An Integrated Product-Process Representation for Deriving Robotic Task Structures using Graph Transformations

Maikel J.S. Brinkhoff[1,2], **Sebastian Esser**[1,2], **and André Borrmann**[1,2]

[1]Chair of Computing in Civil and Building Engineering, Technical University of Munich, Germany
[2]TUM Georg Nemetschek Institute - AI for the Built World, Technical University of Munich, Germany

maikel.brinkhoff@tum.de

Abstract -

This paper introduces a graph-based method for deriving robot-executable task structures from Building Information Models (BIM). While BIM models are widely employed to coordinate design and planning, they are seldom used to facilitate on-site automation. A key challenge is that they typically describe building elements and properties, but not the stepwise procedures required for robotic construction. Existing BIM-to-robot pipelines therefore encode process logic in external artifacts (e.g., schedules, scripts, skill libraries) and connect this logic to BIM elements through ad-hoc mappings, which limits reuse across elements and projects. We propose a workflow that integrates BIM-derived product data with reusable construction method templates and performs systematic refinement into task structures. The BIM model is represented as a labeled property graph, construction methods are modeled as parametrized construction process templates, and graph transformations instantiate element-specific task structures. These graphs capture ordering, dependencies, and required parameters, and are mapped to Behavior Trees (BTs), resulting in an execution-level control structure for on-site robotic tasks. A case study on a multi-story building illustrates that the workflow can be applied at building scale, generating element-specific task structures and corresponding BTs from a single, integrated product-process representation.

Keywords -

BIM; Graph Modeling; Task Decomposition; Behavior Tree; Construction Robotics

1 Introduction

In construction projects, digital design intent must be translated into explicit tasks before it can be executed. Digital models support this transformation by coordinating information across disciplines and project stages. Building Information Models (BIM) have become the dominant representation for this purpose [1], informing planning and scheduling [2] as well as on-site progress monitoring during execution [3]. At the same time, construction work remains largely manual, even as robotic systems are beginning to demonstrate their potential for on-site tasks [4].

Despite its broad use, BIM remains sparsely used to enable on-site automation. These models contain geometric descriptions of building components and semantic information ranging from component properties (e.g., materials) to higher-level objects such as stories and zones. However, they often lack the procedural details needed to derive robot-executable tasks. According to the taxonomy of Everett and Slocum [5], construction processes are typically defined at the activity level, whereas robotic execution needs fine-grained tasks and elemental motions. This mismatch in abstraction leaves a gap between the information available in BIM and the task-level information required for robotic execution.

Recent research has sought to derive actionable information for robotics directly from BIM. Examples include extracting work locations and object context from Industry Foundation Classes (IFC) geometry [6], or combining BIM with onboard sensing to perceive workpiece geometry and update motion plans under changing conditions [7]. While these studies demonstrate the value of BIM-derived information, they remain narrowly scoped. They often address a single task or building element and rely on manually engineered mappings from BIM entities to robot actions. This limits their transfer beyond the specific task setting for which the mapping was designed.

The underlying challenge is representational. Product data, schedules, and robot actions are typically stored in separate artifacts, such as schedules in CSV files, loosely defined process graphs, and hard-coded task scripts. This fragmentation prevents systematic reuse and the consistent capture of process logic at the building scale. Addressing this challenge calls for a unified, higher-level representation that links construction processes and their task structures across building elements, while remaining compatible with robotic execution-level control structures. Graph modeling is identified as a promising approach to represent products, processes, dependencies, and their evolution over time in a single formalism [8].

This paper proposes an integrated product-process representation that links BIM-derived product data, construc-

tion methods, and graph-based process modeling to systematically derive element-specific task structures. These structures remain connected to their source BIM data and are mapped to execution-level representations for robotic systems. The remainder of the paper discusses related research on BIM for construction automation and process modeling with graphs. It then presents the proposed workflow and graph representation and illustrates their application in a case study. It concludes with a discussion of contributions, limitations, and directions for future research.

2 Background

2.1 Deriving Robotic Tasks from BIM Data

Existing BIM-to-robot pipelines mainly differ in where procedural knowledge is represented and at what granularity it becomes executable. A common pattern is to encode process logic in an external artifact and use BIM primarily to supply geometric and semantic parameters. Zhu et al. [9] represent construction intent in a CSV schedule that specifies task sequences per element. Execution parameters, such as IDs or locations, are retrieved from the BIM model. Terzer et al. [10] similarly derive mission-specific parameters from IFC data for wall painting, where high-level missions are decomposed into subtasks and executed using Behavior Trees (BTs) [11]. These approaches demonstrate the value of BIM as a contextual data source for robotic execution. However, process logic is generally defined outside the BIM model and linked to BIM entities through manual mappings. This makes the resulting task structures application-specific and limits their reuse.

