ENGINEERING OF SOFTWARE FACTORIES FOR HIGH TECH INDUSTRY – EXPERIENCE AND CHALLENGES

Ivan Kurtev

ECMFA 2019

ALTRAN

ALTRAN GROUP

World leader in R&D and engineering services

- + 30 countries
- + 45 000 people worldwide

Altran Netherlands

- ~ 1000 people
- 6 locations
- ~ 200 clients

ALTRAN NETHERLANDS EXPERTISE

ALTRAN

MODEL DRIVEN ENGINEERING AT ALTRAN NETHERLANDS

10+ years of experience in applying MDE in industry

Organization:

- Applied Modeling Team
- Model Driven Engineering (MDE) Expertise Center
- MDE Trail: training language engineers
- Innovation Group MDE

Some clients and partners:

- ASML, Philips Healthcare, Thermo Fisher Scientific, ESI, …

MAIN MDE THEMES AND PROJECT TYPES

- Model-driven software construction
- Reconstruction of models for analyzing system performance
- Model-driven software modernization
- Component interface modeling
- Monitoring and runtime verification
- Model-based testing
- MDE tools development

Project types:

- Automation of engineering processes
- Typical duration: 1+ years

AGENDA

Altran MDE solution: Software Factories

Managing a software factory: Language Architecture

Implementing a software factory
- Language development aspects
- Model transformations
- Tools

ALTRAN MDE SOLUTION

TAILORED SOFTWARE FACTORIES FOR ENGINEERING AUTOMATION

ALTRAN

THE MDE APPROACH

MDE promises to:

- Increase productivity in software development
- Improve quality of the product

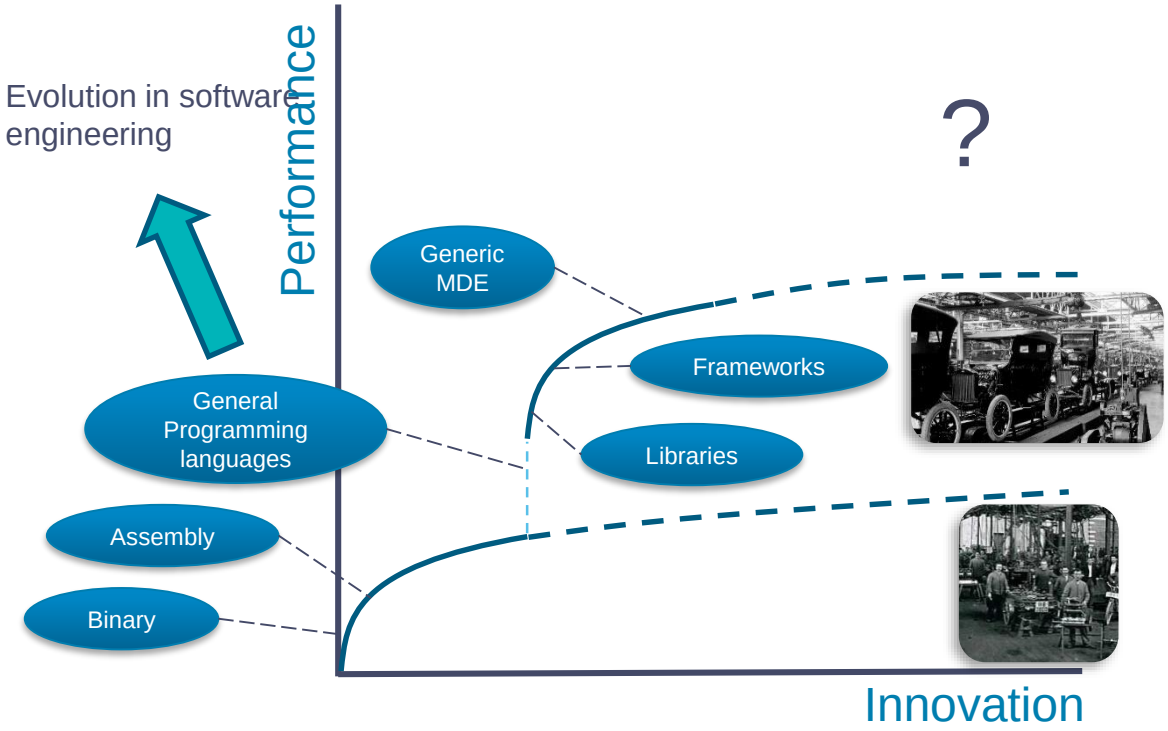
Key ideas:

- Raise the level of abstraction to manage complexity via modeling
- Use of domain-specific modeling and languages

Both the *level of abstraction* and *domain-specificity* admit of degrees

- Spectrum of possibilities in realizing the MDE vision

CHALLENGES IN SOFTWARE ENGINEERING



Productivity Challenges

- Constant change
- Increasing complexity
- Demand for higher quality
- Legacy software
- Hard to find qualified people

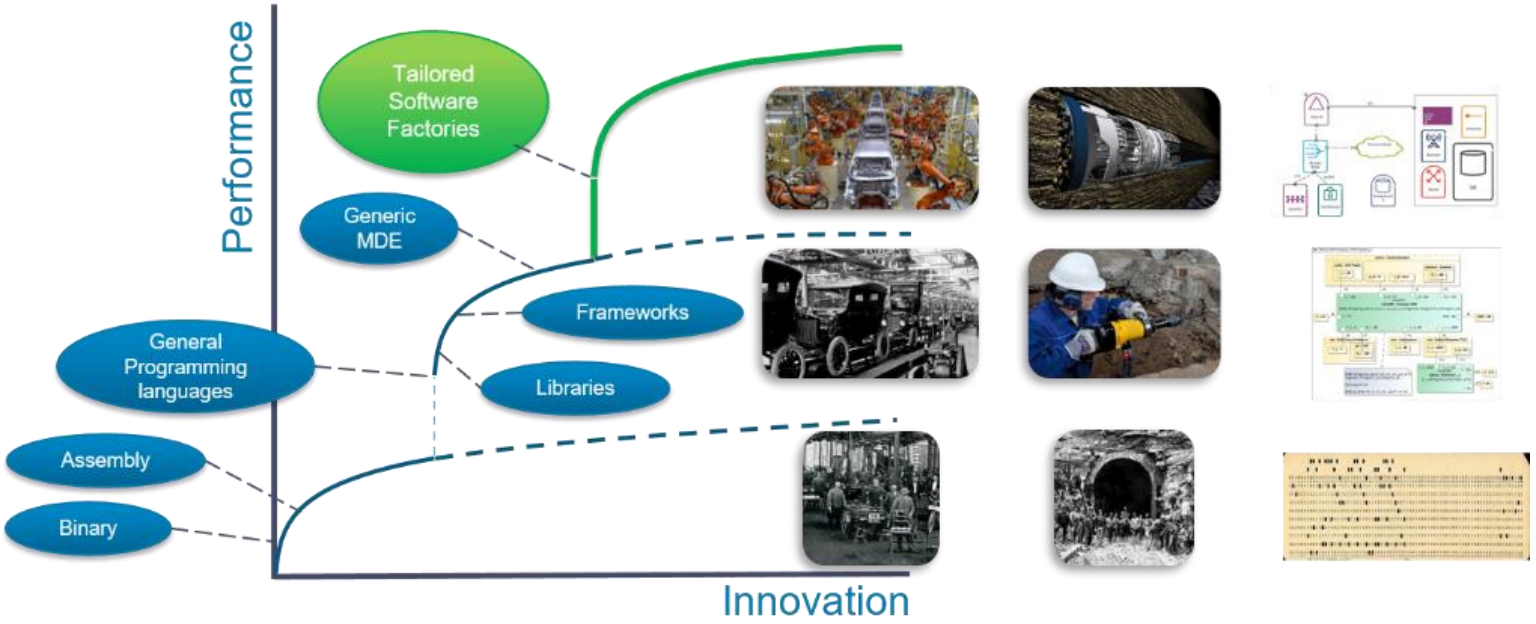
Examples of evolution in some engineering and manufacturing domains

ALTRAN

SOLUTION BASED ON MDE

Tailored Software Factories

Goal-oriented, specialized, optimized, integrated and automated software production



