IASMiRT

Full Paper Transactions, SMiRT 28
Toronto, Canada, August 10-15, 2025
Division III

SMiRT 28
TORONTO, CANADA

FINITE ELEMENT MODELS TO BUILDING INFORMATION MODELS: INTEROPERABILITY SOFTWARE AND DIGITAL TOOLS FOR REAL-TIME CHANGE DETECTION

Will Ashe[1], Saran Srikanth Bodda[2], Harleen Kaur Sandhu[3], Abhinav Gupta[4], and Kevin Han[5]

[1]Graduate Student, Center for Nuclear Energy Facilities and Structures (CNEFS), North Carolina State University (NCSU), USA (wlashe@ncsu.edu)
[2]Research Faculty, CNEFS, NCSU, USA (ssbodda@ncsu.edu)
[3]Former Postdoctoral Research Scholar, CNEFS, NCSU, USA (harleengoku@gmail.com)
[4]Director, CNEFS, NCSU, USA (agupta1@ncsu.edu)
[5]Associate Professor, CNEFS, NCSU, USA (kevin_han@ncsu.edu)

ABSTRACT

Nuclear power plant construction frequently encounters cost overruns and schedule delays. The structural design process for nuclear facilities is complex and demanding due to strict code tolerances and the potential for conflicts between numerous structural and mechanical systems. Engineers are constantly revising designs to ensure safety, functionality, and regulatory compliance. Structural designs are typically completed in Finite Element (FE) software, with ANSYS APDL being a widely used tool for analyzing nuclear structures. Nuclear piping models pose a unique challenge in structural design due to their extensive length, complex routing, and numerous connection points. Even small changes can have a ripple effect throughout the entire system, making it challenging to track and manage design updates accurately. With design changes being a major contributor to construction delays in nuclear projects, it is crucial to develop efficient methods for detecting and managing design modifications in FE platforms and translate them into building information models (BIM) used in construction for quick approval. To address these challenges, this paper will introduce a novel algorithm to accurately identify and visualize engineering design modifications in ANSYS APDL piping models. By automating the design change detection process, this approach enhances design verification, improves engineering workflows, and minimizes the risk of errors. This paper also provides an investigation into APDL-to-BIM conversion to improve the design-construction coordination of nuclear projects. Combining these approaches ultimately contributes to a more efficient and reliable nuclear power plant design process.

INTRODUCTION

Construction of nuclear power plants is a time-consuming process, with many teams working on various aspects of the project. Nuclear power plant construction is notorious for suffering from project budget overruns and construction timeline delays (Eash-Gates et al., 2020). Prime examples of these issues are highlighted in the construction of the VC Summer Nuclear Station in Jenkinsville, South Carolina as well as the Olkiluoto Nuclear Power Plant project in Finland. In both cases, actual costs were well above projected costs, causing the projects to be delayed and temporarily abandoned (Crees, 2018; Greene and Tickell, 2015). A significant factor contributing to these project failures was the necessity for on-site design modifications. A study conducted by Robb Stewart and Shirvan (2023) analyzed the factors contributing to construction timeline delays in nuclear power plants. The research identified that design change modifications were the most significant cause of nuclear project delays. These modifications often occur due to unforeseen engineering challenges and site-specific constraints, requiring substantial rework and additional regulatory approval. The study further found that human error was the second most impactful factor affecting nuclear facility construction schedules. On-site design modifications have a direct impact on the rate of human error, as they pressure engineers and construction teams to make rapid adjustments.

Nuclear facilities are densely packed with mechanical and structural systems, which frequently result in installation conflicts that necessitate redesign. Piping systems are especially prone to this because of their intricate routing in tight areas. When structural system designs are modified on-site, engineers are required to update their simulation models accordingly. Subsequently, a structural analysis must be rerun to assess the impact of these changes on the performance, stability, and safety of the system (Sandhu et al., 2024). The lack of interoperability between FE analysis and BIM platforms hinders design change approval workflows (Ren et al., 2018). The first step in integrating the updated design changes into the BIM is identifying and tracking modifications within the FE model. However, an absence of change-tracking capabilities in FE platforms significantly complicates efficient model organization and version control.

