Journal of Intelligent Instrumentation, Automation, and Control Systems
Volume 1, Issue 2, May-August, 2025

Behavior Trees (Bt) In Plc And Automation Controllers: A Modular Approach To Intelligent Industrial Automation

Dr. Ananya Mehta[1], Prof. Rajesh Kumar Singh[2]
Assistant Professor[1], Professor[2]
[1]Department of Electrical Engineering, [2]Department of Electronics and Instrumentation Engineering
[1]Indian Institute of Technology, Delhi (IIT Delhi), [2]National Institute of Technology, Trichy (NIT Trichy)
Email ID: *ananya.mehta123@gmail.com[1], rajesh.singh.nitc@yahoo.co.in[2]*

ABSTRACT

Behavior Trees (BT) are increasingly emerging as a robust methodology for structuring control logic in automation systems, particularly in programmable logic controllers (PLCs) and advanced industrial controllers. Unlike traditional sequential or ladder-based programming paradigms, BTs offer modularity, scalability, and clear task hierarchies, making them ideal for complex, dynamic, and autonomous industrial operations. This paper explores the foundational concepts of Behavior Trees, their integration into PLC-based automation systems, and the potential benefits and challenges associated with their deployment. The discussion further extends to the practical implications for industries aiming for adaptive and intelligent automation frameworks.

KEYWORDS: *Behavior Trees, PLC, Automation Controllers, Industrial Automation, Modular Control, Intelligent Systems, Task Hierarchy, Reactive Systems*

INTRODUCTION

Overview of Automation Controllers

Programmable logic controllers (PLCs) have long been the backbone of industrial automation, managing processes ranging from simple conveyor systems to highly complex manufacturing lines. Traditional PLC programming often relies on ladder logic, function block diagrams, or structured text, which, while effective, can become cumbersome for highly dynamic or adaptive applications.

Need for Advanced Control Paradigms

With the rise of smart manufacturing and Industry 4.0 initiatives, automation systems increasingly require flexibility, modularity, and decision-making capabilities beyond static control sequences. Traditional PLC logic, while deterministic and reliable, struggles to adapt to unstructured or real-time decision-making requirements. Behavior Trees offer a solution by combining reactive control, modularity, and clear task prioritization.

Behavior Trees in Industrial Context

Originating from the field of robotics and game AI, Behavior Trees structure tasks hierarchically, allowing systems to make decisions dynamically while maintaining predictable behavior. This capability is particularly valuable in industrial environments, where multiple tasks may compete for resources, and operational conditions frequently change. Integrating BTs into PLCs and automation controllers enables more intelligent, maintainable, and scalable systems.

Table 1: Comparison of PLC Programming Paradigms

Feature	Ladder Logic	Function Block	Structured Text	Behavior Trees (BT)
Modularity	Low	Medium	Medium	High
Scalability	Low	Medium	High	High
Reactivity	Low	Medium	Medium	High
Ease of Debugging	Medium	Medium	High	High
Adaptability	Low	Medium	Medium	High

LITERATURE REVIEW

Behavior Trees: Definition and Origins

Behavior Trees are formal, tree-structured representations of system behaviors. Each node in a BT represents a task, condition, or action, organized into a hierarchy where parent nodes define the execution logic of child nodes. Unlike finite state machines, which can become exponentially complex with increasing system states, BTs maintain clarity and modularity.

Applications in Robotics and Gaming

Initially, BTs gained prominence in robotics, enabling adaptive behavior in autonomous systems. Gaming industries adopted BTs for non-player character AI due to their modularity and scalability. These applications highlight BTs' potential for decision-making under uncertainty, reactive behavior, and task prioritization—all crucial attributes for modern industrial automation.

Integration into PLCs and Automation Systems

Recent studies demonstrate the feasibility of implementing BTs on PLCs using structured text or function blocks. The modular nature of BTs complements PLC programming methodologies, allowing engineers to design reusable and maintainable control sequences. BTs also facilitate clearer mapping between high-level operational goals and low-level actuator commands.

Comparative Advantages Over Traditional Logic

Compared to sequential or ladder-based logic, BTs offer several advantages:

- **Modularity:** Individual behaviors can be developed and tested independently.
- **Scalability:** Trees can be expanded without affecting unrelated modules.
- **Reactivity:** BTs inherently support dynamic decision-making based on real-time conditions.
- **Maintainability:** Hierarchical structure simplifies debugging and system updates.

