A Framework for Modular PLC Programming Based on Object-Oriented Design

How software engineering principles can tame the complexity of industrial automation code

Authors
Jonathan Edenburg & Oliver Simonsson

Industry Partner
Etteplan — International Engineering Consultancy

Platform
TwinCAT 3 / IEC 61131-3 (4th ed.)

01 · THE PROBLEM

Industrial automation code is costly to maintain

Programmable Logic Controllers (PLCs) are the computers running the machines that make modern manufacturing possible, such as conveyor belts, packaging lines, and production equipment of all kinds. They run continuously, often for years, and any failure can halt an entire production line at significant cost.

Yet the software controlling these machines is frequently written without consistent structure. Engineers at Etteplan, an international engineering consultancy, identified four recurring problems across their PLC projects that this thesis set out to address:

Unclear program flow

Heavy use of global variables makes it hard to trace what controls what, especially in large or legacy systems.

Low code reusability

Similar logic is rewritten from scratch in every new project due to tight coupling with specific hardware.

Steep onboarding curve

New engineers face steep learning curves because there is no shared structure to orient themselves by.

Limited scalability

Without enforced boundaries, adding new functionality risks breaking other parts of the system.

02 · Our Approach

Research grounded in real engineering practice

This thesis used Design Science Research, a methodology for producing knowledge through the act of building a solution. Rather than studying problems in isolation, we developed the framework iteratively alongside Etteplan, testing our design decisions against both existing literature and the lived experience of practicing engineers.

Three methods were combined to inform the design

1 Literature review

OOP principles, IEC 61131-3 standard (all 4 editions), industrial design patterns.

2 Interviews

Semi-structured interviews with automation engineers; seven recurring themes identified.

3 Codebase review

Analysis of real Etteplan project code to identify structural patterns.

4 Implementation

Partial build in Beckhoff's TwinCAT 3 to validate that the design works in practice

> *"Architecture as communication; how the code is structured tells you as much about the machine as the code itself does."*
> — Recurring theme across all five engineer interviews

05 · Looking Ahead

Future development directions

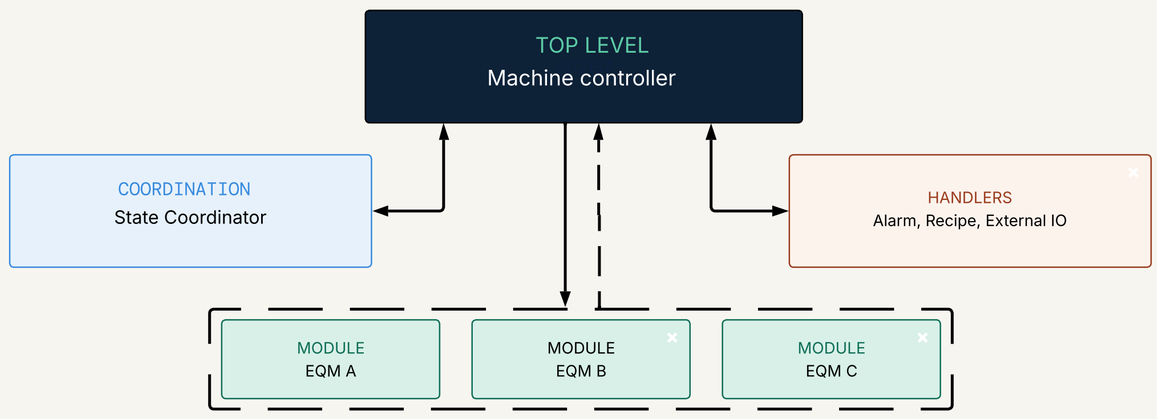
Two of the six core components, the Recipe Handler and the I/O Handler, were fully designed and specified but not implemented within the scope of this thesis. Both are documented and the implemented components are structurally ready to receive them. The most immediate continuation is deploying the framework on a real Etteplan project, where production conditions will surface edge cases that are difficult to anticipate in design.

03 · The Solution

A modular, object-oriented framework for PLC development

The fourth edition of the international standard for PLC programming languages, IEC 61131-3 (released 2025), added or extended support for object-oriented programming constructs. Among these were encapsulation, inheritance, interfaces, and abstract methods. Our framework uses these constructs to enforce consistent structure across projects, not merely through documentation or convention, but through the compiler itself.

The architecture is built around three core layers:



Each Equipment Module (EQM) encapsulates the logic for one physical or functional subsystem of a machine, its internal state, alarm conditions, and HMI data. The State Coordinator governs when transitions between operational phases are permitted, only allowing a transition once every registered EQM signals readiness. The Alarm Handler centralises all alarm data flow, decoupling alarm detection in EQMs from presentation on the operator interface. All communication between layers happens through interface contracts that the compiler enforces making it structurally impossible to skip or bypass the architecture.

04 · Key Results

Structure as a property, not a convention

The designed framework delivers its core promise: any engineer working within it, regardless of experience level, has a foundation to develop code that conforms to the same architecture. Abstract method contracts and compiler-verified interface requirements mean that consistency is no longer something that must be entirely enforced through code review or documentation. It is built into the system itself.

6
Core framework components designed and specified.

3
Components implemented in TwinCAT 3.

5
Engineer interviews conducted; 7 themes identified.

0
Physical hardware needed to test simulation is structural.

A notable consequence of the architecture is hardware-independent testability. Because the Machine Controller is a Function Block rather than a Program, it can be instantiated in a test harness and any EQM replaced by a simulated mock without modifying any surrounding logic. This makes it possible to develop and validate control logic entirely on a desktop, before hardware is ever connected.

CORE TAKEAWAY

The framework proposes an OOP-based foundation for future PLC-projects. It delivers structural consistency rather than relying entirely on convention. In automation, code quality often varies by engineer, but this approach uses abstract method contracts, typed communication, and compile-time module checks to ensure uniform architecture across projects. It could reduce long-term costs from inconsistent code, including onboarding friction, handover errors, and duplicated effort.