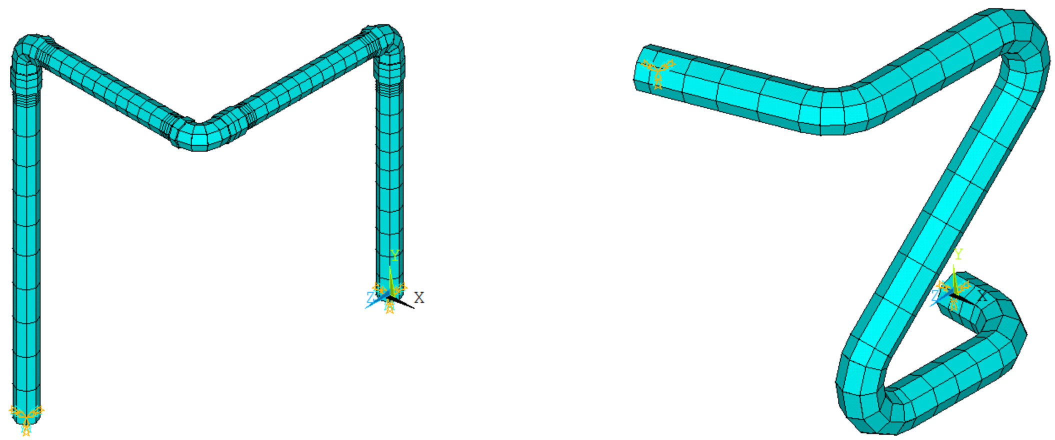
A framework for design change detection has been proposed within the BIM platform by Lin and Zhou (2020). This study tackles inefficiencies in how previously developed design change mechanisms are implemented in BIM. Traditional design change detection methodologies often mistake simple data updates for meaningful design modifications, leading to inaccurate or irrelevant results (Likhitruangsilp et al., 2018). To address these challenges, the researchers developed a new hash-code comparison approach to categorize and detect semantic design changes. In BIM, hash codes are highly effective because objects such as walls, beams, and mechanical components are represented as discrete, identifiable entities with unique Globally Unique Identifiers (GUID) (Arnold and Hempelmann, 2016). Hash-code-based comparison approaches become less valuable when analyzing APDL models because they do not account for the parametric and iterative nature of finite element analysis. This makes direct object identification and tracking more complex, as numerical approximations, boundary condition refinements, and mesh updates can introduce small variations.

To address the challenge of detecting and tracking design changes in FE models, this paper will propose a command recognition algorithm to track and visually identify engineering modifications within APDL piping models. Specifically, the algorithm leverages the ANSYS in-house graphical user interface (GUI) to visually highlight and annotate even the most subtle design modifications. The core functionality of the program involves comparing the previous and updated piping model versions, visually highlighting piping components according to the specific changes detected. Furthermore, the research investigates the conversion of FE piping models to BIM and explores the process of importing these models into BIM software. Combining FE design change detection with FE-to-BIM interoperability enhances coordination, minimizes inconsistencies, and ensures accurate engineering updates throughout the construction lifecycle.

FE DATA STRUCTURE OUTLINE AND EXAMPLE MODEL DESCRIPTIONS

This paper will propose a command-recognition algorithm to provide real-time change detection to FE pipe models created through the APDL format. APDL is a text-based scripting language where each command performs a specific action in the modeling and simulation setup. Commands are typically single words or short phrases followed by a series of arguments or conditions. The APDL format allows for the use of variables and parameters, making the text scripts flexible and reusable. The parametric capabilities of this language facilitate automation, reproducibility, and customization in simulation workflows, making APDL a valuable tool for complex engineering studies. This paper will focus exclusively on the analysis of pre-processing commands related to the construction and modeling of pipe systems.

Two piping models developed with APDL (see Figure 1) will be used to test and demonstrate the functionality of the change-detection algorithm. The first model represents a simple piping system to serve as a foundational test case. The system consists of basic piping components such as three 90-degree elbow joints, with four straight pipe segments oriented along the X, Y, and Z axes. The second model is a Z-pipe system selected from the EBR-II nuclear reactor (Sandhu et al., 2023). The model consists of elbows with various curvatures and a diagonal pipe element oriented along the Y and Z axes. The Z-pipe model presents distinct challenges due to its unconventional elbow configurations and the unique orientations of its straight pipe segments. The combination of these two models enables a comprehensive assessment of the change-detection algorithm's performance across both simple and complex configurations.



(a) Original Lab Model. (b) Original Z-Pipe Model.

Figure 1. Original FE Models for Algorithm Testing.

ALGORITHM INITIATION AND MODEL TRANSFORMATION

The FE model change-detection algorithm is a Python-based program that provides an interactive platform for an intuitive user experience. The user must initiate the program from either the command prompt or a preferred Python Integrated Development Environment (IDE). Upon initiation, the user will be prompted to drag and drop both their old and updated models into the designated boxes. The user can then click ”Run Comparison” to begin the runtime process. The user-program interaction sequence can be seen in Figure 2.