IMPLEMENTATION OF BEHAVIOR TREES IN PLC SYSTEMS

Table 2: Example Behavior Tree Nodes for A Robotic Assembly Line

Node Type	Function	Example in Assembly Line
Root Node	Defines overall system behavior	"Complete Assembly Task"
Sequence Node	Executes child nodes in order	"Pick Part → Inspect Part → Place Part"
Selector Node	Chooses first successful child node	"If Part OK → Assemble, Else → Reject"
Decorator Node	Modifies child node behavior	"Retry task 3 times if failure occurs"

Node Type	Function	Example in Assembly Line
Leaf Node	Performs actual task/action	"Activate Gripper", "Move Robot Arm"

STRUCTURE OF BEHAVIOR TREES (BT) IN PLC AUTOMATION

Behavior Trees consist of several key elements:

1. Root Node:

The Root Node is the top-most node of the Behavior Tree and represents the overarching behavior or primary goal of the system. It acts as the entry point for the evaluation and execution of the entire tree. In a PLC-based system, the Root Node may represent a full operational cycle, such as “Complete Assembly Line Operation” or “Manage Production Workflow.” The Root Node ensures that all child behaviors are evaluated in a structured and hierarchical manner, establishing a clear flow of control.

2. Composite Nodes:

Composite Nodes define the logic of execution for child nodes and determine how tasks are selected and sequenced. There are three common types of composite nodes:

- Sequence Node: Executes child nodes sequentially, moving to the next task only if the current task succeeds. In a PLC-controlled assembly line, a Sequence Node may enforce the order “Pick Part → Inspect Part → Place Part,” ensuring that no step is skipped.
- Selector Node: Evaluates child nodes in order and selects the first one that succeeds. It is used for conditional branching, such as handling defective parts versus acceptable parts, providing adaptive decision-making.
- Parallel Node: Executes multiple child nodes simultaneously, suitable for tasks that can run concurrently, such as monitoring multiple sensors while performing assembly tasks.

Composite nodes are essential for organizing complex behaviors into reusable and modular blocks, which is a key advantage in industrial automation.

3. Decorator Nodes:

Decorator Nodes modify or influence the execution of a child node, often by adding constraints or additional conditions. For instance, a Decorator Node might enforce rules such as:

- Retry a task if it fails.
- Execute a task only if a certain sensor condition is met.
- Limit execution frequency to prevent overloading actuators.

In PLC implementation, Decorator Nodes allow conditional logic to be embedded at the task level, making the system reactive to real-time inputs while maintaining a clear structure.

4. Leaf Nodes:

Leaf Nodes are the terminal nodes of the Behavior Tree and correspond to actionable tasks or decisions executed by the system. These are the concrete operations, such as activating a gripper, moving a robotic arm, or sending a signal to a conveyor motor. Leaf Nodes interface directly with PLC routines, input/output modules, or actuators, bridging high-level decision-making with low-level physical execution.

MAPPING BTs TO PLC ARCHITECTURE

PLC systems are traditionally event-driven or operate in cyclic scan modes. Behavior Trees can be implemented in these systems by translating each BT node into a PLC-compatible routine, function block, or structured text module.

- Root and Composite Nodes: Can be represented as high-level control routines or function blocks, managing execution flow.
- Decorator Nodes: Are translated into conditional checks or control logic embedded in function blocks.
- Leaf Nodes: Map directly to PLC I/O operations, such as turning on a motor, reading a sensor, or sending a signal to a robotic actuator.

Scan Cycle Integration:

PLC scan cycles provide a deterministic timing framework. During each cycle, the BT evaluation occurs, checking conditions, executing nodes, and handling failures. This ensures reliable and predictable operation even in dynamic or real-time environments. The hierarchical structure of BTs also allows non-blocking execution, meaning that the system can continue evaluating alternative paths without halting the main process.

