Chair of Computing in Civil and Building Engineering
TUM School of Engineering and Design
Technical University of Munich

Modeling:
Mathematics:
Programming:
Science:

Software Lab:

Interactive IFC-to-Graph Viewer for Construction Process Logic and Robot Execution

Description

This project develops a browser UI to explore and inspect construction knowledge in the Information Backbone for Robotic Construction (IBRC) [1]. It uses the existing representations and outputs described in [1] and [2] without modifying or extending them. The focus is on making these outputs explorable and interpretable for different roles, including research, robotics engineering, and site planning. The project visualizes two key aspects:

1. ***Product–process graphs** [2], which connect building elements to task structures, including relations such as spatial decomposition, containment, adjacency, and precedence (and optionally resource links).*
2. ***Behaviour Trees** (BTs) [3], which represent the executable control structure derived from these task structures.*

In the IBRC setting, this information is stored as structured data (e.g., in a graph store, or served through an API). However, without strong visual tooling, this knowledge remains rather abstract, and difficult to inspect, debug, or communicate across roles. To address this gap, the main deliverable of this project is an interactive UI (for example, implemented with React) that provides:

- an **IFC-and-Graph view** for navigating the product–process representations, and
- a **Behaviour Tree view** for inspecting execution structure and parameters.

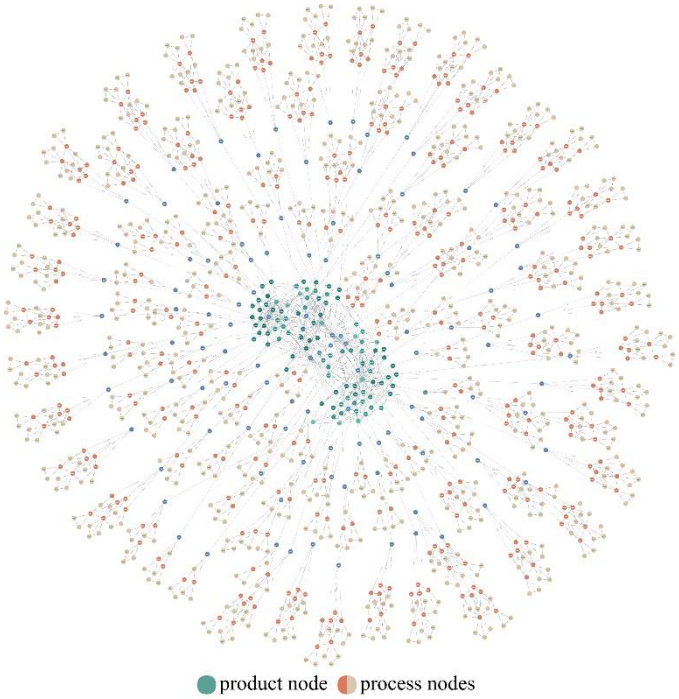

Figure 1. Product-Process graph of a simple structure (adapted from [1]).

Fig. 1 shows a product-process graph for a three-story building. For larger projects, the graph grows quickly and makes the global structure and individual task structures cumbersome to track. In addition, it is difficult to track to *which* construction elements the processes refer to. Pure IFC viewers, on the other hand, present the building geometry but do not expose the underlying construction processes and dependencies linked to each element. For this reason, the UI should combine both perspectives [4]: the IFC viewer provides geometric context, and the graph view provides construction logic. In [4], users select building elements in the interactive IFC view, and the corresponding node is automatically focused and highlighted in the graph view, creating a direct visual link between building components and their associated construction logic. An example of this is presented in Fig. 2.