A second pattern brings task planning closer to the BIM environment by embedding task definitions in scheduling workflows and linking actions to robotic skills. Oyediran et al. [12] integrate robotic task planning into 4D BIM by associating actions with skill identifiers stored in an external database. This database also specifies additional modeling inputs, such as pickup locations or installation reference frames, which have to be manually added to the BIM model. While this improves traceability and ties tasks more directly to building elements, it still depends on manual modeling inputs and does not provide a general, reusable representation of process logic.

Across these approaches, BIM serves as a repository of product data and execution parameters, while procedural knowledge is spread across heterogeneous artifacts (e.g., schedules, mission scripts, skill libraries, BTs). This split complicates the systematic refinement of product-level descriptions into reusable task structures. It therefore motivates representations that connect element context with explicit process logic in a unified form.

2.2 Graph Representations in BIM and Construction Robotics

Graph representations have increasingly been explored as a way to integrate heterogeneous information and to make relationships between products, processes, and resources explicit. Knowledge graphs are used to support semantic integration and reasoning by translating IFC-derived scene information into graph form and linking it to task and agent models. Chen et al. [13], for example, propose the iSTA (Scene-Task-Agent) model for inspection tasks. The model represents scene context extracted from IFC in a Resource Description Framework (RDF) format combined with robotics ontologies and a custom task ontology to support reasoning and task allocation. In a related fashion, Zhu et al. [9] use a Linked Building Data (LBD) approach to integrate building models and construction schedules by converting IFC and CSV schedule data into RDF graphs. The information stored in the LBD graph is queried by a Robot Construction Action Node (RCAN) system, which serves as an intermediate layer between BIM and robotic execution in ROS and supports task allocation and execution. These approaches highlight how graphs can unify and query contextual data, rather than generating task structures for robotic execution.

Beyond knowledge graphs, task graphs have been used to encode process dependencies more directly. Atanasova et al. [8] introduce a directed graph model for cooperative human-robot assembly. Unlike prior works focused on inspection or static task sequences, their approach captures real-time task dependencies, agent assignments, and evolving assembly states. This example demonstrates how graph models can capture dynamic process logic and dependencies relevant for execution.

Taken together, prior work shows that BIM information can parametrize robotic actions, while graph representations can make process relations and dependencies explicit. Yet these research directions remain insufficiently connected at the step where procedural knowledge is linked to element context. What remains unclear is how this connection can be established systematically and represented in a form that supports robotic execution.

2.3 Control Structures for Execution

Behavior Trees (BTs) organize robotic behavior in a modular hierarchy that separates control flow from execution nodes and provides runtime reactivity [11]. Complex missions are decomposed into executable actions, organized by control logic and supplied with the parameters needed for those actions. This makes BTs relevant not only as an execution formalism, but also as an indicator of what task representations must contain to support robotic execution.