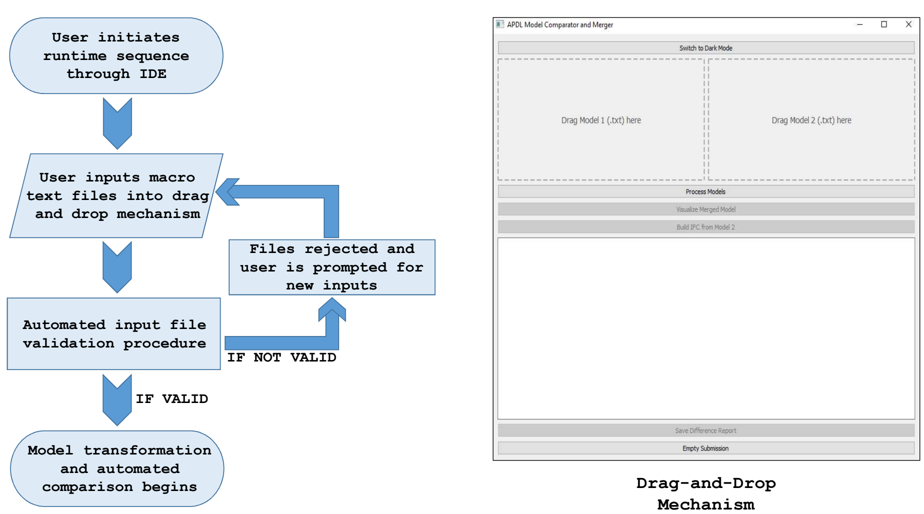
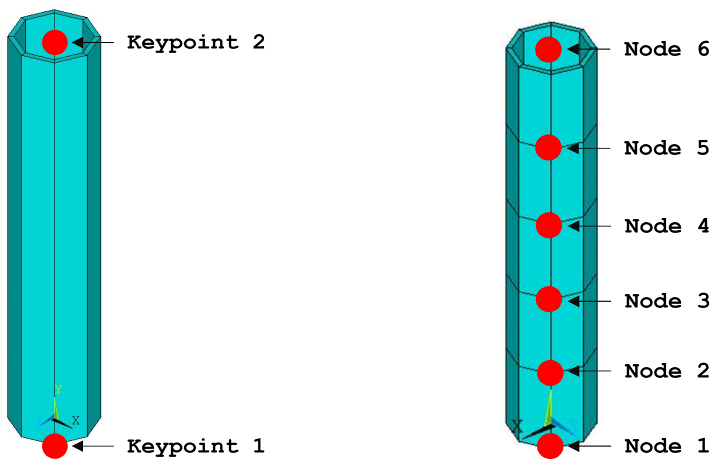


Figure 2. User-Program Initiation Sequence.

Once the code is executed by the user, the code initiates a two-part transformation process as its first step. The transformation process converts input files into a structured and readable layout that the design change detection algorithm can systematically analyze. These transformation processes will be outlined in the subsequent sections.

Keypoint Mesh to Node Transformation

The first stage of the model comparison algorithm implements a transformation process for a *keypoint mesh-to-node* conversion. In APDL, the beginning and end points of unmeshed elements are specified using keypoint coordinates, which define the geometric layout and orientation of the element. When the user sets an element mesh size for finite element analysis, the keypoints are discretized into nodes, effectively dividing the element into equally spaced smaller segments. A visualized depiction of the keypoint to node relationship can be seen in Figure 3.



(a) Unmeshed Pipe Element. (b) Meshed Pipe Element.

Figure 3. Keypoints to Nodes Transformation.

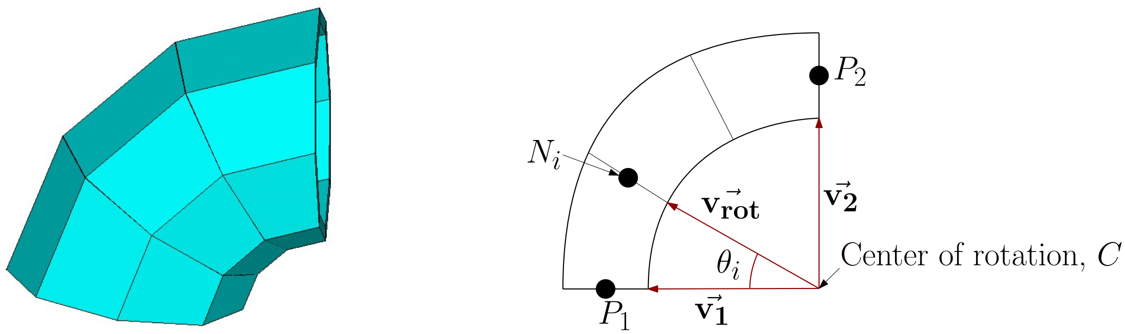
To compare the analysis models to one another, the algorithm must first extract the nodal coordinates from the user-defined keypoints. Given the keypoint coordinates of two points P_1 and P_2, the x, y, and z coordinates of a node N_i in a straight pipe section can be computed using Equation Set 1.