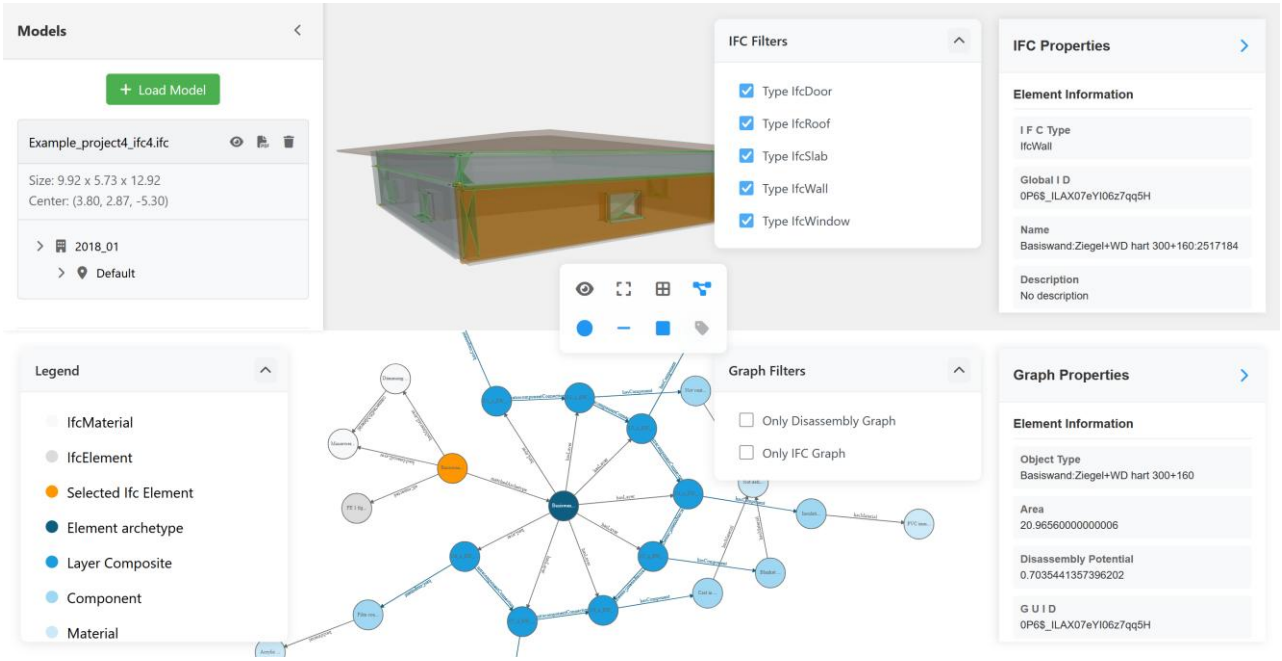

Figure 2. Exemplary IFC-and-Graph viewer [4].

Behaviour Trees follow from the product–process graph, but face a similar issue: without a dedicated viewer, they are hard to inspect and understand. In this project, the Behaviour Trees are available as XML, which quickly becomes cumbersome to read when trees grow larger. Existing tools like Groot2 [5] provide useful ideas for interaction (tree layout, search, and parameter inspection), but they run as desktop applications rather than in the browser. The project, therefore, includes a browser BT view that visualizes the XML as an interactive tree and exposes the node parameters. The BT view should at least support expanding/collapsing subtrees, searching by node name/ID, and showing each node's parameters.

Task / Work Packages

- Requirements & Use Cases: Define 2-3 concrete use cases (e.g., "inspect method assigned to a wall element", "visualize dependencies for a process step", "trace execution parameters and constraints for an action node").

- Data Interface (read-only): Implement a data interface to **consume** existing IBRC outputs:
  - Load product-process graph + metadata from **Neo4j** or from an **IBRC API**
  - Load BT definitions (XML) linked to graph entities, parse them into an internal tree model for rendering
    - Focus: stable IDs, versioning, and fast incremental loading for the UI.

- IFC + Graph Visualization Module: Build a combined viewer with:
  - IFC model view with element selection and highlighting
  - Interactive graph view (node/edge rendering, pan/zoom, collapsible subgraphs)
  - Search and filters (by element type, relation type, method, dependency)
  - Inspection panel showing properties and links (selected element, related tasks, dependencies, states)
  - Cross-linking: selection in the IFC view focuses the corresponding graph node, and selection in the graph highlights the element in 3D

- Behaviour Tree Visualization Module:
  - Tree rendering (sequence/fallback/decorator/action/condition nodes, based on the exported XML)
  - Expand/collapse, readable execution flow, parameter inspection for each node, and step highlighting
  - Cross-linking between BT nodes and graph entities (tasks / elements / optional resources), so users can jump between the BT subtree and the corresponding part of the graph

- Evaluation & Documentation: Evaluate the prototype against the initial use cases (qualitative + small performance notes). Deliver final documentation, a demo walkthrough, and a short final report.

Expected Deliverables

- Working prototype (web app) with IFC + graph visualization + behaviour tree visualization, including cross-linking between the views
- **Documented data contracts** for graph + BT (types and mapping rules), usable by IBRC services, including how stable IDs are used to connect elements, graph nodes, and BT nodes.
- Import/export support (at least one): **Neo4j**, **JSON-LD**, **RDF**, or standardized JSON
- Filters and inspection tools (e.g., “show only dependencies”, “only method/execution edges”, “show constraints”)
- Short report (approach, architecture, limitations, evaluation)
- Demo (live or recorded walkthrough)

Recommended Skills

Solid programming skills (TypeScript/Node or Python), basic API integration (working with REST endpoints, JSON), and interest in data modelling and visualization. Experience with React and web UI dev is helpful, as well as familiarity with graph data (e.g., Neo4j).

Supervisors

Brinkhoff Maikel, Chair of Computing in Civil and Building Engineering, maikel.brinkhoff@tum.de
Wrabel Tamira, Chair of Computing in Civil and Building Engineering, tamira.wrabel@tum.de

References

[1] Wrabel, T., Esser, S., & Borrmann, A. (2026). A Framework Architecture for the Information Backbone for Robotic Construction. In *43rd Proceedings of the International Symposium on Automation and Robotics in Construction (ISARC)*. Submitted manuscript, under review.

[2] Brinkhoff, M., Esser, S., & Borrmann, A. (2026). An Integrated Product–Process Representation for Deriving Robotic Task Structures using Graph Transformations. In *43rd Proceedings of the International Symposium on Automation and Robotics in Construction (ISARC)*. Submitted manuscript, under review.

[3] Colledanchise, M., & Ögren, P. Behavior Trees in Robotics and AI: An Introduction. CRC Press, Clermont, Florida, 2018. doi:10.1201/9780429489105.

[4] Emmenegger, P., & Forth, K. (2025). *IFC and Neo4j Viewer* [source code]. Retrieved from https://github.com/cea-ethz/disassembly-graph-viewer?tab=readme-ov-file

[5] Auryn Robotics. (2026). *Groot2 - The most advanced IDE to create and debug Behavior Trees*. Retrieved from https://www.behaviortree.dev/groot/