Maximize Efficiency: Leveraging Emulate3D for Automated PLC Code Creation

Agenda

- WiseSyde introduction
- Motivation
- Context
- Our vision
- Design tools
- Code generation
- Achievements
- Impact

Wise Technical ltd

- Consultancy and Engineering in aviation sector since 2014
- Based in London

Invesyde PM&A

- 20 years of experience in Logistics automation
- Emulate3D VAR since 2015
- Based in Madrid

Motivation

Automate tasks that

- Add low value
- Are repetitive
- We don´t like

Improve quality

- Better code, more readable, more maintainable
- Fewer errors

Improve performance

- Build faster
- Reduce troubleshooting time
- Reduce testing time

Context

Electrical / Controls upgrade Project in major airport

- Panels
- Wiring
- Controls equipment including Siemens Safety CPUs
- Field networks

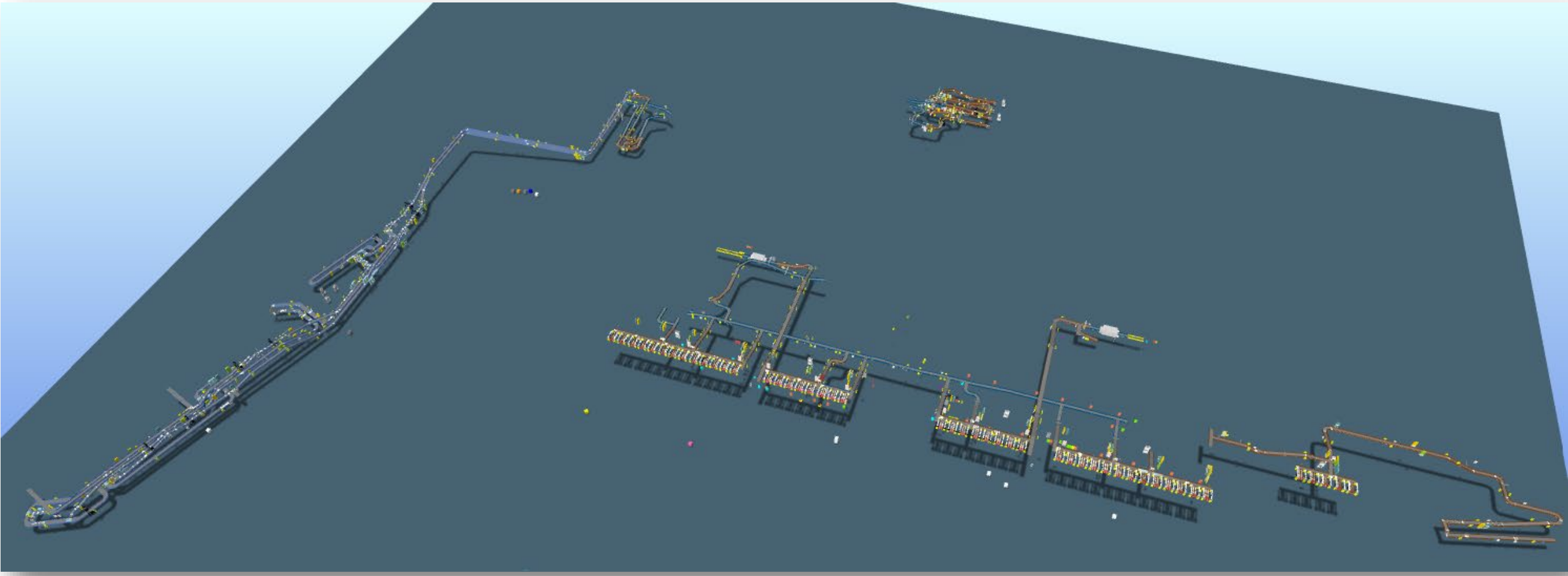
Scope

- 12 PLCs
- Check-in / Transfers / Sortation / make-up areas

Constraints

- Align to latest airport standards including PLC blocks library and SCADA interface
- Keep existing interface with HLC systems
- Keep existing interface with surrounding PLCs and third party equipment

Context



Our vision

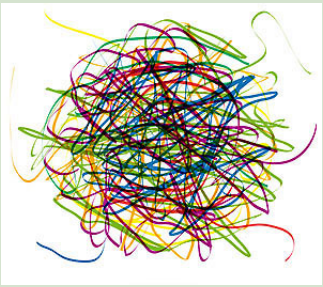
≈ 2.500.000 years ago

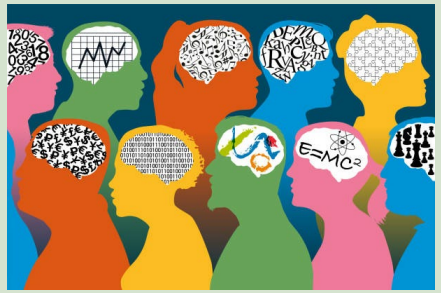
PRESENT

Our vision

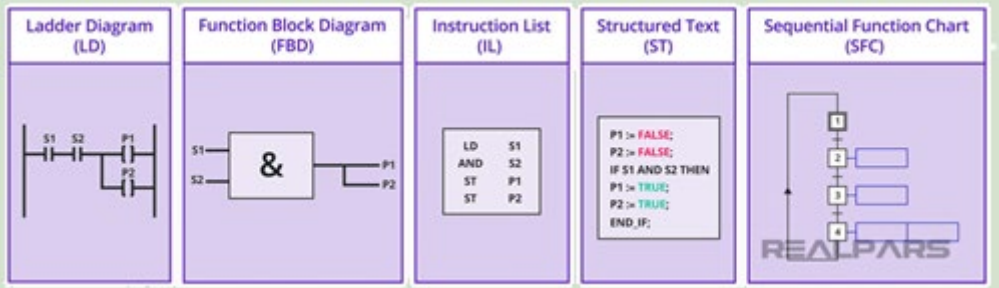
IEC 61131-3

BEFORE





AFTER



1993

PRESENT

Our vision

FB_Line.scl

```
VAR_INPUT
   inLineConfig : "UDT_Line_Config";   // Line config
   inModelConnected : Bool;   // Model connected to PLC
   inGlobalFaultReset : Bool;
   inGlobalStatisticsReset : Bool;
   inSemiautoMode : Bool;
END_VAR

VAR_OUTPUT
   outRouteStatus : "UDT_Route_Status";   // Route status
   outRouteStatusModel : "UDT_EQUIPMENT_STATUS_MODEL";   // Route status model
   outRouteAvailable : Bool;   // Route availability
   outConveyorOosAvailable : Array[1..100] of Bool;   // Conveyors out of service available
   outConveyorOos : Array[1..100] of Bool;   // Conveyors out of service
END_VAR

VAR_IN_OUT
   inOutGeneralStart : Bool;   // Command general decision point start
   inOutCommandLsb : Bool;   // Command least significant byte
   inOutCommandMsb : Bool;   // Command most significant byte
   inOutInstantStop : Bool;   // Instant stop
```

