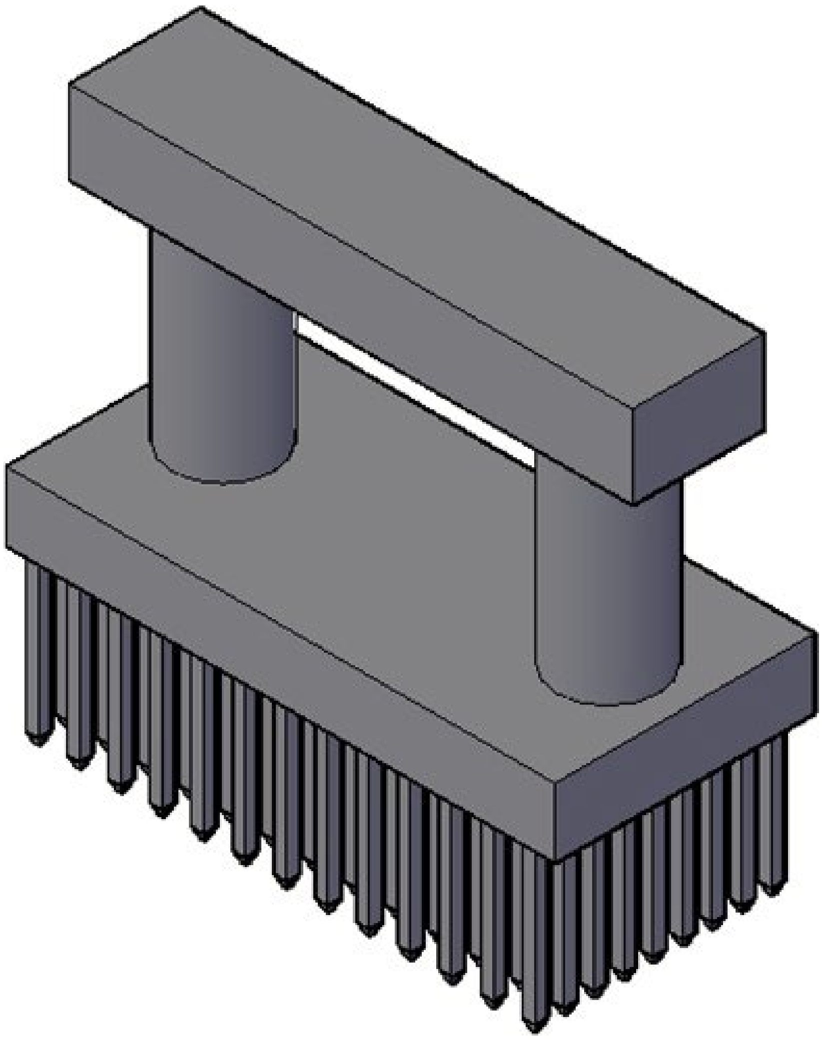


Figure 37: Pier scheme

# 21 Results

This study shows a clear path for using our designed method to improve 3D models. With further research, it should be possible to build simple and flexible building blocks that are accessible on a website. While creating the web portal is out of the scope of this research, the most straightforward option can be found in the screenshot below. Following these templates, many more elements could be added similarly. At the same time, existing models and classes could be gradually improved and enhanced by many more valuable functions, such as material quantity calculations, more code validation and export to other formats.