SOFTWARE FACTORY

A software factory automates engineering processes

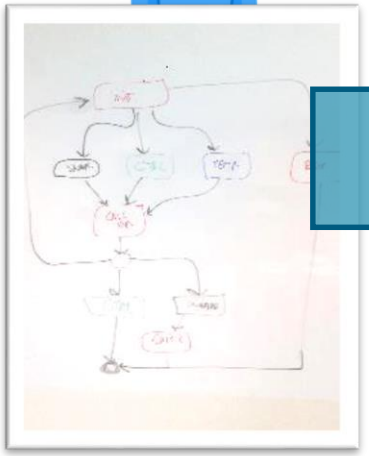
Reflects the context and the working assumptions:

- in an industry domain
- in a specific company
- for a specific product in a given company

Software factories are highly-specialized for a given context

THE ENGINEERING CHALLENGE

Domain expert

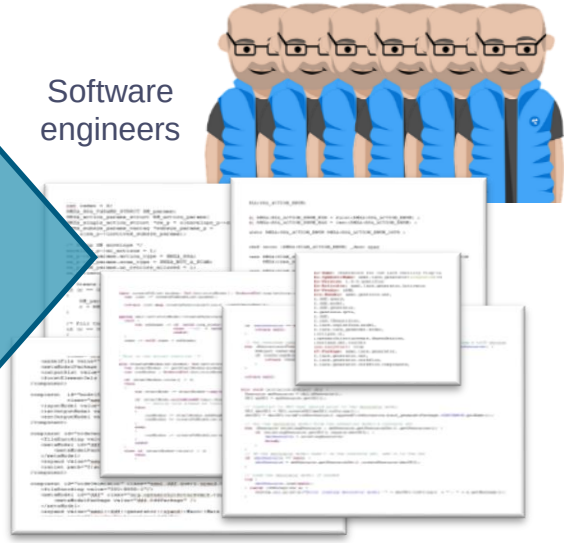


Requirement specification

Design gap

Goal: bridge the gap as much as possible

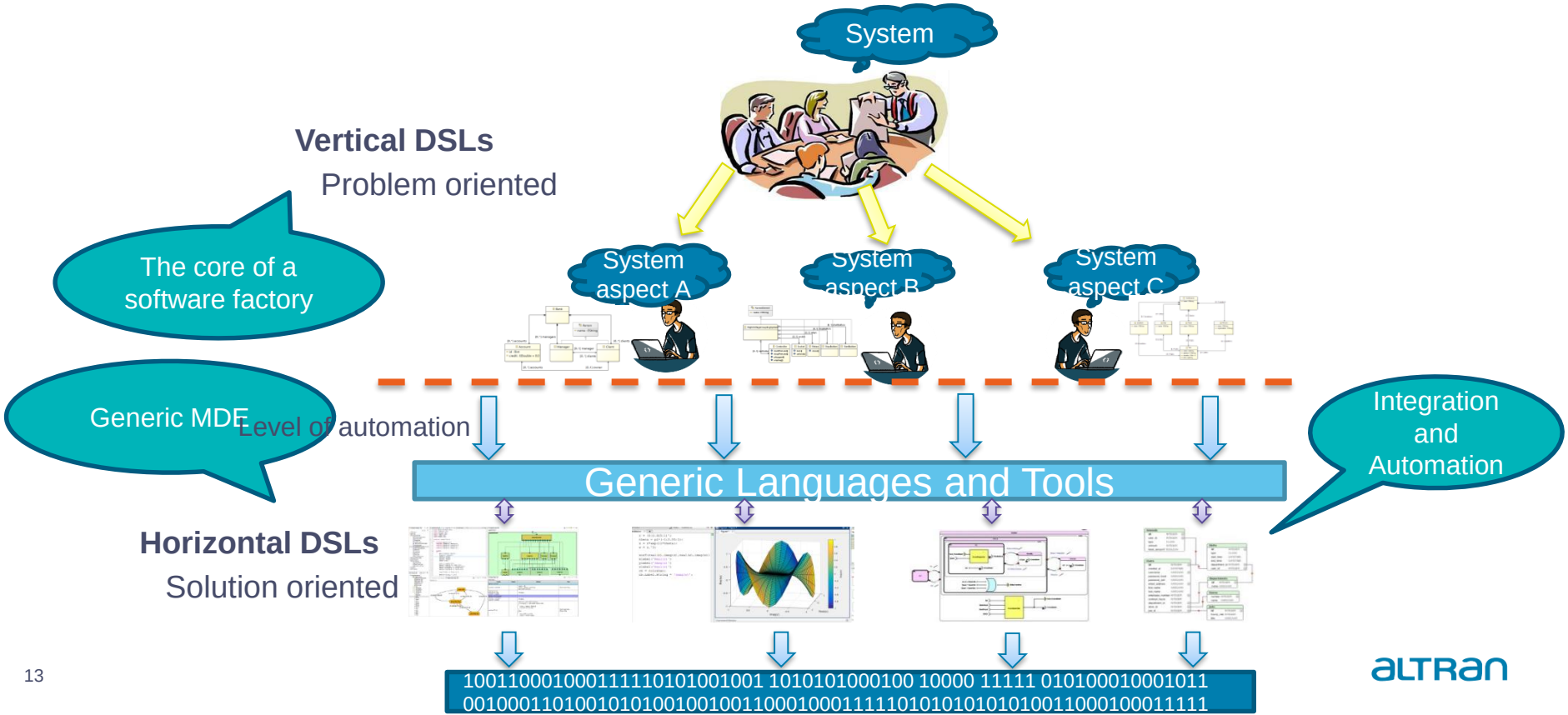
Software engineers



Software engineering and testing

ALTRAN

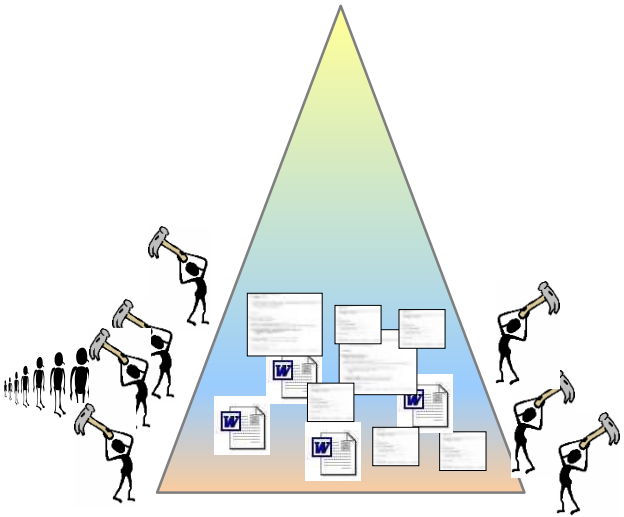
ENGINEERING AUTOMATION WITH DOMAIN SPECIFIC LANGUAGES



ALTRAN

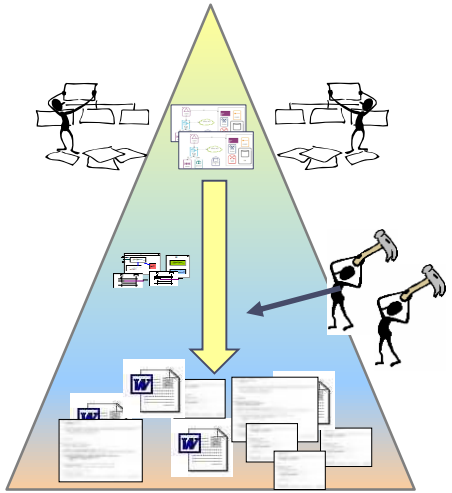
DEPLOYMENT OF SOFTWARE FACTORIES IN ENGINEERING PROCESS

Application of Software Factories requires a change in the engineering process



… typically from a document-based and GPL code-based approach

‘climb’ or ‘jump’?