FB_Line.xml

```
<Section Name="Input">
  <Member Name="inLineConfig" Datatype="&amp;quot;UDT_Line_Config&amp;quot;">
    <Comment>
      <MultiLanguageText Lang="en-GB">Line config</MultiLanguageText>
    </Comment>
  </Member>
  <Member Name="inModelConnected" Datatype="Bool">
    <Comment>
      <MultiLanguageText Lang="en-GB">Model connected to PLC</MultiLanguageText>
    </Comment>
  </Member>
  <Member Name="inGlobalFaultReset" Datatype="Bool" />
  <Member Name="inGlobalStatisticsReset" Datatype="Bool" />
  <Member Name="inSemiautoMode" Datatype="Bool" />
</Section>
<Section Name="Output">
```

OPC FOUNDATION

<AutomationML/>

2006

PRESENT

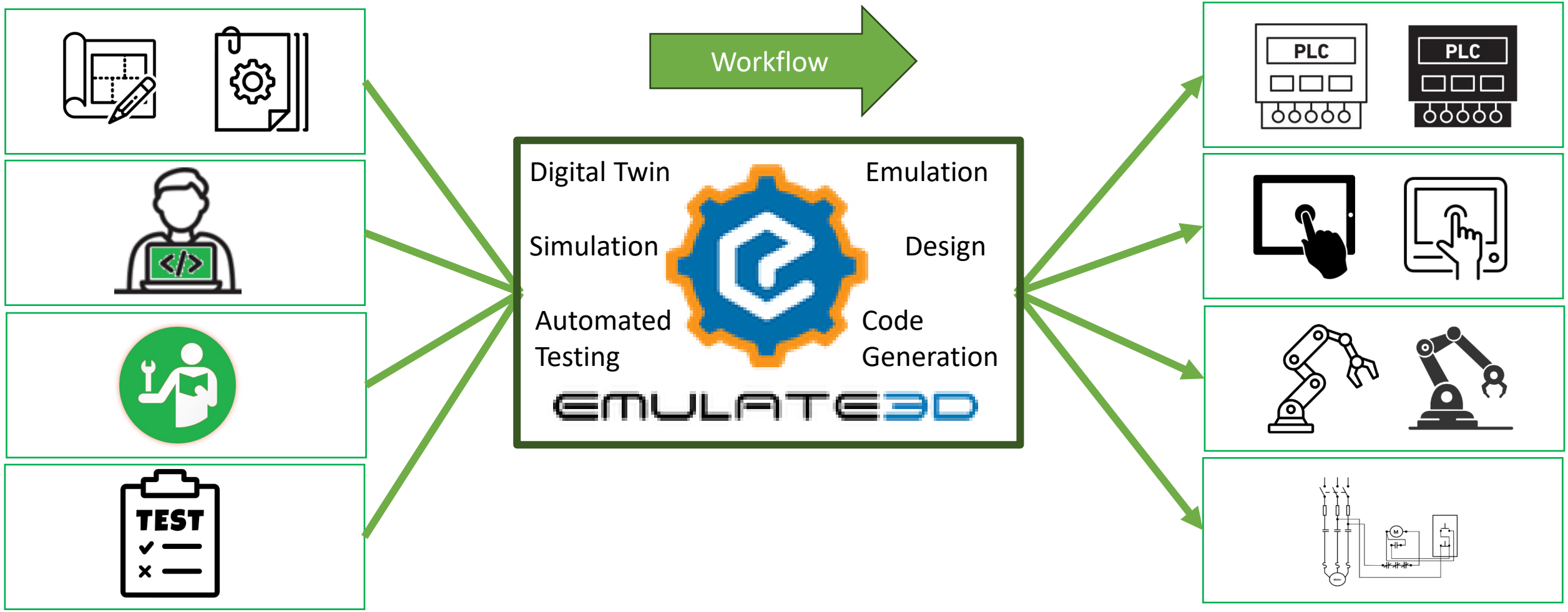
Our vision

- **100% of the relevant automation information is present in E3D**
- **Develop and provide Design methodology**
- **Generate using model information**

2023

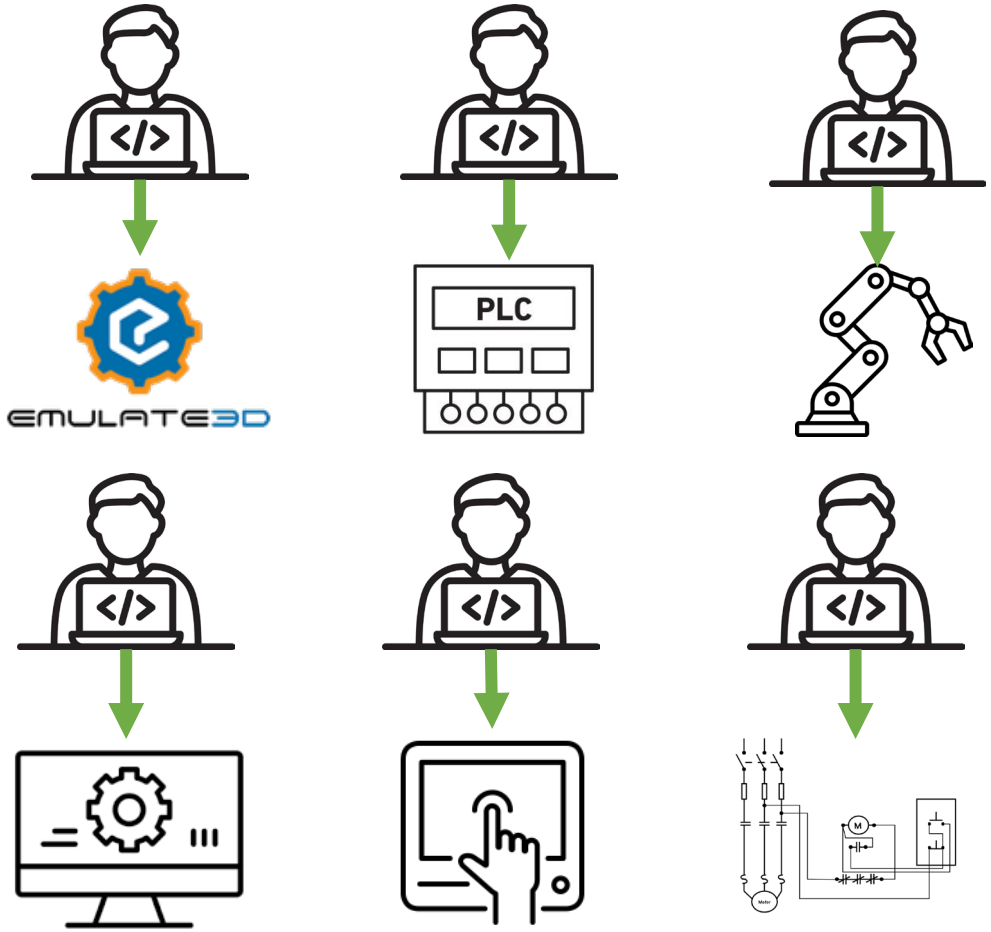
PRESENT

Our vision: What if?