$$x_i = x_1 + \frac{x_1 - x_2}{m_i} \cdot i \qquad y_i = y_1 + \frac{y_1 - y_2}{m_i} \cdot i \qquad z_i = z_1 + \frac{z_1 - z_2}{m_i} \cdot i \tag{1}$$

Variable Definitions

- x_i, y_i, z_i: Coordinates of node N_i.
- m_i: Mesh size associated with node N_i.
- i: Step count or index node in the interpolation process.
- x_1, y_1, z_1: Coordinates of the starting keypoint P_1.
- x_2, y_2, z_2: Coordinates of the ending keypoint P_2.

The transformation process for converting elbow keypoints to nodes is more complex and involves additional steps due to the interpolation of coordinates over an arc. Figure 4 provides a depiction of a meshed elbow element and a section view diagram, highlighting the relevant variables that will be used and computed during the transformation process.



(a) Meshed Elbow Element.

(b) Elbow Variable Diagram.

Figure 4. Elbow Coordinate Transformation Outline.

In the original APDL text file, the user is required to define the elbow's start keypoint, end keypoint, and center of rotation. Given the coordinates of P_1, P_2, and C from the input model file, we can extract the elbow nodal coordinates by implementing the following process (Piña, 2011):

- Calculate vectors from C to points P_1 and P_2

$$\tilde{\mathbf{v}}_\mathbf{1} = P_1 - C \quad (\tilde{\mathbf{v}}_\mathbf{1} = x_1 - x_C, y_1 - y_C, z_1 - z_C) \tag{2}$$

$$\tilde{\mathbf{v}}_\mathbf{2} = P_2 - C \quad (\tilde{\mathbf{v}}_\mathbf{2} = x_2 - x_C, y_2 - y_C, z_2 - z_C) \tag{3}$$

- Calculate the bend angle θ between the vectors $\tilde{\mathbf{v}}_\mathbf{1}$ and $\tilde{\mathbf{v}}_\mathbf{2}$

$$\theta = \cos^{-1}\left(\frac{\tilde{\mathbf{v}}_\mathbf{1} \cdot \tilde{\mathbf{v}}_\mathbf{2}}{|\tilde{\mathbf{v}}_\mathbf{1}||\tilde{\mathbf{v}}_\mathbf{2}|}\right) \tag{4}$$

- Calculate the cross product $\vec{\mathbf{N}}$ of the vectors $\tilde{\mathbf{v}}_\mathbf{1}$ and $\tilde{\mathbf{v}}_\mathbf{2}$

$$\vec{\mathbf{N}} = \tilde{\mathbf{v}}_\mathbf{1} \times \tilde{\mathbf{v}}_\mathbf{2} \tag{5}$$

- Compute the step angle θ_i

$$\theta_i = \frac{i \cdot \theta}{n_e} \tag{6}$$

- Calculate the rotated vector $\vec{\mathbf{v}_{\text{rot}}}$ using Rodrigues' Rotation Formula

$$\vec{\mathbf{v}_{\text{rot}}} = \tilde{\mathbf{v}}_\mathbf{1}\cos(\theta_i) + (\vec{\mathbf{N}_{\text{unit}}} \times \tilde{\mathbf{v}}_\mathbf{1})\sin(\theta_i) + \vec{\mathbf{N}_{\text{unit}}}(\vec{\mathbf{N}_{\text{unit}}} \cdot \tilde{\mathbf{v}}_\mathbf{1})(1 - \cos(\theta_i)) \tag{7}$$