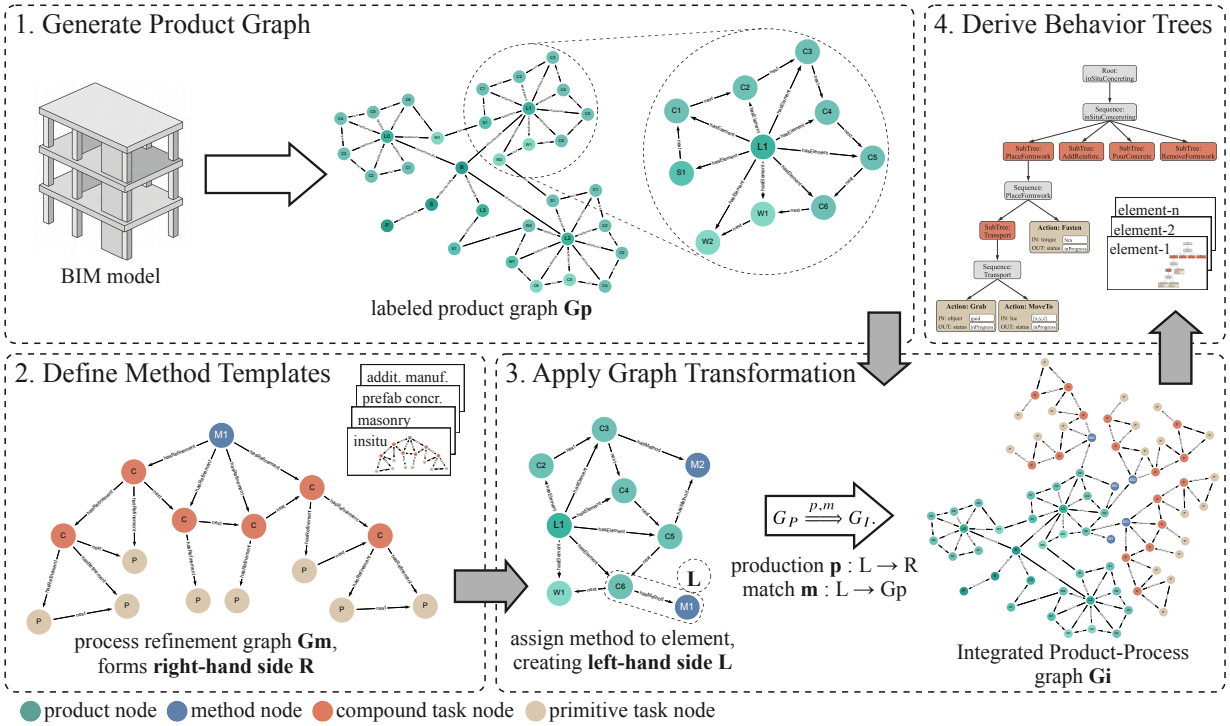

Figure 1. **Overview of the proposed method**: (1) represent a BIM model as a labeled product graph G_p; (2) decomposed construction processes as method templates G_m; (3) user-assigned methods link each product to its method and transform G_p into an integrated product-process graph G_I; and (4) derive Behavior Trees.

In construction robotics, BTs are used as execution structures for tasks parametrized from BIM. Terzer et al. [10] use BIM to provide semantic and geometric parameters for wall painting missions. Similarly, Tandon et al. [14] apply BTs to inspection missions, where BIM data provides target elements and inspection poses, and subsequently generate BTs for navigation and visual inspection. These examples show that BTs are a suitable target structure for describing construction processes. This motivates structuring the task graphs discussed in sec. 2.2 so that element context and process logic can be mapped to BTs, supporting a refinement workflow from BIM-derived product data to execution-level control.

3 An Integrated Product-Process Representation for Robotic Construction

This study proposes a unified workflow that uses graphs to link product data derived from BIM with procedural knowledge and refine this link into task structures suitable for robotic execution. As shown in Fig. 1, the workflow represents the BIM model as an LPG and models construction methods as parametrized templates. Binding these templates to elements triggers graph transformations that create task structures for each building element, yielding an integrated product-process graph. The resulting graph is then converted into a robot-executable format using Behavior Trees. The following sections detail these stages in turn, beginning with product representation, progressing to process structuring and generation, and finally deriving execution-level control structures.

3.1 Representing BIM Data as a Product Graph

We represent the building as a product graph, modeled as an LPG from semantic and geometric information contained in Industry Foundation Classes (IFC) models. This choice aligns with prior work showing that graphs can unify heterogeneous information within a single model [13, 8]. The LPG provides a transformable basis for deriving task structures that can later be converted into robot-executable control structures. Because we focus on construction methods for load-bearing elements, the resulting graph inherently represents the building's structural shell.

The graph follows the IFC spatial hierarchy, grouping building elements under their respective building stories, as illustrated in the upper-left section of Fig. 1. Nodes are labeled as either *SpatialElement* or *BuildingElement*, with their IFC subtype retained as a second label. Spatial elements include the site, building, and its stories, while

relevant building elements include, but are not limited to, beams, walls, columns, and slabs. The representation can be extended with *Resource* nodes (e.g., robots, equipment, or material supply points). The combined labels enable queries not only over spatial and containment relations, but also over specific element types.

Edges capture spatial decomposition and containment relationships from IFC, ensuring that each element remains embedded in its spatial and physical context. We additionally enrich the graph with adjacency relations between neighboring elements, and inter-element ordering relations that can be used to represent a construction sequence. When the graph is extended with process information, typed edges (e.g., *needsResource*) can connect tasks to required resources to support later allocation and feasibility checks. Together, the labeled nodes and edges form the product graph $\boldsymbol{G_p}$. This graph is fine-grained enough to associate construction methods and task structures with individual elements, yet abstract enough to be applicable to typical IFC models. Coarser representations, such as zones, do not expose individual elements as task targets. In contrast, finer decompositions, such as fasteners, are often absent in design-phase models and therefore cannot be assumed.