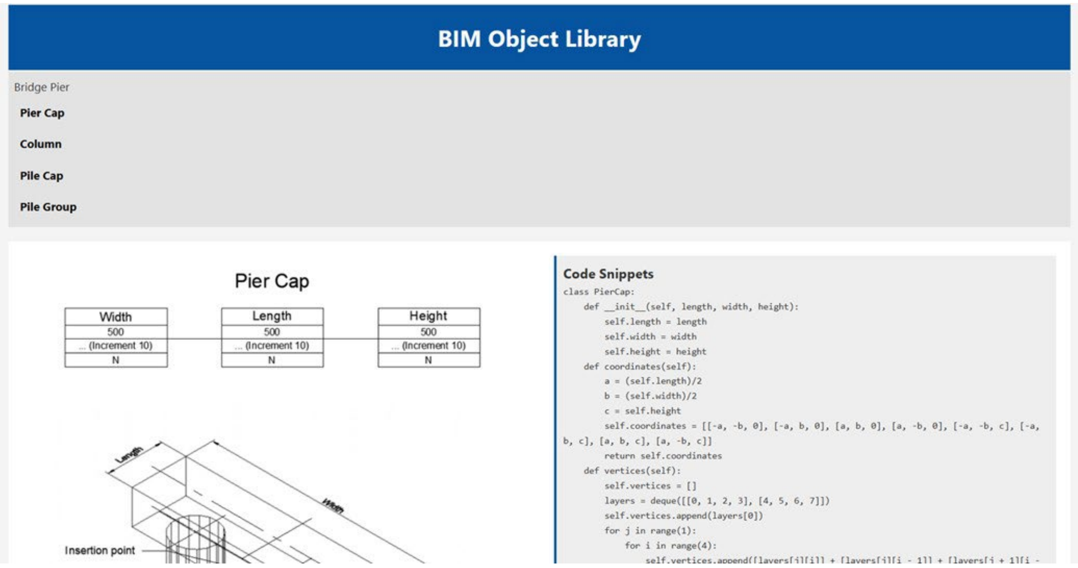



Figure 38: Website Template

## 22 Further Research

Only a fraction of the bridge decomposition was scrutinised and classified throughout this research. There is a great variety of different arrangement types of substructure components, other major elements necessary to model a bridge (abutments, supporting wall), and another world of elements that combined represent the bridge superstructure. The influence of the road parameters and loads could not be examined in this paper, even though in bridge engineering, these are the main defining factors of any bridge design. Reinforcement modelling should also be improved to represent everything required by the code. The library should also include a section on the technology of construction and temporary elements, such as launching noses, sheet piles, scaffolding, and temporary piers in time. Technological processes may also become another solid addition to the approach, allowing the monitoring of specialised software procurement of the materials, their movements on the construction site, their testing in the laboratory, etc.

## 23 Discussions

The elements and classes created during this research could be used in various ways to produce 3D models. The most straightforward approach would be using these schemes as a guide to collect the primitive 3D model in Blender using, for instance, geometry nodes. This does not require any knowledge about coding, and the code snippets can be largely ignored; the schemes showing how different elements interact with each other and their defining parameters allow clusters of nodes to be set up, reproducing the desired 3D geometry.

The most significant advantage of this approach is that Blender already incorporates all the necessary geometrical adjustment functions. It easily allows the generation of simple shapes, further adjustment of their external geometry, and all possible functions to set them up properly in desired coordinates in 3D space. Geometry nodes could easily be copied, so a cluster of nodes that forms a Pile Group could be reproduced as many times as the project requires, establishing other Pile Groups for other piers in seconds. Blender also allows one to instantly see what has been generated that way, confirming that the model aligns with the design. Another option Blender offers is using Python code directly in its shell. That way, 3D models can be generated straight from the Python code text document in the program and then adjusted according to the program's function as needed, including copying objects and moving and rotating them. In this research, a model for a Circular Column was produced that way and illustrated in FIGURE 23. A third option, probably

