Journal of Computational Design and Engineering, 2021, 8(1), 343–361

doi: 10.1093/jcde/qwaa083
Journal homepage: www.jcde.org
Advance Access Publication Date: 18 December 2020

RESEARCH ARTICLE

Development of a rule-based system to enhance the data consistency and usability of COBie datasheets☆

Vishal Kumar* and Ai Lin Evelyn Teo

Department of Building, National University of Singapore, 4 Architecture Drive, Singapore 117566, Singapore

☆Selected paper from the International Congress and Conferences on Computational Design and Engineering 2019, 7-10 July 2019, Penang, Malaysia
*Corresponding author. E-mail: e0011149@u.nus.edu

Abstract

Building information modeling (BIM) for facilities management (FM) has been gaining considerable attention. Construction operations building information exchange (COBie) datasheets are conceptualized as an electronic format of data for FM handover extracted from the BIM model and supplemented with information from other sources. To build an efficient COBie datasheet, it is advocated to build and verify data at all stages of design and construction, commonly known as data drops. Nevertheless, data consistency verification is a difficult task pertaining to COBie's complex structure and data representation. This study aims to understand the challenges associated with the COBie datasheet verification and consistency checking process, especially during data drop stages, and develop a solution to mitigate these challenges. The study uses a combined methodology of design thinking and waterfall model from the software development process. The outcome of the research study manifests in a prototype application. The prototype application can help in verifying COBie datasheet consistency during data drop stages. Additionally, this study proposes a new dimension to utilize the COBie datasheet to track various asset-related changes in a project by comparing COBie datasheets and visualizing this data in a visually interactive manner using a property graph model.

Keywords: building information modeling; COBie; COBie data drops; information management

1. Introduction

Building information modeling (BIM) has been exemplary in transforming how construction data is handled, exchanged, and collaborated between different stakeholders in the architectural, engineering, and construction (AEC) industry. BIM authoring tools have rendered more significant benefits to AEC professionals through features like clash detection, cost analysis, quantity take-off, design options, energy analysis, occupational behavior modeling, optimized construction sequencing, algorithm-based design, and automated code checking (Kim, Anderson, Lee, & Hildreth, 2013; Choi, Kim, & Kim, 2015; Pärn, Edwards, & Sing, 2017; Anand, Sekhar, Cheong, Santamouris, & Kondepudi, 2019; Caetano & Leitão, 2019; Kim, Lee, Shin, & Choi, 2019). With BIM gaining adoption at a fast pace, especially during design and constructions stages, there is growing interest in using BIM during post-occupancy stages of a building such as facilities management (FM), renovations, and waste recycling (Mohandes, Abdul Hamid, & Sadeghi, 2014; Volk, Stengel, & Schultmann, 2014; Patacas, Dawood, Vukovic, & Kassem, 2015; Kumar & Teo, 2019a). This will ensure that BIM will be used in the building's complete life cycle (Herr & Fischer, 2019). Additionally, it will ensure a reduced operational cost, access to accurate information, a central repository of operational data, and reduced reliance on the workforce for data know-how (Becerik-Gerber, Jazizadeh, Li, & Calis, 2011).

With several potential benefits highlighted in various publications, a wide range of challenges were also identified with

Received: 5 January 2020; **Revised:** 13 October 2020; **Accepted:** 30 October 2020

BIM for FM usage. It includes inconsistent guidelines for asset data capture, inadequate data integration due to multiple and different schemas, syntax, semantics, and lack of systematic procedure for information integration from various sources (McArthur, 2015; Yalcinkaya & Singh, 2019a). Teicholz (2013) highlighted that most of the issues associated with BIM for FM use could be mitigated effectively using information management and information exchange standards. Industry Foundation Class (IFC), a vendor-neutral data model, is developed as a standardized digital description of assets for collaboration and information sharing during the entire project life cycle (Vanlande, Nicolle, & Cruz, 2008).

Nevertheless, IFC is too generic and complex, designed to cater to many different configurations and levels of detail (Koo & Shin, 2018). Model view definition (MVD) is a subset of IFC schema, designed for a specific purpose and narrows the information required for the specified workflow (Chipman, Liebich, & Weise, 2012). Some examples of such configuration of data requirements for specific purposes are clash detection, costing, fire calculations, and scheduling. Construction operations building information exchange (COBie) is an MVD designed for FM and asset management (East, Nisbet, & Liebich, 2012). The focus of COBie is to capture "minimum common data" required for maintenance activities during the FM stage. It emphasizes the need for collecting data during the entire design and construction stages in a specified format, which can be used by FM professionals during operation and maintenance (O&M) stages.

COBie has been gaining significant attention from academia, industry, and government bodies. Professionals have started realizing the importance of information capturing inside the BIM model besides its most intended use as a 3D visualization of the project. Such a shift in perception has ensured that BIM models need not be "over-detailed" in terms of geometrical representation, and it must strike a balance between level of detail (LoD) and level of information (LoI). Since COBie is a non-geometrical data, it emphasizes more on high LoI. To facilitate capturing data for COBie, COBie guide and COBie responsibility matrix help identify what information is needed, at what stages, and by which stakeholders (East, 2013a; East & Carrasquillo-Mangual, 2013). Thus, there is a significant reliance on the professionals to ascertain that the data collected is in the right format and verified for consistency.

East (2013a) highlighted that the current FM data handover is highly disconnected from the design stages. Various relevant information, which could have been useful during the O&M stages, is lost, as it is not captured efficiently during design stages. A similar view of capturing FM data right from the beginning of the design stage was supported by other researchers (Lavy & Jawadekar, 2014; Liu & Issa, 2014). In the current scenario, while the LoD increases as the construction stage progresses, the LoI remains a less priority during the design stages and only sees a surge in capturing information before the handover stage (Kumar & Teo, 2020a). To mitigate this issue, East (2013a) proposed the concept of COBie "Data Drops." In this process, the COBie datasheet is filled and extracted at pre-defined stages to ascertain that the COBie data capturing is well within the set guidelines. COBie responsibility matrix can be used to ascertain what data in the COBie spreadsheet need to be captured during different stages.

Though the idea of data drops is benevolent, its implementation is challenging and needs extra time and effort. The data drops are designed around the concept that as the design will progress, LoI related to each asset will increase, and therefore, it is being populated gradually. On the other hand, design changes are synonymous with construction projects. Such changes arise from various factors that can be external such as environmental, political, economic, or internal such as design change by owners, consultants, engineers, unforeseen circumstances on-site, or safety requirements (Yana, Rusdhi, & Wibowo, 2015). Incurring design changes means that data inside the COBie datasheet at each data drop will change from the previous stage, contradicting the normative belief that it is only addition to the previously captured data. This implies that each data drop is a mixture of new and old data and also means that each time several manual entries need to be done to complete the COBie deliverables at each data drop. Even though manual entries give the flexibility to complete the COBie datasheet, it is highly prone to error. Furthermore, data inside the COBie datasheet is spread across multiple workbooks with connected links; ascertaining data consistency due to manual entry is a big task.

This paper is aimed at enhancing the usability aspect of COBie data drops. COBie data drops at any stage should not be seen in isolation but in relation to previous data drops. This entails two challenges around COBie data drops: first, verification of consistent data inside the COBie datasheet at each data drop, and second, what data has changed from previous stages. Identifying data changes inside the COBie datasheet and verifying it is a daunting task. COBie datasheet can be composed of thousands of rows of data, spreading across multiple workbooks. COBie data drops should not be seen as an intermediate step toward a single final FM data handover goal only. A practical data mining of COBie data drops can help in providing useful project-related information.

This paper begins with a brief review of the literature, which highlights the complexities and challenges associated with the COBie datasheet. The challenges identified during the literature review were verified using exploratory research, in which a simulation study was conducted with a BIM model. Following the review, an analysis of possible technical solutions was brainstormed, and finally, an application framework was developed, followed by a prototype. The prototype has been validated using a BIM model provided by the National Institute of Building Sciences (NIBS) (East, 2011). NIBS provides these BIM models for testing all COBie-related developments.

This research is a part of ongoing development, termed "COBie Dataset Management System (CDMS)." CDMS consists of various modules. The modules discussed in this paper are part of the proposed CDMS. Discussion on some of the other modules of CDMS was presented earlier in other published articles (Kumar & Teo, 2019a, Kumar & Teo, 2019b). Likewise, a discussion on the entire proposed CDMS system and some additional modules will be presented in forthcoming articles. This paper is solely dedicated to developing one of the modules (verifier and compare) as it requires the full attention of its own. Henceforth, only information related to this development is presented in this paper (including literature review). For the development of the entire CDMS system, a questionnaire survey was conducted among COBie professionals to identify the benefits and issues associated with COBie datasheet handling. The survey details can be found in our article (Kumar & Teo, 2020b). In addition to the verifier and compare module, a brief introduction to the visualizer module is discussed. The visualizer module introduction is necessary because data output from the verifier and compare module can be envisaged in a node-link diagram using the visualizer module. However, the visualizer module's technical development will be discussed separately in future upcoming article as it needs attention of its own.

2. Research Design and Approach

The following objectives were set forth for the study, to identify the challenges associated with COBie data drops and develop a

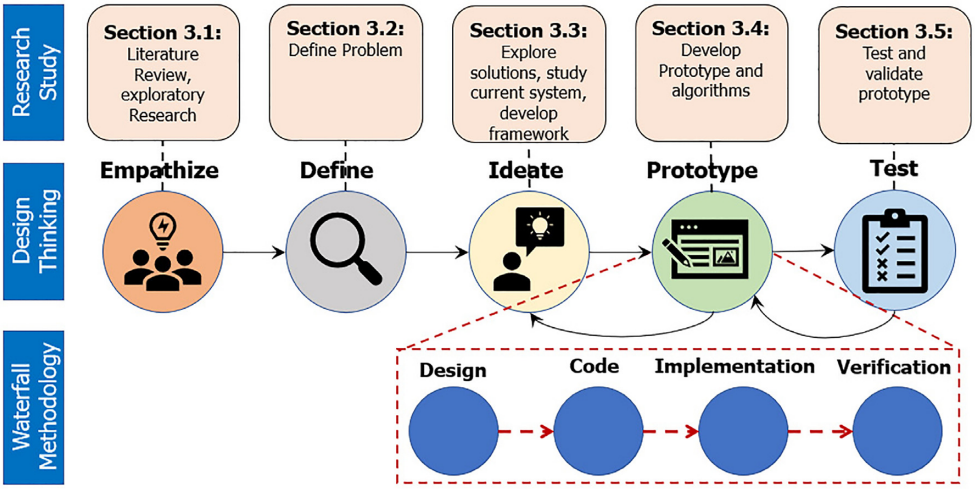

Figure 1: Research methodology and steps.

framework (with a working prototype): (i) review the complexities and challenges associated with COBie data drops; (ii) identify the common design changes in a project and its effect on the COBie datasheet; (iii) study existing tools available in the public domain and identify the challenges; (iv) develop a framework for a prototype tool to address the identified challenges; and (v) validate the prototype using a BIM model in a simulated environment.

