

A Model-driven Tool Environment for Automation and Control Application Development – Transformation Assisted, Extendable Approach

Timo Vepsäläinen, David Hästbacka, Seppo Kuikka
Tampere University of Technology
Department of Automation Science and Engineering (ASE)

NW-MODE 2009
28. August 2009



TAMPEREEN TEKNILLINEN YLIOPISTO
Department of Automation Science and Engineering

Presentation includes:

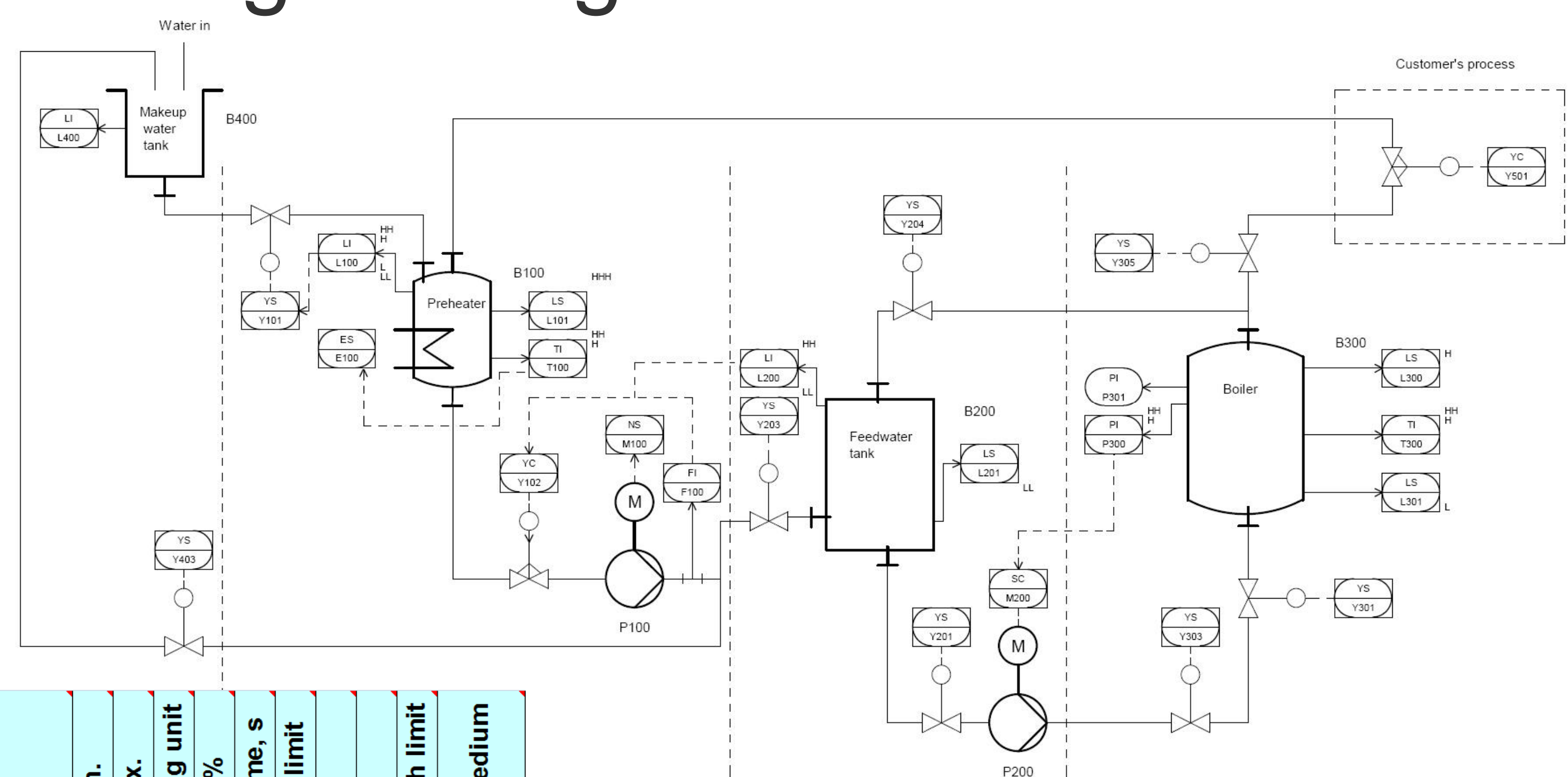
1. Introduction
2. The AUKOTON development process
3. UML AP Tool
4. AUKOTON Transformations
5. Conclusions and discussion



1. Introduction

- Collaboration of professionals with multidisciplinary backgrounds: manufacturing system, process, control, automation, and software engineering

An example of a piping and instrumentation diagram (P&ID).