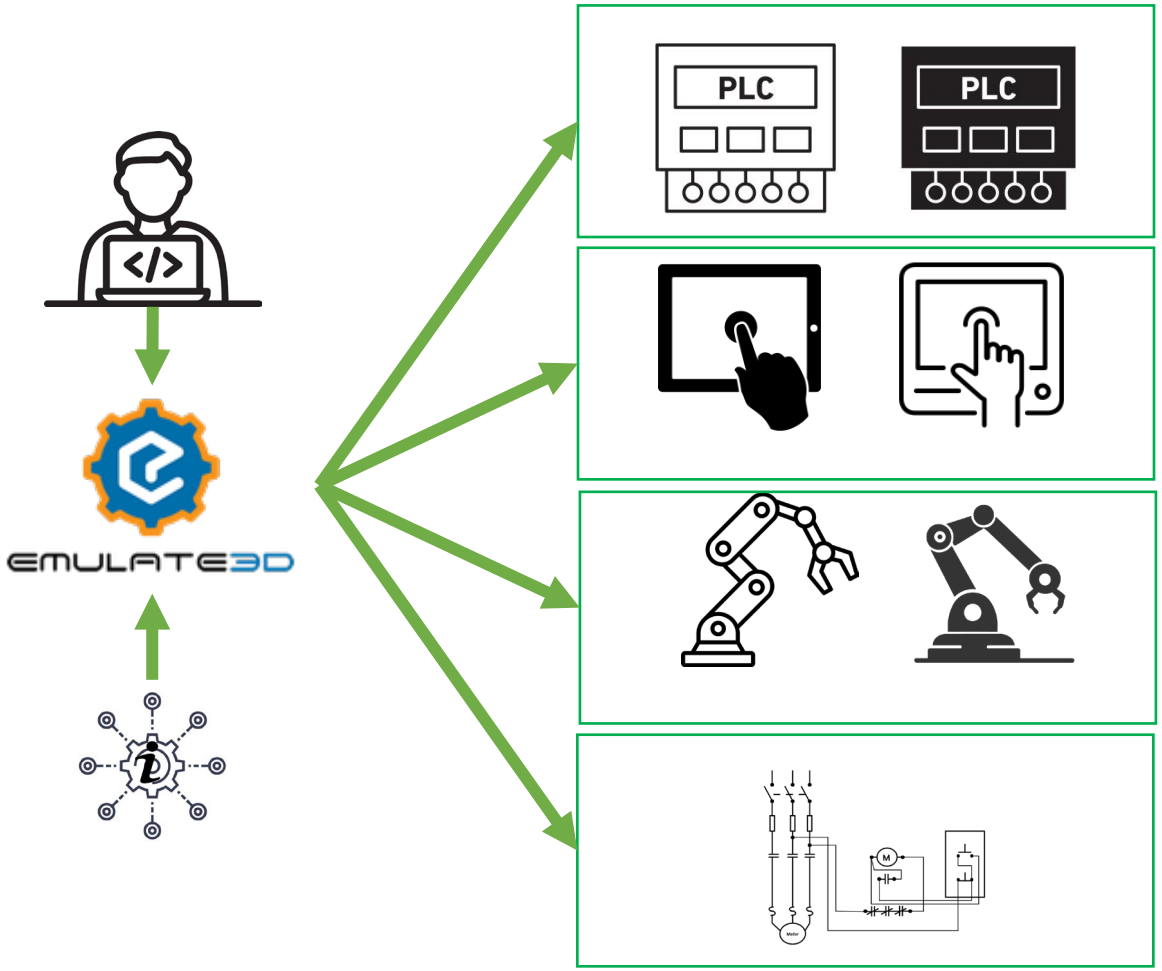


Our vision: What if?

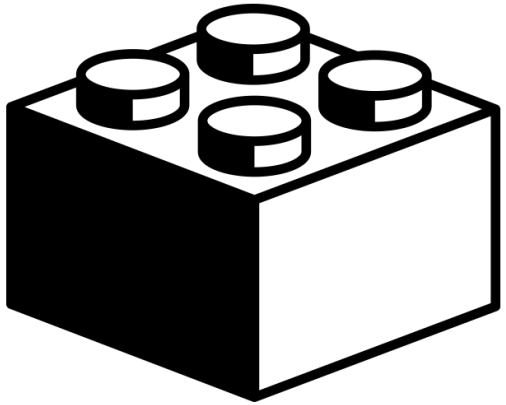
Before

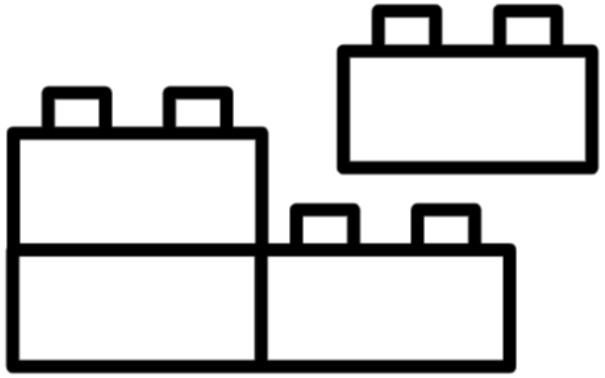


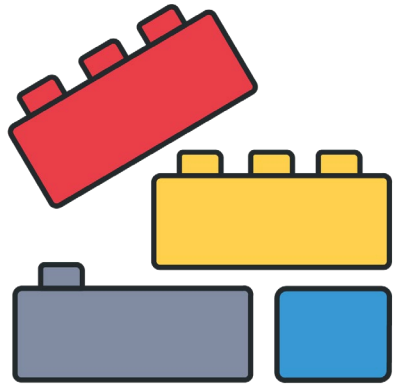
After

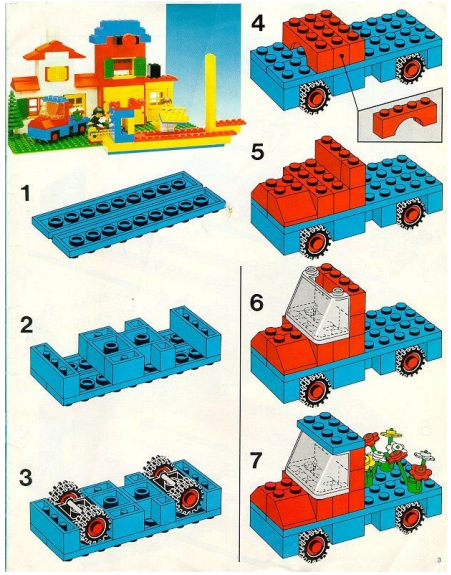


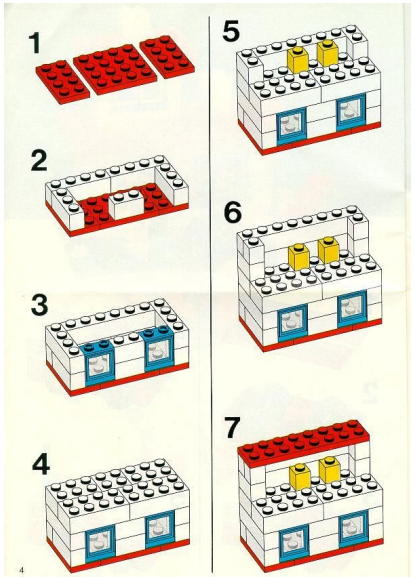
Our vision: But how?