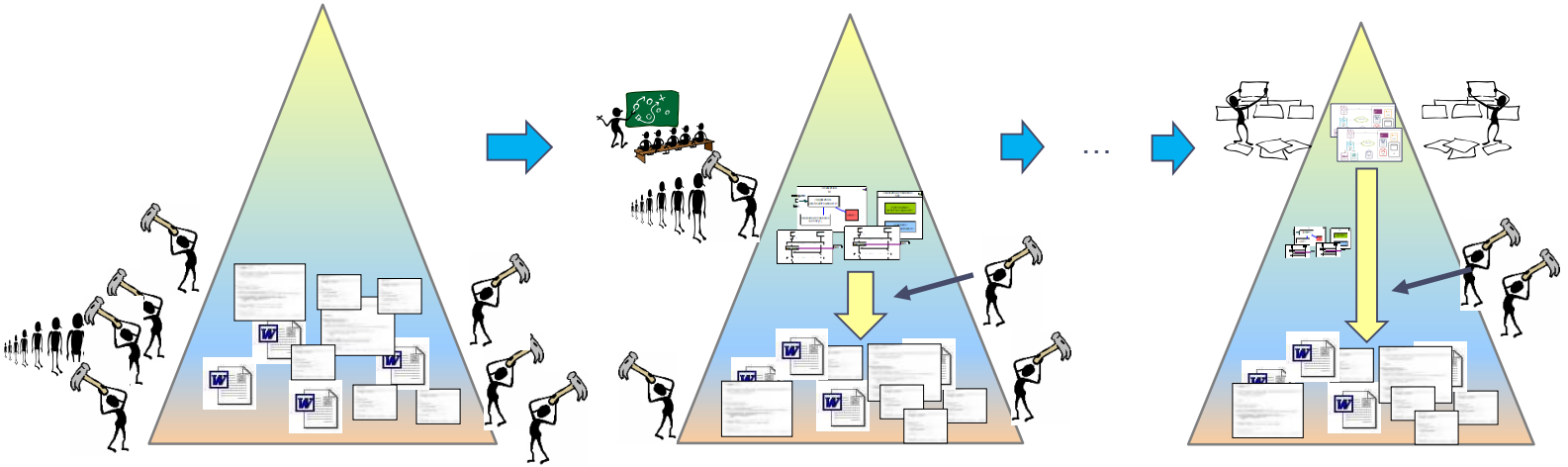


… to a complete automated factory with integrated vertical DSLs

ALTRAN

“CLIMB” THE MOUNTAIN

‘Climb’ by gradually raising the level of abstraction



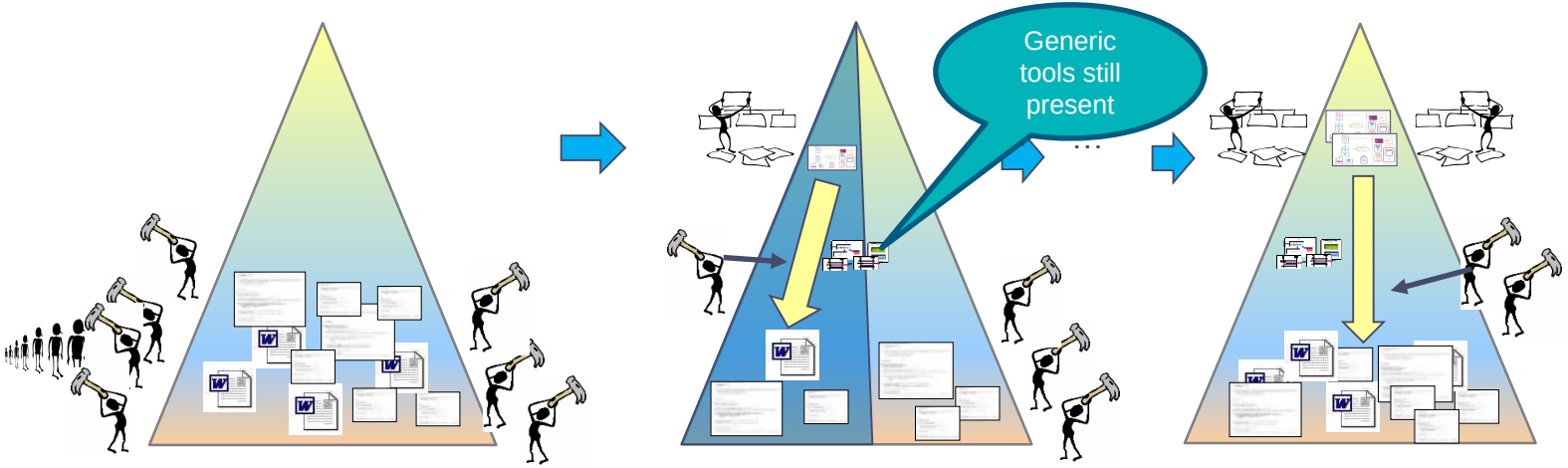
Generic modeling with horizontal DSLs

ALTRAN

“JUMP” THE MOUNTAIN

‘Jump’ by starting at the right level of abstraction

‘Slices’ of vertical DSLs



Experts do integration of solution technologies

ALTRAN

‘CLIMB’ VS ‘JUMP’

Gradually raising the level of abstraction

- Engineers need to be trained in the used generic technologies (possibly more than once, every time when a new technology is adopted)
- Adaptations of generic tools is usually a significant effort
- Design gap (problem – solution distance) still large

Starting at the right level of abstraction

- Identification of the right concepts is challenging
- Efforts needed in integration of the solution technologies and development of solution generators

We advocate the “Jump the mountain” approach: start building a tailored software factory from the very beginning

‘CLIMB’ VS ‘JUMP’: IN SEARCH OF SYSTEMATIC EVIDENCE

Both approaches are possible realizations of the MDE vision

- Employ modeling
- Reports exist that confirm the MDE promises are met at different degrees

Comparisons are usually between non-MDE and MDE-based solutions

It is challenging to systematically compare the two approaches at a large scale and in long term

EXPERIENCE WITH SOFTWARE FACTORIES

Improved productivity compared to traditional (no model-based) approach

- In the range 3-10 times reduction of development time

Inexperienced engineers learn faster

Training efforts are less than in the case of introducing a generic MDE tool

WHAT TO MEASURE

In order to empirically assess an MDE-based process we need measurements

An assessment model is required

- Measure development effort per activity in a MDE process

(Model-Based Software Engineering: A Multiple-Case Study on Challenges and Development Efforts, Jolak et al., MODELS2018)

- Measure maintenance effort

But also …

- Measure the effort for developing a software factory
- Measure the effort for maintaining a software factory

SOFTWARE FACTORY ANATOMY

LANGUAGE ARCHITECTURE

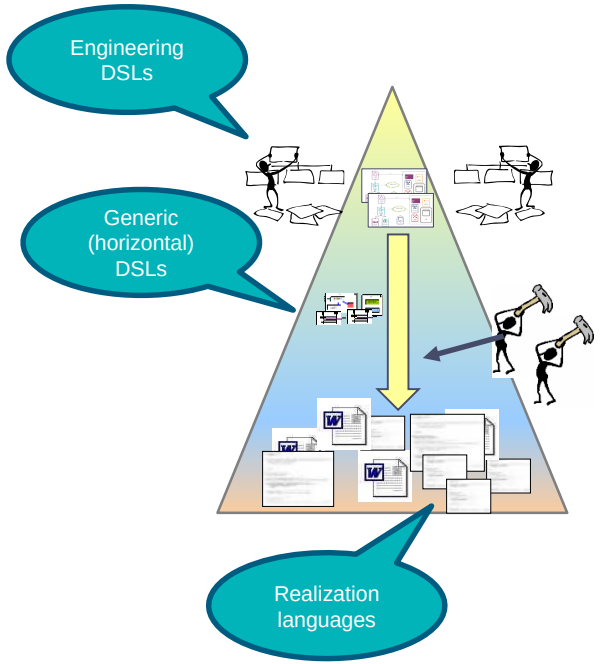
LANGUAGE ECOSYSTEMS

An engineering process is based on an ecosystem of languages

Languages come en masse

Figures from real projects:

15 core DSLs 14 DSLs
89 transformations >10 major transformations



How do we manage such an ecosystem?