Process system	Process item type	Process function type	ID	Description	IO type	IO connection type	Instrument type	Range, min.	Range, max.	Engineering unit	Accuracy, %	Filtering time, s	Lower limit	Low limit	High limit	Higher high limit	Process medium
CWS.T1	tank	level	LI100	Cooling water level	AI	4-20 mA	level transmitter	0.0	2.5	m	2	3	1.0	1.2	1.8	2.0	Water
CWS.T1	tank	level	LS100_1	Cooling water level low	BI	24 V	level switch										Water
CWS.T1	tank	level	LS100_2	Cooling water level low	BI	24 V	level switch										Water
CWS.V4	block valve	position	V101-S1	Make-up water valve closed	BI	24 V	limit switch										Water
CWS.V4	block valve	position	V101-S2	Make-up water valve open	BI	24 V	limit switch										Water
CWS.V4	block valve	position	V101-Q1	Make-up water solenoid valve	BO	24 V	solenoid valve										Water

An example of a spreadsheet presentation of instrumentation information.



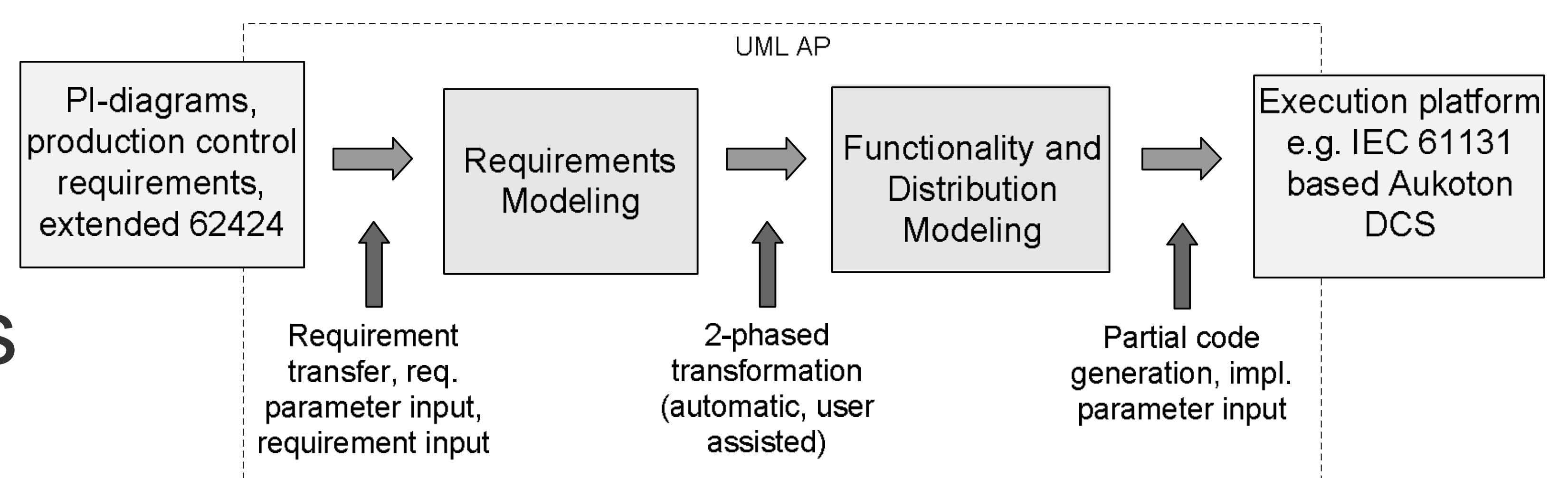
1. Introduction – challenges

- Different engineering disciplines (and companies) do not usually have common, structured presentations of design information.
 - Development practices may vary even within same field of engineering.
 - Different DCS-platforms and collections of type circuits.
- ➔ Design becomes platform (and project) dependent early, reuse of solutions is difficult and the development requires manual, error-prone transfer of design information.



2. The AUKOTON development process

- The intention of the process is to ease some of the challenges with use of
 - Domain-specific modeling concepts of the UML AP
 - Model-driven engineering approach
- Three main phase in the development: Requirements, Functionality and platform specific modeling.
- Automatic (and/or user assisted) transformations between the phases and concepts