EXAMPLE APPLICATION IN MANUFACTURING

Consider a robotic assembly line where parts are inspected, assembled, and packaged:

- Selector Node: Handles alternate paths based on part quality. For example, if a part is defective, it is routed to a reject bin; if acceptable, it proceeds to assembly. This dynamic decision-making allows the system to respond to real-time changes without manual intervention.
- Sequence Nodes: Define the precise order of operations. For example, “Pick Part → Inspect Part → Assemble → Place → Package” ensures tasks are executed in a logically consistent order.
- Decorator Nodes: Add constraints such as “only assemble if part quality is acceptable” or “retry assembly up to three times on minor errors.”

Dynamic Adaptation:

The Behavior Tree structure enables the PLC-controlled system to adapt to changes like machine downtime, defective parts, or priority orders without rewriting the core logic. For instance, if a conveyor stops, the BT can automatically reroute tasks or pause non-critical operations until the issue is resolved. This flexibility reduces downtime, improves reliability, and simplifies maintenance.

Key Benefits in Manufacturing:

- Modular and reusable task definitions.
- Clear separation of decision logic and execution logic.
- Enhanced scalability for large or complex production lines.
- Simplified debugging and maintenance due to hierarchical structure.

ADVANTAGES OF BEHAVIOR TREES IN INDUSTRIAL AUTOMATION

1. Enhanced Modularity and Reusability

Behavior Trees promote a highly modular structure where individual tasks or behaviors are encapsulated into discrete, reusable nodes. Each node, whether it is a leaf, decorator, or composite node, can be developed, tested, and maintained independently. This modularity allows engineers to reuse common behaviors across multiple automation sequences without rewriting the logic.

For example, a “pick-and-place” operation can be implemented as a leaf node and reused across different assembly lines, whether the tasks involve handling small electronic components, mechanical parts, or packaging items. If the logic for handling a part changes—for instance, the gripper type is upgraded—the corresponding node can be updated without affecting other parts of the tree. This reduces programming time, minimizes errors, and ensures consistency across different automation processes.

Additionally, modularity enables the creation of libraries of standardized BT nodes. These libraries can accelerate development for future projects and support a scalable approach for increasingly complex manufacturing operations.

2. Improved System Reliability

Behavior Trees implement a hierarchical decision-making framework that inherently prioritizes tasks based on their importance and conditions. High-priority operations, such as emergency shutdowns, quality checks, or safety interlocks, are always evaluated before lower-priority actions.

This hierarchical control improves system reliability in several ways:

- Dynamic Task Management: The BT can respond immediately to changing conditions, such as machine malfunctions or material shortages, by rerouting or pausing lower-priority tasks.
- Fault Tolerance: Failure in a non-critical node does not halt the entire system. Alternative branches can be activated to maintain operation continuity.
- Predictable Behavior: The clear, structured execution flow ensures that critical tasks are not skipped, even in complex environments with concurrent activities.

By ensuring that tasks are executed in a controlled and prioritized manner, BTs reduce unplanned downtime and prevent cascading failures that could disrupt production.

3. **Facilitation of Human-Machine Collaboration**

A key advantage of BTs is their transparency. Unlike traditional ladder logic, where complex interconnections can obscure system behavior, BTs present tasks and decision-making logic in a hierarchical tree format that is visually interpretable.

This transparency benefits human operators and engineers in multiple ways:

- Easier Monitoring: Operators can quickly understand which tasks are active, pending, or blocked.
- Simplified Intervention: If an anomaly occurs, engineers can identify the responsible node and intervene without needing to stop the entire system.
- Collaborative Automation: In human-robot interaction scenarios, BTs allow humans to set high-level objectives while the system autonomously manages task execution. For instance, an operator may instruct a robotic line to prioritize urgent orders, and the BT ensures the correct sequence of tasks is executed.

This collaborative approach enhances safety, productivity, and operational flexibility.

4. Simplified Debugging and Maintenance

Behavior Trees separate conditions, decision-making logic, and action execution into distinct nodes. This separation enables engineers to isolate, test, and debug specific modules without affecting the overall system operation.

Advantages for maintenance include:

- Fault Localization: Engineers can quickly identify the failing node (e.g., a sensor check, actuator command, or decision condition) and correct it without extensive system downtime.
- Incremental Testing: Individual nodes or branches can be tested independently before integration into the full tree, ensuring reliability from the ground up.
- Ease of Updates: System modifications, such as adding new tasks, updating sequences, or changing priorities, can be implemented at the node level without requiring a full system redesign.