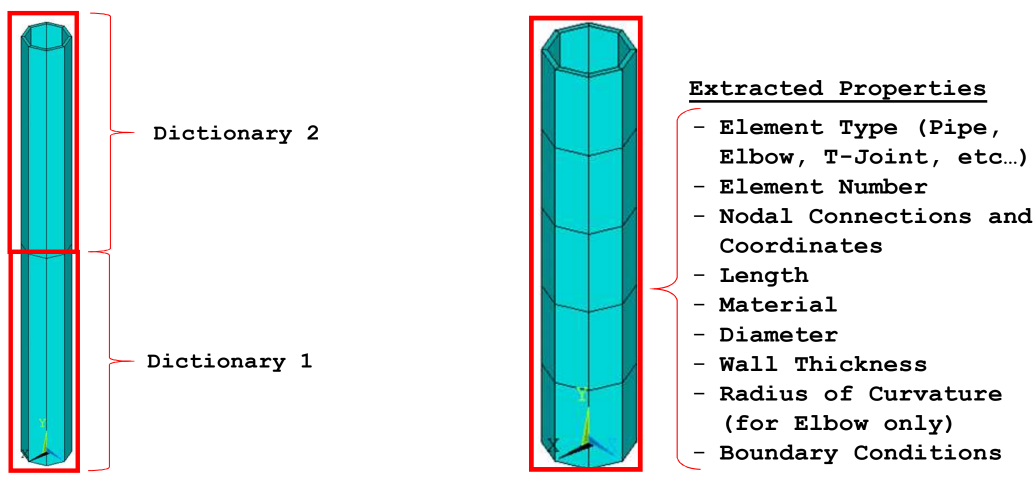
- Compute the new coordinates for node N_i (x_i, y_i, z_i)

$$N_i = C + \vec{\mathbf{v}_{\text{rot}}} \tag{8}$$

The purpose of first transforming the keypoints into nodes is to enable the algorithm to accurately connect the corresponding mesh elements at the nodal points. The nodes serve as the foundation for building the final annotated APDL model once the comparison process is complete. The nodes can also be compared individually to assess any changes in mesh element length, orientation, or connectivity. Upon transformation of keypoints to nodes, the nodal numbers and coordinates are stored in a Python list to be used later on in the comparison process. From here, the algorithm moves into part two of the transformation process: *unmeshed to meshed element mapping*. Part two of the transformation procedure will be outlined in the subsequent section.

Unmeshed to Meshed Element Mapping

When the user assigns a mesh size to unmeshed elements, ANSYS automatically assigns a unique number to each element within the mesh. These element numbers and associated properties can be retrieved through a backend coding process for tracking purposes. The function of meshed element mapping is to track and store element properties in the Python algorithm's internal database for future comparison. Specifically, the program leverages Python dictionaries to store elemental properties and track any engineering changes made over time. A single dictionary for each unmeshed element will be created for both models. Within each unmeshed element local dictionary are unique element property entries. The properties include element type, number, nodal connections, material, geometry, and boundary conditions. Figure 5 provides a list of the properties that are extracted for meshed elements.



(a) Dictionary Creation Process.

(b) Dictionary Entry Outline.

Figure 5. Meshed Element Property Extraction.

This extraction process is performed for each unmeshed element individually. For example, in a pipe system with 10 piping components, there will be 10 unique dictionaries with property entries for each component. This ensures that similar elements will still be compared if mesh sizes and element quantities change between models. It is important to note that these transformation procedures are executed twice—once for each input model. After completing the extraction of element properties, the algorithm transitions to the element comparison phase. The comparison stage serves as the core component of the comparison algorithm, where the structural, mechanical, and geometric differences between the meshed elements of the input models are systematically analyzed. By organizing the extracted data, the algorithm ensures that the comparison phase is both accurate and efficient, setting the stage for a structured approach to analyzing model changes.

COMPARISON ALGORITHM METHODOLOGY AND EXAMPLE USAGE

This section will provide an overview of the model comparison process followed by example implementations of the algorithm. The comparison algorithm systematically evaluates the input models on unmeshed element local dictionaries to identify and track structural, mechanical, and geometric variations within the piping system. Specifically, the algorithm compares the properties of elements with identical numbers from each input model.

The comparison stage works to identify and visualize any engineering changes that are made in the updated model. When the algorithm execution concludes, the user will see two outfile text files in their working directory. The first output file will be the final APDL model file that contains annotations and colored elements based on the changes that were detected. Geometry and material changes are highlighted in orange, existing element orientation or location changes are highlighted in red, added elements are highlighted in green, and deleted elements are highlighted in purple. The second output file is a text document that details any detected modifications, including a specific identification of the elements impacted by these changes. See Figure 6 for a holistic overview of the algorithm's execution.