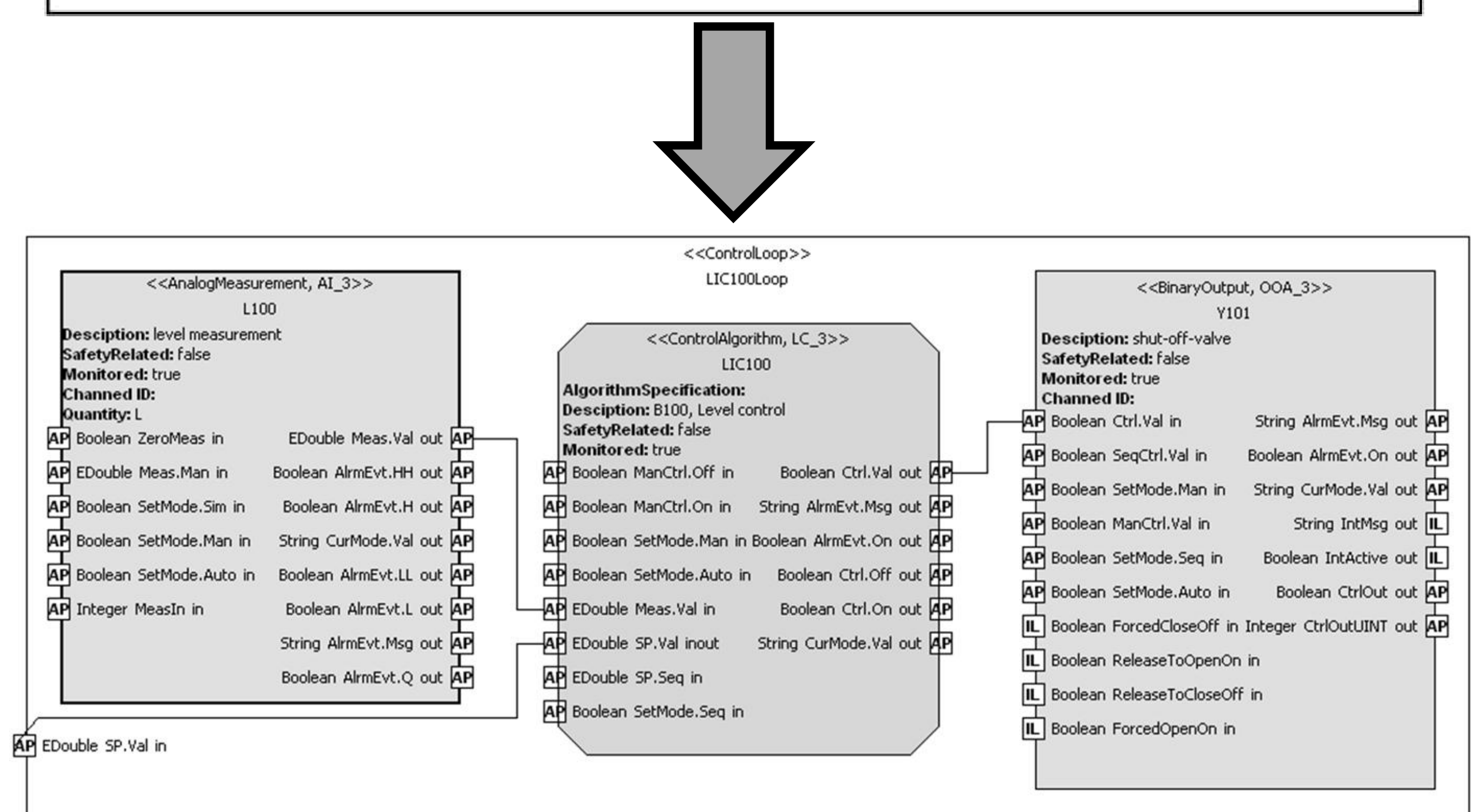
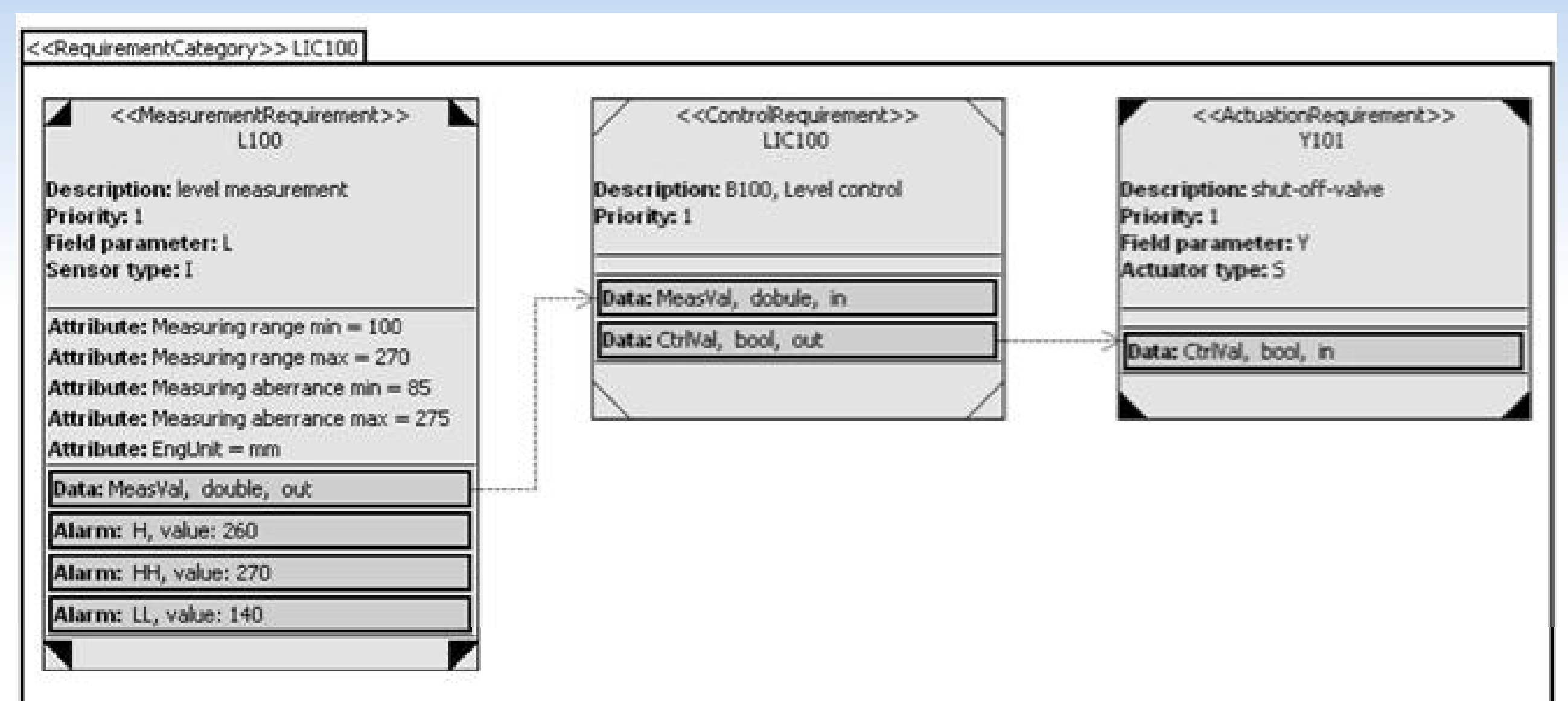
2. The AUKOTON development process