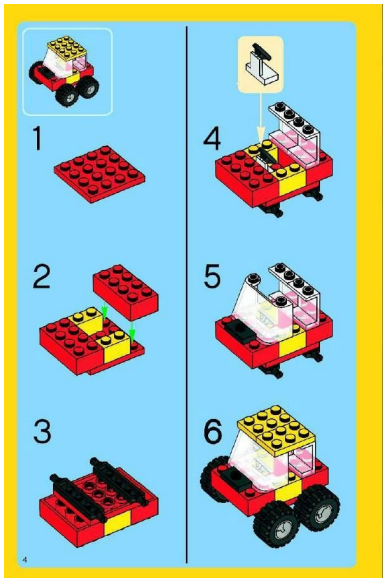






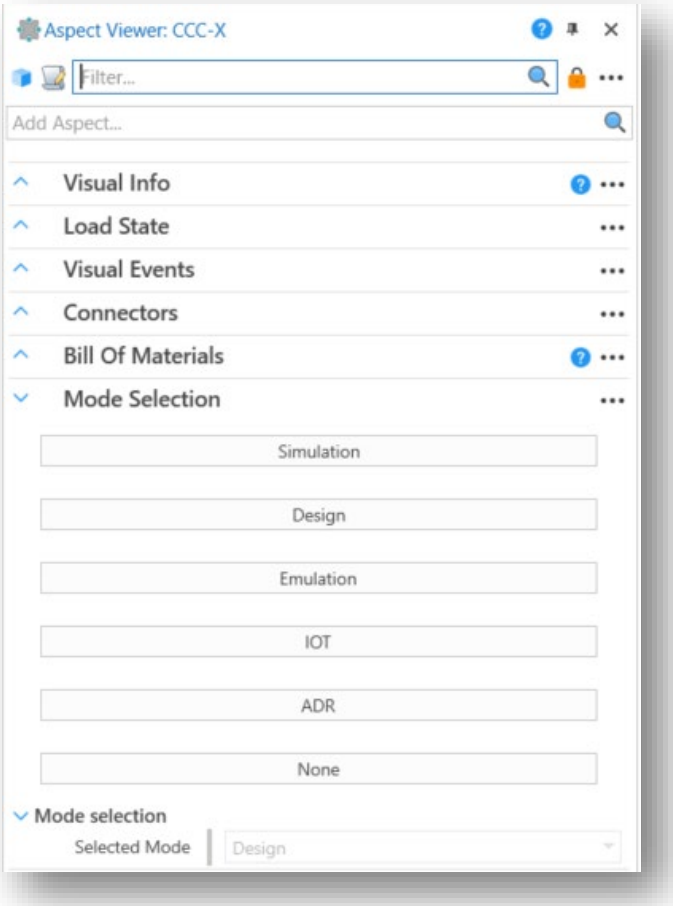




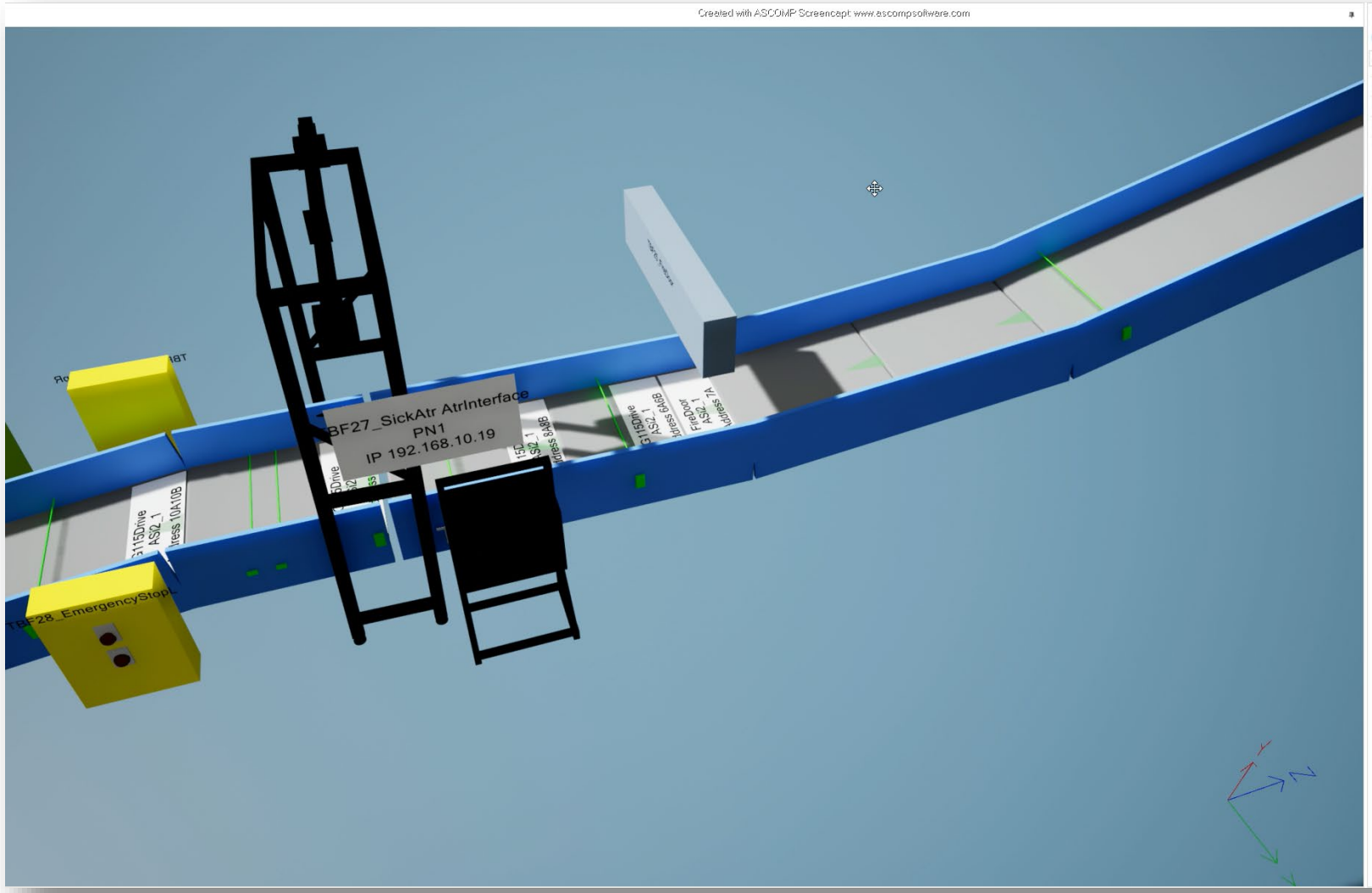


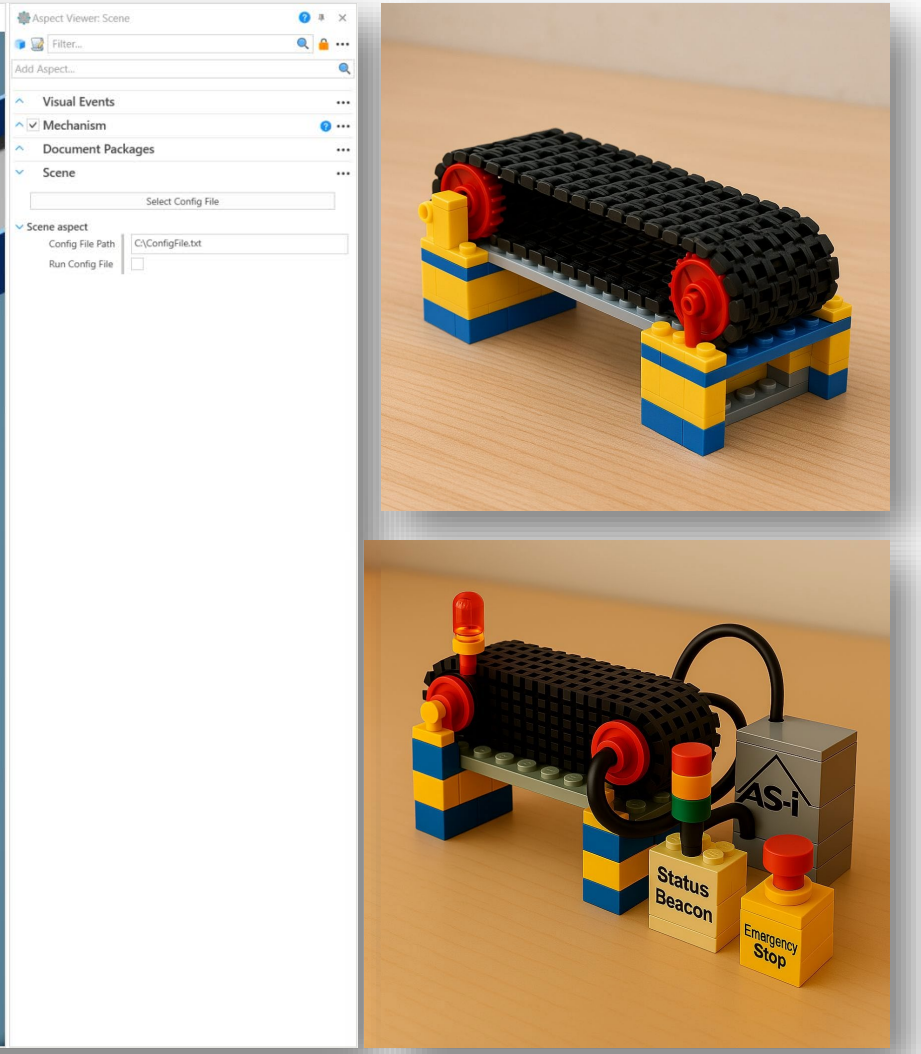
Design tools

- Provide Design methodology
- Facilitate design process
- Minimize design errors
- Improve communication
- Provide visual representation of Design outcome
- Emulation model as by-product of Design
- Test as by-product of Design