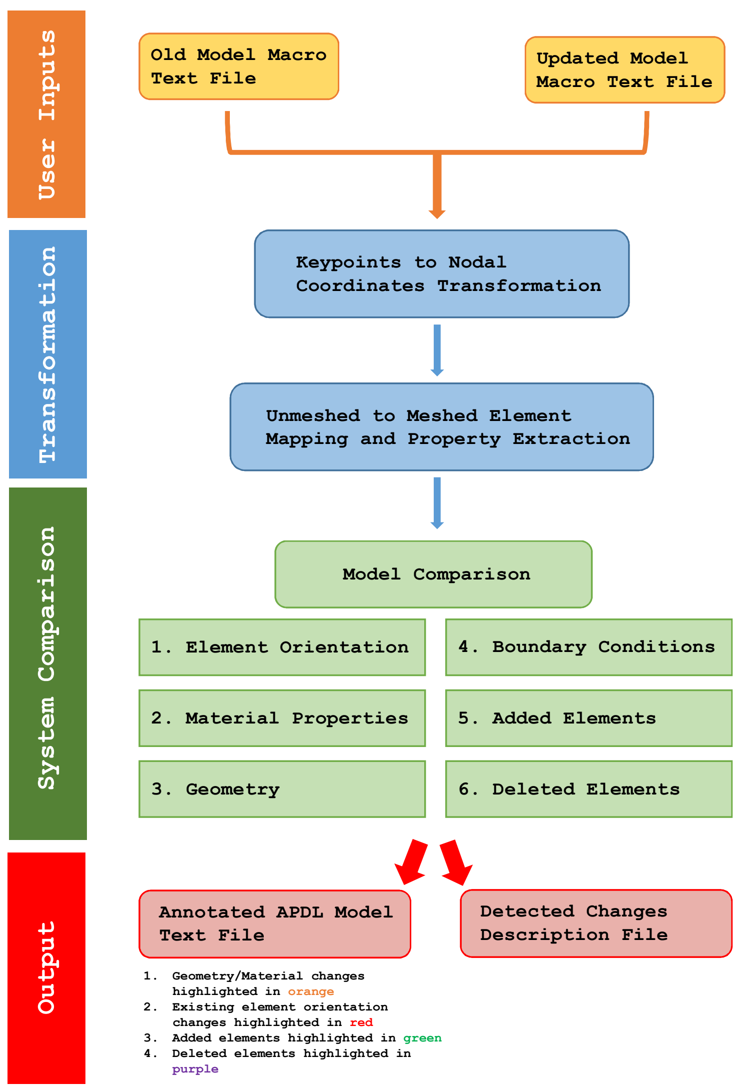


Figure 6. Model Comparison Algorithm Flowchart.

With the output annotated APDL text file, the user can copy and paste its contents directly into the APDL command prompt. When the model finishes rendering in the ANSYS GUI, the user will see highlighted elements based on the engineering changes identified by the algorithm. If elements were added or deleted on either end of the piping system, the user will see the updated location of the boundary conditions, if applicable. To better describe this, two example implementation cases have been provided in Figure 7 and Figure 8.

Example 1: Simple Piping Model

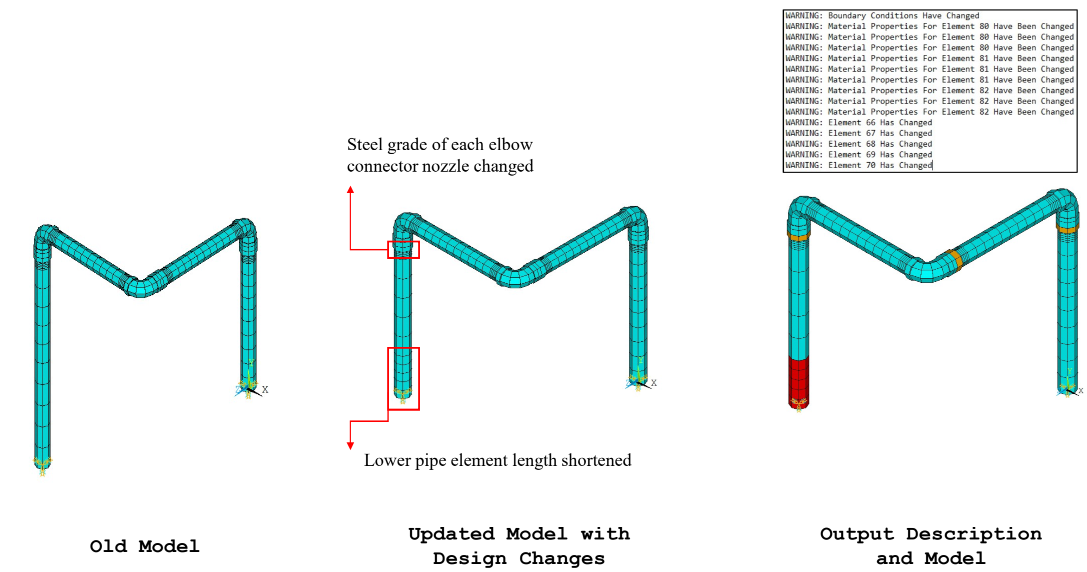


Figure 7. Simple Pipe Model Output Example.