Since the study required a more in-depth study about the existing issues with iteration over the possible solutions, a combination of design thinking methodology and the waterfall model (for prototype development) was used in this search. Design thinking was applied as an exploratory research process that helped in formulating the research problem. It is defined as an iterative problem-solving approach, meant to explore diversified ideas (Dorst & Cross, 2001). Both the methodologies (design thinking and waterfall) share similar stages; however, design thinking is more focused on developing and exploring divergent ideas (Lindberg, Meinel, & Wagner, 2011). The waterfall model is a type of software development process based on sequential steps (Balaji & Murugaiyan, 2012). Since the research study outcome manifests in a tool, based on the requirements developed using the design thinking process, the waterfall model of development was used for tool scripting and validation. The waterfall model of development is more suited for a time-bound development where the requirements are set (Petersen, Wohlin, & Baca, 2009). Using the waterfall model, the development of flowcharts, codes, implementation, debugging, and verification was conducted. Section 3.4 discusses in detail the activities conducted using the waterfall model. Figure 1 shows how both the methodologies were merged along with activities during each incremental step.

3. Development of Framework and Prototype

This section will discuss the various steps followed in this research study. As mentioned earlier, the study involved five necessary steps: (i) *Empathize*: understanding the problem context. (ii) *Define* the formulation of questions for the desired solution. (iii) *Ideate*: an exploration of ideas to develop the necessary solutions. (iv) *Prototype*: development of ideas into a workable, tangible form. (v) *Test*: prototype testing and feedback for refinement.

3.1 Empathize: understanding the problem context

An in-depth literature review was conducted to understand the problem context. The purpose of the literature review was to understand what are the current challenges associated with the COBie datasheet. A thorough review of published articles (journals and conferences), blogs, forums, and user groups was conducted to identify the challenges. The identified challenges were then discussed with experts who are using COBie in their project to cross verify. Additionally, an exploratory study was conducted using a BIM model provided by NIBS (mentioned earlier). The identified issues were simulated inside the BIM model, and respective COBie datasheets were extracted (to simulate data drops) using the COBie responsibility matrix as a guideline. The summary of the identified issues is discussed in the following sub-sections.

3.1.1 *COBie data representation in spreadsheet format*

COBie data can be exported in three different formats, i.e. IFC STEP (standard for the exchange of product model data) Physical File Format, ifcXML, or SpreadsheetML (open XML schema used by spreadsheet applications) (Yalcinkaya & Singh, 2019a). COBie datasheet is a complex representation of connected data, spreading across 20 workbooks, each having multiple columns, and each column of data is either unique or is connected with other workbooks (East, 2013a). Currently, such representation poses multiple challenges in understanding the data. For example, even though the data inside COBie datasheet is connected, professionals need to figure out themselves using multiple filtering and sorting inside different workbooks to find the relevant data (Yalcinkaya, Singh, Nenonen, & Junnonen, 2016; Kumar & Teo, 2019a). Similarly, identifying the missing link in the connected data is difficult, as this needs to be done manually (Kumar & Teo, 2020a).

Yalcinkaya et al. (2016) highlighted that the spreadsheet format is the most widely used among the three file formats in which COBie can be delivered. The primary data inside the COBie sheet is meant to come from a BIM model; nevertheless, to have a complete deliverable, data from other sources need to be input inside the COBie datasheet. A typical COBie deliverable contains 20 sheets, whereas BIM authoring software COBie export function supports data for fewer workbooks (e.g. Autodesk Revit can export only data for 10 workbooks). The data inside the

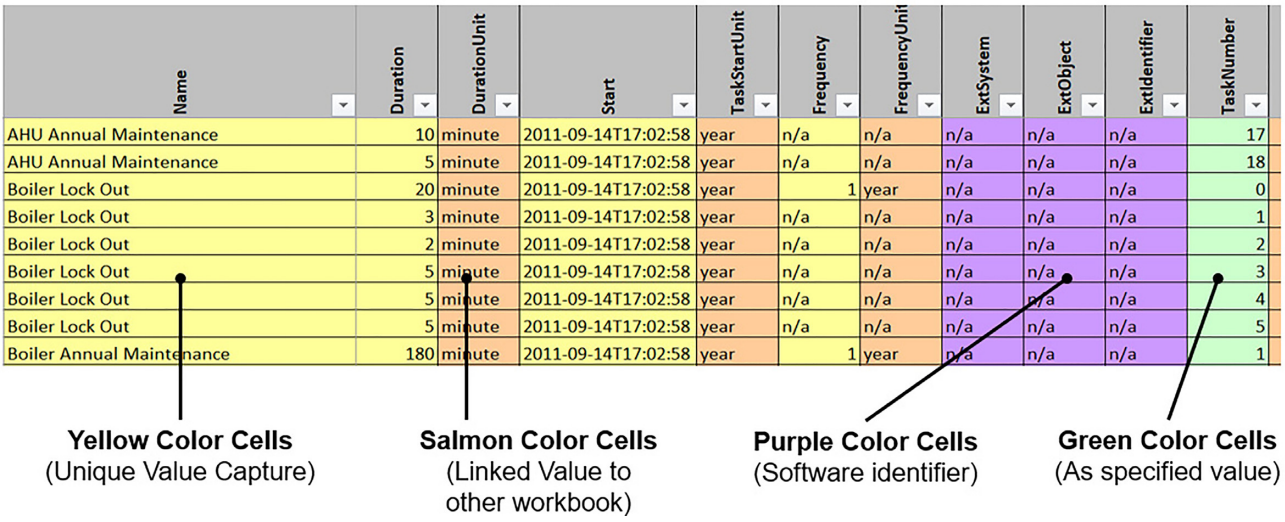

Figure 2: A typical COBie datasheet color-coding system (East, 2013a).

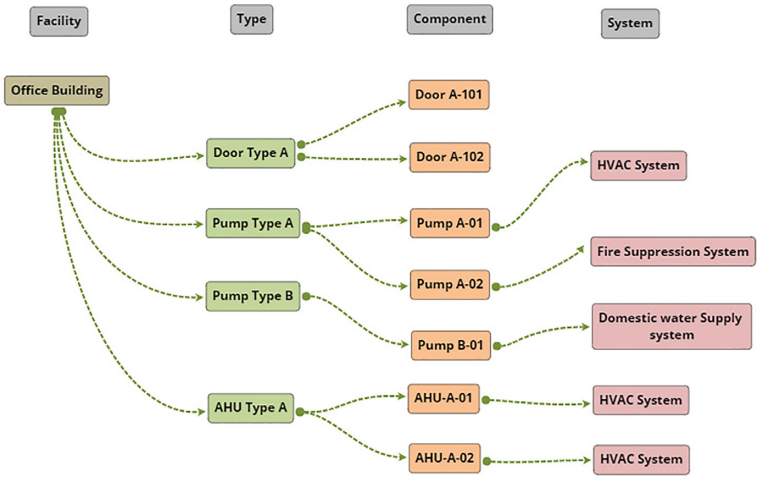

Figure 3: Example of COBie equipment organization (East, 2013b).

COBie datasheet is color-coded to signify different connections. Figure 2 shows a typical COBie workbook with different colors highlighting different columns.

The data representation of a COBie datasheet is strict and does not allow users to change its representation format. The order of the workbooks, columns inside the workbooks and color-coding, is in the same order always and need not be changed.

Most importantly, even though the data inside the COBie datasheets are represented separately, they are connected to data in other workbooks (East & Carrasquillo-Mangual, 2013). So, the salmon-colored columns represent connected data, but a user needs to know where to find the connected data. Finding connected data inside the COBie datasheet is a daunting task (Yalcinkaya & Singh, 2019b). The connected data is spread across different workbooks. Figure 3 shows an example of how a piece of information spread across different workbooks is connected.

3.1.2 *Challenges in handling and verifying large datasheets*

COBie datasheet collects a vast amount of data, which is interconnected and spreads among different workbooks. Once the project progresses, the data inside the COBie datasheet also changes. These changes occur in various workbooks since the change in one data affects others. Identifying these changes in the COBie workbook is a considerable task. Identifying a change will mean that a person needs to conduct multiple actions such as zoom in/out, scroll down, and filtering and sorting. These challenges related to spreadsheet format have been highlighted previously by various researchers (Henderson Jr & Card, 1986; Woods & Watts, 1997; Yalcinkaya & Singh, 2019a). When a large amount of data is stored in a spreadsheet format, identifying linked data changes becomes challenging (Yalcinkaya, 2017).

Not all data is captured inside the COBie sheet extracts from BIM model automatically. A considerable amount of data needs to be captured manually. Manual capturing of data can lead to an error because a user needs to ascertain that all manually captured data dependency and linkage are adequately represented in all workbooks (Kumar & Teo, 2020b). Identifying such changes and ascertaining that all relevant data is captured accurately are challenging tasks to conduct and need high cognitive understanding to verify data accuracy.

3.1.3 *Issues with COBie data drops*

COBie data drop involves extracting the COBie datasheet at pre-defined stages to ascertain and verify the COBie capturing

process and data consistency. To guide a COBie data drop, various documents such as the COBie guide and COBie responsibility matrix are used. Even though the process looks benevolent in its idea, COBie data drops have multiple challenges. Nevertheless, before discussing these challenges in detail, it is necessary to understand the process of COBie data drop.

Different authors and organizations have described the stages of data drops in different ways. For example, East (2013a) identified the data drops stages at as-planned, as-designed, as-constructed, as-occupied, as-built, and as-maintained (six stages). Likewise, East and Carrasquillo-Mangual (2013) mentioned only four stages for data drops. They defined them as design development deliverables (35% design), construction document design deliverable (100%), beneficial occupancy construction deliverable, and as-built construction deliverable. BIM task group (2012) proposed five data drop stages with elaborate descriptions of expected outcomes. They linked the data drops with the RIBA plan of work (RIBA PoW)-2007 stages for better clarity. These previous studies did not link all the RIBA stages with data drops, creating confusion about the remaining stages. RIBA PoW organizes the "process of briefing, designing, constructing, and operating building projects into eight stages and explains the stage outcomes, core tasks and information exchanges required at each stage" (RIBA, 2013; Sinclair, 2019). Alnaggar and Pitt (2019) have combined PMI stages, RIBA PoW, and Cobie data drop highlighting that it is critically important to map COBie data drop with RIBA PoW to identify the roles and responsibilities of all stakeholders clearly at all stages. However, in all these studies, there is a lack of transparent methodology to connect COBie data drops with RIBA PoW, especially not justifying why certain stages were not part of data drops and what broad activities should happen at these stages. This study intends to fill this gap and proposes some new additions to the COBie data drops process sequentially with RIBA PoW-2013 and AIA PoW-2012 (discussed in detail in Section 3.3.1).

Moreover, each time a COBie data drop is generated, it needs an extensive verification to establish that the COBie datasheet is consistent (Yalcinkaya et al., 2016; Kumar & Teo, 2019a). An exploration study by using BIM Model provided by NIBS for COBie-related research was conducted (East, 2011). The purpose of the exploration study was to understand the structure, linkage, and data source in the COBie datasheet. Additionally, testing of several verifiers available in the market, such as ONUMA COBie validator (Onuma, 2013) or the COBie QC tool (East, 2016), was conducted. As a result of this exploratory study, it was identified that the current verifiers are designed to check the consistency and linkage values for validation. Nevertheless, these verifiers miss identifying several missing and incorrect values inside the COBie datasheet. For example, identifying missing attributes, incorrect data format, cross verifying with picklists, logical verifications such as steps in jobs workbook, cross verifying from previous data drops, identifying unlinked data, and so on.