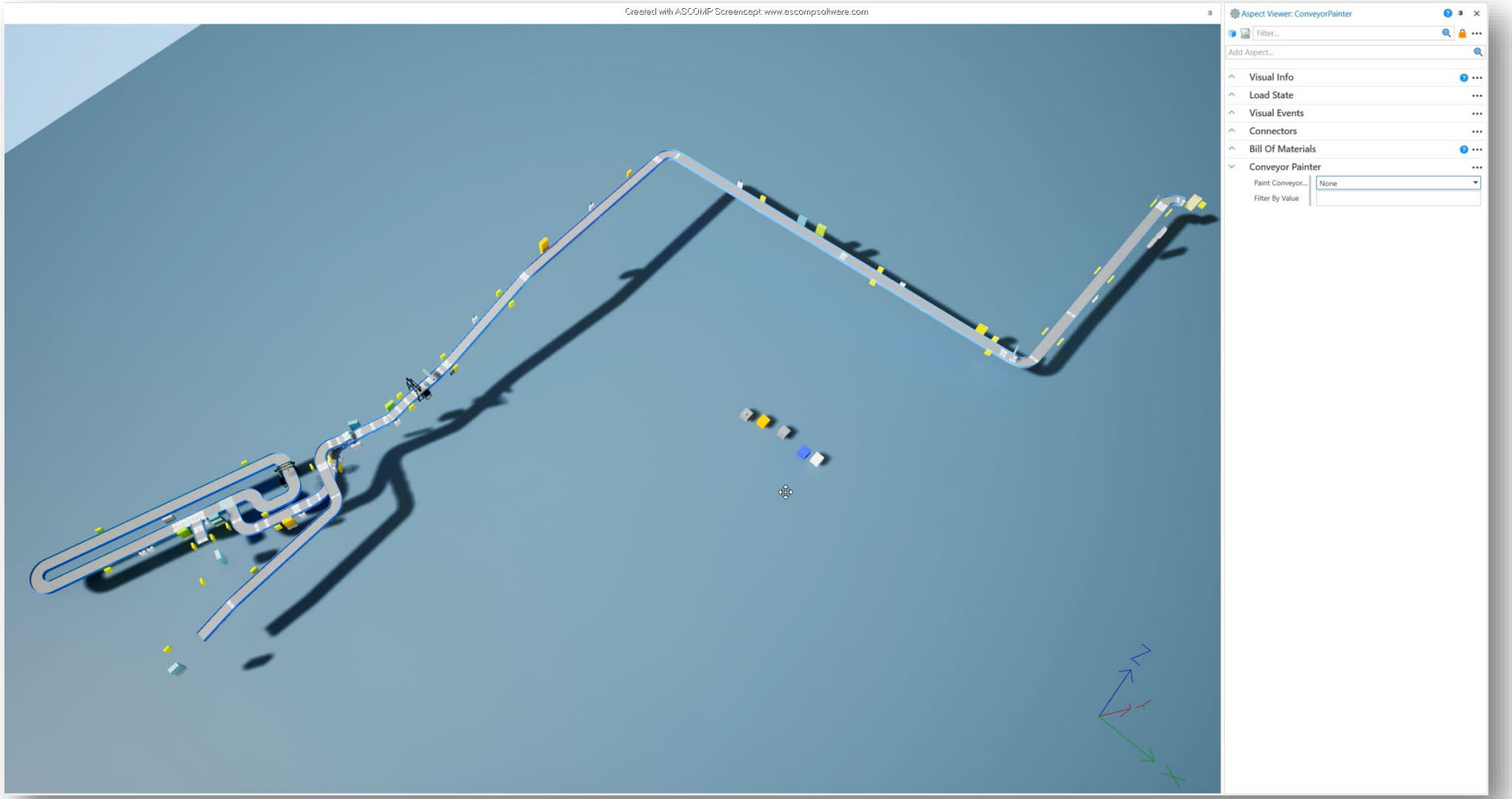


Design tools: In action

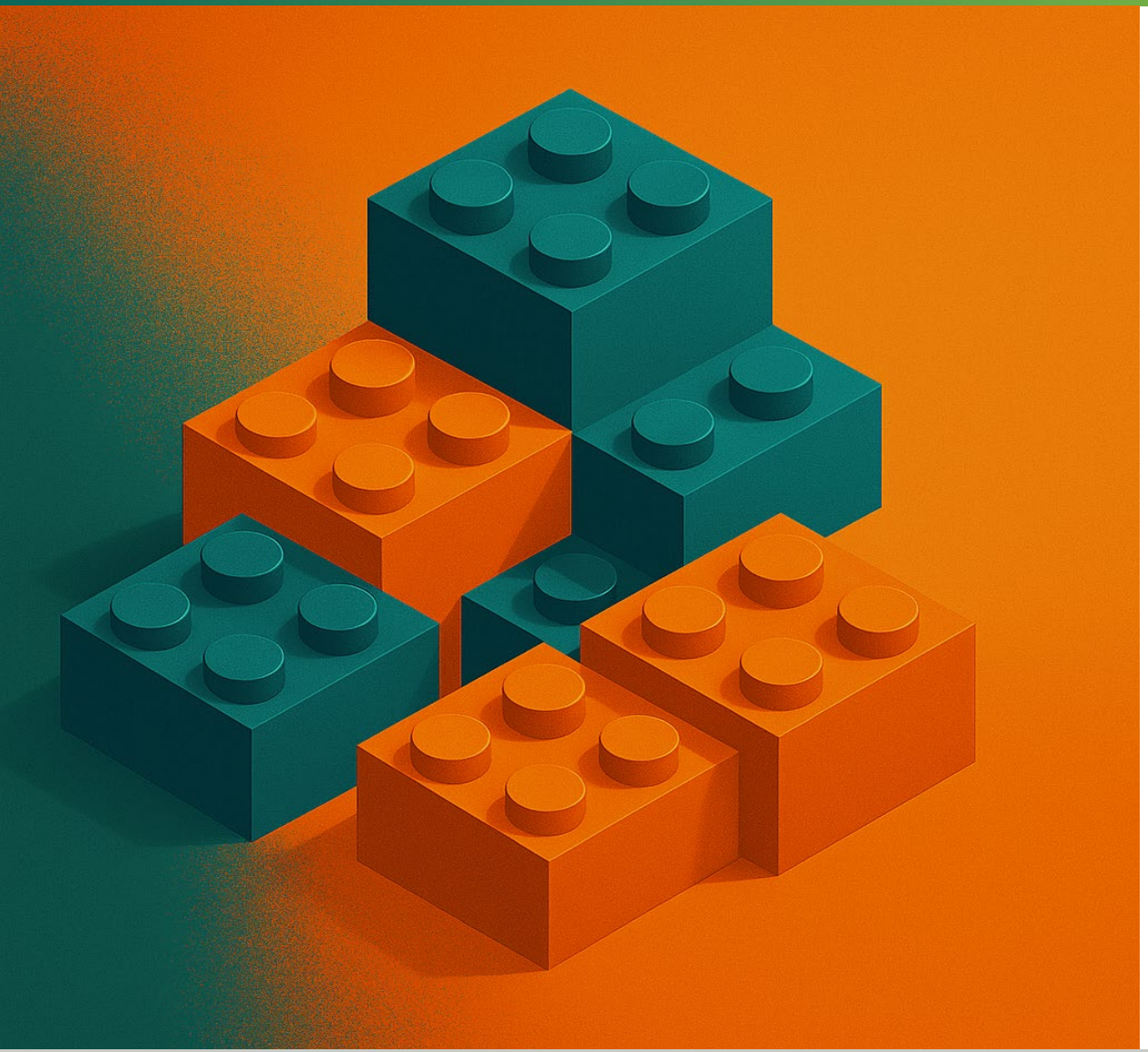


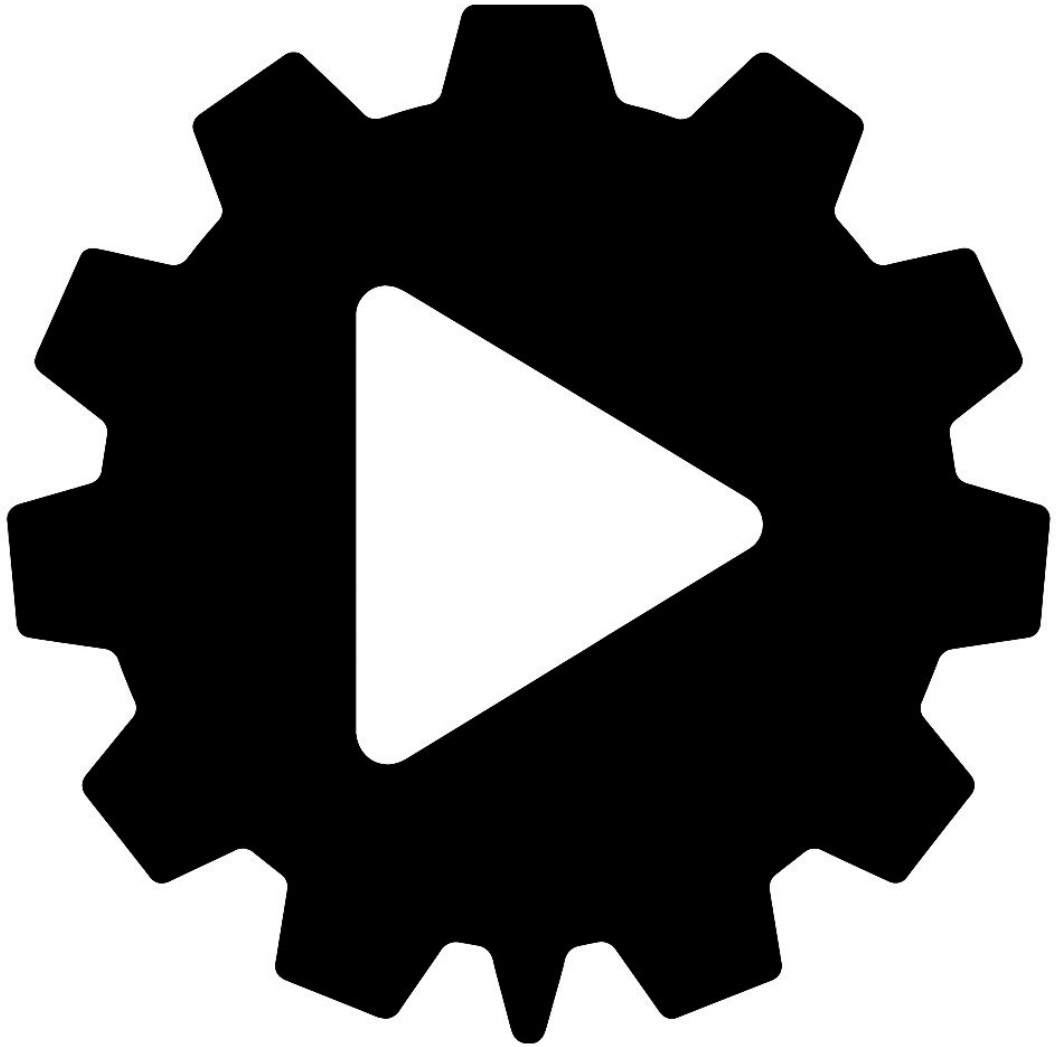


Design tools: In action



Code generator





Code generator: Objectives

Streamline PLC Code Creation

- Develop a tool for automated PLC code generation from Demo3D models, enhancing efficiency and accuracy.