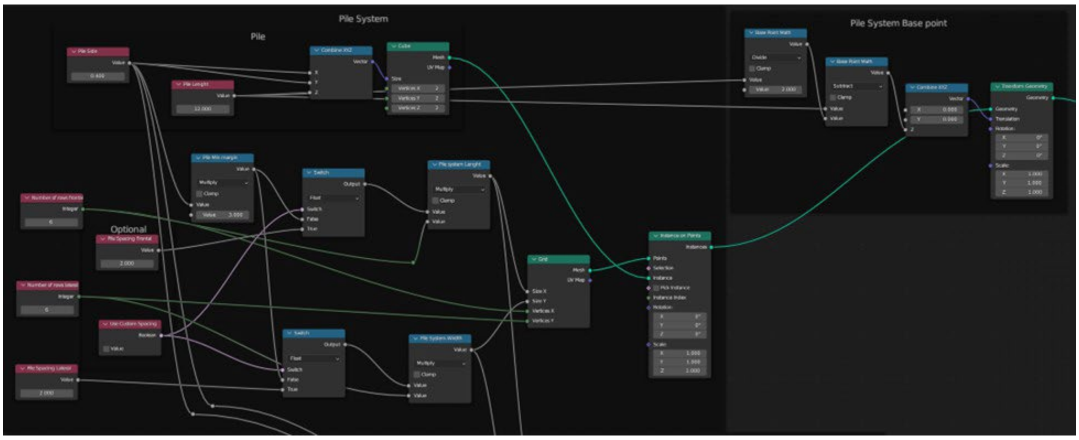



Figure 39: Blender screenshot, geometry nodes

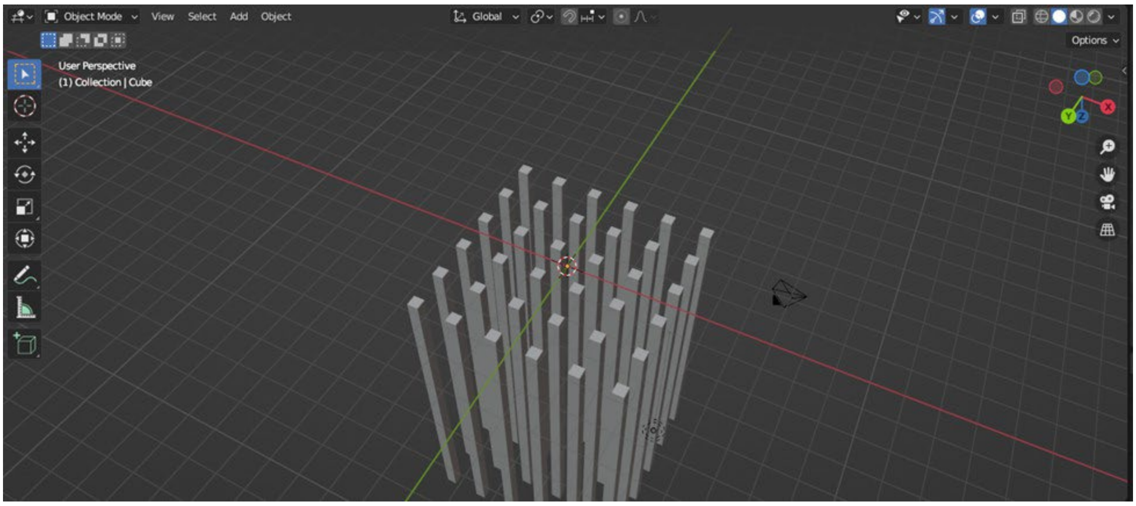



Figure 40: Blender screenshot, 3d model generated from the geometry nodes

the most time-consuming, would be establishing custom geometry nodes in Blender. This requires some additional code and updating the program, but overall, it is a possible solution. Each class can have its geometry node with prebuilt functions and outputs. This will make clusters much more compact, allowing easier adjustments of user inputs. Another approach is using Python Script nodes in Dynamo. For instance, it is possible to generate a 3D object with any geometry, combining functions of coordinates and vertices to produce a mesh. While the functions prepared in this research will not instantly work with Dynamo, a simple script can convert Python lists to the appropriate format.

This screenshot shows the generation of a Driven Pile Mesh from the Python Class we've established. The code includes two instances of the Python Script to extract indices and coordinates, as the current version of Dynamo only allows one Output from any Python script.

The following simple snippet of code allows for extracting coordinates from the pile class to be used in Dynamo and converting measurements into meters.

```
coordinates = []
j=0
for i in IN[0]:
    pt=Point.ByCoordinates(i[j]/1000, i[j+1]/1000, i[j+2]/1000)
    coordinates.append(pt)

OUT = coordinates
```

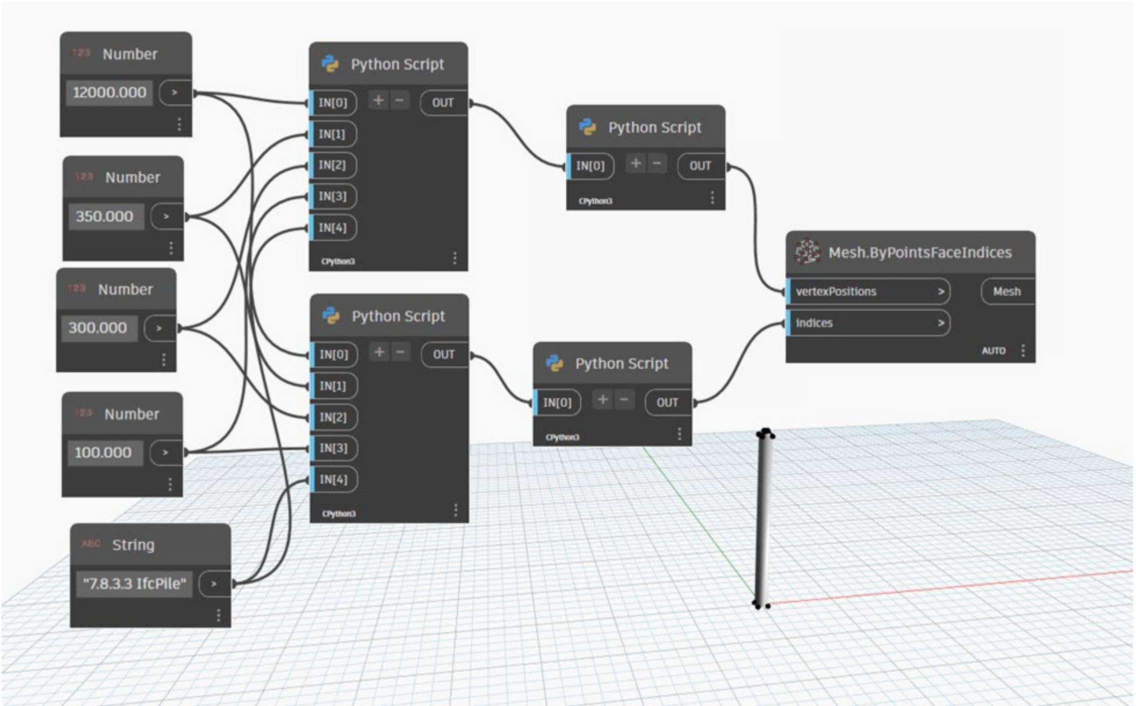



Figure 41: Dynamo screenshot, nodes and 3D object geometry

Another vital benefit of such approach is digital storage of the data. Using the classes approach, all attrubites are easily accessible, which allows any other program to refer to these and get access to them instantly. Other methods could also easily use the established classes, simplifying many other engineering tasks such as materials calculations and 3D geometry clash detection, allowing for time saving with additional automation, as all the coordinates of all elements are stored in digital form. All the dimensions and the whole geometry has an absolute precision, which improves the accuracy of overall design. Usage of library of elements also saves costs in production of drawing or documentation, as everything that gets uploaded to a library can be reused in multiple other projects.