LANGUAGE ARCHITECTURE

A language ecosystem is captured in a *language architecture*

- Language Architecture: an Architecture Language for MDE, Brouwers, Hamilton, Kurtev, Luo. Modelsward 2017

Language Architecture is a *megamodel*

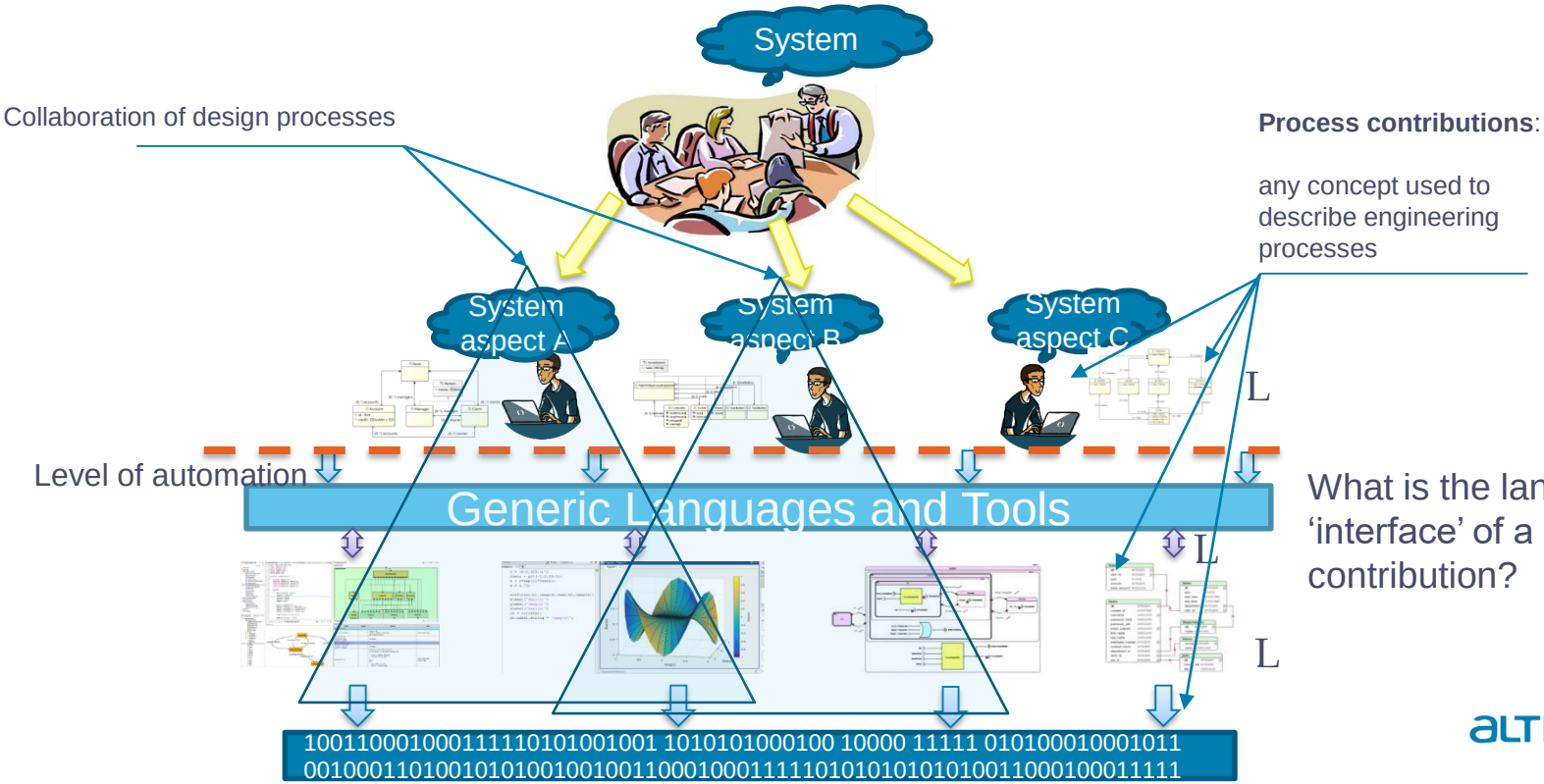
Brings *language-centric view* on the engineering process

Other works aim at recovering a language architecture

- Systematic Recovery of MDE Technology Usage, Di Rocco, Di Ruscio, Hartel, Iovino, Lammel, Pieranotnio, ICMT 2018

LANGUAGE CENTRIC VIEW ON ENGINEERING PROCESSES

Language Architecture captures linguistic aspects of automated engineering processes

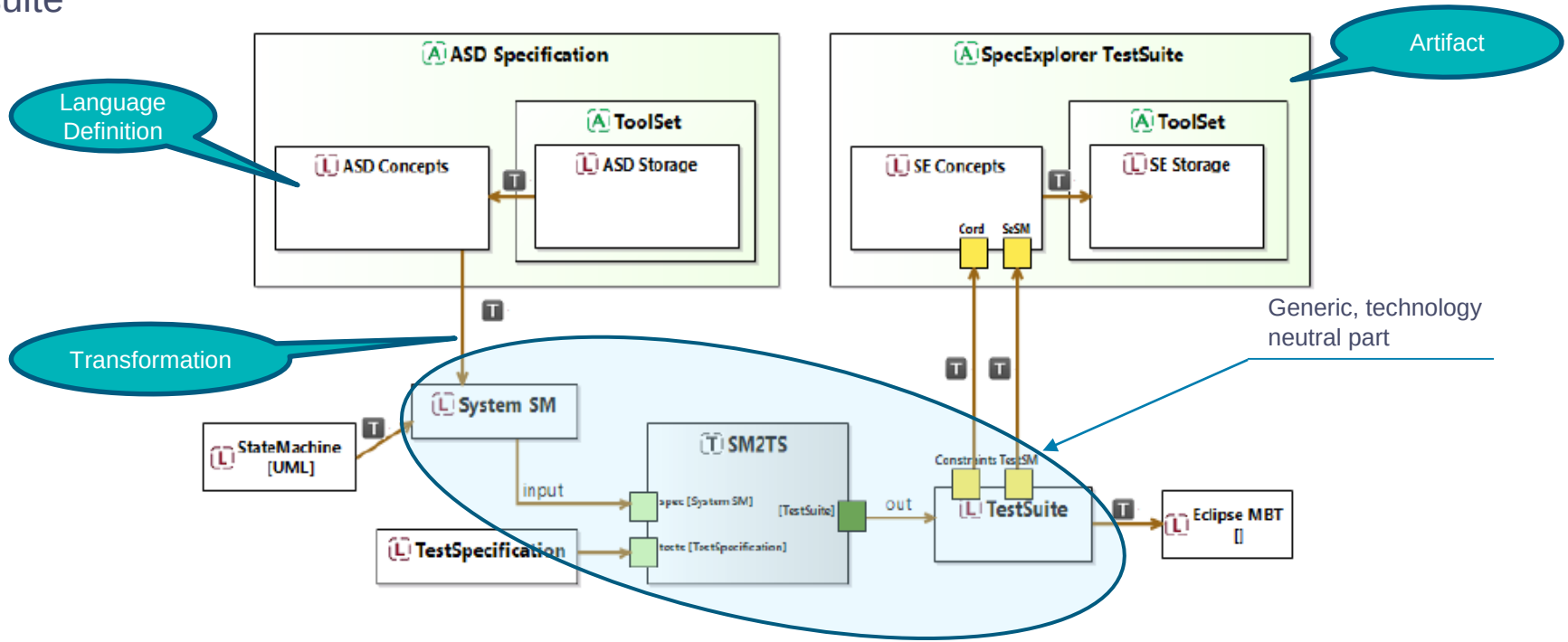


What is the language ‘interface’ of a process contribution?