3.2 Capturing Construction Processes as Method Templates

A construction method describes how a building element is realized in practice, expressed as an ordered sequence of processes. In this work, we adopt this notion for four construction methods: *in-situ concreting, prefabricated concreting, masonry,* and *3D concrete printing*. Each method is modeled as a parametrizable template that organizes its process steps into a reusable structure. The template is element-agnostic at first, becoming element-specific task structures only when bound to a building element. This separation allows the same method to be reused across multiple elements while still adapting to their individual properties. However, construction processes are still too abstract to directly instruct robots [5]. We therefore refine them into tasks (e.g., Grab), which ultimately provide the lower-level units from which executable actions are derived.

3.2.1 Task Decomposition

We structure method templates using Hierarchical Task Network (HTN) planning [15] principles. HTN offers a hierarchical modeling approach that breaks high-level activities down into ordered subtasks. In HTN terminology, high-level activities are *compound* tasks, which can be refined into other compound tasks or into *primitive* tasks. Primitive tasks are directly executable actions such as *Grab, MoveTo,* or *Fasten*. In a graph representation, compound tasks map to internal nodes, and primitive tasks map to leaf nodes. Primitive task nodes specify the required operational properties, such as locations, tolerances, and element IDs. These properties are instantiated, that is, receive their values, when the method template is bound to a specific building element.

In the proposed representation, a construction method is formalized as a directed acyclic graph (DAG) $\boldsymbol{G_m}$, shown in the bottom-left quadrant of Fig. 1. The DAG consists of compound and primitive task nodes linked by refinement (parent-child) relations. For each compound task, the order of its child tasks defines the local execution sequence among siblings. This captures hierarchical decomposition and task ordering in a form that can later be converted into an executable control structure. A partial template example for in-situ concreting is shown below.

Partial in-situ template

```
InSituConcreting
├─ PlaceFormwork
│  ├─ Transport
│  │  ├─ Grab
│  │  └─ MoveTo
│  └─ Fasten
├─ AddReinforcement
├─ PourConcrete
└─ RemoveFormwork
```

3.3 Integrating Product and Process Data

Up to this point, product data and procedural knowledge have been represented separately. The transition from abstract method templates to actionable task structures requires associating these templates with the product data of each element. As shown in the bottom-right quadrant of Fig. 1, we achieve this in two steps: first, a method template is bound to a building element, and second, graph transformation rules instantiate task properties and transfer the template structure, yielding an element-specific task structure.

3.3.1 Method Binding

Assigning construction methods to elements is treated as a design decision and is therefore user-defined. Each assignment is stored in the product graph as a directed *hasMethod* edge from the element node to a *ConstructionMethod* node representing the chosen method. Elements that use the same method reference the same *ConstructionMethod* node, which reflects that the method is a reusable template rather than an element-specific object. Each element-method pair then forms the left-hand side pattern matched by the transformation rules.

3.3.2 Graph Transformation

After assigning a construction method to an element, we apply graph transformation rules [16] to generate element-specific task structures:

$$p : L \rightarrow R, \tag{1}$$

where $\boldsymbol{L}$ denotes the element-method pattern, and denotes $\boldsymbol{R}$ the right-hand side graph that introduces or enriches task nodes and edges according to the method template. We consider four kinds of rules: (i) rules that create task instances for all tasks in the method template and designate the root task for the bound element, (ii) rules that transfer refinement and sequencing relations from the template to the instances, (iii) rules that instantiate primitive task properties using product data, and (iv) rules that link instantiated task nodes to *Resource* nodes where resource requirements are specified. A match

$$m : L \rightarrow G_P, \tag{2}$$

selects an occurrence of the pattern $\boldsymbol{L}$ in $\boldsymbol{G_p}$ (created by the element-method assignment). Applying a rule $\boldsymbol{p}$ under a match $\boldsymbol{m}$ transforms the product graph $\boldsymbol{G_p}$ into an integrated product-process graph $\boldsymbol{G_I}$:

$$G_P \overset{p,m}{\Longrightarrow} G_I. \tag{3}$$

The transformation does not delete or rewrite the product side. Instead, it extends the graph with task structures anchored to building elements. The resulting integrated product-process graph $\boldsymbol{G_I}$ links spatial and physical building information with sequenced construction tasks and forms the basis for the execution model.

