Model-Driven Engineering of Industrial Process Control Applications

Tomaž Lukman[1], Giovanni Godena[1], Jeff Gray[2], Stanko Strmčnik[1]

[1] Jožef Stefan Institute, Department of Systems and Control, Ljubljana, Slovenia
[2] University of Alabama, Department of Computer Science, Tuscaloosa, AL, USA

IEEE ETFA' 10
13.09.2010, Bilbao, Spain

Outline

- ✓ Introduction (industrial process control applications)
- ✓ Engineering challenges
- ✓ Model-Driven Engineering (MDE)

- ✓ A MDE approach for process control applications
- ✓ Case study
- ✓ Discussion

- ✓ Future work
- ✓ Conclusion

Introduction

- ✓ **Industrial process control systems** – control a specific industrial process
  - Used in several industrial sectors -> improve production, optimize process, and reduce time and costs
- ✓ Research context: engineering of such systems, emphasis on **software**
- ✓ The most common hardware platform for these systems are **Programmable Logic Controllers (PLC)**
- ✓ The programming languages for PLCs are defined by the IEC 6113-3 standard:
  - Instruction list (IL), Structured text (ST), Ladder diagram (LD), Function block diagram (FBD), Sequential function chart (SFC)
  - None are object-oriented

Engineering Challenges

- ✓ Current **challenges** of process control applications engineering are:
  - i. **Lack of automation** – the use of informal approaches or of general-purpose approaches (e.g., UML without profiles) does not facilitate automatic transformation
  - ii. **The migration challenge** – closed proprietary development environments hinder migration
  - iii. **The use of inadequate abstractions** – low-level solution-specific instead problem-specific abstractions
  - iv. **The lack of verification and validation** – exclusive reliance on testing (especially on-site)
  - v. **Developer specifics** – programming skills (IEC languages), no object-oriented and no modeling knowledge

Model-Driven Engineering (MDE)

- ✓ **Model-Driven Engineering** (MDE)
  - promotes the systematic and disciplined use of models throughout the lifecycle, shifts the attention from code to models
  - has the potential to address the identified challenges
- ✓ MDE relies on:
  - **Modeling Languages** – Domain-Specific Modeling Languages (DSMLs) use concepts (e.g., symbols) of a specific domain and formalize the application structure, behavior, and requirements
  - **Model transformations** – specify how target artifacts are generated
  - **Specialized tools** – enable modeling with a DSML and execution of model transformations

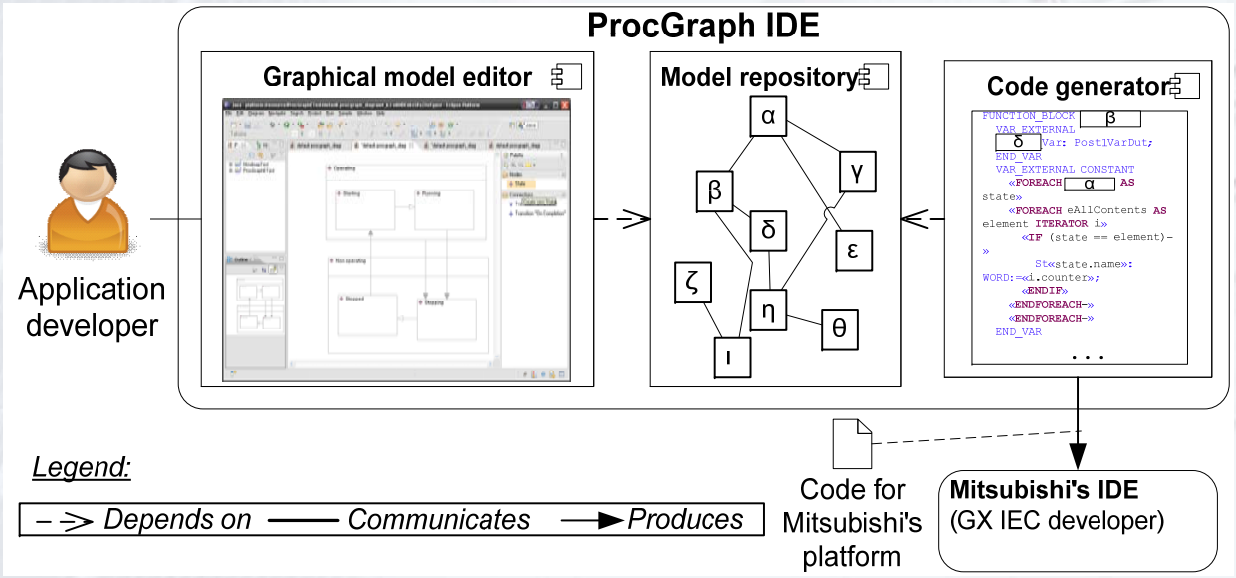
A MDE approach for process control applications