ALTRAN

EXAMPLE

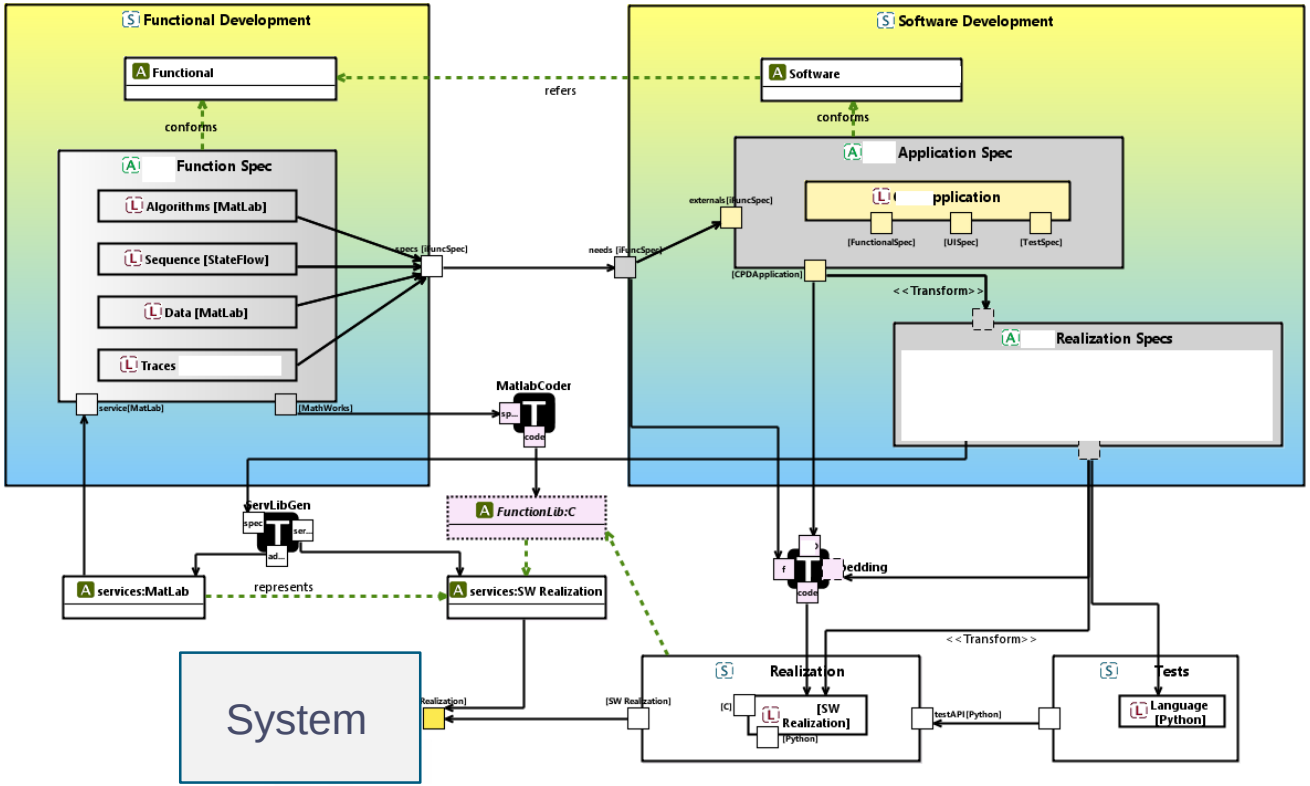
Toolchain for model-based testing: from ASD interface specifications to SpecExplorer test suite



ALTRAN

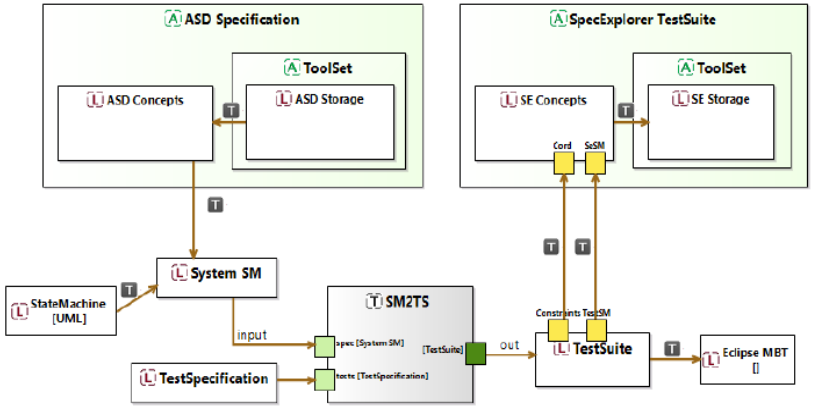
EXAMPLE

Functional specification and software realization phases in a language architecture



ALTRAN

LANGUAGE ARCHITECTURE: APPLICATIONS



Language architecture applications:

- Identification of points for automation
- Inventory of knowledge and expertise

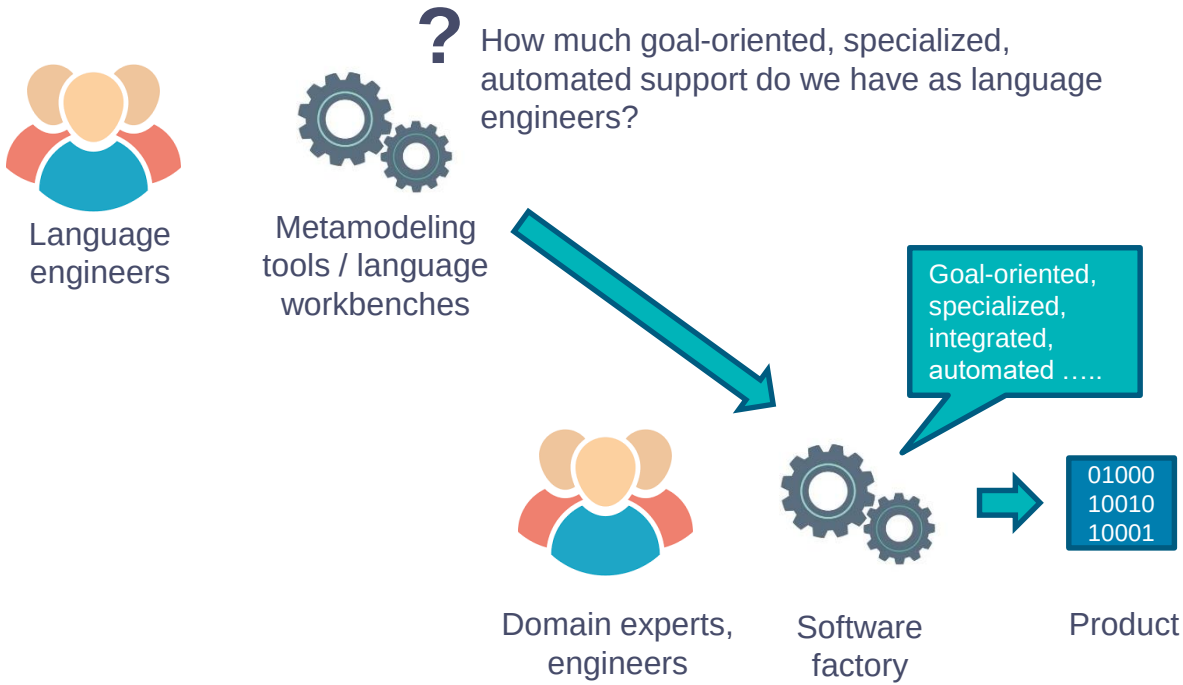
Makes the role of languages in engineering processes explicit