Automation Requirements

- Required instrumentation, controls, interlockings etc.
- From multiple sources (various stakeholders)

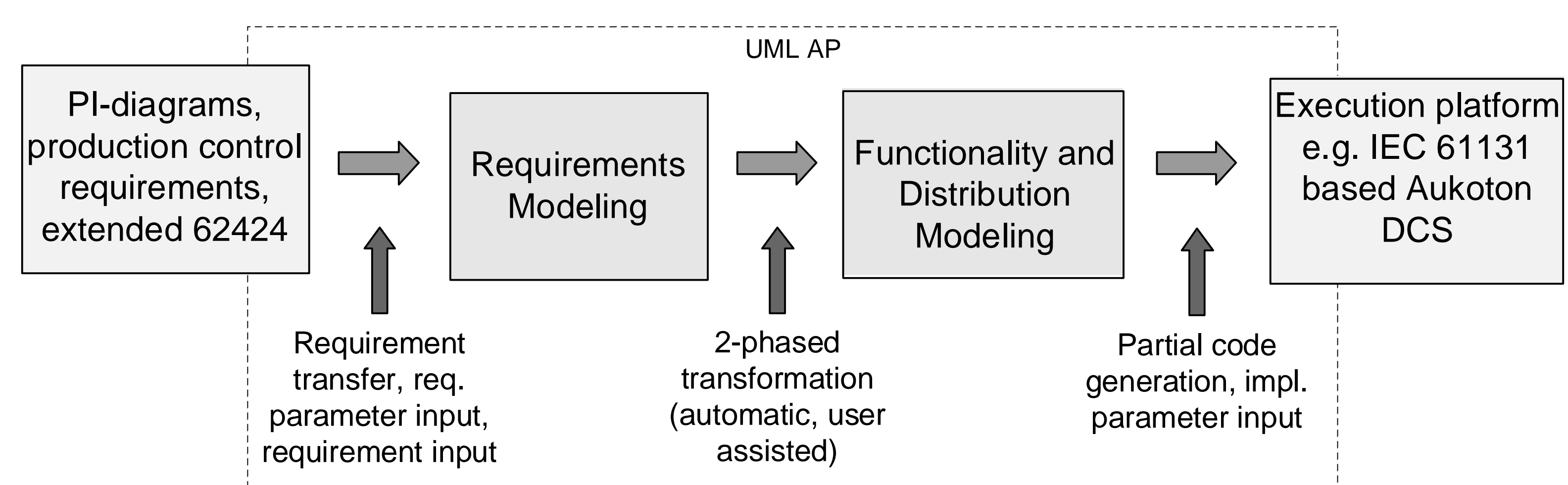
Automation Functions

- platform independent pieces of functionality of applications.
- Can be linked, combined and refined to platform specific constructs.