Based on these preliminary identifications through exploratory study, brainstorming efforts were performed. The brainstorming efforts were held at two intervals. It involved five members, including the authors and three industry experts who have prior experience in handling COBie datasheets. All three professionals have over eight years of experience in the construction industry using BIM and more than three years of handling COBie. In the first brainstorming part, the challenges were discussed, and broad categories were identified for verification purposes. In the second part, three broader ideas of development that should be linked together, i.e. development of a verifier module, which can check for data consistency; development of a compare module; and development of a visualization module, which can help in visualizing the identified data interactively, were identified.

Based on the discussion in the brainstorming efforts, the following five categories were identified on which a "COBie verifier" should function to conduct a deep-level verification:

(i) *Duplicate data*: Since COBie requires merging of data from all trades (architecture, MEP (Mechaincal,Electrical and Plumbing), landscape, infrastructure) into one single file and manual entry is possible in COBie datasheet; duplication of data is possible.
(ii) *Incomplete data*: Manually entered data are prone to missing linkage data inside the COBie datasheet.
(iii) *Missing data*: Since data is captured manually, there is a possibility of data missing from future data drops that were captured in previous data drops because they are not coming from the BIM model.
(iv) *Incorrect data*: Data captured in the COBie data drop is the incorrect format.
(v) *Data dependency and link*: Data captured at every workbook is linked correctly to various workbooks.

Besides identifying these categories on which the verification should work, it was also identified that the verification system should not only identify the incorrect data but also inform users about various missing information that should be captured in the COBie datasheet. This will ensure a better data capturing process. A COBie capturing process requires COBie capturing guidelines and information capturing requirements, which can differ based on the owner's preferences. Alternatively, one can refer to the COBie guide for equipment and its minimum parameters that need to be captured. Therefore, the proposed verification system is supported by several clusters of a database. The database captures the COBie guide parameters, owner's requirements of parameters, and stores information about standardized data that need to capture in all the projects. This database intends to support the verification process more holistically as well as makes users carry out informed decisions.

Similarly, for "COBie compare," the identified functions were to compare two COBie datasheets to identify changes. For COBie visualizer, the identified functions were to visualize data in a node-link diagram and store historical data.

3.1.4 *Understanding changes in COBie datasheet*

Design changes are widespread in a construction project, implying that each time a design changes, the data inside the COBie datasheet will also change, resulting in a mix of old data, new data, and deleted data. In many cases, such changes are difficult to identify from the drawings itself because they might be happening at the specification level, which is not easily visible in drawings.

To identify such changes, a COBie datasheet needs to be compared with the previous COBie datasheet to see what data has changed. Nevertheless, such comparison is challenging because the addition/deletion of data results in a random movement of data (cell location) inside the workbooks when extracted from BIM software as data drop. Moreover, such changes affect multiple workbooks inside a COBie datasheet. Therefore, the usual compare functions of spreadsheet software cannot be applied to it.

Even though such comparison is challenging, performing a comparison of data can have multiple benefits. For example, during a tendering stage, designers often issue an addendum to design changes. Merely comparing the COBie datasheets can

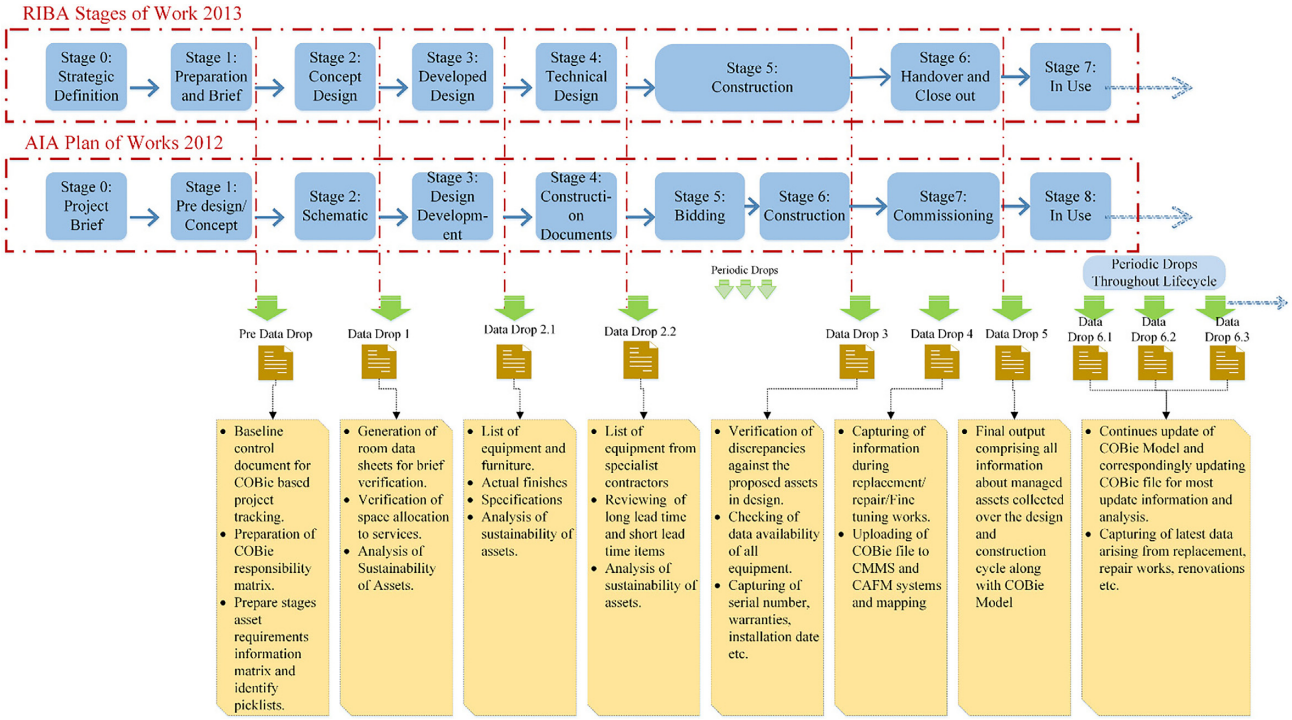

Figure 4: Proposed COBie data drop in synchronization with the RIBA plan of works 2013 and AIA plan of works 2012.

help understand what changes have been made in the design, especially at the asset level. Stewart (2014) mentioned that COBie could be used for carbon calculation. Thus, by comparing the COBie datasheet, one can ascertain how carbon calculation has changed. Similarly, Biswas and Krishnamurti (2012) highlighted that COBie data could be linked to LEED calculations. Thus, we can ascertain how LEED performance has changed for a project by using the compare function.

3.2 Define: delineating the problem statement

Section 3.1 discussed the problem context of COBie datasheets and COBie data drops. Based on these understanding, the following problems were identified that needed attention:

- Difficulty in verifying the data consistency of a COBie datasheet during various data drops.
- Difficulty in finding what information has changed inside the COBie datasheet during data drops
- Finding a reverse relation and link between data changes inside a COBie datasheet to the architectural design changes, i.e. identifying architectural design changes using COBie datasheet as a base document.
- Visualizing the data changes inside a COBie datasheet in a visually interactive manner for easy understanding.

3.3 Ideate: determining the algorithmic functionality

In the ideation stage, a series of options were explored to devise solutions for the problem statements. This included reviewing the existing verification systems available in the market, analyzing their process flaws, and identifying gaps. Based on the analysis, the functionalities of the proposed modules were identified. Feedback from three professionals was taken on the identified functionalities for these modules. The below sub-sections discuss in detail the conceptualization of the verifier, compare, and visualizer module. However, before discussing the modules' conceptualization, Section 3.3.1 discusses the new proposed data drop process flow, in line with RIBA and AIA PoWs addressing the gap highlighted in Section 3.1.3.

3.3.1 *Proposing new data drops stages*

As highlighted in Section 3.1.3, a new outline for COBie data drop in line with RIBA PoW-2013 and AIA PoW-2012 is proposed in this research study. The selection of RIBA PoW-2013 and AIA PoW-2012 as the baseline has multiple reasons. The primary reason is that there is already a significant discussion about RIBA PoW and COBie synchronization in the published literature. Second, RIBA PoW steps are synced with the American Institute of Architects plan of work (AIA PoW-2012) (Jung, Häkkinen, & Rekola, 2018), but no published literature links AIA PoW to COBie data drops. COBie is more widely used in the UK and the USA since these countries have adopted COBie since 2011 (East, 2013a). Therefore, by proposing synchronized COBie data drops with RIBA PoW and AIA PoW, this study tries to connect COBie data drops to the most widely used and developed Plan of Works for better clarity of the data drop process (Fig. 4).

In this proposed data drop process, a brief description of broad information, which needs to be captured, is presented. Additionally, few new items are proposed inside the process flow. First, there should be periodic data drops between data drop 2.2 and data drop 3. This will ensure that the data capturing process is smooth and not rushed before the handover stage. For large complex projects, there are hundreds of equipment. Periodic data drops will ensure that data is captured continuously, accurately, and systematically rather than at the end. Second, the data drop process should continue beyond the construction stage to the O&M stage. This will ensure that a COBie datasheet can be used for future renovation and demolition work (Kumar & Teo, 2019a). The updated COBie datasheet will also help to maintain an up to date database of managed assets and help in

scenarios where managing agents are changed after specific durations. Similarly, proposed data drop 4 is intermediate before the final data drop 5, which is at handover and closeout. Data drop 4 will happen after the trial run of equipment. This will ascertain that any updates after the trial run has been updated inside the COBie datasheet.

This proposed process flow highlights that for building a comprehensive COBie deliverable as an outcome, a COBie datasheet needs to be built continuously (Lavy & Jawadekar, 2014). Nevertheless, creating each data drop is not an easy process. Since it is semi-autonomous (i.e. only partial data is extracted from the BIM model), manual entry is required for a complete COBie datasheet. For example, Stewart (2014) mentioned that 40% of the data was entered manually in a project for COBie deliverable in a case study. Also, it is not bi-directional, i.e. the data entered at any data drop will not be captured in the BIM model. This implies that the datasheet at every COBie data drop stages need to be verified every time to ascertain that all the data have been captured from the previous data drop (Kumar & Teo, 2020a).

3.3.2 *Conceptualizing a verification module*

As discussed in Section 3.1.3, COBie data drops need an extensive verification module to ascertain that the data captured inside it is consistent. Thus, a need for the database was identified to support the verification function. The lack of a database limits the verification process only to the checking of value formats. The data verification needs to be conducted at two different levels: first, the data need to be checked for its consistency among different workbooks, and second, data need to be verified in comparison with previous COBie data drops. As mentioned previously, a COBie datasheet is not entirely extracted from the BIM model. Data from other sources need to be inserted manually. Such manual insertions are prone to error.

Different data drop stages require a certain number of columns to be captured as essential. Therefore, a COBie data drop selection function is added, which verifies data according to COBie data drop. The essential data requirements for each data drops are outlined in the COBie responsibility matrix, which is followed in this research as a baseline (East, 2013a).