Reduce Manual Effort & Errors

- Automate the generation of PLC projects to minimize manual input and reduce human error.

Enable Unattended Operation

- Design the tool to run autonomously, allowing for seamless integration into existing workflows (I.E. AzureDevOps).

Code generator: Requirements

Ensure High-Quality & Compliant Code

- Guarantee adherence to the client's most stringent industry standards, ensuring robust and reliable code

Flexibility

- The tool should be capable of extracting any necessary information from the model and generating its corresponding PLC code

Expandability

- The system should allow for easy creation and integration of new elements to be generated

Compatibility with manual operations

- The system must maintain manual modifications between iterations

Open solution

- Applicable to all sectors and platforms capable of importing/exporting data in xml format

Code generator: Our approach

Establish Reusable & Standardized Building Blocks

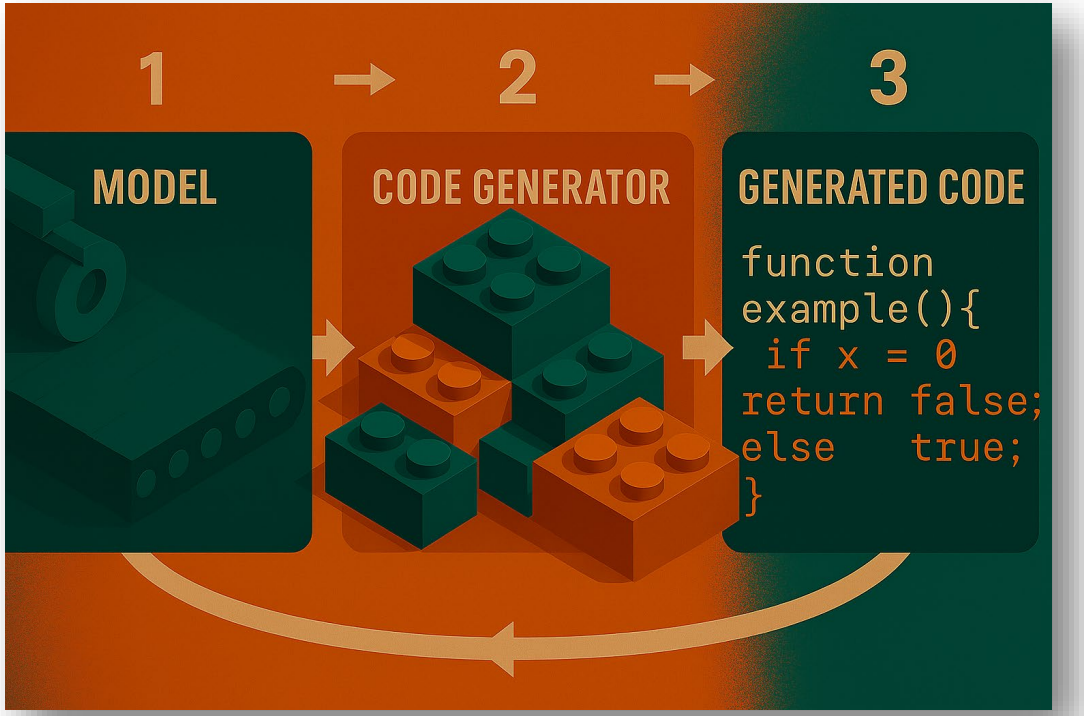
- Define clear standards and create a library of reusable PLC code templates to ensure consistency and accelerate development

Model creation

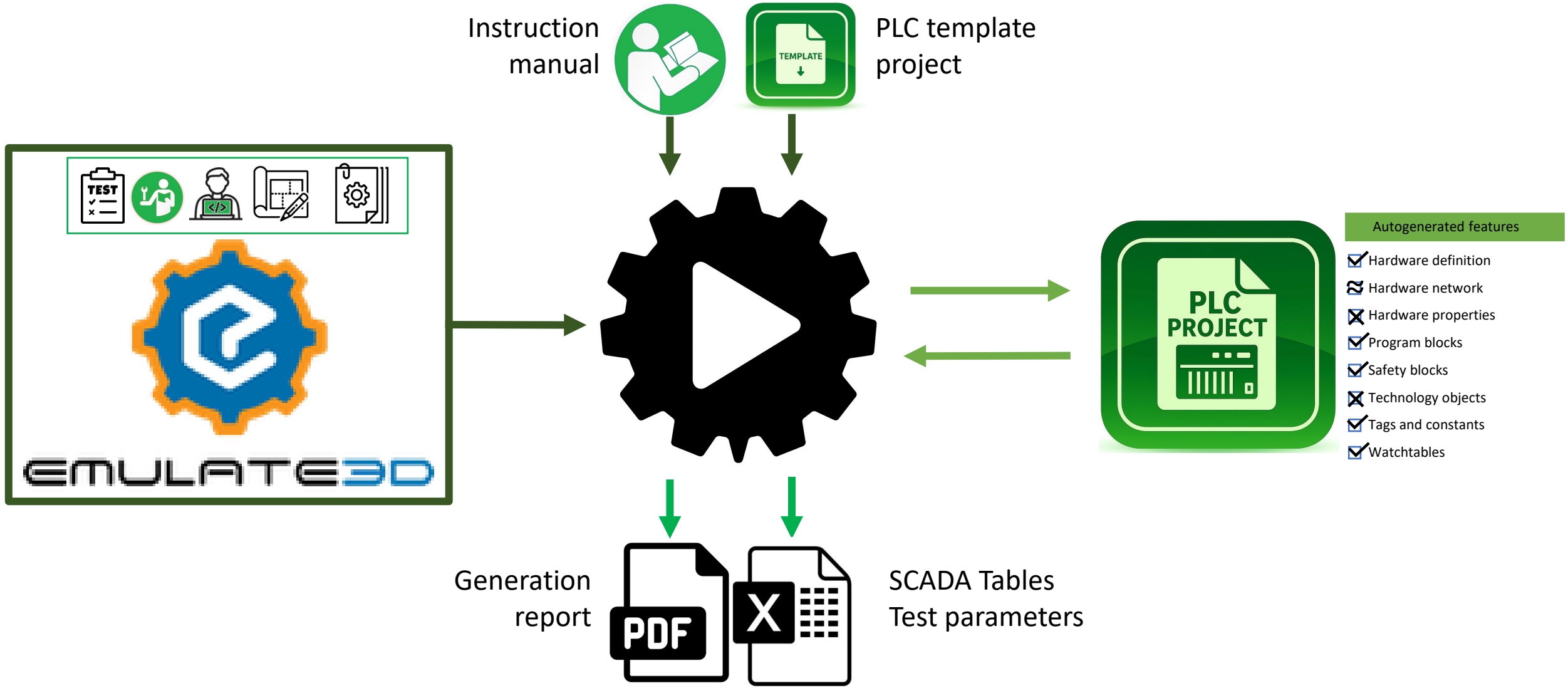
- Utilize design tools that can translate elements within the Demo3D model into these predefined building blocks