- ✓ A few MDE attempts for the engineering of process control applications already exist
  - They are not widely adopted
    - » immature and
    - » do not properly address the identified challenges
- ✓ The answer: we developed our own MDE approach, which contains of two levels:
  - **Infrastructure development level** – **domain experts** (i.e., expert application developers) develop and evolve **DSML(s)** and **tool developers** develop the **tool infrastructure** that enables the MDE approach at the application development level
  - **Application development level** - **application developers** develop the **process control applications** based on the application requirements and with the use of the provided tool infrastructure

Infrastructure development level

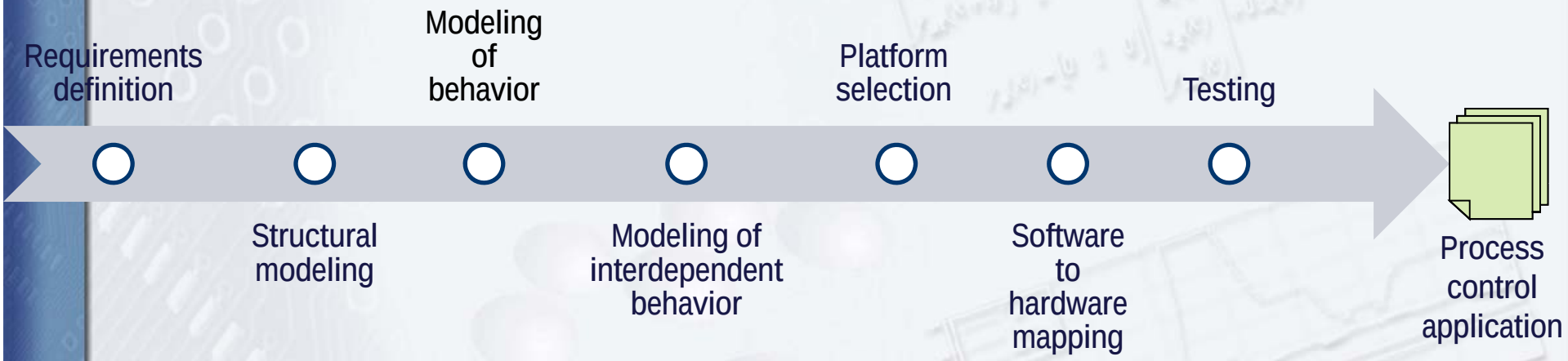
✓ The infrastructure that enables our MDE approach:

- **ProcGraph language** – this already existing semi-formal DSML had to be formalized to be useful for MDE -> through metamodeling
- **Model repository** – the EMF (Eclipse Modeling Framework) tool generated the repository from the ProcGraph metamodel
- **Graphical model editor** – in GMF (Graphical Modeling Framework) a notation, tooling, mapping and editor model were defined to generate a basic editor. It was extended by custom code.
- **Code generator** – into a combination of FBD and ST for Mitsubishi PLCs. Encoded into code generation templates for the openArchitectureWare tool. The Mitsubishi import format had to be decoded.



The MDE process
(Application development level)