3.4 Deriving a Robot-Executable Control Structure

The integrated product-process graph specifies which tasks are required and how they are ordered, but its task structures still need to be expressed as an execution-level control structure. We therefore map the task structures to Behavior Trees (BTs) [11] that preserve the higher-level process logic captured in the graph. While typically the internal control program of a single robot, we use BTs as a shared representation of the construction process.

Compound tasks become *control* nodes (e.g., sequence or parallel) that organize their children according to the refinement and precedence relations in the DAG $\boldsymbol{G_m}$, preserving the hierarchical and ordered structure introduced in Sec. 3.2.1. Primitive tasks become *action* nodes that carry over the element-specific properties. To promote reuse and modularity, recurring compound tasks are represented as standalone *subtrees* that can be referenced from higher-level Behavior Trees (see Fig. 2). A project-level tree then composes the element-specific trees and orders them according to inter-element dependencies, providing a single entry point for execution. The resulting BT is generated per project from the integrated product-process graph, preserving traceability to elements and task data while providing a robot-executable control structure.

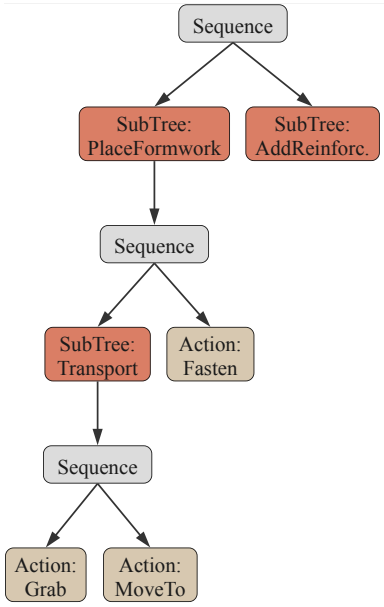

Figure 2. Part of an exemplary Behavior Tree for an in-situ concreting method, illustrating control nodes (sequence), action nodes, and reusable subtrees.

4 Case Study

To illustrate the proposed workflow, we apply it to a multi-story building model and demonstrate how BIM-derived product data, construction method templates, and graph transformations are combined to generate element-specific task structures and Behavior Trees.

4.1 Setup

The case study uses an IFC model of a three-story building, shown in Fig. 3. We restrict the scope to load-bearing elements and consider 51 beams, 36 columns, six walls, and four slabs. The product graph preserves the IFC spatial hierarchy and is enriched with adjacency relations by checking intersections between element axis-aligned bounding boxes. Four construction methods are considered: *in-situ concreting, prefabricated concreting, masonry,* and *3D concrete printing*. Their method templates follow the structure described in sec. 3.2. For the case study, method bindings are assigned manually to illustrate the workflow, for example, in-situ concreting for

slabs and beams, 3D concrete printing for columns and walls, and masonry for walls. The workflow was implemented in Python, using IfcOpenShell for IFC processing, Neo4j for graph storage and transformation, and a custom Python script to query the graph and write Behavior Tree XML files.

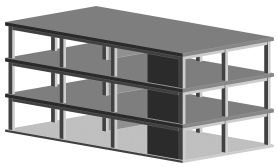

Figure 3. BIM model used for case study.

4.2 Instantiating the Product-Process Graph

The resulting product graph $\boldsymbol{G_p}$ contains seven spatial nodes with six spatial decomposition relationships, 97 building-element nodes and containment relationships, 223 adjacency relationships, and 96 precedence relationships between elements. We obtain the element precedence relationships through a simple centroid sorting along the Z, Y, and X axes. These relations are not intended to represent an accurate construction sequence, but provide an initial project-level order that can later be replaced by more explicit sequencing rules.

Method binding is realized using an interactive IFC viewer [17], where users select building elements and assign a construction method to each of them. Applying graph transformation rules yields the integrated product-process graph $\boldsymbol{G_I}$, where method templates are instantiated and each element is linked to a rooted task structure. For the case study model, the transformations generate 1,138 task nodes, comprising 396 compound tasks and 742 primitive tasks, connected by 1,480 refinement edges and 792 precedence edges. On average, the in-situ concreting, prefabricated concreting, and masonry templates contribute 10, 7, and 5 primitive tasks per element, respectively, reflecting differences in method complexity. The resulting task structures are acyclic and anchored to their elements, indicating that the method templates can be reused consistently at the building scale.