# 24 Conclusion

As the construction industry evolves, adopting BIM and parametric modelling becomes increasingly vital. Developing a standard, software-independent BIM object library is essential for efficient information transmission and management throughout the life cycle of infrastructure projects. Such standardisation will streamline the construction process and promote consistency and quality across the industry. As this technology continues to mature, it holds the promise of reshaping the landscape of civil engineering, driving the industry towards a more sustainable, efficient, and innovative future. The organisation of engineering knowledge on a free collaborative platform would push forward BIM usage in infrastructure and allow for the gradual improvement of existing models. It will also level the playing field for smaller companies, allowing them to use existing snippets of code and template projects to create their own BIM models with a fraction of current labour costs. These library elements would allow the Client to use inbuilt functions for their calculations, allowing greater control over the project. Realising the full potential of BIM in bridge construction necessitates a concerted effort among all industry stakeholders, including policymakers, professionals, and academics. Together, these entities must work towards a common framework that facilitates the adoption and optimisation of BIM practices. BIM is a beacon of innovation in the construction industry, offering a path toward more efficient, sustainable, and collaborative construction projects. This thesis's findings underscore BIM's transformative potential and chart a course for its broader adoption and integration. As the industry moves forward, embracing BIM and fostering an environment of collaboration and standardisation will be crucial for achieving the next leap in civil engineering excellence.

# References


[1] S. A. Biancardo, N. Viscione, C. Oreto, R. Veropalumbo, and F. Abbondati, "Bim approach for modeling airports terminal expansion," *Infrastructures*, vol. 5, May 2020. doi: 10.3390/infrastructures5050041. (visited on 04/01/2021).

[2] B. Will, P. C. Max, A. Brommer, *et al.*, "Open bim approach for efficient serial bridge modeling," *Report*, Jan. 2020. doi: 10.2749/wroclaw.2020.0983. (visited on 07/16/2024).

[3] *From tech investment to impact: Strategies for allocating capital and articulating value*, Deloitte Insights. [Online]. Available: https://www2.deloitte.com/us/en/insights/topics/leadership/maximizing-value-of-tech-investments.html.

[4] *Tech in construction - report bundle (6 reports)*, Market Research Reports Consulting — GlobalData UK Ltd. [Online]. Available: https://www.globaldata.com/store/report/tech-in-construction-bundle/.

[5] *Investing in the right construction technology to improve financial performance, productivity, and reputation- and the mindset shift that's driving top roi 2020 fmi industry report where construction firms are finding value*. [Online]. Available: https://mkt-cdn.procore.com/downloads/reports/2019FMIReport_RPT_US.pdf.

[6] J. C. Cheng, Q. Lu, and Y. Deng, "Analytical review and evaluation of civil information modeling," *Automation in Construction*, vol. 67, pp. 31–47, Jul. 2016. doi: 10.1016/j.autcon.2016.02.006.

[7] Y. Zou and S. X. Zhou, "A model-based bim framework for bridge engineering," *Applied Mechanics and Materials*, vol. 587-589, pp. 1339–1343, Jul. 2014. doi: 10.4028/www.scientific.net/amm.587-589.1339. (visited on 03/08/2020).

[8] Allplan, *The revolution that is changing bim for bridges*, blog.allplan.com. [Online]. Available: https://blog.allplan.com/en/the-revolution-that-is-changing-bim-for-bridges (visited on 07/16/2024).

[9] R. Hajdin and V. Samec, "Bim and bms: Current status and challenges," *Report*, pp. 710–715, Jan. 2022. doi: 10.2749/prague.2022.0710. (visited on 07/16/2024).

[10] D. Isailovi´c and R. Hajdin, "Geometry as a common ground for bms and bim," *Report*, Jan. 2022. doi: 10.2749/prague.2022.0720. (visited on 10/27/2023).

[11] A. Girardet and C. Boton, "A parametric bim approach to foster bridge project design and analysis," *Automation in Construction*, vol. 126, p. 103 679, Jun. 2021. doi: 10.1016/j.autcon.2021.103679. (visited on 04/08/2021).

[12] K. G. Ahmed, "Augmented reality in remote learning: A proposed transformative approach for building construction education," Dec. 2020. doi: 10.1109/econf51404.2020.9385491. (visited on 09/05/2023).

[13] S. Kaewunruen, M. AbdelHadi, M. Kongpuang, W. Pansuk, and A. M. Remennikov, "Digital twins for managing railway bridge maintenance, resilience, and climate change adaptation," *Sensors*, vol. 23, p. 252, Jan. 2023. doi: 10.3390/s23010252. [Online]. Available: https://www.mdpi.com/2031778.