Furthermore, the need for a function inside the verification module helps users identify missing data based on a rule-based database. The purpose of these features is not only to verify the data but also to help users in making a complete COBie datasheet.

The verification module was developed to carry out the following broad functions:

- *Check for duplicate data*: Various data inside the COBie datasheet is unique in nature. Such data is highlighted in yellow-colored cells. A function inside the module check that the data is unique in nature and no duplication has taken place. The uniqueness of data inside COBie is defined at two levels. In the first case, the data inside a column has to be uniquely called the primary key. Moreover, in some cases, the data after combining cell values from multiple cells have to be unique, hence called a compound key. For example, the "name" column inside the type, component, space workbook has a unique value. This means that all the values inside the column should not be repeated in the same column. Alternatively, the "name" column of the "attribute" workbook is not unique by itself. After concatenating the values from two columns, i.e. "name" and "RowName," the value should be unique. The verifier should check for duplication as according to the mapping rules defined by NBS-US mapping rules (NIBS, 2017). In addition to these values, other columns were identified in which duplicate values should be avoided. We have categorized them as warnings. Some examples are: *Type worksheet* (Warning): check model numbers of equipment; *Component workbook* (Warning): check the serial number of equipment, check tag number, check barcode number, asset identifier number; *Spare workbook* (Warning): check set number, Check part number.
- *Check for incomplete data*: Manual entry inside the COBie datasheet is common because BIM authoring tools do not export all the workbook information. Nevertheless, the data inside COBie are all linked, so manually input information must be verified with their dependencies. For example, a data entry inside the "document" workbook needs to be cross verified, whether such equipment is captured in the "Type" workbook. In a COBie datasheet, certain information can be categorized as "standardized information" that will not drastically vary with the project. For example, various equipment have standard job procedures such as "Annual AHU maintenance" or "semi-annual sprinkler maintenance." Similarly, the document workbook should have some required number of documents for different equipment types. For example, a "door" should have "O&M Manual," "Product Data," and "Warranty." Such information requirements are stored in the proposed database. The function uses this database to verify standardized data. The database can be customized to add any values which should be checked. When the function finds any equipment with an equal value in type workbook in the "name" and "category" column, it will check whether a "job description" or "document" for the same has been covered in the "job" or "document" workbook or not. For such deep-level verifications, some identified workbooks are *Resource workbook*: check all resources are linked; *Job workbook*: check jobs captured for all required equipment from the database, check the sequence of task numbers and priors; *Document workbook*: check all type of equipment has document covered, check document requirements for various type of equipment with the database.
- *Check for missing data*: COBie datasheet need not be seen as an independent datasheet during COBie data drop stages. At each data drop, a large amount of data is captured manually. Therefore, it is essential to compare the latest COBie datasheet with the previous COBie datasheet to ascertain whether data captured previously is captured in the current version or not. Furthermore, the COBie guide defines the minimum attributes that need to be captured for any equipment. Owners can define additional attributes for capturing information. The proposed database related to the verifier module stores standard attribute requirements for various equipment as prescribed in the COBie guide and owner specific requirements. By verifying the information from the COBie datasheet with the database, this function identifies the missing data. This function's algorithm is designed to search for partial name matching also for better catching of words. For such identifications, some workbooks that should be explicitly checked are *Document workbook*: check whether any document captured before is captured in the current data drop; *Attribute workbook*: check all attribute are covered with database requirements; *Coordinate workbook*: check whether the same defining attribute is missing from other equipment or space.
- *Check for incorrect data format*: Data in all the cells of COBie are written in a specific format. This format is defined by COBie

(a)

Name	CreatedBy	CreatedOn	Category	Description	AssetType	Manufactur	ModelNumb	
AC Unit Type 1	danielle.r	2011-09-1	23-75 10 2	Horiz. D.X. Fan Coil	Fixed	sales@mitsubishielectric.com	PC24EK1	sales@n
AC Unit Type 2	danielle.r	2011-09-1	23-75 10 2	Horiz. D.X. Fan Coil	Fixed	sales@dataaire.com	DAPA-2.5	sales@d
AHU	danielle.r	2011-09-1	23-75 35 1	63300000 J	Fixed	sales@York.com	AP-500	sales@Y
Air Compressor - Dental	mariange	2013-01-2	23-65 55 1	Duplex Packaged Assembly with Dryer	Fixed	manufacturerx@genericmanufacturer.com	DAC-M55	manufa
Air Cooled Chiller	danielle.r	2011-09-1	23-75 10 2	633-703 kW	Fixed	sales@York.com	YCAS0150EC46Y	sales@Y
Air Cooled Condensing Unit- Large	danielle.r	2011-09-1	23-75 10 2	3500 Watt Capacity	Fixed	sales@dataaire.com	DRCU-0312	sales@d
Air Cooled Condensing Unit- Small	danielle.r	2011-09-1	23-75 10 2	2500 Watt Capacity	Fixed	sales@mitsubishielectric.com	PUG24BKB	sales@n

(b)

Name	CreatedBy	CreatedOn	Category	Description	AssetType	Manufacture	ModelNumb	
AC Unit Type 1	danielle.r	2011-09-1	23-75 10 2	Horiz. D.X. Fan Coil	Fixed	sales@daikin.com.sg	ALDPV2883J	sales
AC Unit Type 2	danielle.r	2011-09-1	23-75 10 2	Horiz. D.X. Fan Coil	Fixed	sales@panasonic.com	PAN-2500	sales
AHU	danielle.r	2011-09-1	23-75 35 1	840002JK	Fixed	sales@GRUNDFOS.com	GR-F-2500	sales
Air Compressor - Dental	mariange	2013-01-2	23-65 55 1	Duplex Packaged Assembly with Dryer	Fixed	manufacturerx@genericmanufacturer.com	DAC-M55	manu
Air Cooled Chiller	danielle.r	2011-09-1	23-75 10 2	633-703 kW	Fixed	sales@York.com	YCAS0150EC46Y	sales
Air Cooled Condensing Unit- Large	danielle.r	2011-09-1	23-75 10 2	3500 Watt Capacity	Fixed	sales@dataaire.com	DRCU-0312	sales
Air Cooled Condensing Unit- Small	danielle.r	2011-09-1	23-75 10 2	2500 Watt Capacity	Fixed	sales@mitsubishielectric.com	PUG24BKB	sales

Figure 5: Sample excerpt of (a) before and (b) after of COBie sheet (type workbook) resulting from design changes (highlighted in purple).

Table 1: Examples of proposed rules and functionality of the modules (Kumar & Teo, 2020a).

Example rule	Functionality (affected workbooks)
1: Change in location of equipment	Component, system, coordinate
2: Complete removal of the equipment type	Type, component, systems, spare, job, document, attribute, coordinate
3: Component removal of an equipment type	Component, system, attribute, coordinate
4: Change in the system of equipment	System
5: Change in the zone of space	Zone
6: Change in properties of equipment	Type, spare, job, document
7: Change in room number of the room	Space, zone, component, document
8: Change in space name of the space	Space, zone, component, document
9: Addition of room or space	Space, zone, component, document
10: Addition of new equipment type	Type, component, systems, spare, job, document, attribute, coordinate

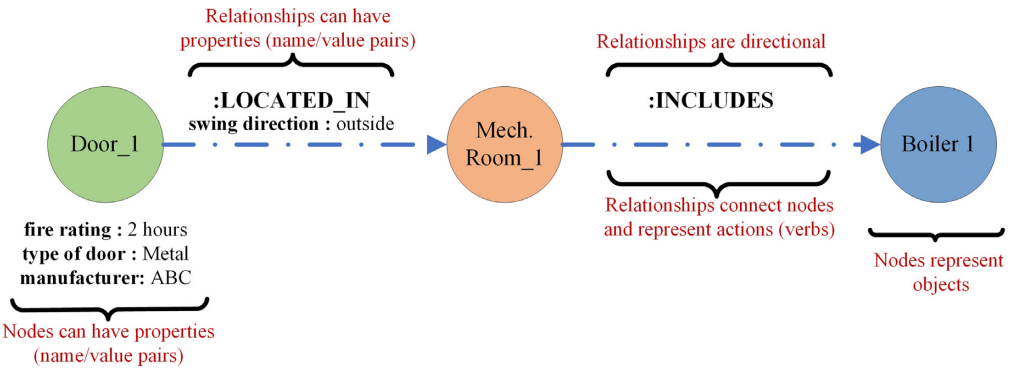

Figure 6: Example of building blocks of the property graph model.

NBS-US mapping rules (NIBS, 2017). For example, inside the type workbook, the "NominalLength" column values should be numeric. Therefore, the value in it should be only numbers such as "10.5" and not "10.5 m" or "10.5 meters." Similarly, each cell value should have a specific type of values such as alphanumeric, numeric, ISO date, or choose from picklists. This function's algorithm is designed to search the format of the data and find the discrepancy. For checking data format, it is necessary to match the data from the picklist. So, the function uploads the picklist into the memory and then runs to check the data format.

- *Check for data dependency and link*: Data presented inside the COBie sheet is linked to other workbooks. Therefore, all dependent data needs to be verified inside the COBie datasheet.

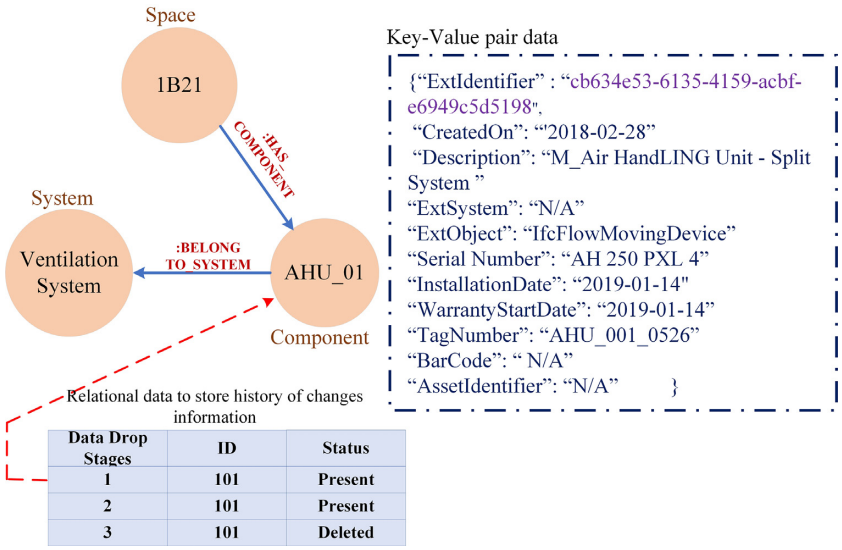

Figure 7: Representation of COBie data in the property graph model format.