Facilitate Continuous Improvement & Refinement

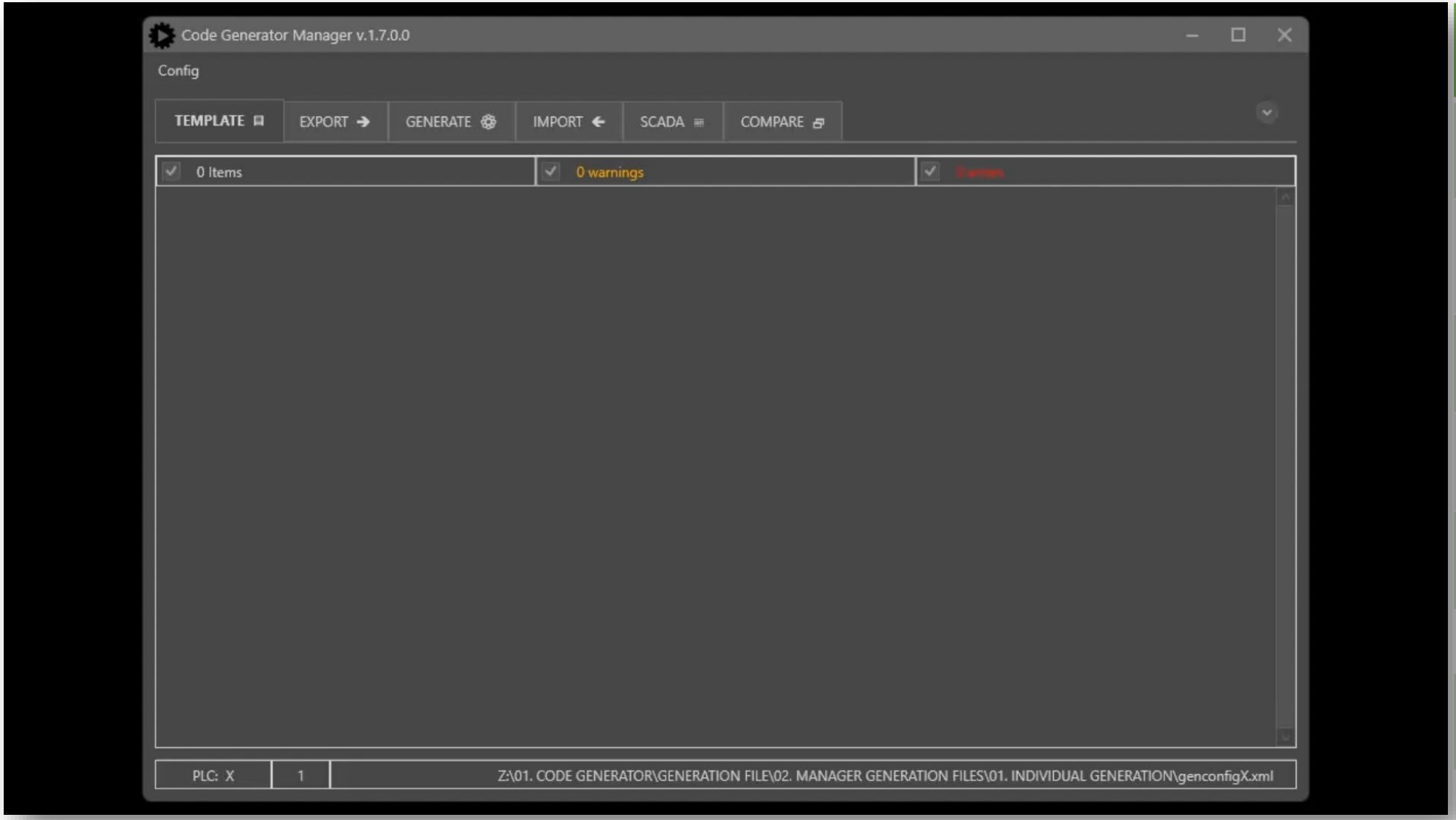
- Enable PLC programmers to iteratively adjust and optimize generated code and models, ensuring the final project meets all requirements



Code generator



Code generator: In action



Action	Duration
Configuration (*)	≈30 seconds
Update template (*)	≈40 seconds
Export (*)	≈40 seconds
Generate	≈3 minutes
Import	≈2 minutes
Total	≤ 7 min

Achievements

Automatic code generation

- PLC tags 100%
- Standard code > 98%
- Safety code > 95%
- HW configuration > 90%
- Watch tables
- SCADA interface documentation 100%

Code Quality

- Amount of loose code reduced to < 2%

FAT testing

- Completed for 11/12 PLCs (1 on hold due to layout design changes)

Impact in engineering workflow

Before

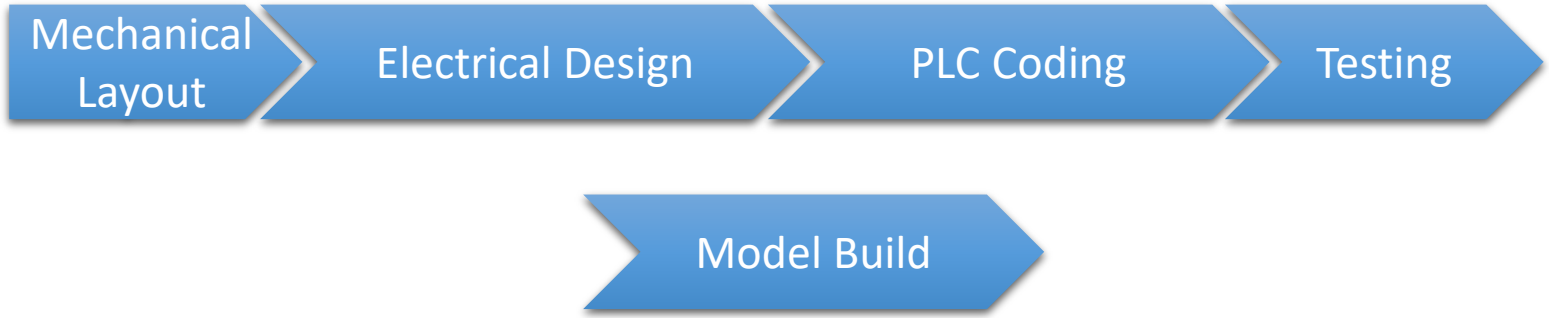
- Emulation model build starts when Design completes
- Model is built from Design documentation
- Alignment between PLC tags and IO browsers required
- Emulation testing starts when both model and PLC Code are built
- High impact of copy/paste errors in testing
- Rollout of code fixes is manual
- Version control of individual PLC instances

After

- Emulation model build finishes when Design completes
- Model produces Design documentation
- PLC tags generated automatically from model, no Excel IO browsers, no alignment required
- Emulation testing starts when Design completes and first version of PLC is generated
- Low impact of copy/paste errors in testing
- Rollout of code fixes is automatic
- Version control of PLC template Project and model

Impact in engineering workflow

Before



After

Impact in code quality

Before

- Inconsistencies across PLC instances
- Excesive use of loose code

After

- Consistent code across PLCs
  - Tag and object naming
  - Comments
  - Network names
  - Block calls
- Higher code encapsulation
- Minimize copy/paste errors
- Simplify version control

THANK YOU!