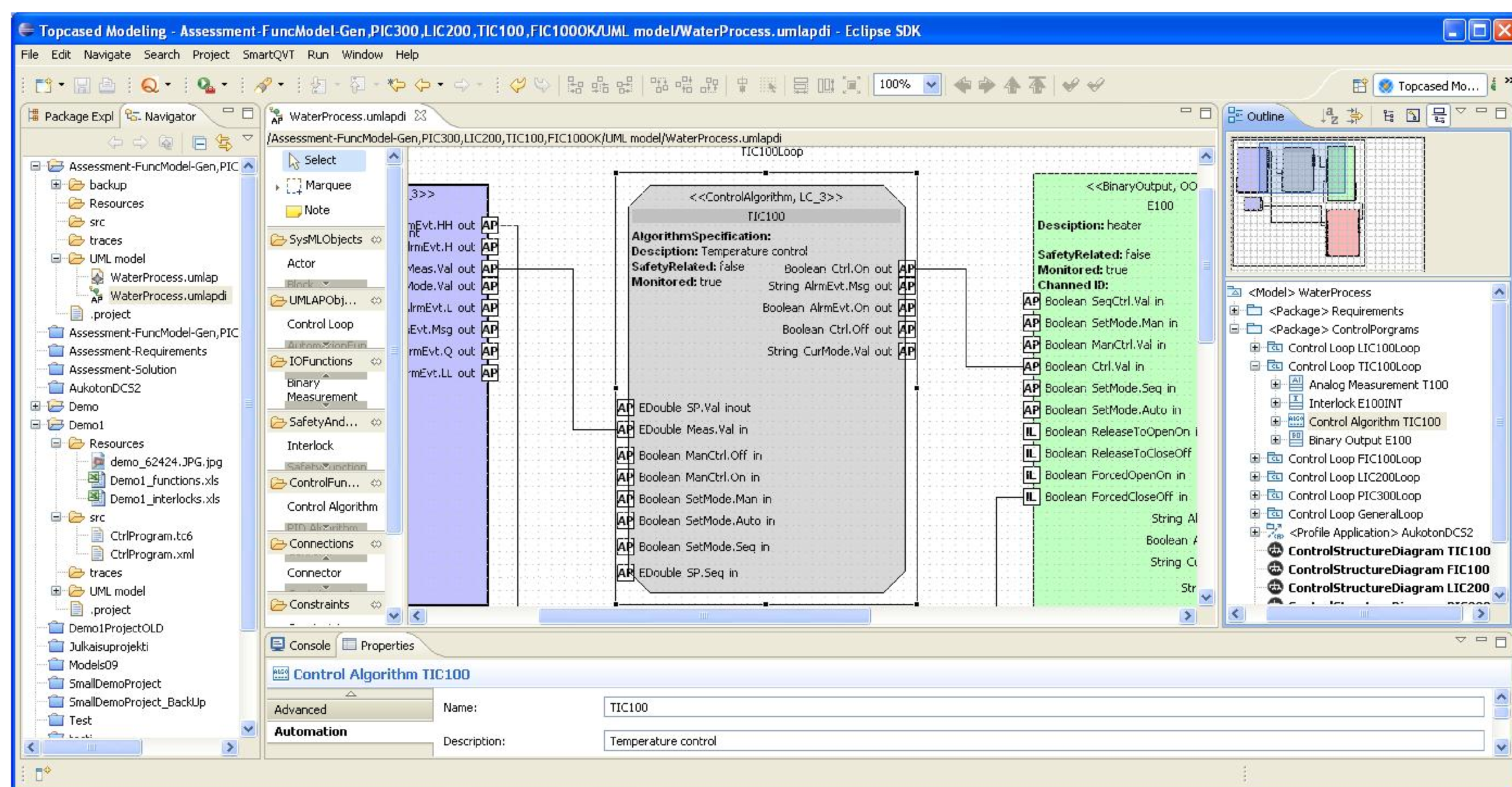
2. The AUKOTON development process

- 3 different kinds of transformations for:
 - Importing source information
 - Intra-model transformations between modeling concepts (Automation Requirements and Functions) and phases
 - Exporting models with code generators
- Need for:
 - flexible integration of transformation tools
 - ability to target transformations to user-selected model packages



3. UML AP Tool

- An Eclipse based development tool for supporting the *modeling concepts* (UML Automation profile) and the *development process*



3. UML AP Tool – extension interface

- 3 extension points for different types of transformations
 - Ability to search and utilize transformation tools contributing their services to the extension interface of the tool
 - A Java interface for each transformation
 - Initiation of transformations with targeting information
 - Transformation descriptions
- Implementation as a separate plug-in
- Transformation actions tied to outline view of the Eclipse platform.