Example 2: Z-Pipe System Model

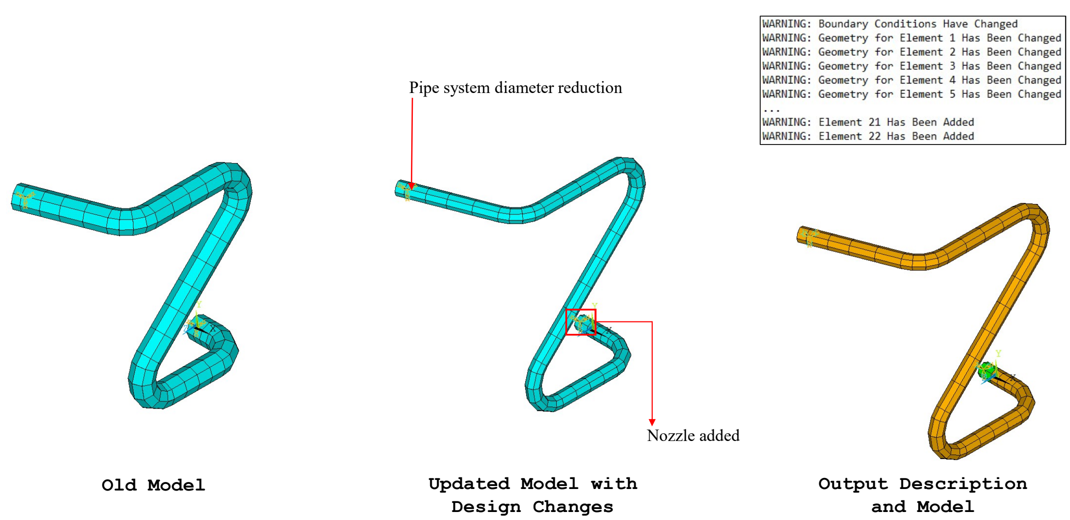


Figure 8. Z-Pipe Output Example.

INVESTIGATION INTO APDL TO IFC CONVERSION

This section will provide a brief investigation into APDL to IFC4 conversion. Many studies have been conducted on the forward conversion end of BIM-FEM, using IFC as a vehicle for data exchange (Beirnaert and Lippens, 2018; Crowder, 2023; Giangiulio et al., 2022; Ren et al., 2018). The Industry Foundation Classes (IFC) is an open, standardized data format designed for interoperability in the AEC industry. IFC4, the latest major version, introduces improvements in geometry handling, parametric modeling, and enhanced support for structural analysis data. However, limited attention has been given to the backward conversion of FEM-to-BIM, specifically in the context of achieving interoperability between APDL-based finite element models and BIM platforms. In principle, the IFC schema can be leveraged to provide bidirectional compatibility between heterogeneous data platforms.

This paper successfully demonstrates the conversion of the simple piping system from APDL to IFC, highlighting the feasibility of data interoperability between these platforms. The APDL-IFC conversion process is completely automated and implemented as an extra step on the backend of the model comparison algorithm. In IFC, APDL pipe elements are translated into *IfcPipeSegment* geometric objects. When the *IfcPipeSegment* is imported into Revit, a BIM software developed by Autodesk, the object is recognized in its context as an IFC geometric object. Figure 9 displays the conversion process of the simple piping system into IFC and Revit.

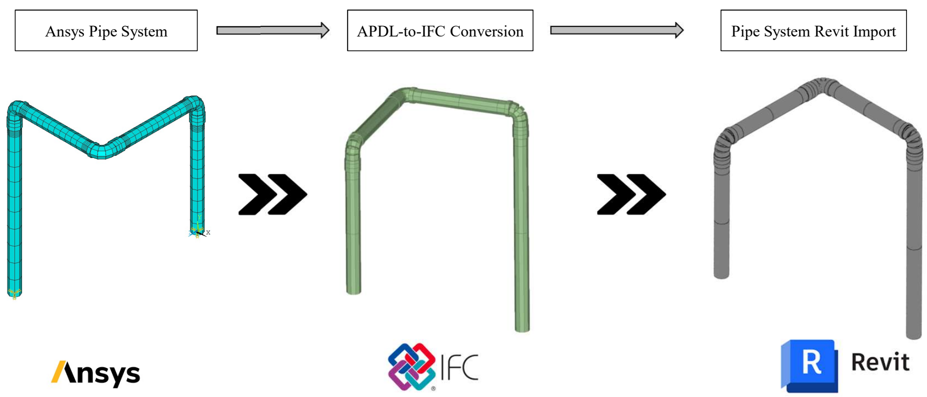


Figure 9. APDL to IFC4 to Revit Import.