[14] P. E. Love, J. Matthews, I. Simpson, A. Hill, and O. A. Olatunji, "A benefits realization management building information modeling framework for asset owners," *Automation in Construction*, vol. 37, pp. 1–10, Jan. 2014. doi: 10.1016/j.autcon.2013.09.007.

[15] D. W. Chan, T. O. Olawumi, and A. M. Ho, "Perceived benefits of and barriers to building information modelling (bim) implementation in construction: The case of hong kong," *Journal of Building Engineering*, vol. 25, p. 100 764, Sep. 2019. doi: 10.1016/j.jobe.2019.100764.

[16] C. M. Eastman, P. M. Teicholz, R. Sacks, and G. Lee, *BIM handbook : A guide to building information modeling for owners, managers, designers, engineers and contractors*. Wiley, 2018.

[17] M.-I. O. Ali, Y. Mohamed, H. Taghaddos, and R. Hermann, "Bim obstacles in industrial projects : A contractor perspective," Jun. 2015. doi: 10.14288/1.0076376. (visited on 09/07/2023).

[18] M. Lahiani, "Benefits of bim implementation in the french construction industry," *IOP Conference Series: Earth and Environmental Science*, vol. 588, no. 4, p. 042 055, Nov. 2020. doi: 10.1088/1755-1315/588/4/042055. [Online]. Available: https://dx.doi.org/10.1088/1755-1315/588/4/042055.

[19] R. Samimpay and E. Saghatforoush, "Benefits of implementing building information model-

ing (bim) in infrastructure projects," *Journal of Engineering, Project, and Production Management*, vol. 10, pp. 123–140, May 2020. doi: 10.2478/jeppm-2020-0015.

[20] B. Gercek, O. B. Tokdemir, M. E. Ilal, and H. M. Gunaydin, "Bim execution process of construction companies for building projects," *Proceedings of International Structural Engineering and Construction*, vol. 4, E. Pellicer, J. M. Adam, V. Yepes, A. Singh, and S. Yazdani, Eds., Jul. 2017. doi: 10.14455/isec.res.2017.5. (visited on 11/09/2020).

[21] A. Lymath, *The top five barriers to bim implementation*, NBS, Dec. 2014. [Online]. Available: https://www.thenbs.com/knowledge/the-top-five-barriers-to-bim-implementation.

[22] J. Baillie, "Bim and its benefits explained," *Health estate*, vol. 70, pp. 19–22, Sep. 2016.

[23] *Digital transition survey 2017 by construction it alliance - issuu*, issuu.com, Nov. 2017. [Online]. Available: https://issuu.com/constructionitalliance/docs/3rd_national_survey_to_benchmark_ (visited on 07/16/2024).

[24] K. Ullah, I. Lill, and E. Witt, "An overview of bim adoption in the construction industry: Benefits and barriers," *10th Nordic Conference on Construction Economics and Organization*, vol. 2, pp. 297–303, May 2019. doi: 10.1108/s2516-285320190000002052. [Online]. Available: https://www.emerald.com/insight/content/doi/10.1108/S2516-285320190000002052/full/html.

[25] C. Harrison and D. Thurnell, "Bim implementation in a new zealand consulting quantity surveying practice," *International Journal of Construction Supply Chain Management*, vol. 5, pp. 1–15, Jul. 2015. doi: 10.14424/ijcscm501015-01-15.

[26] A. T. Hall, S. Durdyev, K. Koc, O. Ekmekcioglu, and L. Tupenaite, "Multi-criteria analysis of barriers to building information modeling (bim) adoption for smes in new zealand construction industry," *Engineering, Construction and Architectural Management*, Jun. 2022. doi: 10.1108/ecam-03-2022-0215.

[27] L. Liao, E. A. L. Teo, R. Chang, and X. Zhao, "Diffusion of building information modeling in building projects and firms in singapore," *Sustainability*, vol. 12, p. 7762, Sep. 2020. doi: 10.3390/su12187762. [Online]. Available: https://www.mdpi.com/2071-1050/12/18/7762/pdf.

[28] *Action plan for the digital strategy for aotearoa*, New Zealand Digital government, Sep. 2022. [Online]. Available: https://www.digital.govt.nz/digital-government/strategy/digital-strategy-for-aotearoa-and-action-plan/action-plan-for-the-digital-strategy-for-aotearoa/ (visited on 07/16/2024).

[29] *The infrastructure action plan, the new zealand government, 2023*. [Online]. Available: https://www.treasury.govt.nz/sites/default/files/2023-05/infrastructure-action-plan-2023.pdf.

[30] *The new zealand bim handbook 2023 appendix a modelling and documentation best practice*. [Online]. Available: https://www.biminnz.co.nz/s/BIM-Handbook_APPENDIX_Av3-rw35.pdf (visited on 07/16/2024).

[31] M. Antwi-Afari, H. Li, E. Pärn, and D. Edwards, "Critical success factors for implementing building information modelling (bim): A longitudinal review," *Automation in Construction*, vol. 91, pp. 100–110, Jul. 2018. doi: 10.1016/j.autcon.2018.03.010.