Blueprint for the development of Software Factory

LANGUAGES

ONE DAY IN THE LIFE OF A LANGUAGE ENGINEER

SOFTWARE FACTORIES FOR LANGUAGE ENGINEERING

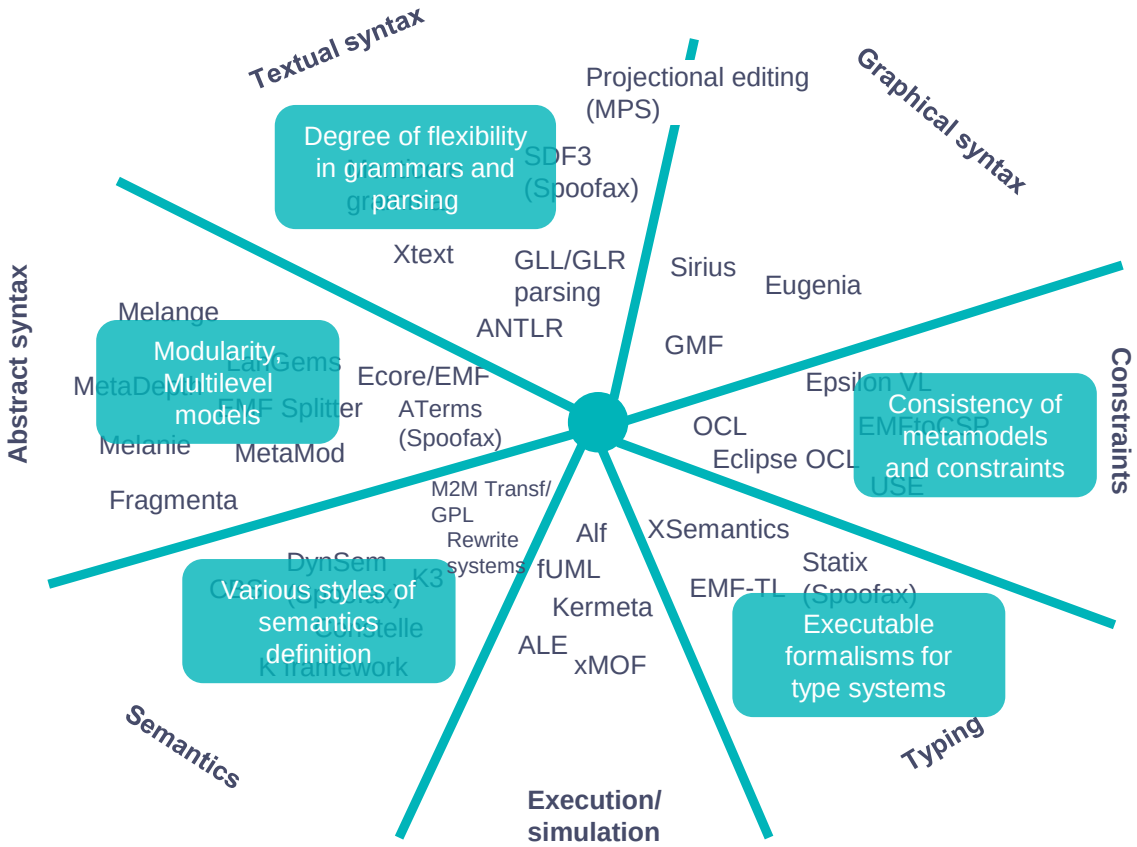


Languages and their supporting tools are also products of an engineering process

Language engineering process can itself be supported by a software factory

ALTRAN

LANGUAGE ASPECTS AND SOME TECHNOLOGIES

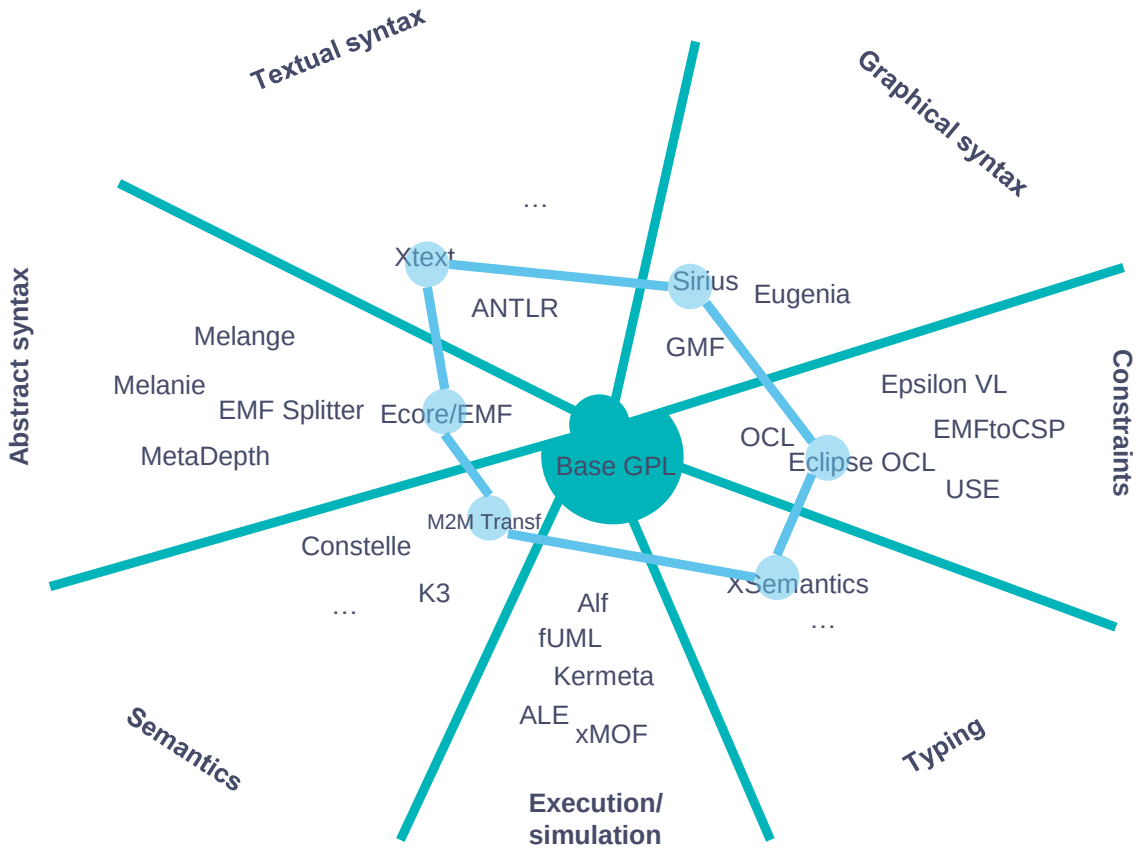


Various:

- workbenches that integrate most or all aspects
- theoretical underpinnings
- level of maturity
- degree of industrial adoption

ALTRAN

LANGUAGE ENGINEERING PRACTICE AT ALTRAN



Language engineering based on Eclipse/EMF platform

- uses industrial state-of-the-art technologies in this platform

Note: usage of model transformations is discussed further

ALTRAN

OBSERVATIONS

Once a commitment to a language workbench/platform is made it is difficult to use and integrate aspects from other platforms

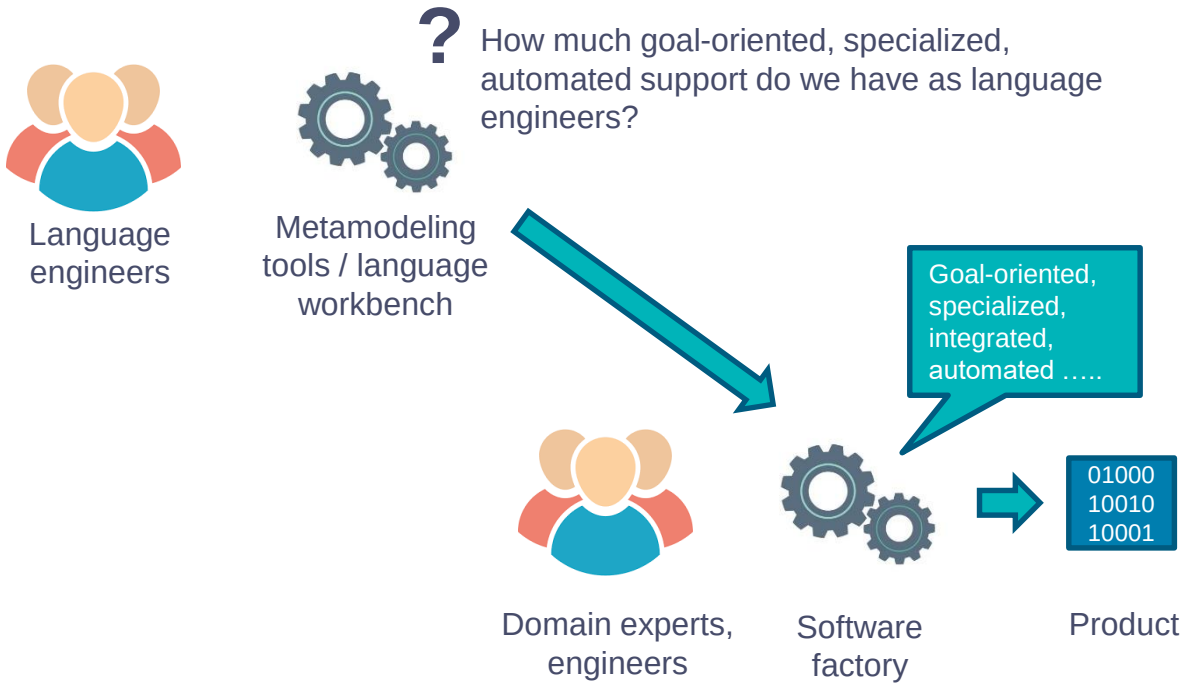
Mature support for model creation: textual and visual editors

Limited support for language reuse and composition

We are nearly empty handed concerning tools in semantics and simulation aspects

- GPL code and M2M transformations are commonly used

SOFTWARE FACTORIES FOR LANGUAGE ENGINEERING



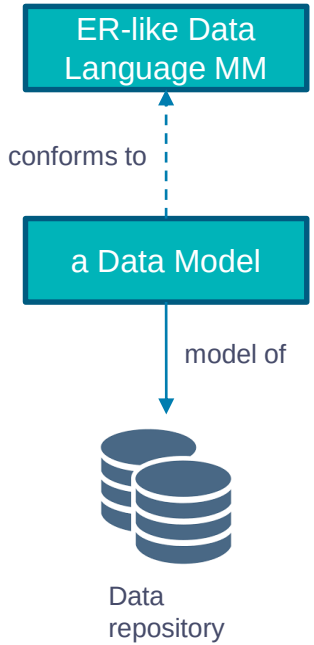
Some aspects are tackled at a generic level

Is this a problem?