<<interface>> IRequirementImporter
+ getRequirement (rootPackage : Package, traces : IFolder) : Boolean + getDescription () : String

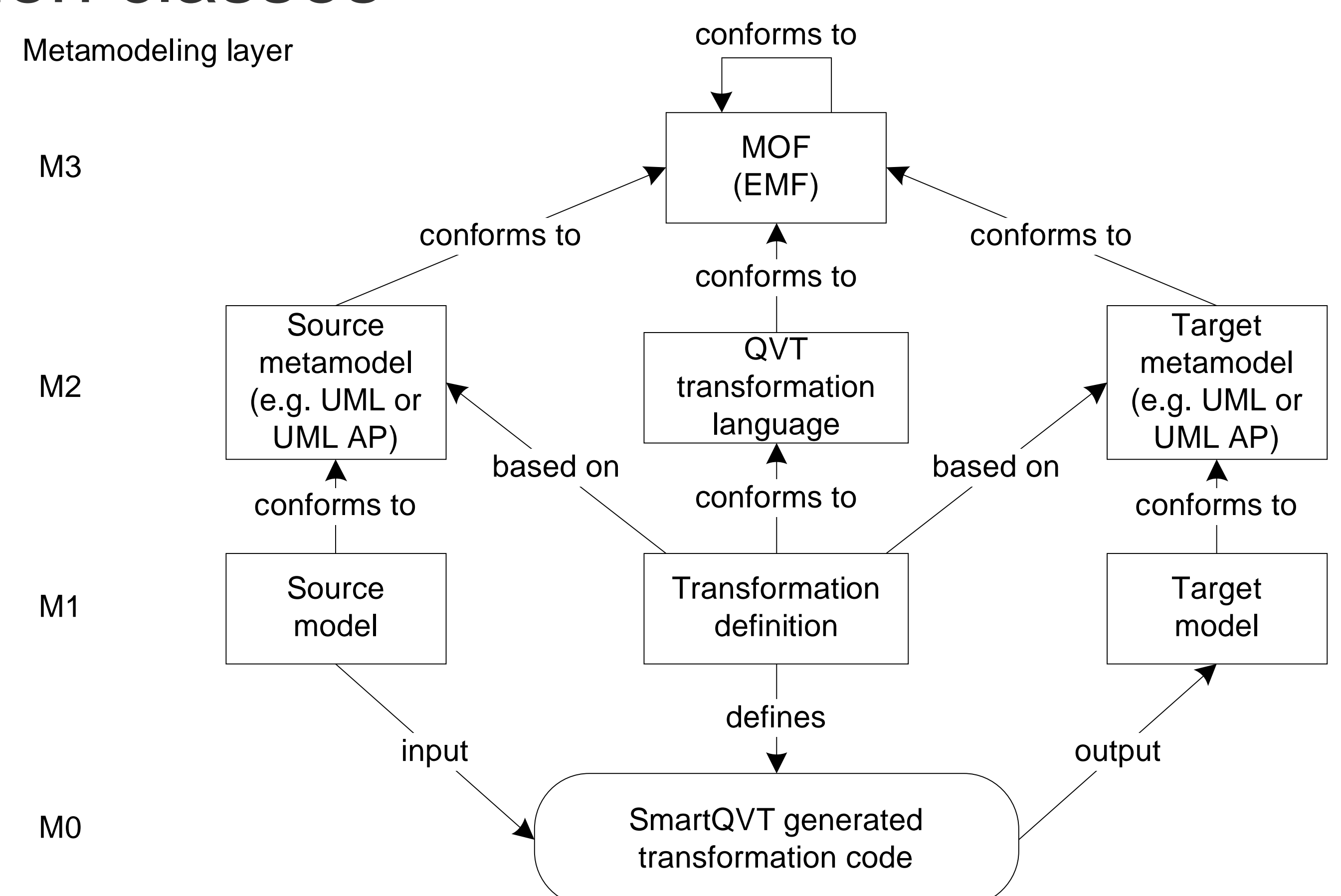
<<interface>> ITransformer
+ transform (sourcePackage : Package, targetPackage : Package, traces : IFolder) : Boolean + getDescription () : String