4.3 Deriving Behavior Trees from Task Structures

The task structures in $\boldsymbol{G_I}$ provide the basis for deriving Behavior Trees that describe the construction process independently of a specific robot platform. They specify *what* must be done and in which order, while robot selection determines *how* tasks are executed. This separation is important because the considered methods may be realized using different platforms (e.g., in-situ concreting via a pump, prefabrication using lifting and placement equipment such as a crane, masonry with a bricklaying robot, and 3D concrete printing with a mobile printing system).

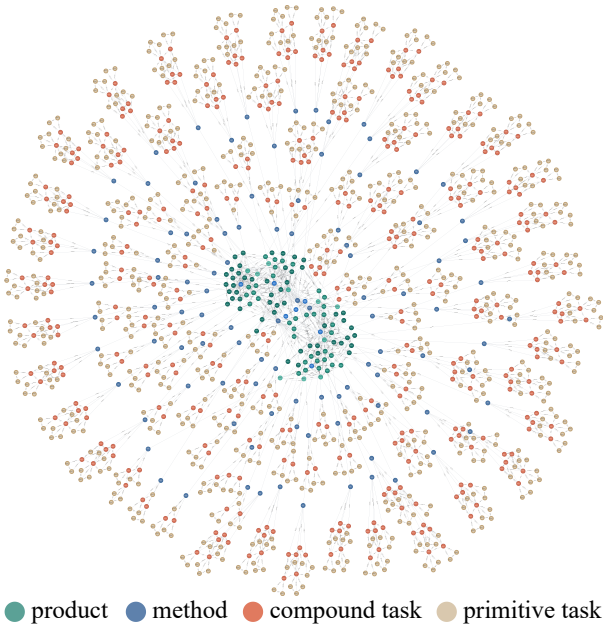

Figure 4. Product-Process Graph for a Simple Building with Multiple Methods.

For each element-method pair, the rooted task structure is mapped to a BT subtree. Compound tasks become control nodes, while primitive tasks become parametrized action nodes. This mapping is applied to all elements with assigned methods. Fig. 5 shows a representative BT for an element assigned to the in-situ concreting template. Its action (leaf) nodes carry the parametrized values of a particular element, for instance the element ID and target location used by the *GrabObject* and *MoveTo* actions.

Across the case study, the workflow generates 98 BTs: one BT per element in scope plus one project-level BT. These BTs preserve the method template structure and remain linked to their corresponding building elements. This demonstrates that the refinement of BIM-derived product data and procedural knowledge into task structures carries through to an execution-level control structure.

5 Discussion

The case study shows that construction method templates can be reused to generate element-specific task structures systematically. A small set of methods is applied across the load-bearing elements, and graph transformations instantiate rooted task subgraphs for each element from the corresponding method binding. This indicates that reusable method templates are applied consistently at