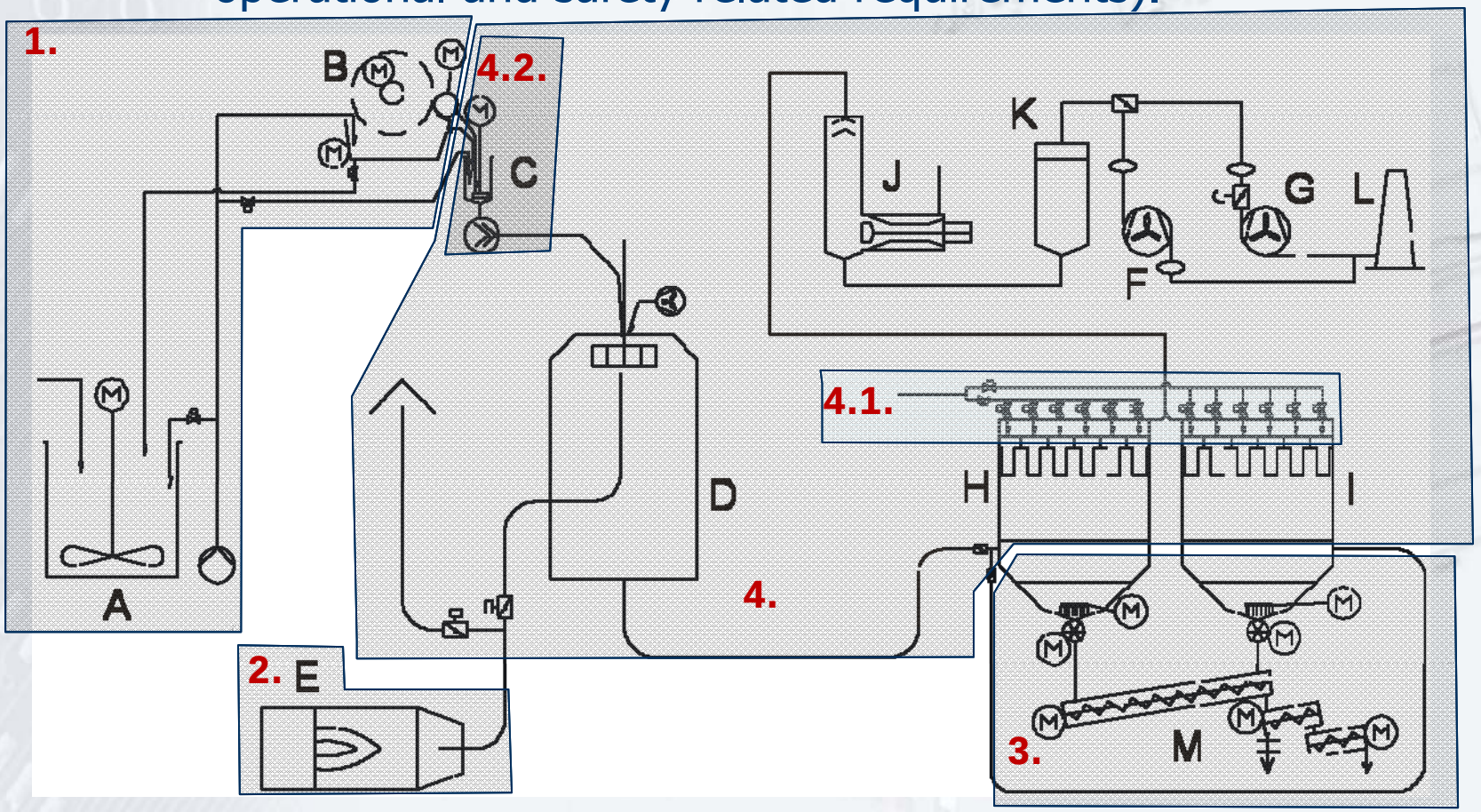
- ✓ The development process activities:



- ✓ These development activities are presented through a case study:
  - On a TiO_2 (titanium dioxide) pigment production subprocess

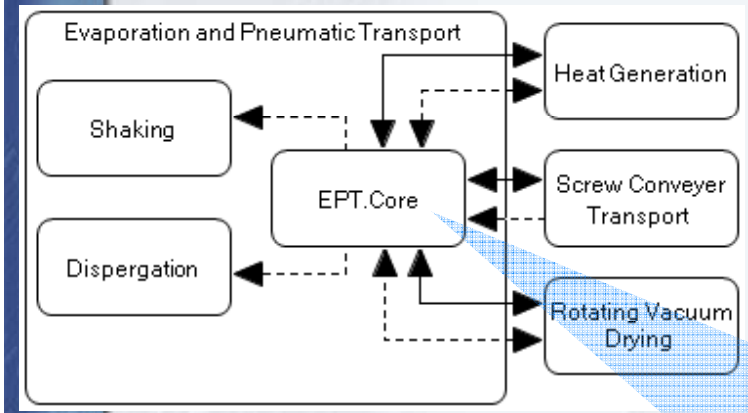
Case study (1/4)

- ✓ Requirements definition
  - Defined through a P&ID (Piping and Instrumentation Diagram) and supporting documents (e.g., informal operational and safety related requirements).