<<interface>> IModelExporter
+ runGenerator (modelPackage : Package, src : IFolder, traces : IFolder) : Boolean + getDescription () : String



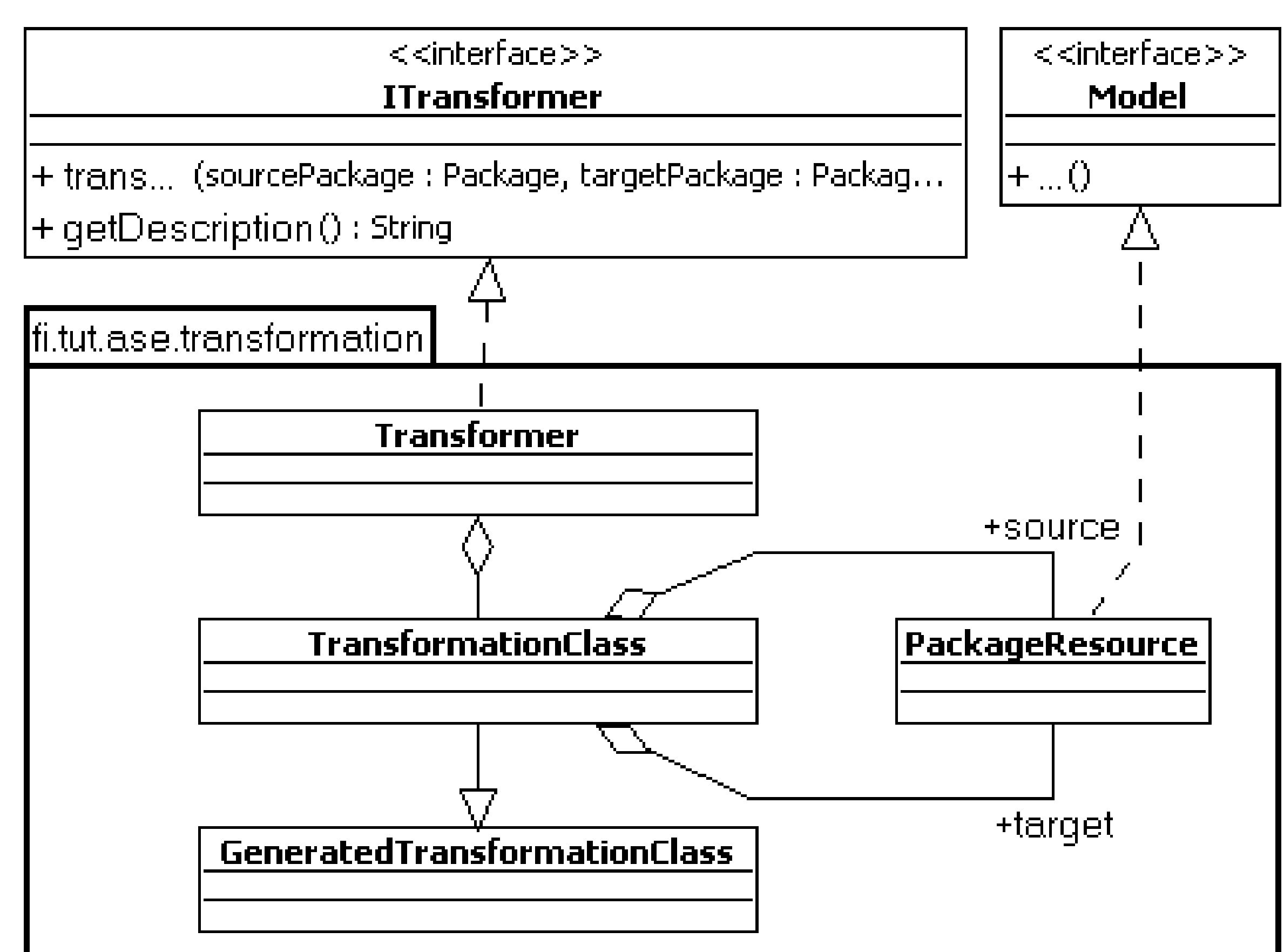
4. AUKOTON Transformations - SmartQVT

- Implementation of QVT (OMG Query/View/Transformation) Operational Mappings language
 - EMF based metamodels (e.g. UML, SysML, UML AP, ...)
 - Generates Java transformation classes
- Issues to consider:
 - Integration to the tool
 - Targeting of transformations by black boxing or *restricting the scope of transformation*
 - Simultaneous access to the models
 - Maintainability