By improving the maintainability of automation systems, BTs reduce engineering workload, minimize operational disruptions, and support long-term system scalability.

CHALLENGES IN IMPLEMENTING BEHAVIOR TREES IN PLC SYSTEMS

Computational Overhead

Complex BTs with numerous nodes may increase PLC cycle times, particularly in systems with limited processing power. Efficient node evaluation strategies and prioritization mechanisms are essential.

Compatibility with Existing PLC Architectures

Not all PLCs natively support structured or hierarchical programming needed for BTs. Implementing BTs may require additional software layers or firmware modifications.

Training and Skill Requirements

Engineers accustomed to traditional ladder logic may require training to design, implement, and debug BT-based control systems. Bridging this skill gap is critical for successful adoption.

Scalability Concerns in Extremely Large Systems

While BTs are modular, extremely large trees may become difficult to manage without rigorous design standards and documentation practices.

SCOPE AND FUTURE PROSPECTS

Integration with Industry 4.0

BTs align closely with Industry 4.0 goals, such as adaptive manufacturing, predictive maintenance, and autonomous decision-making. Coupled with IoT sensors, AI algorithms, and cloud-based analytics, BTs can enable highly intelligent PLC systems capable of self-optimization.

Hybrid Approaches

Future research may explore hybrid control architectures combining BTs with traditional PLC logic, finite state machines, or neural networks to leverage the strengths of multiple methodologies.

Expansion to Multi-Agent Systems

BTs are suitable for coordinating multiple robotic agents or subsystems, allowing decentralized decision-making while maintaining overall system coherence.

Potential for Standardization

As adoption grows, developing standard libraries of BT modules for common industrial tasks can significantly reduce implementation complexity and promote best practices.

CONCLUSION

Behavior Trees present a powerful, modular, and flexible approach for PLC and automation controller programming. Their hierarchical structure, reactivity, and clarity make them ideal for complex industrial processes requiring adaptive behavior. While challenges such as computational load and skill requirements exist, the benefits in terms of modularity, maintainability, and intelligent decision-making are substantial. As industries increasingly adopt smart manufacturing principles, BTs are poised to become an essential tool in the evolution of intelligent industrial automation systems. Future developments, including hybrid architectures and standardization, will further enhance their utility and adoption in industrial automation.

REFERENCES

1. Sidorenko, A., Rezapour, M., Wagner, A., & Ruskowski, M. (2024). *Towards using behavior trees in industrial automation controllers*. Procedia CIRP, CIRP Conference on Manufacturing Systems. dfki.de+2Emergent Mind+2
2. Ögren, P., & Sprague, C. I. (2022). *Behavior trees in robot control systems*. arXiv. arXiv
3. Biggar, O., Zamani, M., & Shames, I. (2020). *A principled analysis of Behavior Trees and their generalisations*. arXiv. arXiv
4. Obermeier, M., Braun, S., & Vogel-Heuser, B. (2022). *A model driven approach on object oriented PLC programming for manufacturing systems with regard to usability*. arXiv. arXiv
5. Hallén, M., Iovino, M., Sander-Tavallaey, S., & Smith, C. (2024). *Behavior trees in industrial applications: A case study in underground explosive charging*. arXiv. arXiv
6. Hutter-Mironovova, M., Blumhofer, B., Schneider, C., & Wagner, A. (2024). *Behavior Tree as a decision planning algorithm for industrial robot*. In *Advances on P2P, Parallel, Grid, Cloud and Internet Computing* (3PGCIC-2024). Springer. springerprofessional.de
7. Zhang, Q., Yao, J., Yin, Q., & Zha, Y. (2018). Learning Behavior Trees for autonomous agents with hybrid constraints evolution. *Applied Sciences, 8*(7), 1077. https://doi.org/10.3390/app8071077 MDPI+1
8. Colledanchise, M., Parasuraman, R., & Ögren, P. (2015). *Learning of Behavior Trees*

for autonomous agents. arXiv. arXiv+1

9. Scheide, E., … (2021). Behavior Tree Learning for Robotic Task Planning Through Monte Carlo Tree Search. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*. research.engr.oregonstate.edu
10. Fusaro, F., Lamon, E., De Momi, E., & Ajoudani, A. (2021). *An integrated dynamic method for allocating roles and planning tasks for mixed human-robot teams*. arXiv. arXiv