- In many cases, M2M transformations are sufficient
- In other cases, dedicated support saves efforts

ALTRAN

EXAMPLES FROM PRACTICE: A DATA MODELING LANGUAGE



Data description language

Relatively intricate interplay of multiplicities, access control for CRUD operations

Repository must always be consistent wrt the data model

Mechanism like garbage collection

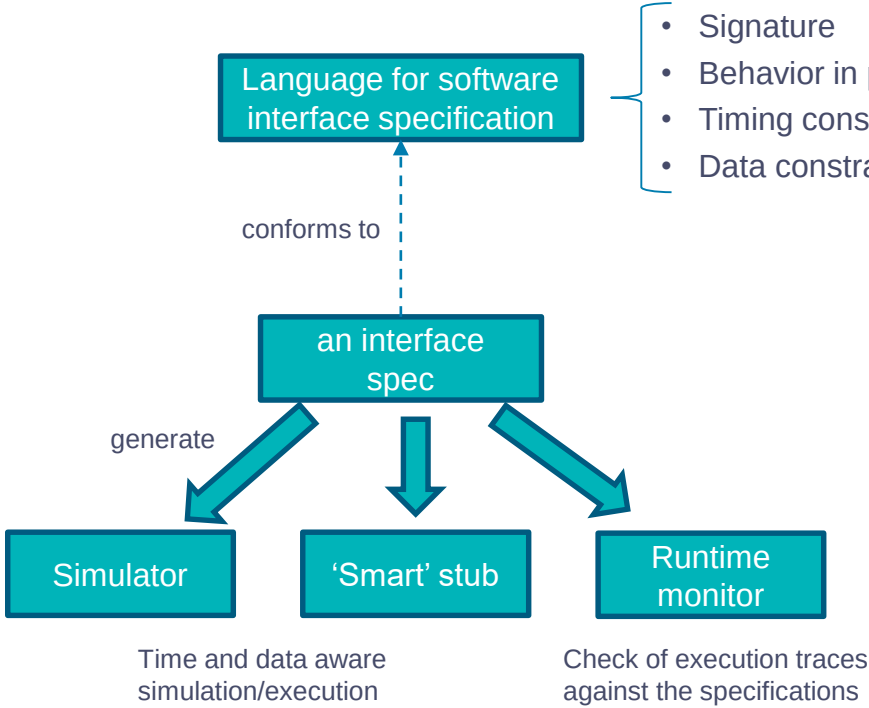
We need to:

- Define OCL constraints that eliminate inconsistent data models
- Generate test data repositories based on partial information

Challenges:

- Do the OCL constraints eliminate all data models that model possibly ill-behaving repositories? (similar to *safety* property of a type system)
- Model completion of partial models

EXAMPLES FROM PRACTICE: TIME AND DATA AWARE SIMULATOR



- Signature
- Behavior in protocol state machines
- Timing constrains (response time, periodicity)
- Data constraints (allowed values, data dependency)

- Theories (solution domain) are known
- Semantics was given in mathematical way for a common understanding

Challenges:

- Development of the tools still labor intensive with a lot of manual work

ALTRAN

OBSERVATIONS

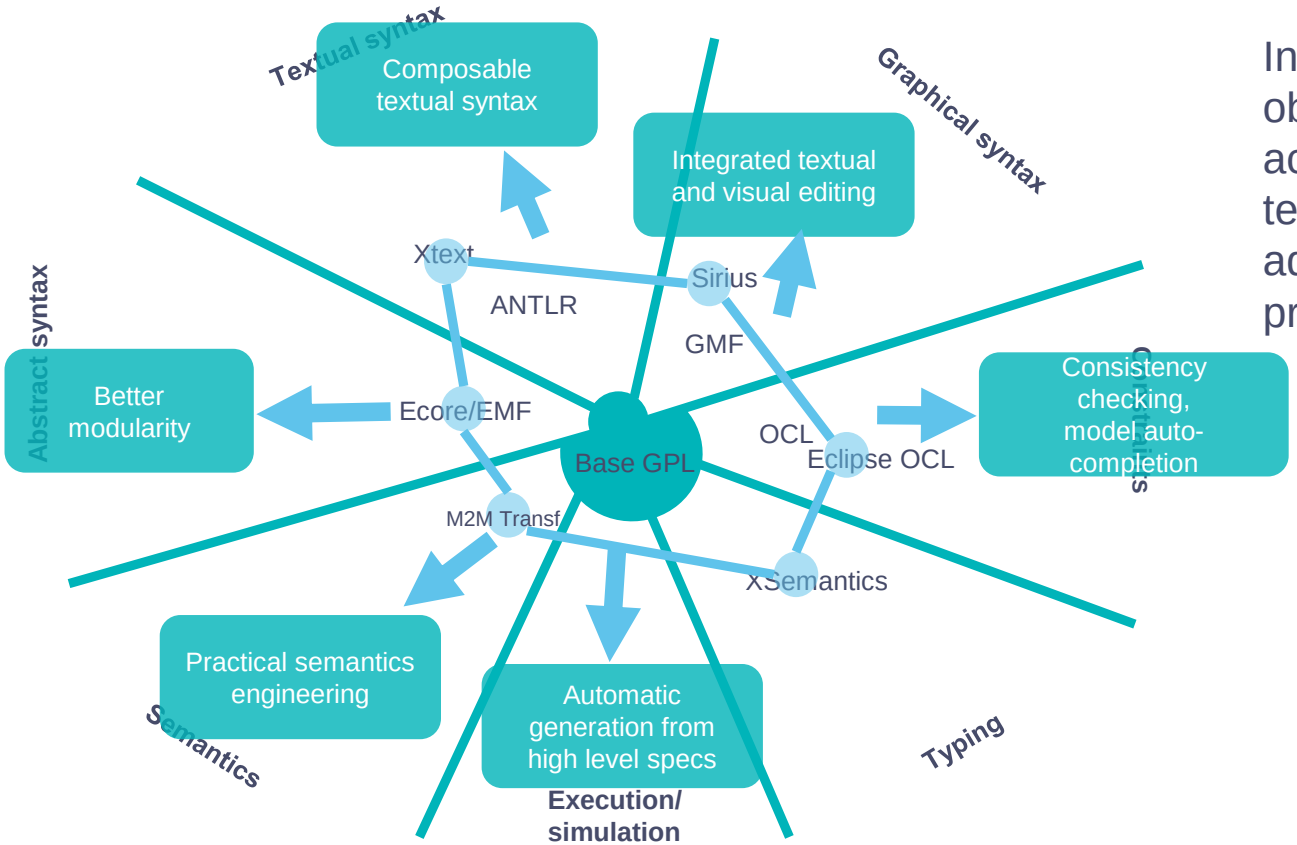
Significant amount of language semantics, execution and simulation issues can be solved with generic tools

- Model interpreter in a GPL
- Model transformation to a known language tool (translational semantics)

However, we encounter some intricate cases:

- Automated reasoning tools are of great value
- Support for model execution is too laboursome and tools are lacking