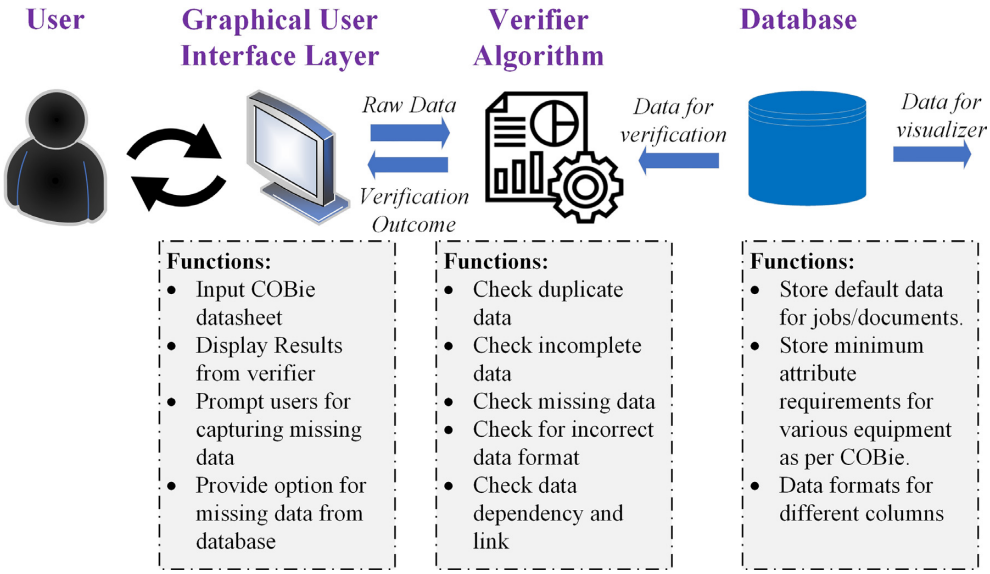

Figure 8: Implementation architecture for verifier module.

This functionality is significant because the COBie datasheet allows manual capturing, and data can be modified after it has been extracted.

Moreover, it is essential to note that the amount of data inside the COBie sheet increases as the project progresses. Therefore, the verification module was designed to check data based on the data drop stages. It ensures that the module does not generate errors beyond the relevant data drop stage. The development of the prototype application is discussed in Section 3.4.1.

3.3.3 *Conceptualizing a compare module*

While the verifier module was conceptualized to verify the data's reliability inside the COBie datasheet, the compare module was designed to understand how data has changed inside the COBie datasheet between two data drops. The central idea behind developing this module is to use and demonstrate that by comparing two COBie datasheets, we can ascertain the design changes happening in a project.

For the development of the compare module, a detailed analysis was conducted to understand how changing a workbook's value will affect the data inside other workbooks. A thorough study of each column of COBie workbooks was carried out. After that, a simulation study was conducted whereby design changes were simulated inside the BIM model, and their effect on the COBie datasheet was studied. After analyzing the data, the identified changes were clubbed into rule sets. Each rule sets run through different columns of different workbooks to perform the comparison. An example of the before and after extract has been shown in Fig. 5. This module's details are already covered in our previously published article (Kumar & Teo, 2020a). Therefore, only a brief description of the module is presented in this article.

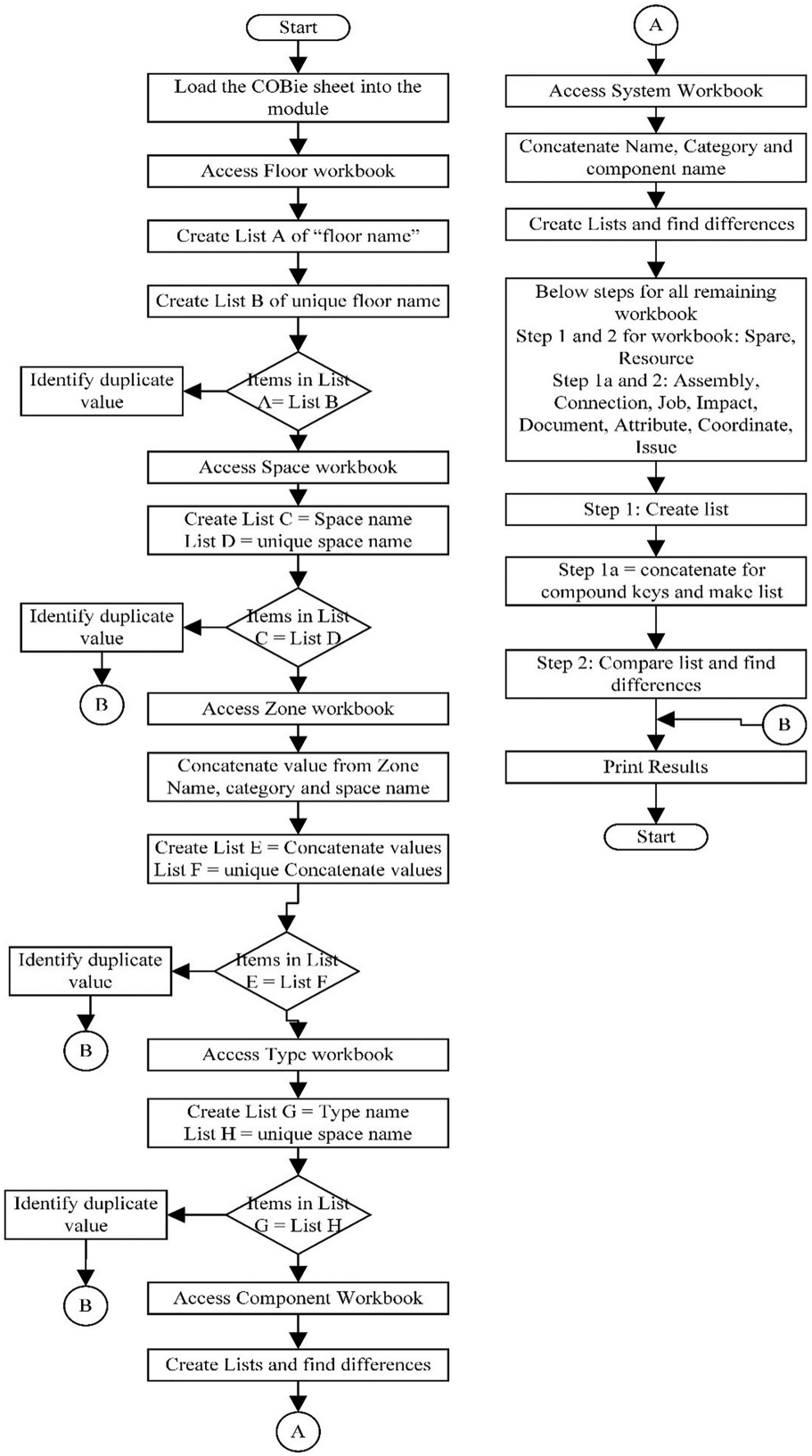

Figure 9: Sample flowchart developed for deduplication during verification.

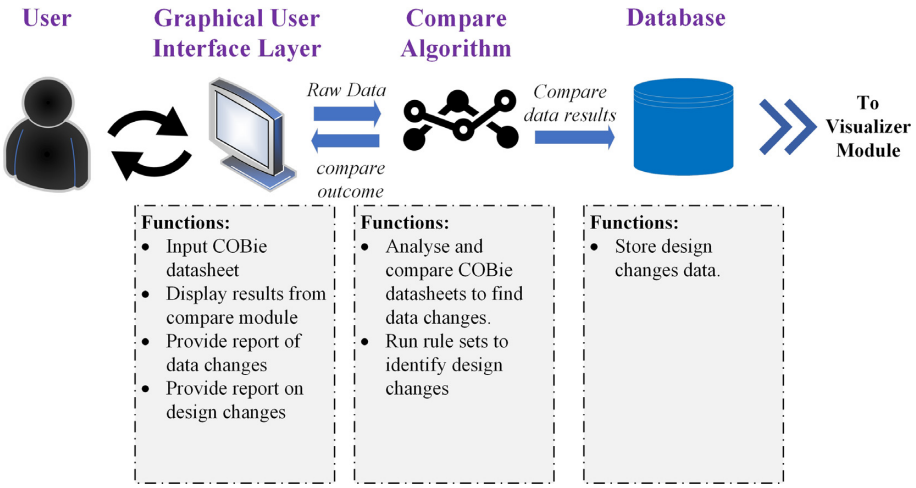

Figure 10: Implementation architecture for compare module.

An example of the rule sets (not inclusive) has been shown in Table 1. The table shows the design changes identified, and the COBie workbooks affected by the individual design changes. To make an optimized module, columns were identified that would be affected due to such design changes. For example, in Rule 1, a change in location of equipment will affect the columns "*Space*" in the component workbook, which might affect the "*Name*" in system workbook, and "*CoordinateXAxis*," "*CoordinateYAxis*," and "*CoordinateZAxis*" columns in the coordinate workbook.

These rules baseline both the verifier and compare modules. For many verification processes, it is necessary to conduct a compare function to ascertain data accuracy. For example, suppose a data is found to be deleted using the compare module. In that case, the verifier module uses this data to check the back and forth linkage to see whether all relevant linked data are deleted or not.

3.3.4 *Conceptualizing a visualization module*

While data inside a COBie datasheet is represented in a linked and dependent format, such representation is partially absent in a COBie datasheet. The current spreadsheet format of the COBie datasheet lacks visibility of data dependencies and semantics, resulting in high memory load to find relevant data, besides lacking query capabilities (Yalcinkaya et al., 2016). A visualization module was developed to tackle such issues, which is on the graph model (node-link). The module's development will be discussed separately in another article as it needs its own attention. Still, a brief description of the visualization module is discussed here. For this development, a type of graph model known as the property graph model was used. The property graph model is a type of node-link representation. It provides flexibility in storing data at nodes and edges that connect the nodes. An example of the property graph model is shown in Fig. 6.

To convert the COBie datasheet into a property graph model database, custom algorithms were developed. These algorithms helped in storing the data into the nodes. In addition, nodes are connected through edges whose relationships were defined. These custom algorithms can be used to convert any COBie datasheet into a property graph model automatically. Figure 7 shows an example of how the data from the COBie sheet is stored in the property graph model. Two nodes can be linked to each other through multiple relationships stored over edges. A SQL-based query can be run over this database to find relevant information based on any relationship edges. While this visualization module can be used independently to find any relevant information by running a query, the compare module's data will also be converted and stored inside the same database. Custom queries can run to see the results from the compare module in a visually interactive manner. A sample test case (test case 6) has been shown in Table 2 to showcase the functionality.

3.4 Prototype: development of algorithms for the modules

In this section, the process flow for the development of algorithms/scripts of the two modules, i.e. verifier and compare, will be discussed. Both the modules are developed using python programming language version 3.7. The choice for python was based on multiple accounts: (i) python is an open-source language; (ii) the modules will have the possibility to integrate with BIM authoring tools as common BIM authoring tools such as Revit, and ArchiCAD does support python language, and (iii) python has well-developed libraries to deal with spreadsheet format such as openpyxl (Gazoni & Clark, 2018).

3.4.1 *The system architecture of the modules*

Verifier module: The implementation architecture of the verifier module is shown in Fig. 8. The graphical user interface (GUI) layer interacts with the user. At this layer, the user can load the COBie datasheets and see the verifier module outcome. The inaccessible layer behind the GUI is the algorithm of the verifier module. The algorithm runs over the loaded COBie datasheet and analyzes the data consistency. The verifier algorithm interacts with a database, which adds a semantic layer to the verifier module. The database has been divided currently into three clusters. The first cluster stores information about the documents, standard job requirements, default attributes, custom attributes, and coordinates names required for COBie capturing. The second cluster store information about standard data formats which need to be checked. This is in addition to the picklists, which the program store in memory and flush it after the program run, verify and generate results. The third cluster store data of verification results and pass it to the visualizer module.