[32] *Building information modelling (bim) for road infrastructure: Tem requirements and recommendations — unece*, unece.org. [Online]. Available: https://unece.org/transport/publications/building-information-modelling-bim-road-infrastructure-tem-requirements-and.

[33] K. K. Fallon and M. Johnson, "Common object library description," *US Army Corps of Engineers: Engineer Research and Development Center (Knowledge Core)*, Aug. 2012. doi: 10.21236/ada570085. (visited on 07/16/2024).

[34] D. Pasini, V. Caffi, B. Daniotti, S. Lupica Spagnolo, and A. Pavan, "The innovance bim library approach," *Innovative Infrastructure Solutions*, vol. 2, May 2017. doi: 10.1007/s41062-017-0062-y. [Online]. Available: https://paperity.org/p/79869821/the-innovance-bim-library-approach.

[35] , *Welcome to the national bim standard - united states — national bim standard - united states*, Nationalbimstandard.org, 2017. [Online]. Available: https://www.nationalbimstandard.org/.

[36] T. Tschepke, *Bim im bahnbau: Daten, standards und qualifizierung sind die größten her-*

*ausforderungen*, System——Bahn, Nov. 2021. [Online]. Available: https://www.system-bahn.net/aktuell/bim-im-bahnbau-daten-standards-und-qualifizierung-sind-die-groessten-herausforderungen/ (visited on 07/16/2024).

[37] B. Daniotti, A. Pavan, S. L. Spagnolo, V. Caffi, D. Pasini, and C. Mirarchi, "Standardized guidelines for the creation of bim objects," *Springer tracts in civil engineering*, pp. 39–70, Nov. 2019. doi: 10.1007/978-3-030-32889-4_3.

[38] *The history of the national building specification*, www.thenbs.com. [Online]. Available: https://www.thenbs.com/about-nbs/history-of-nbs/.

[39] *Uni 11337 building and engineering work—coding criteria of works, activities and resources—identification, description and interoperability*, 2009.

[40] A. Pavan, S. Lupica Spagnolo, V. Caffi, C. Mirarchi, and B. Daniotti, "National bim digital platform for construction (innovance project)," in *Digital Transformation of the Design, Construction and Management Processes of the Built Environment*, B. Daniotti, M. Gianinetto, and S. Della Torre, Eds. Cham: Springer International Publishing, 2020, pp. 3– 15, isbn: 978-3-030-33570-0. doi: 10.1007/978-3-030-33570-0_1. [Online]. Available: https://doi.org/10.1007/978-3-030-33570-0_1.

[41] *Case study: Kiwirail - transforming new zealand's rail network with technology — construction accord*, www.constructionaccord.nz. [Online]. Available: https://www.constructionaccord.nz/good-practice/beacon-projects/case-study-kiwirail/.

[42] *Pace digital engineering — beca*, www.beca.com. [Online]. Available: https://www.beca.com/what-we-do/projects/transport-and-infrastructure/pace-digital-engineering (visited on 07/16/2024).

[43] S. Hazlegreaves, *Scaling up collaboration to support the delivery of new zealand's rail network transformation*, Planning, Building Construction Today, Oct. 2022. [Online]. Available: https://www.pbctoday.co.uk/news/digital- construction- news/bim- news/collaboration - new - zealands - rail - network - transformation / 112784/ (visited on 07/16/2024).

[44] *Digital engineering framework version 4*. [Online]. Available: https://www.kiwirail.co.nz / assets/ Uploads / Who - we - are/ Innovation/ Downloadable - documents / Updated - DE-Framework-Versions/KiwiRail-Digital-Engineering-Framework.pdf (visited on 07/16/2024).

[45] *Digital engineering - kiwirail*, GitHub. [Online]. Available: https://github.com/KiwiRailDigiEng (visited on 07/16/2024).

[46] *Kiwirail asset definition viewer*, white-bay-079a98d00.1.azurestaticapps.net. [Online]. Available: https://white- bay- 079a98d00.1.azurestaticapps.net/#/Viewer (visited on 07/16/2024).

[47] D. Pupeikis, A. A. Navickas, E. Klumbyte, and L. Seduikyte, "Comparative study of construction information classification systems: Cci versus uniclass 2015," *Buildings*, vol. 12, p. 656, May 2022. doi: 10.3390/buildings12050656. (visited on 11/14/2022).

[48] M. Häußler, S. Esser, and A. Borrmann, "Code compliance checking of railway designs by integrating bim, bpmn and dmn," *Automation in Construction*, vol. 121, p. 103 427, Jan. 2021. doi: 10.1016/j.autcon.2020.103427. (visited on 02/05/2021).

[49] *Bim-content fü r auftragnehmer*, infoplattform-personenbahnhoefe.deutschebahn.com. [Online]. Available: https://infoplattform-personenbahnhoefe.deutschebahn.com/pbhf/Vorgaben-zur-Anwendung-der-BIM-Methodik/BIM-Content-fuer-Planer-7719364#:∼:text=Daf%C3%BCr%20stellen%20wir%20dem%20Auftragnehmer (visited on 07/16/2024).