4. AUKOTON Transformations – plug-in structure

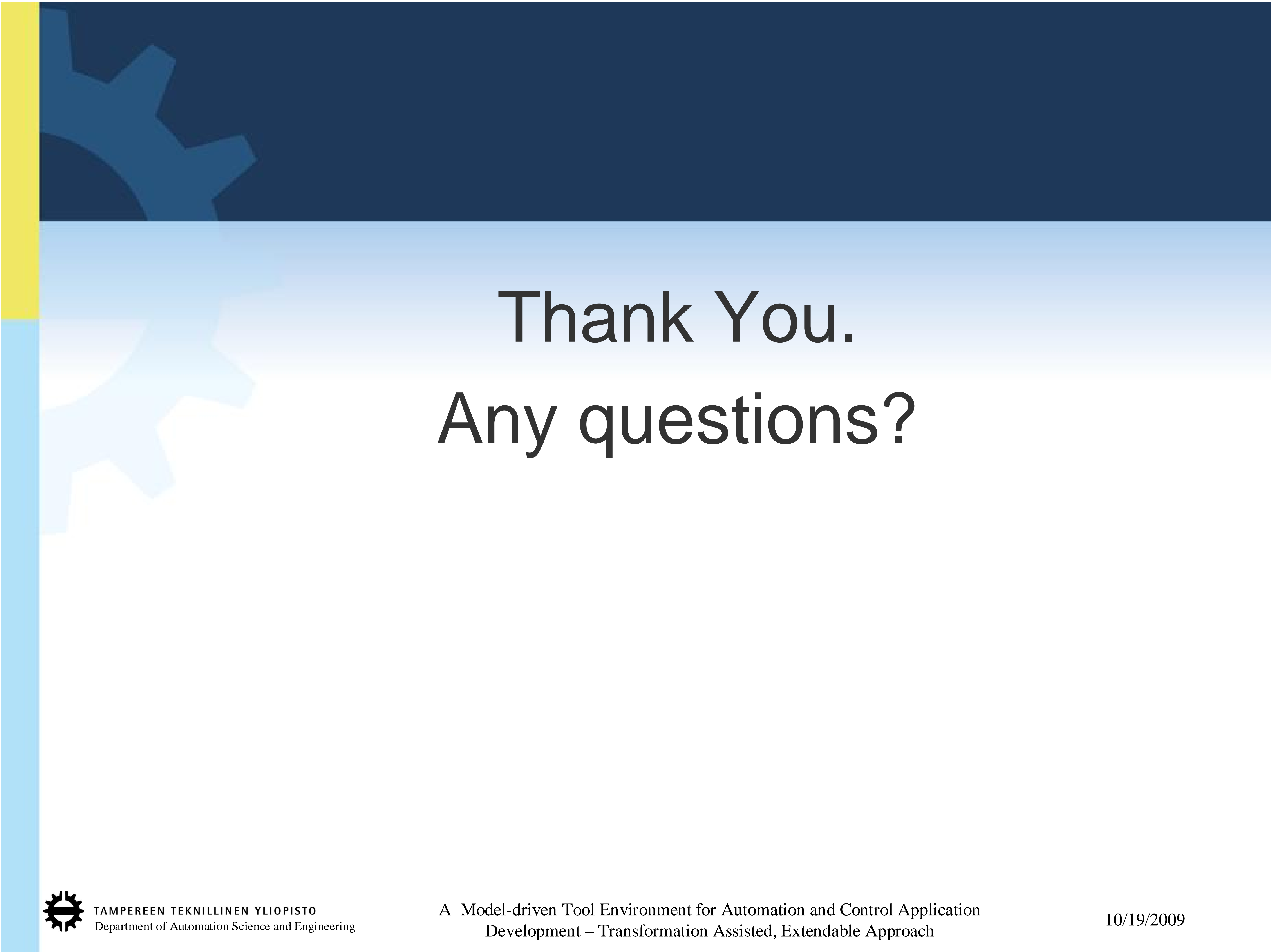
- Transformer: implements the ITransformer interface of the tool, initializes the actual Transformation class.
- TransformationClass: extends the generated transformation class, implements black boxes, initializes package-wide resource classes.
- PackageResource: restricts direct access to only elements contained by user selected packages, implements Model interface required by SmartQVT.
- Benefits: easy to maintain, targeting of transformations, black boxes etc.



5. Conclusions and discussion

- + Several practical transformations utilizing both the extension interface and the plug-in structure.
- + The approach appears to be extendable to some other modeling and transformation tools.
- + With use of EMF, the approach can be extended to handle some XML serialized files as source or target models of transformations.
- The approach only deals with the model elements, not the graphical presentations of models, i.e. amount and contents of graphical diagrams. → Need for auto-layouts.





Thank You.
Any questions?