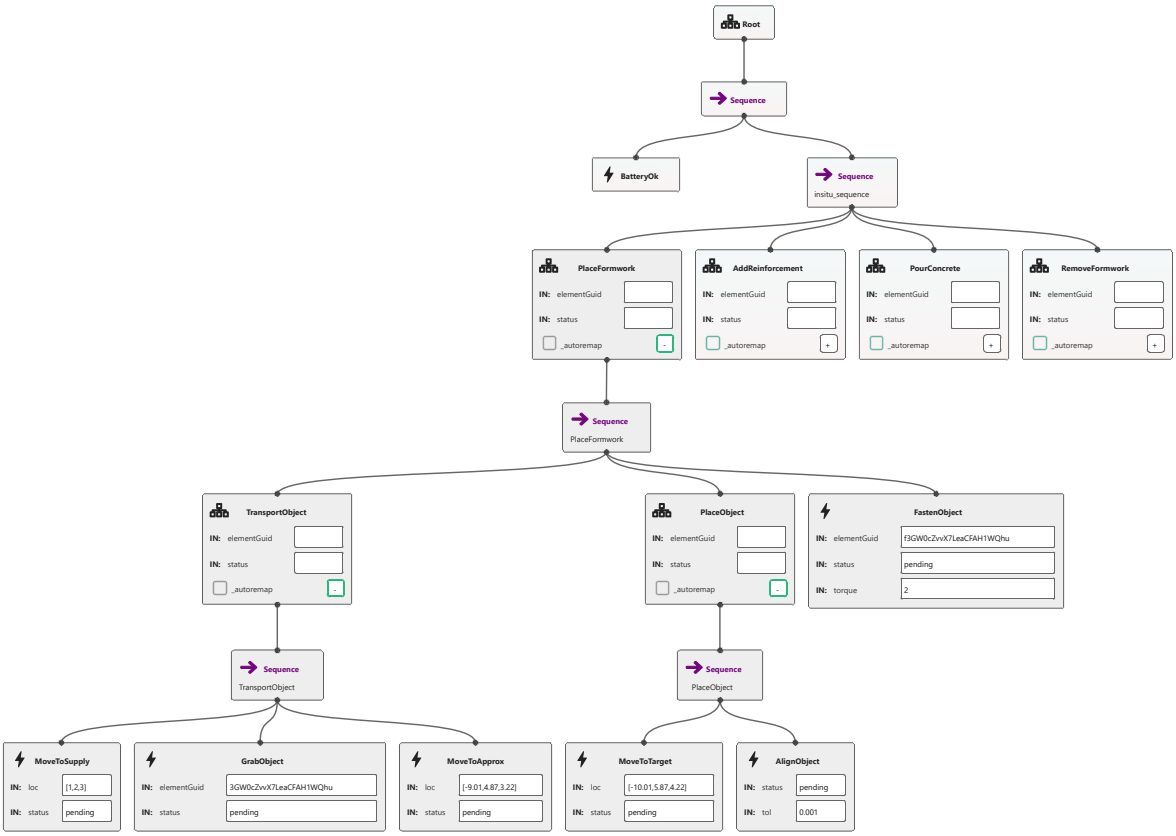

Figure 5. Behavior Tree derived from product-process graph.

the project level.

The proposed representation complements knowledge graph approaches in BIM [9, 13] by linking product information and process knowledge in a form that can be mapped to execution-level control structures. Instead of distributing process information across schedules, mission scripts, or skill libraries, the integrated product-process graph stores task decomposition and ordering together with element context. Queries can therefore access process semantics (e.g., predecessors, successors, refinement relations) alongside product attributes. The same graph structure can be extended with additional dimensions such as resources or temporal constraints. This creates a flexible layer for future functionalities such as robot-task allocation or discrete-event simulation.

Despite its merits, the current approach has limitations. LPGs are flexible and easy to extend, but their schemaless nature offers limited guarantees about validity and structure. A closer alignment with the IFC schema would not only provide more structure but also support more complex reasoning. For example, this would enable spatial reasoning, where queries could infer whether a robot can reach a target pose by passing through doors of sufficient width, or traverse terrain such as stairs. Such queries would extend the current lightweight representation beyond element-specific task generation toward a richer representation of construction context. Additionally, method templates are defined manually, which raises questions of modeling effort and scalability as more construction methods are added. Finally, the present work stops at Behavior Tree generation and does not yet validate runtime execution. Future work will therefore focus on more systematic template definition and on evaluating the generated structures in simulation and with physical robots.

6 Conclusion

This paper presents a refinement workflow that links BIM product data with procedural knowledge to generate element-specific task structures for robotic construction. Product data is represented as a labeled property graph (LPG), and construction methods as parametrized templates. Graph transformations instantiate element-specific task structures, which are then mapped to Behavior Trees (BTs) as execution-level control structures. We evaluate the approach on a multi-story building model. For the load-bearing elements, we build an integrated product-process graph, reuse a small set of method templates across

many elements, and generate one BT subtree for each element to match the intended process logic. This reduces the need for hand-crafted BTs and keeps each subtree explicitly linked to its source element. Because tasks and product context share a single graph, the model can be extended with resource and timing constraints, used for robot-task allocation, or updated during execution by integrating perception feedback.

In summary, the proposed workflow consolidates fragmented product, process, and execution artifacts into a single product-process model that yields robot-executable task structures. It supports systematic refinement and reuse across elements and provides a foundation for reuse beyond a single project, paving the way for more scalable BIM-to-robot workflows in construction robotics.

Acknowledgments

We gratefully acknowledge the support of the German Research Foundation (DFG) for funding the project under grant FOR 5672, subproject A "Generic framework for simulating and executing robotized construction", grant no. 517965147. Furthermore, we acknowledge the support of the TUM Innovation Network "Co-Construct", the DFG Transregio 277 "Additive Manufacturing for Construction" (grant 414265976), and the DFG project 510047432 as part of the program SPP 2187 "Adaptive modularized constructions made in flux" (grant 423969184).