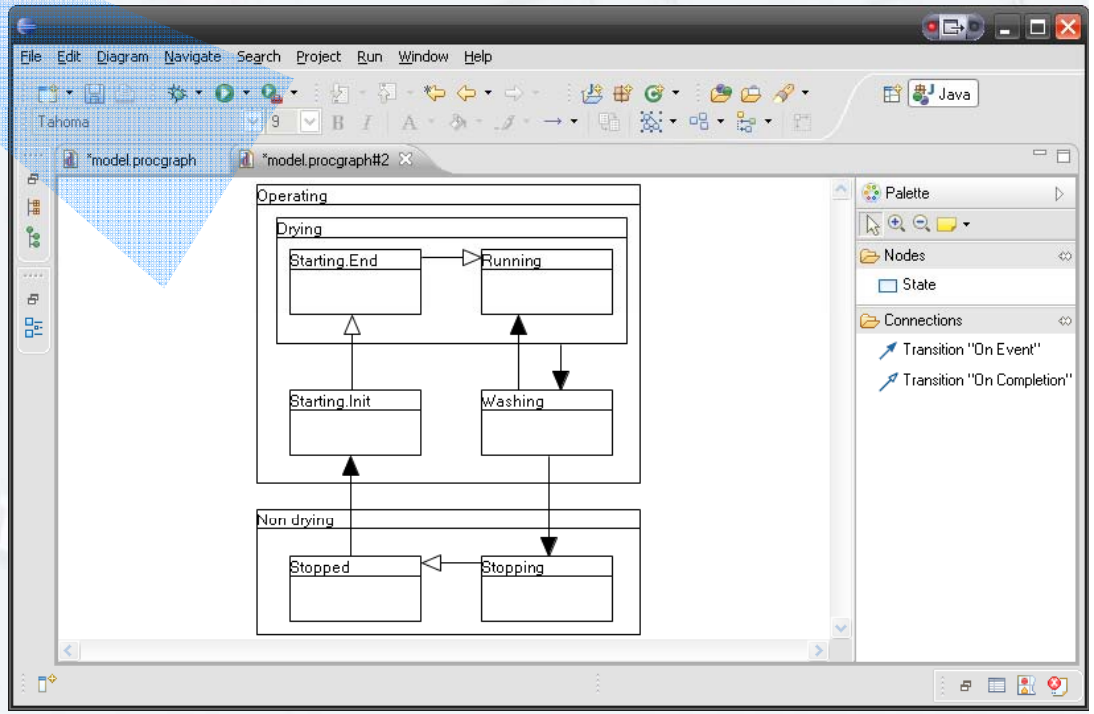


Case study (2/4)

✓ Structural modeling



- Identify Procedural Control Entities (PCEs) – main abstractions of ProcGraph (e.g., a process, an operation or an activity)
- Identified by a system analyst based on high cohesion and low coupling criteria
- Show on an Entity dependencies diagram