CONCLUSIONS

The architecture, engineering, and construction (AEC) industry faces a growing demand for the digitalization of finite element modeling (FEM) change-tracking and data exchange processes to improve workflow efficiency. Frequent design modifications during construction necessitate the continuous refinement and validation of structural systems. However, the absence of integrated change-tracking tools in finite element software presents significant barriers, hindering the transition from design to construction. Therefore, this paper introduces an algorithm that identifies and tracks structural modifications within ANSYS Parametric Design Language (APDL) piping models. By enabling a systematic comparison process between models, the proposed program equips engineers with a robust platform to detect subtle, nuanced changes in complex piping systems. Providing a design change detection platform is particularly valuable for managing iterative design processes, ensuring model accuracy, and streamlining workflows in projects with intricate structural configurations. This ultimately decreases repetitive work for engineers by automating the detection of changes, reducing the risk of errors, and ensuring that models remain consistent and up-to-date throughout the design and construction lifecycle.

ACKNOWLEDGMENTS

This research was supported under a Department of Energy Office of Nuclear Energy University Nuclear Leadership Program Graduate Fellowship and by the CNEFS center at NC State University. Resources for the Center come from the dues paid by member organizations and from the Civil Engineering Department and College of Engineering. Any opinions, findings, conclusions, or recommendations expressed in this paper are those of the author and do not necessarily reflect the views of the DOE Office of Nuclear Energy.

REFERENCES

Arnold, G. and Hempelmann, H. (2016). "Hash-based change tracking for software make tools." US Patent 9,280,331.

Beirnaert, F. and Lippens, A. (2018). "Analysis of the interoperability from BIM to FEM."

Crees, A. (2018). "The failed V.C. Summer nuclear project: A timeline." *Choose Energy*.

Crowder, N. C. (2023). *Improving Digital Engineering Workflows by Enabling BIM Interoperability for Structural Analysis and Facilitating Accurate Modeling of Building, Equipment, and Piping Systems*. Ph.D. thesis, North Carolina State University.

Eash-Gates, P., Klemun, M. M., Kavlak, G., McNerney, J., Buongiorno, J. and Trancik, J. E. (2020). "Sources of cost overrun in nuclear power plant construction call for a new approach to engineering design." *Joule*, 4(11), pp. 2348–2373.

Giangiulio, M., Granitzer, A.-N., Tschuchnigg, F. and Hoffmann, J. (2022). "BIM-to-FEM: development of a software tool to increase the operational efficiency of dam construction projects." In *International Conference on Trends on Construction in the Post-Digital Era*, pp. 182–195, Springer.

Greene, J. and Tickell, O. (2015). "Finland cancels Olkiluoto 4 nuclear reactor." *Ecologist*.

Likhitruangsilp, V., Handayani, T. N. and Yabuki, N. (2018). "A BIM-enabled change detection system for assessing impacts of construction change orders." In *17th International Conference on Computing in Civil and Building Engineering*.

Lin, J.-R. and Zhou, Y.-C. (2020). "Semantic classification and hash code accelerated detection of design changes in BIM models." *Automation in Construction*, 115, p. 103212.

Piña, E. (2011). "Rotations with Rodrigues' vector." *European journal of physics*, 32(5), p. 1171.

Ren, R., Zhang, J. and Dib, H. N. (2018). "BIM interoperability for structure analysis." In *Construction Research Congress 2018*, pp. 470–479.

Robb Stewart, W. and Shirvan, K. (2023). "Construction schedule and cost risk for large and small light water reactors." *Nuclear Engineering and Design*, 407, p. 112305, ISSN 0029-5493, doi: https://doi.org/10.1016/j.nucengdes.2023.112305.

Sandhu, H. K., Ashe, W., Nicholas, C., Gupta, A. and Han, K. (2024). "Automation in Digital Engineering Solutions for Nuclear Design-Construction Integration using Fem-Bim Interoperability." In *International Conference on Structural Mechanics in Reactor Technology*, IASMiRT.

Sandhu, H. K., Bodda, S. S., Sauers, S. and Gupta, A. (2023). "Condition monitoring of nuclear equipment-piping systems subjected to normal operating loads using deep neural networks." *Journal of Pressure Vessel Technology*, 145(4), p. 041901.