References

[1] A. Borrmann, M. König, C. Koch, and J. Beetz. *Building Information Modeling: Technology Foundations and Industry Practice*, volume 1. Springer International Publishing AG, 2018. doi:10.1007/978-3-319-92862-3.

[2] P.F. Ahmadi and M. Arashpour. An analysis of 4d-bim construction planning: advantages, risks and challenges. In *Proc. of the 37th Int. Symp. on Automation and Robotics in Construction (ISARC 2020)*, pages 163–170, 2020. doi:10.22260/ISARC2020/0025.

[3] F. Pfitzner, S. Hu, A. Braun, A. Borrmann, and Y. Fang. Monitoring concrete pouring progress using knowledge graph-enhanced computer vision. *Automation in Construction*, 174, 2025. doi:10.1016/j.autcon.2025.106117.

[4] C.J. Turner, J. Oyekan, L. Stergioulas, and D. Griffin. Utilizing industry 4.0 on the construction site: Challenges and opportunities. *IEEE Transactions on Industrial Informatics*, 17(2), 2021. doi:10.1109/TII.2020.3002197.

[5] J.G. Everett and A.H. Slocum. Automation and robotics opportunities: Construction versus manufacturing. *Journal of Construction Engineering and Management*, 120, 1994. doi:10.1061/(ASCE)0733-9364(1994)120:2(443).

[6] S. Kim, M. Peavy, P. Huang, and K. Kim. Development of bim-integrated construction robot task planning and simulation system. *Automation in Construction*, 127, 2021. doi:10.1016/j.autcon.2021.103720.

[7] K.M. Lundeen, V.R. Kamat, C.C. Menassa, and W. McGee. Autonomous motion planning and task execution in geometrically adaptive robotized construction work. *Automation in Construction*, 100, 2019. doi:10.1016/j.autcon.2018.12.020.

[8] L. Atanasova, J.C. Kenny, E.V. Alexi, F. Chovghi, D. Mitterberger, and K. Dörfler. Building (with) human–robot teams: fabrication-aware design, planning, and coordination of cooperative assembly processes. *Construction Robotics*, 9(2), 2025. doi:10.1007/s41693-025-00158-w.

[9] A. Zhu, P. Pauwels, E. Torta, H. Zhang, and B. de Vries. Data linking and interaction between bim and robotic operating system (ros) for flexible construction planning. *Automation in Construction*, 163, 2024. doi:10.1016/j.autcon.2024.105426.

[10] M. Terzer, T. Flatscher, M. Magri, S. Garbin, J. Emig, and A. Giusti. A facilitated construction robot programming approach using building information modelling. In *10th Int. Conf. on Control, Decision and Information Technologies (CoDIT)*, pages 2656–2661, 2024. doi:10.1109/CoDIT62066.2024.10708285.

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

[12] H. Oyediran, W. Turner, K. Kim, and M. Barrows. Integration of 4d bim and robot task planning: Creation and flow of construction-related information for action-level simulation of indoor wall frame installation. *Journal of Information Technology in Construction*, 30, 2025. doi:10.36680/j.itcon.2025.015.

[13] J. Chen, W. Lu, Y. Fu, and Z. Dong. Automated facility inspection using robotics and bim: A knowledge-driven approach. *Advanced Engineering Informatics*, 55, 2023. doi:10.1016/j.aei.2022.101838.

[14] A. Tandon, J. Stührenberg, and K. Smarsly. Automated building inspections coupling building information modeling and behavior trees. In *Proc. of the 2025 European Conference on Computing in Construction*, 2025. doi:10.35490/EC3.2025.360.

[15] K. Erol, J. Hendler, and D.S. Nau. HTN Planning: Complexity and Expressivity*. In *12th AAAI Conf. on Artificial Intelligence*, pages 1123–1128, 1994.

[16] A. Corradini, U. Montanari, F. Rossi, H. Ehrig, R. Heckel, and M. Löwe. *Handbook of Graph Grammars and Computing by Graph Transformation*, chapter Algebraic Approaches to Graph Transformation - Part I: Basic Concepts and Double Pushout Approach, pages 163–245. World Scientific, 1997. doi:10.1142/9789812384720_0003.

[17] T. Wrabel, S. Esser, and A. Borrmann. A framework architecture for the information backbone for robotic construction. In *Proc. of the 43rd Int. Symp. on Automation and Robotics in Construction (ISARC 2026)*, 2026.