POTENTIAL ADVANCEMENTS

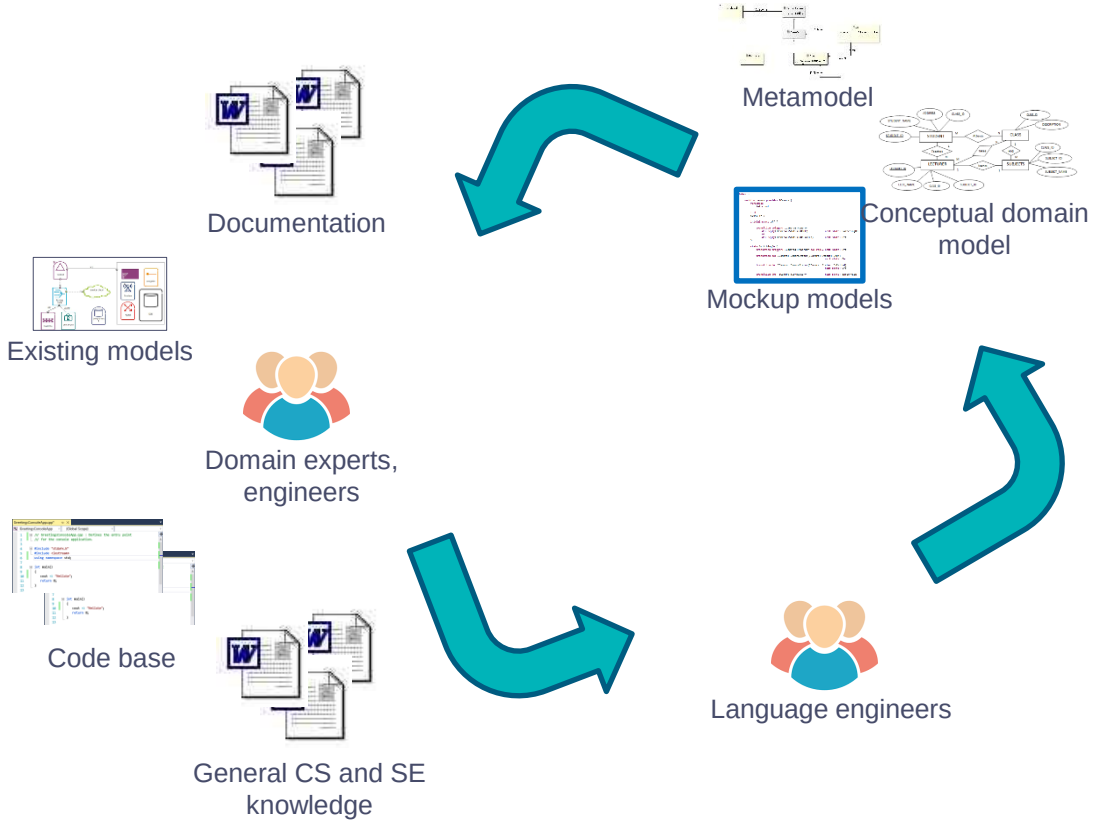


In all language aspects we observe experimental and academic tools and techniques that may advance the state-of-practice

ALTRAN

LANGUAGE DISCOVERY

An iterative and interactive process



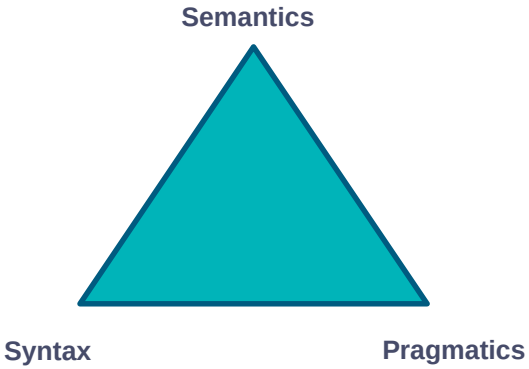
The limits of my language are the limits of my world

L. Wittgenstein

Can this process be sped up and partially automated?

ALTRAN

PRAGMATICS: THE SECOND CLASS CITIZEN



- Pragmatics: the study of how a language is used; how context contributes to meaning;
- Pragmatics = Meaning – Semantics
- Often not treated equally to syntax and semantics
- Emphasizes the relationship ‘language’ <-> ‘user’

Plays a crucial role in designing user-friendly languages!

MODEL TRANSFORMATIONS

MODEL TRANSFORMATIONS IN ALTRAN PRACTICE

We use QVTo to write model-to-model transformations

- Transformations are the heart of the generators
- Codify solution knowledge

Motivation:

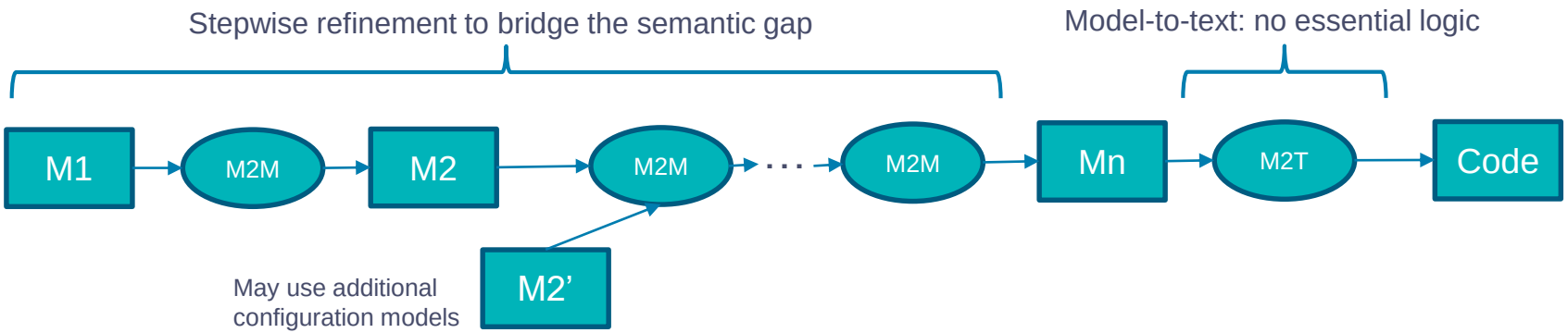
- Model transformation is a recurring task, ergo, it benefits from a DSL-based approach
- Well recognized benefits of using M2M transformation language over using a GPL
- QVTo is a standardized language

Application:

- Code generation
- Bridging different technology domains
- Model extraction from data

MODEL TRANSFORMATIONS IN ALTRAN PRACTICE

Transformation chain pattern for code generation:



ALTRAN

CHALLENGES

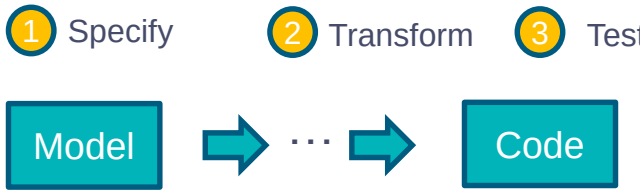
Dealing with variability in the result models in case of many configuration parameters

- Separation of concerns and design challenge

Performance and scalability of model transformations

- Execution time and factors that influence it
- QVTo profiler was developed in 2013-2014

Specifying and verifying model transformations



I trust my transformation and result
because the result tests pass

I trust my result
because I trust my model and transformation

THE PRESENT OF MODEL TRANSFORMATION LANGUAGES

Model transformations are needed

The present of model transformation languages in practice?

- Still largely unknown and probably not bright
- Checking on ICMT conference series, very few studies about the industrial adoption

THE PRESENT OF MODEL TRANSFORMATION LANGUAGES

Recent study:

"… results show no statistically significant benefit of using a dedicated transformation language over a modern general-purpose language. However, we were able to identify several aspects of transformation programming where domain-specific transformation languages do appear to help, including copying objects, context identification, and conditioning the computation on types."

Model Transformation Languages under a Magnifying Glass: A Controlled Experiment with Xtend, ATL, and QVT. Hebig, R., Seidl, C., Berger, T., Pedersen, J.K., Wasowski, A. ESEC/FSE 2018

THE PRESENT OF MODEL TRANSFORMATION LANGUAGES

- As a MT community, we probably missed to convince the practitioners and to demonstrate the usefulness of the transformation languages

- What are the factors in choosing a technology for solving a model transformation problem?

TOOLS

TOOLS

“We need better tools”

(stated at many events by many people over the last years)

Are the Tools really a Problem?: Yes and No

(*Industrial Adoption of Model Driven Engineering*, J. Whittle et al. MODELS 2013)

We have to recognize that we can already do a lot with the current tools

- Tools improvements and maturation did happen at some degree

TOOLS

Plethora of experimental and academic tools

- The grains of future advanced tools (?)
- Already useful at the level of reuse of knowledge
- Very often do not consider the industrial context in which may be integrated

Significant gap in terms of required engineering efforts for tool completion and maturity

- How to bridge this gap?

CONCLUSIONS

Multiple ways to apply MDE-based solution in practice

Software Factories: automated software production, unique solution for a given context

Empirical assessment needed:

- MDE-based vs code-based development
- Comparison among MDE approaches

CONCLUSIONS

Language engineering: an engineering process that requires its own highly optimized tools

Significant potential for advancing industrial language engineering tools

We can achieve a lot but should always strive for better

QUESTIONS?

ALTRAN