For developing the verifier module, a two-step process was followed. In the first step, for all different algorithms, a detailed process flow chart was developed. An example of such flowchart

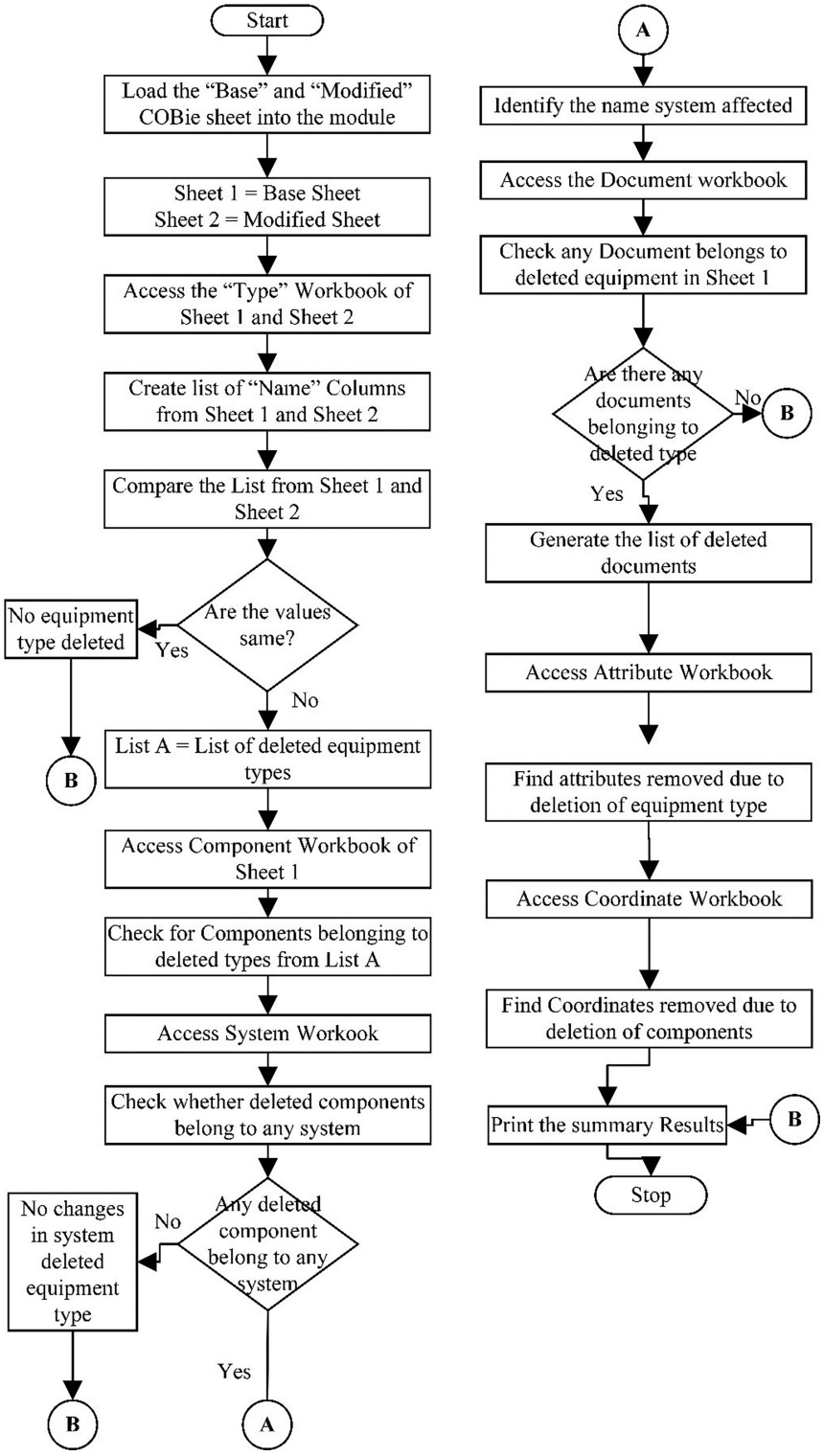

Figure 11: Sample flow chart for rule 2 (removal of equipment type).

is shown in Fig. 9. Since deduplication is one of the functions inside the verifier module, a flowchart is developed similar to Fig. 9. The process flow consists of checking workbooks against the uniqueness of the cell values for different workbooks. As discussed earlier, the uniqueness can be at the primary key level or the compound key level. So, for some workbooks such as space, type, or component, the uniqueness is at the primary key level, whereas for attributes, or coordinate workbooks, the uniqueness is at the compound key level. For verifying these workbooks, a concatenation of few column values is required before they are

being checked for uniqueness. Similar flowcharts are developed for other verifier functions such as checking for missing data, incomplete data, or incorrect format.

In the second step, the actual coding (scripting) has been done for all different algorithms based on these flowcharts. The codes inside python were written in small component scripts, and then these components are called inside the main script. This helped in the management of scripts and validation easier. The development of scripts has been done around a few useful concepts inside python programming. These are classes, loop, lists, and dictionaries. Pseudo-Code 1 shows the excerpt of the code developed for the deduplication process.

Psuedo-Code 1:
 # Verify the duplicate data (Type workbook)
load *sheet 1 = New COBie sheet*
access *the type workbook of sheet1*
create *a list A of cell values from column "Name" of the workbook*
create *a list of unique values from list A into list B*
if *the number of items in list A = = list B:*
    print ("No duplicate values in Type workbook")
else:
    dictionary A = counter (list A)
    access the items and make a list of keys having number more than 1.
    print ("key" value is having duplicate entry at n locations at following rows")
    print (cell.value (key)).
 # Verify the duplicate data (Attribute workbook)
load *sheet 1 = New COBie sheet*
access *the Attribute workbook of sheet1*
concatenate *values from the column "name" and column "RowName."*
create *a list C of cell values from the concatenated values*
create *a list of unique values from list C into list D*
if *the number of items in list A = = list B:*
    print ("No duplicate values in Attribute workbook")
else:
    dictionary C = counter (list C)
    access the items and make a list of keys having number more than 1.
    print ("key" value is having duplicate entry at n locations at following rows")
    print (cell.value (key)).

Compare module: The implementation architecture of the compare module has been shown in Fig. 10. The GUI interact with the user. The user needs to upload two COBie files to compare. Behind the GUI is an inaccessible layer that runs the compare algorithm. The compare algorithm analyzes and compares the two COBie datasheets. It runs the rule sets defined in the algorithm and finds the data that has changed. The result from the compare algorithm is stored in a database. The database stores the changed data, and this data is used further in the visualizer module to see it in the property graph model.

Similar to the verifier module, the algorithm is developed in the two-step process for the compare module. In the first step, the process flow chart has been developed. Figure 11 shows an example of the flow chart developed for "Rule 2: removal of equipment type" from Table 1. The algorithm makes a "list" of equipment from both the COBie sheets and compares it for differences. Then, the script makes a "list" of the removed equipment. The script then looks for any components removed from the type workbook but still present in the component workbook. Similarly, it checks for consistency in system, document, and coordinate workbooks.

In the second step, the actual coding has been done based on these flowcharts. Pseudo-Code 2 shows the excerpt of the script developed for "rule 2: removal of equipment type."

Psuedo-Code 2:
 # Compare the "Type" sheet
load *sheet 1 = Old COBie sheet*
load *sheet 2 = New COBie sheet*
access *the type workbook of sheet1 and sheet2*
create *a dictionary of cell values from Column "Name" and column "extIdentifier" for both the sheets*
compare *the "value" of both the dictionaries and for each different "value" in the dictionary, extract the value of "Key"*
    print (list A of "keys" as "type" removed from the Type worksheet)
 # Check whether components belonging to removed equipment type is removed from the Component workbook
access *the "component" worksheet of sheet1 and sheet2*
generate *a "list B" of component names from sheet 1 belonging to equipment type from List A*
generate *list C of component names from sheet 2*
find *the items from list B in list C*
if:
    items from list B is found in list C
    print ("Component name" belonging to deleted equipment Type (from list A) need to be removed from sheet 2, as the equipment type is removed)
else:
    print (The data is consistent with the equipment type)
#Check whether components belonging to the removed equipment type is removed from the system workbook
Continued……

3.5 Test: validating the prototype

For the validation of the verifier and compare module, the test case analysis method was used. Test cases are step-by-step instructions to verify the functionalities of a system. For this validation, the Clinic BIM model (Test cases 1 to 5) and Apartment BIM model (Test case 6) with their supporting documents were used as a base (East, 2011). The test case process flow used in this research is shown in Fig. 12. The aim was to ascertain whether the modules can identify the missing data and find the various design changes purposely done inside the BIM model to test the modules. All the reference components, types, names, or numbers are directly referred to as the COBie datasheet

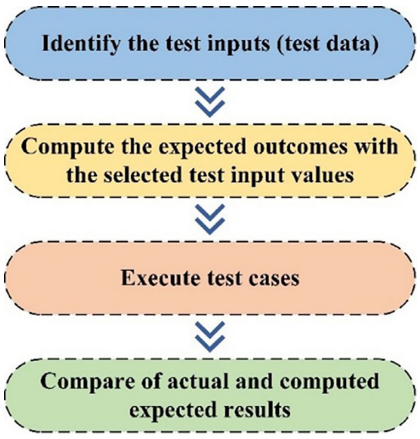

Figure 12: Test case process flow for this study.

Table 2: Proposed examples of test cases.