[50] *Bim-anwendung und implementierung*, infoplattform-personenbahnhoefe.deutschebahn.com. [Online]. Available: https://infoplattform- personenbahnhoefe.deutschebahn.com/pbhf/Vorgaben-zur-Anwendung-der-BIM-Methodik/BIM-Anwendung-und-Implementierung--10681436#:∼:text=Anlage%20A%20 (visited on 07/16/2024).

[51] *Vorgaben zur anwendung der bim-methodik db infrago ag*, infoplattform-personenbahnhoefe.deutschebahn.com. [Online]. Available: https://infoplattform- personenbahnhoefe.deutschebahn.com/pbhf/Vorgaben-zur-Anwendung-der-BIM-Methodik/Vorgaben_zu_Anwendung_der_BIM-Methodik-7719352 (visited on 07/16/2024).

[52] Baustandards and G. Personenbahnhöfe, *Vorgaben zur anwendung der bim-methodik anlage a -digitale bauteilbibliothek*. [Online]. Available: https://infoplattform-personenbahnhoefe.deutschebahn.com/resource/blob/7717102/50134ccc1abacab3537e223b15f1b57e/BIM-Vorgaben-Anlage-A-data.pdf (visited on 07/16/2024).

[53] D. c. Staff, *7 key benefits of open source software*, Developer.com, Aug. 2017. [Online]. Available: https://www.developer.com/project-management/open-source-software-benefits/.

[54] *What to know about open-source software: Benefits and advantages*, Heavybit. [Online]. Available: https://www.heavybit.com/library/article/open-source-software-benefits-advantages/.

[55] *What is open source software (oss)?* GitHub Resources. [Online]. Available: https://resources.github.com/open-source/what-is-open-source-software/.

[56] J. Reock, *Top 5 benefits of open source software*, OpenLogic by Perforce, Apr. 2020. [Online]. Available: https://www.openlogic.com/blog/top-5-benefits-open-source-software.

[57] *Australia and new zealand bim best practice guidelines*, 2018. [Online]. Available: https://bim.natspec.org/images/Article_files/Resources/Partner_documents/ANZIQS_2018__BIM_Best_Practice_SoftCopy_FINAL.pdf.

[58] *Building information modelling (department of transport and main roads)*, sc-tmrwcmgr-cd.azurewebsites.net. [Online]. Available: https://www.tmr.qld.gov.au/business-industry/Technical-standards-publications/Building-Information-Modelling.

[59] *Building information modellinge - buildings department*, Bd.gov.hk, 2018. [Online]. Available: https://www.bd.gov.hk/en/resources/codes-and-references/building-information-modelling/index_statutory_submissions.html (visited on 08/14/2024).

[60] A. J. Reis and, *Bridge Design*. John Wiley  Sons, Apr. 2019.

[61] J. Mitchell, *Main parts of a bridge - explained — engineeringclicks*, EngineeringClicks, Aug. 2018. [Online]. Available: https://www.engineeringclicks.com/main-parts-of-a-bridge/.

[62] C.-E. Tolmer, C. Castaing, Y. Diab, and D. Morand, “Adapting lod definition to meet bim uses requirements and data modeling for linear infrastructures projects: Using system and requirement engineering,” *Visualization in Engineering*, vol. 5, Nov. 2017. doi: 10.1186/s40327-017-0059-9.

[63] *Natspec bim - open bim object standard*, bim.natspec.org. [Online]. Available: https://bim.natspec.org/documents/open-bim-object-standard (visited on 07/16/2024).

[64] S. N. Zealand, *Timber framed buildings : superseding NZS 3604:1999*. Standards New Zealand, 2011.

[65] *Construction specification for bored and driven piles*, New Zealand Geotechnical Society. [Online]. Available: https://www.nzgs.org/libraries/construction-specification-for-bored-and-driven-piles/ (visited on 07/16/2024).

[66] F. H. Administration, *Design and Construction of Driven Pile Foundations Volume I*. Lulu.com, Nov. 2012.

[67] *Si units*, Measurement Standards Laboratory. [Online]. Available: https://www.measurement.govt.nz/metrology/si-units/.

[68] X. Zhou, J. Zhao, J. Wang, *et al.*, “Outdet: An algorithm for extracting the outer surfaces of building information models for integration with geographic information systems,” *International Journal of Geographical Information Science*, vol. 33, pp. 1444–1470, Feb. 2019. doi: 10.1080/13658816.2019.1572894. (visited on 03/10/2020).

[69] C. S. Ogueta and M. C. Sweden, “Object interaction query: A context awareness tool for evaluating bim components’ interactions,” Jan. 2013. doi: 10.5151/despro-sigradi2013-0050. (visited on 07/16/2024).