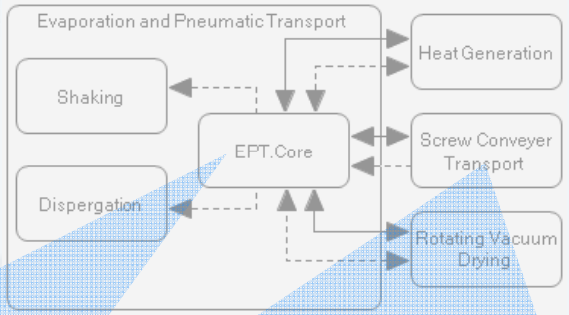


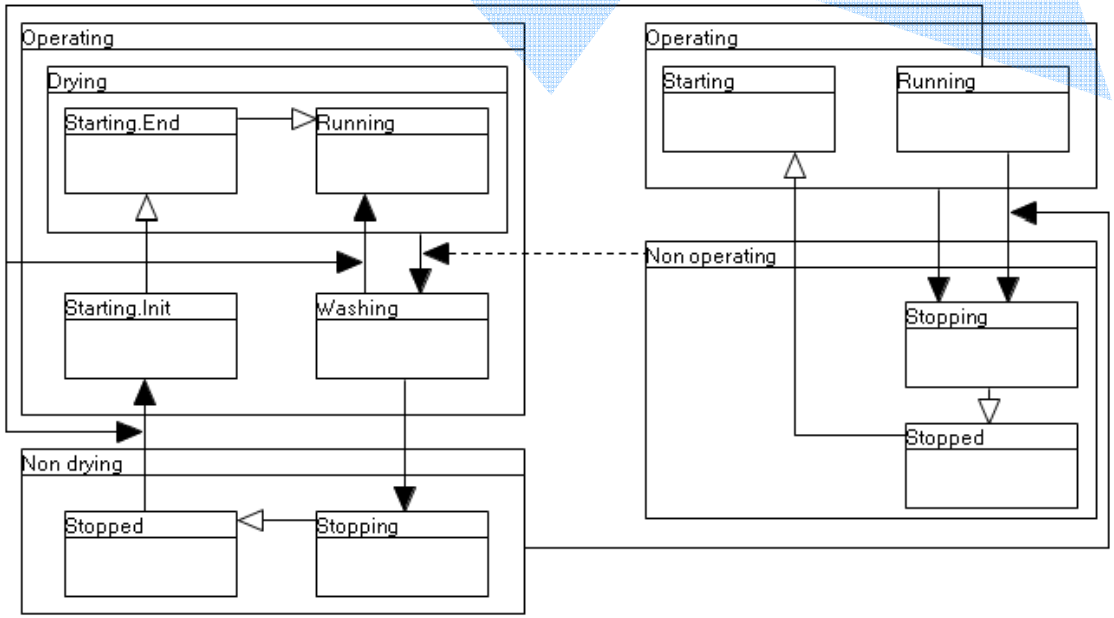
✓ Modeling of behavior

- Extended state transition diagrams define the behavior of each PCE

Case study (3/4)

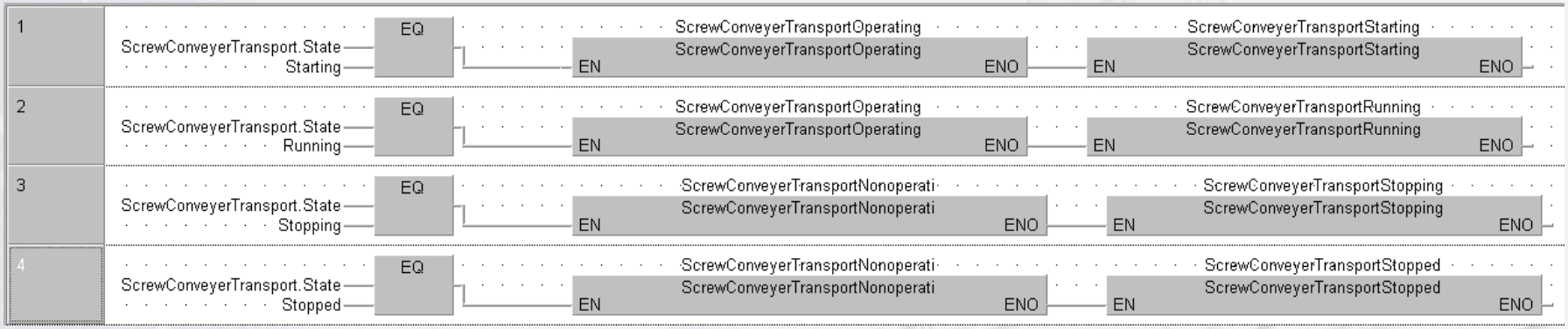
- ✓ Modeling of interdependent behavior
  - A dependencies state transition diagram defines the mutual behavior dependencies between two PCEs
  - Two dependence relationships exist:
    - » conditional dependency, which is denoted by a normal line with a filled arrowhead
    - » propagation dependency, which is denoted by a dashed line with filled arrowhead





Case study (4/4)

- ✓ Platform selection
  - A platform is determined through the selected code generator
  - We developed a generator for Mitsubishi PLCs



- ✓ Mapping of software onto the hardware
  - No visual modeling – adjusted in the code generator or in the development environment of the PLC vendor
- ✓ Testing
  - The code has to be imported, compiled and uploaded on the PLC

Discussion

- ✓ The presented MDE approach brings these **benefits**:
  - **Increased software quality** – automatic code generation, without human coding errors, ProcGraph enables a better system decomposition
  - **Increased productivity** – automatic code generation *(-> challenge "i.")* and inherent reuse of domain knowledge through ProcGraph
  - **Platform independence and platform migration** – migration is achievable through the development of a new code generator *(-> challenge "ii.")*
  - **Improved communication and interaction between development participants** – less misunderstandings between developers, because ProcGraph is defined more formally

Future work

- ✓ Develop tools that support additional development tasks
  - Verification tool – enables a "correct-by-construction" process instead of a "construct-by-correction" *(-> challenge "iv.")*
- ✓ Quantitative evaluation of the benefits of our MDE approach

Conclusion

- ✓ A MDE approach for the engineering of industrial process control applications was introduced
- ✓ The challenges of engineering such applications were identified
- ✓ The developed infrastructure that enables our MDE approach was described
- ✓ The MDE process was presented though a case study
- ✓ The experienced benefits were presented

Contributions:

- ✓ A MDE approach for process control applications that is aligned with the identified challenges
- ✓ The developed infrastructure that enables this approach (i.e., supporting tools)