Example of test cases	Functionality	Test data	Expected outcome	Module outcome	Result
Test case 1	Merge two rooms and combine the equipment	Merged interaction station room number 1D09, 1D10 and 1D11	Affected workbooks: i. Space: Compare and identify 1D11 and 1D10 space have been removed, dimensions of 1D09 have changed. ii. Zone: Verify that 1D09,1D10 have been removed from the zone workbook. iii. Component: Verify that components in space 1D10, 1D11 have been moved to 1D09. iv. Space: Check whether any document was there for previous spaces that are deleted. v. Coordinate: The coordinates of spaces 1D10 and 1D11 have been deleted, and the coordinate of space 1D09 has changed.	All changes have been identified correctly by compare and verifier module.	Pass
Test case 2	Equipment added to the BIM model (New equipment Type)	The following equipment type is added: a) "Sink P-7000" b) "Cabinet Type D"	Affected workbooks: i. Type: Identify the newly added equipment. ii. Component: Find the newly added components belonging to these equipment type. iii. System: Identify new components belonging to the new type added to the existing or new system. iv. Spare: Identify any spare for the new component. v. Job: Verify new jobs for the added equipment type. vi. Document: verify whether new documents for the added types are added or not. vii. Attributes: Identify the newly added COBie attributes for the equipment type. viii. Coordinate: Identify component. Coordinate: Identify that all the components coordinated have been captured.	All data has been identified by the verifier and compare module effectively.	Pass (Fig. 13)
Test case 3	Deleting equipment type from the BIM model	The following equipment type are removed from the BIM Model and COBie datasheet:	Affected workbooks:	All workbook changes have been identified correctly.	Pass

Table 2: Continued

Example of test cases	Functionality	Test data	Expected outcome	Module outcome	Result
		a) Air Cooled Chiller b) Boiler c) Fan-Sidewall Type 1	i. Type: Identify that equipment Air Cooled Chiller, Boiler, Fan-Sidewall Type 1 have been deleted from the workbook. ii. Component: Verify that all components belong to this equipment should be absent. iii. System: Verify that all components belonging to the deleted equipment type should be removed. iv. Document: Verify that all documents related to three equipment types should be removed. v. Attribute: Verify that all the attributes belonging to these deleted equipment types are removed.		
Test case 4	Verification of Missing data	Added new sink type "P-7000" but no component and documents added	Affected workbook: i. Type: Identify new type added. ii. Component: Identify no new component added for the new sink type "P-7000." iii. Document workbook: Identify no new document added for sink type "P-7000."	Identify all expected items.	Pass (Fig. 13)
Test case 5	Verification of required data	Find fire sprinkler in type and component but no jobs mentioned	The system to identify based on a database that fire sprinkler type has "Semi-Annual Maintenance" Job listing is missing from COBie file.	The System prompts user that the "Semi-Annual Maintenance" Job listing is missing, and the user should consider adding it.	Pass (Fig. 14)
Test case 6	Shows changes in COBie data	Deleted two components: Sink Type C-1 and Shower Stall-1 from System Apartment A Plumbing	It should show in the graph that the two components are removed accordingly.	Showed that the two components were removed from the system using edge relationship arrow	Pass (Fig. 15)

available with the clinic model (East, 2011). The functionality test method was used for the testing method, whereby a system is tested against the functional requirements and specifications. The purpose of the functional testing is to ensure that the requirements are adequately satisfied by the application. In this functional testing, white box testing is used. White box testing is a method where the internal structure/design/implementation of the item is known to the tester (Nidhra & Dondeti, 2012).

4. Discussion

COBie datasheet is a rich source of centralized information. Currently, the COBie datasheet is envisaged and developed as a datasheet to be used as a handover document to reduce the need for various hardcopy manuals, handed over during the handover stage. Various authors have advocated the need for capturing critical information throughout the design and construction stages, which can be relevant during O&M stages (Lavy & Jawadekar, 2014). COBie data drops are in line with the same concept, whereby a COBie datasheet is compiled throughout the design and construction stages and not only before the handover stage. However, through literature reviews, a gap in the outline of the COBie data drop process was identified. Therefore, in this paper, a new standardized process flow aligned with AIA and RIBA PoWs is proposed. The purpose of this alignment is to have a better picture of the data drop process in alignment with two established and well-developed PoWs used in the construction industry.

```
ython Files\Case 11_Addition of New Equipment Type.py
The following Equipment Type has been added into the project:
Cabinet Type D
Sink P-7000
--------------------------------------------------------------------------------
--------------------------------
Following Equipment type has no component referenced in the project:
--------------------------------------------------------------------------------
--------------------------------
Following Component has been added in the project with newly added equipment
Types:
Row No: 245: Equipment Type Cabinet Type D with Component Name Upper Cabinet-
triple Door-Wall:.7112mm:.7112mm:340892 has been added to space 2C12
Row No: 246: Equipment Type Cabinet Type D with Component Name Upper Cabinet-
triple Door-Wall:.7112mm:.7112mm:341187 has been added to space 2C12
Row No: 247: Equipment Type Cabinet Type D with Component Name Upper Cabinet-
triple Door-Wall:.7112mm:.7112mm:341273 has been added to space 2C12
Row No: 248: Equipment Type Cabinet Type D with Component Name Upper Cabinet-
triple Door-Wall:1170mm:1170mm:338233 has been added to space 2C17
Row No: 249: Equipment Type Cabinet Type D with Component Name Upper Cabinet-
triple Door-Wall:1170mm:1170mm:339446 has been added to space 2C17
Row No: 250: Equipment Type Cabinet Type D with Component Name Upper Cabinet-
triple Door-Wall:600mm:915mm:336423 has been added to space 2C17
Row No: 251: Equipment Type Cabinet Type D with Component Name Upper Cabinet-
triple Door-Wall:600mm:915mm:336513 has been added to space 2C17
Row No: 252: Equipment Type Cabinet Type D with Component Name Upper Cabinet-
triple Door-Wall:600mm:915mm:336684 has been added to space 2C17
Row No: 253: Equipment Type Cabinet Type D with Component Name Upper Cabinet-
triple Door-Wall:762mm:762mm:343193 has been added to space 2C22
Row No: 254: Equipment Type Cabinet Type D with Component Name Upper Cabinet-
triple Door-Wall:915mm:915mm:324018 has been added to space 1D06
```

```
Component Upper Cabinet-triple Door-Wall:915mm:915mm:328573 doesnt belong to
any system
Component Upper Cabinet-triple Door-Wall:915mm:915mm:328637 doesnt belong to
any system
Component Upper Cabinet-triple Door-Wall:915mm:915mm:331680 doesnt belong to
any system
--------------------------------------------------------------------------------
--------------------------------
--------------------------------------------------------------------------------
--------------------------------
Document Missing for the Newly added Equipment Types:
Document for Sink P-7000 is missing and has not been added into the Document
Workbook
Equipment Type Cabinet Type D is added to the document workbook with followin
g details:
 Document Name: Cabinets Type D Product Data
 Document Cateogory: Product Data
 Document file name: Cabinets Type A,C,D_Product Data.pdf
--------------------------------------------------------------------------------
--------------------------------
```

Figure 13: Sample excerpt from the validation results (not all-inclusive).

Nevertheless, building a COBie datasheet using COBie data drops is not just about adding the extra set of information over to the previous COBie datasheet. Design changes are frequent in construction projects. Such design changes mean that data inside COBie datasheet at each COBie data drop will vary a lot. Changes in data mean a lot in terms of the COBie datasheet. Data inside the COBie datasheet is coming from various sources and goes beyond the BIM software export. A substantial amount of data inside the COBie datasheet is entered manually. Thus, a COBie datasheet needs to undergo rigorous verification for data consistency. This verification should not be done in isolation over the current COBie datasheet but should be compared with the previous sheets. Such deep-level verification will ensure that manual entries have been consistent, covered all the dependencies, and any data deleted have been deleted entirely with all dependencies.

In the paper, three modules have been described to enhance the usability of COBie and data drops. These are COBie verifier, COBie compare, and COBie visualizer. For the issue of data consistency, in this research, a more semantic-level verifier module

```
\Python\COBie Verifier- Standard Procedure.py
Fire Sprinkler has been found in the type Sheet
The Sprinkler Head Semi-Annual Maintenance is not added in the job Sh
eet
------------------------------------------------
 Suggested Job description for sprinkler:
1.Inspect and identify damaged, bent, corroded, painted, whitewashed,
or weeping sprinkler heads, all of which will need replacement.
2. Remove any insect nests attached to sprinkler heads.
3. Place light coat of oil on sprinkler heads located in areas conduc
ive to insect nests.
4. For sprinkler heads which are subject to above normal temperature
(adjacent to steam pipes, furnaces, ovens, hot gas, vents, exhaust, e
tc.), check color coding of sprinkler heads for the proper temperatur
e range.  Identify the sprinkler heads with improper temperature rang
es, all of which will need replacement.
5. Check that minimum distance of 18 to 36 inches is maintained betwe
en sprinkler heads and racks, palletized stock, or other stacked mate
rials.
6. Remove any ladders, stock or material which is being supported by
the sprinkler piping.
7. Replace broken pipe hangers and refasten any that have come loose.
8.  Check that all water is drained from low points on dry piping sys
tems.
```

Figure 14: Sample screenshot showing results from python code for test case 5.

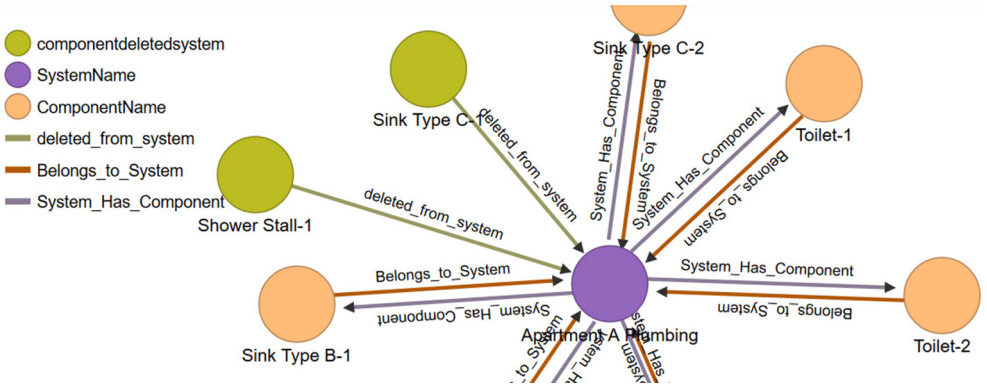

Figure 15: Example of a SQL query to find all deleted components from the system "Apartment A plumbing" (Test case 6).

is developed. This module aims at checking the missing data, incomplete data, incorrect format data, data format, and missing data links. Nevertheless, a deep-level verification is incomplete without a supporting database. This database should include information about the attribute requirements, default requirement of jobs, documents for various types of equipment, and the default list of data formats for appropriate verification. A similar database is proposed in this research to support the verification process.

In addition to this, COBie data drops should not only be seen as an intermediate process of creating a final handover FM document. By effectively mining COBie datasheet at different data drop stages, useful information about the project progress and subsequent design changes can be gathered. In this study, a compare module is also developed to compare the COBie datasheet to identify changes. The compare module is conceptualized to harness the rich amount of data stored inside the COBie datasheet. This module highlights the importance of not looking at a COBie datasheet in-silo but harnesses its potential by comparing it with previous data drops. Some examples of such changes are equipment being deleted or added, specification changes, equipment moved from one room/space to another, deletion, or merger of rooms.

Besides the two modules, an introduction to the visualizer module is presented in this paper. The visualizer module intent is to visualize the COBie datasheet using property graph model (a type of node-link diagram) and store historical data in a structured database. The necessary data such as equipment deleted/added, or specification changes can be visualized using a visualizer module in a property graph model. This database can also be queried using SQL to find the historical data changes about any particular equipment. A more detailed discussion about the functioning and development of the visualizer module will be presented in an upcoming article.

These modules were validated using test case designing to foresee whether the modules can identify the anomalies. A total of 19 test cases were designed to test the compare, verifier, and visualizer modules. A sample of the test case format is provided in Table 2. Various test scenarios were created inside the BIM model, and respective COBie datasheets are extracted. These COBie outputs were compared and verified using the developed modules to ascertain whether they can identify the changed data. All expect two of the test cases passed the trials. The remaining two test cases failed partially, which were rectified at the coding level.

5. Conclusion, Limitations, and Future Works

This research is a part of a broader study aimed at identifying the critical issues associated with a COBie datasheet while it passes through various data drop stages. The entire research used a combined methodology of design thinking and waterfall

model from the software development process. Through an extensive literature study supplemented by an exploratory study using a BIM model, a range of issues associated with the COBie data drops were identified. In this paper, the issues associated with the COBie datasheet's verification are highlighted, as the COBie datasheet passes through various data drop stages. Likewise, in this paper, new proposed stages for COBie data drops have been presented, aligning with the RIBA plan of work 2013 and AIA plan of work 2012.