[70] M. E. Gendy, H. Ibrahim, and I. E. Arabi, “Analysing rectangular pile using one dimensional finite element,” *Geotechnical and geological engineering*, vol. 38, pp. 4617–4635, Apr. 2020. doi: 10.1007/s10706-020-01314-5. (visited on 07/16/2024).

[71] *As 5100.3:2017 bridge design foundation and soil-supporting structure*, www.intertekinform.com. [Online]. Available: https://www.intertekinform.com/en-au/standards/as-5100-3-2017-99574_saig_as_as_209333/ (visited on 07/16/2024).

[72] S. Lobo-Guerrero and L. E. Vallejo, “Influence of pile shape and pile interaction on the crushable behavior of granular materials around driven piles: Dem analyses,” *Granular Matter*, vol. 9, pp. 241–250, Mar. 2007. doi: 10.1007/s10035-007-0037-3. (visited on 10/17/2021).

[73] *Iso/cd 16739:2011(e)*, standards.buildingsmart.org. [Online]. Available: https://standards.buildingsmart.org/IFC/RELEASE/IFC4/FINAL/HTML/link/ifcpile.htm (visited on 07/16/2024).

[74] *What are the sizes of precast piles?* Centrum Pile Ltd, May 2021. [Online]. Available: https://www.centrumpile.co.uk/what-are-the-sizes-of-precast-piles/.

[75] *Hume concrete - hume rc square pile*, Sep. 2014. [Online]. Available: https://chuanhuat.com.my/w3/wp-content/uploads/2014/09/concretePiles.pdf.

[76] *Precast Concrete Handbook*. 2009.

[77] P. Standards Association of Australia. Committee CE/18, *Piling - Design and Installation*. 1995.

[78] A. Abebe and I. Smith, *Pile foundation design: A student guide*.

[79] M. Ekström, "Pile group design: An investigation and development of a software to optimize the design of pile groups," *odr.chalmers.se*, 2021. [Online]. Available: https://odr.chalmers.se/items/c5b7c90a-7bab-4771-8aed-8a7503241360 (visited on 07/16/2024).

[80] P. Guyer, *An Introduction to Pile Foundations for Structures*. Guyer Partners, Dec. 2017.

[81] A. R. Association, *Bridge Design: Concrete (AS 5100.5-2004)*. 2004.

[82] S. N. Zealand, *Concrete Structures Standard*. 2017.

[83] K. Rao, *Foundation Design*. John Wiley Sons, Dec. 2010.

[84] C. E. Reynolds and J. C. Steedman, *Reinforced Concrete Designer's Handbook*. 1976.

[85] *Understanding the basics: What is a pile cap in construction?* Utilities One. [Online]. Available: https://utilitiesone.com/understanding-the-basics-what-is-a-pile-cap-in-construction.

[86] A. A. O. S. Highway and T. Officials, *AASHTO LRFD bridge design specifications, Part I: Sections 1-6*. American Association Of State Highway and Transportation Officials, 2017.

[87] S. B. Magade and R. K. Ingle, "Influence of clear edge distance and spacing of piles on failure of pile cap," *Iranian Journal of Science and Technology, Transactions of Civil Engineering*, vol. 44, pp. 1265–1281, Jun. 2019. doi: 10.1007/s40996-019-00285-9. (visited on 02/27/2021).

[88] A. R. Association, *Bridge Design: Steel and composite construction (AS 5100.6-2004)*. 2004.

[89] *Russiangost—official regulatory library - sp 24.13330.2011*, www.russiangost.com. [Online]. Available: https://www.russiangost.com/p-20780-sp-24133302011.aspx (visited on 07/16/2024).

[90] G. Toniolo, M. Di Prisco, and M. W. T. Mak, *Reinforced concrete design to Eurocode 2*. Springer, 2017.

[91] M. Angar, *Reinforcement layout in concrete pile foundations a study based on non-linear finite element analysis kth royal institute of technology mohammad mustafa angar reinforcement layout in concrete pile foundations kth 2020*. [Online]. Available: https://kth.diva-portal.org/smash/get/diva2:1453339/FULLTEXT01.pdf.

[92] S. A. O. Australia, *AS 3600-1988 concrete structures*. Standards Assoc. Of Australia, 1988.

[93] S. A. of Australia, *Concrete Structures for Retaining Liquids*. 2001.

[94] S. A. (Organization and S. N. Zealand, *Precast concrete pipes (pressure and non-pressure)*. Standards Australia, 2007.

[95] D. Mcdonald, E. Ariyachandra, B. Eliasson, and J. Kwong, *Concrete cover -the what, where and when*. [Online]. Available: https://www.ciaconference.com.au/concrete2021/pdf/full-paper_74.pdf.

[96] W.-F. Chen and L. Duan, *Bridge engineering handbook*. Crc Press, Taylor Francis Group, 2014.

[97] R. Hammond, *Dictionary of Civil Engineering*. 1968.

[98] J. J. Zhao and D. E. Tonias, *Bridge Engineering, Third Edition*. McGraw Hill Professional, Apr. 2012.