In this paper, three modules, i.e. COBie verifier, COBie compare, and COBie visualizer module, are introduced, which are a part of overall CDMS development. The verifier module caters to ascertain the data consistency, whereas the compare module highlights the importance of data mining COBie datasheet to track project design changes. The visualizer module is used to visualize the data in a node-link format.

Nonetheless, the modules have some limitations that need to be considered while reviewing the findings. The system is tested on only a few BIM models that are available through the NIBS website. It needs more rigorous testing to fine-tune the performance of the system to achieve an industrial scale. Currently, it is in a proof-of-concept stage. Additionally, the modules have been developed using an open-source language. Though COBie is a vendor-neutral datasheet, it does contain information coming from BIM authoring software. These modules are tested assuming a single BIM authoring software will be used during the entire COBie data drop process and does not account for BIM authoring software changes. This paper's presented modules are part of more significant development, centric to address the various issues associated with the COBie datasheet. Though, in this paper, only issues important to this study have been presented.

This study contributes to both the body of knowledge and industrial application. While on the body of knowledge front, it highlights the importance of looking at a COBie datasheet compared with previous data drops. It highlights that data mining the connection between various data drops, useful insight about a project can be harnessed. Additionally, a comprehensive verification framework has been proposed in this study. This framework proposes a wider verification umbrella to be used over a COBie datasheet to check its consistency. On the industrial application front, the development manifests into a prototype software application. Such an application can be used by the AEC industry to harness the potential and reduce the cognitive load on professionals in the verification of COBie datasheets. By using the compare module, even a non-BIM user in a project can identify the project changes by using a COBie datasheet.

Conflict of Interest Statement

None declared.

References

Alnaggar, A., & Pitt, M. (2019). Towards a conceptual framework to manage BIM/COBie asset data using a standard project management methodology. *Journal of Facilities Management*, *17*(2), 175–187.

Anand, P., Sekhar, C., Cheong, D., Santamouris, M., & Kondepudi, S. (2019). Occupancy-based zone-level VAV system control implications on thermal comfort, ventilation, indoor air quality and building energy efficiency. *Energy and Buildings*, *204*, 109473.

Balaji, S., & Murugaiyan, M. S. (2012). Waterfall vs. V-model vs. Agile: A comparative study on SDLC. *International Journal of Information Technology and Business Management*, *2*(1), 26–30.

Becerik-Gerber, B., Jazizadeh, F., Li, N., & Calis, G. (2011). Application areas and data requirements for BIM-enabled facilities management. *Journal of Construction Engineering and Management*, *138*(3), 431–442.

BIM task group. (2012). COBie data drops: Structure, uses, and examples. *COBie – Data Drops for Standard MoJ Cell*. Retrieved from http://www.bimtaskgroup.org/cobie-data-drops/.

Biswas, T., & Krishnamurti, R. (2012). Data sharing for sustainable building assessment. *International Journal of Architectural Computing*, *10*(4), 555–574. doi:10.1260/1478-0771.10.4.555.

Caetano, I., & Leitão, A. (2019). Integration of an algorithmic BIM approach in a traditional architecture studio. *Journal of Computational Design and Engineering*, *6*(3), 327–336.

Chipman, T., Liebich, T., & Weise, M. (2012). *mvdXML: Specification of a standardized format to define and exchange Model View Definitions with Exchange Requirements and Validation Rules*. *V*, *1*, 34.

Choi, J., Kim, H., & Kim, I. (2015). Open BIM-based quantity takeoff system for schematic estimation of building frame in early design stage. *Journal of Computational Design and Engineering*, *2*(1), 16–25.

Dorst, K., & Cross, N. (2001). Creativity in the design process: Co-evolution of problem–solution. *Design Studies*, *22*(5), 425–437.

East, B. (2011). *Common Building Information Model Files and Tools*. Retrieved from https://www.nibs.org/page/bsa_commonbimfiles.

East, B. (2013a). *COBie Responsibility Matrix*. National Institute of Building Sciences (NIBS).

East, B. (2013b). Using COBie. In P. Teichloz (Ed.), *BIM for facilities managers*. New Jersey: John Willey & Sons.

East, B. (2016). *COBie QC Reporter Command Line Tool*. Retrieved from https://github.com/OhmSweetOhm/CobieQcReporter/releases/tag/1.0.

East, B., & Carrasquillo-Mangual, M. (2013). *The COBie guide: A commentary to the NBIMS-US COBie standard*. Engineer Research and Development Center, Champaign, IL.

East, E. W., Nisbet, N., & Liebich, T. (2012). Facility management handover model view. *Journal of Computing in Civil Engineering*, *27*(1), 61–67.

Gazoni, E., & Clark, C. (2018). openpyxl - A Python library to read/write Excel 2010 xlsx/xlsm files. Retrieved from https://openpyxl.readthedocs.io/en/stable/.

Henderson, D. A., Jr, & Card, S. (1986). Rooms: The use of multiple virtual workspaces to reduce space contention in a window-based graphical user interface. *ACM Transactions on Graphics (TOG)*, *5*(3), 211–243.

Herr, C. M., & Fischer, T. (2019). BIM adoption across the Chinese AEC industries: An extended BIM adoption model. *Journal of Computational Design and Engineering*, *6*(2), 173–178.

Jung, N., Häkkinen, T., & Rekola, M. (2018). Extending capabilities of BIM to support performance based design. *ITcon*, *23*, 16–52.

Kim, H., Anderson, K., Lee, S., & Hildreth, J. (2013). Generating construction schedules through automatic data extraction using open BIM (building information modeling) technology. *Automation in Construction*, *35*, 285–295.

Kim, H., Lee, J.-K., Shin, J., & Choi, J. (2019). Visual language approach to representing KBimCode-based Korea building code sentences for automated rule checking. *Journal of Computational Design and Engineering*, *6*(2), 143–148.

Koo, B., & Shin, B. (2018). Applying novelty detection to identify model element to IFC class misclassifications on architectural and infrastructure building information models. *Journal of Computational Design and Engineering*, *5*(4), 391–400.

Kumar, V., & Teo, E. A. L. (2019a). *Conceptualizing "COBieEvaluator": An Application for Data Mining COBie Datasets to Track*

Asset Changes Throughout Project Lifecycle. Paper presented at the International Congress and Conferences on Computational Design and Engineering 2019 (I3CDE 2019), Penang.

Kumar, V., & Teo, E. A. L. (2019b). *Towards a More Circular Construction Model: Conceptualizing an open-BIM based Estimation Framework for Urban Mining*. Paper presented at the CIB World Building Congress 2019, Hong Kong.

Kumar, V., & Teo, E. A. L. (2020a). Conceptualizing "COBieE-valuator": A rule based system for tracking asset changes using COBie datasheets. *Engineering, Construction and Architectural Management*, *27*(5), 1093–1118. doi:10.1108/ECAM-04-2019-0216.

Kumar, V., & Teo, E. A. L. (2020b). Perceived benefits and issues associated with COBie datasheet handling in construction industry. *Facilities*, (ahead-of-print).

Lavy, S., & Jawadekar, S. (2014). A case study of using BIM and COBie for facility management. *International Journal of Facility Management, 5*(2), 64.

Lindberg, T., Meinel, C., & Wagner, R. (2011). Design thinking: A fruitful concept for it development?In *Design thinking*(pp. 3–18). Berlin, Heidelberg: Springer.

Liu, R., & Issa, R. (2014). Design for maintenance accessibility using BIM tools. *Facilities, 32*(3/4), 153–159.

McArthur, J. (2015). A building information management (BIM) framework and supporting case study for existing building operations, maintenance and sustainability. *Procedia Engineering*, *118*, 1104–1111.

Mohandes, S., Abdul Hamid, A., & Sadeghi, H. (2014). Exploiting building information modeling throughout the whole lifecycle of construction projects. *Journal of Basic and Applied Scientific Research, 4*(9), 16–27.

NIBS. (2017). National BIM Standard – United States, V3.

Nidhra, S., & Dondeti, J. (2012). Black box and white box testing techniques: A literature review. *International Journal of Embedded Systems and Applications (IJESA), 2*(2), 29–50.

Onuma. (2013). *Onuma and COBie-COBie Validator*. Retrieved from https://onuma.com/products/OpsAndCobieValidate.php.

Pärn, E., Edwards, D., & Sing, M. (2017). The building information modelling trajectory in facilities management: A review. *Automation in Construction*, *75*, 45–55.

Patacas, J., Dawood, N., Vukovic, V., & Kassem, M. (2015). BIM for facilities management: Evaluating BIM standards in asset register creation and service life planning. *Journal of Information Technology in Construction, 20*(10), 313–318.

Petersen, K., Wohlin, C., & Baca, D. (2009). *The Waterfall Model in Large-Scale Development*. Paper presented at the International Conference on Product-Focused Software Process Improvement.

RIBA. (2013). RIBA plan of works. Retrieved from https://www.architecture.com/knowledge-and-resources/resources-landing-page/riba-plan-of-work.

Sinclair, D. (2019). *Guide to Using the RIBA Plan of Work 2013*: Routledge.

Stewart, G. H. (2014). COBie: The practicalities of application. Retrieved from http://iug.buildingsmart.org/resources/itm-and-iug-meetings-2014-stockholm/conference/building/cobie-the-practicalities-of-application/view.

Teicholz, P. (2013). *BIM for facility managers*. John Wiley & Sons.

Vanlande, R., Nicolle, C., & Cruz, C. (2008). IFC and building lifecycle management. *Automation in Construction, 18*(1), 70–78.

Volk, R., Stengel, J., & Schultmann, F. (2014). Building Information Modeling (BIM) for existing buildings—Literature review and future needs. *Automation in Construction*, *38*, 109–127.

Woods, D. D., & Watts, J. C. (1997). How not to have to navigate through too many displays. In *Handbook of Human–Computer Interaction (Second Edition)*(pp. 617–650). Elsevier.

Yalcinkaya, M. (2017). Understanding the technical and cognitive challenges, and closing the gaps in architectural, engineering, construction-facility management standards.

Yalcinkaya, M., & Singh, V. (2019a). Exploring the use of Gestalt's principles in improving the visualization, user experience and comprehension of COBie data extension. *Engineering, Construction and Architectural Management*, *26*(6), 1024–1046.

Yalcinkaya, M., & Singh, V. (2019b). VisualCOBie for facilities management: A BIM integrated, visual search and information management platform for COBie extension. *Facilities, 37*(7/8), 502–524.

Yalcinkaya, M., Singh, V., Nenonen, S., & Junnonen, J.-M. (2016). *Evaluating the Usability Aspects of Construction Operation Building Information Exchange (COBie) Standard*. Paper presented at the CIB World Building Congress, At Tampere, Finland. https://www.researchgate.net/publication/303811016_Evaluating_the_Usa bility_Aspects_of_Construction_Operation_Building_Information_Exchange _COBie_Standard.

Yana, A. G. A., Rusdhi, H., & Wibowo, M. A. (2015). Analysis of factors affecting design changes in construction project with Partial Least Square (PLS). *Procedia Engineering*, *125*, 40–45.