AdaCore

April 10th, 2013

Lean R&D for Open Qualifiable Tools

Matteo Bordin
Project Manager

Thanks!
Cyrille Comar (AdaCore)
Arnaud Dieumegard (IRIT)
Tonu Naks (Krates)
Marc Pantel (IRIT)
Frederic Pothon (ACG)
Elie Richa (AdaCore)
Jose' Ruiz (AdaCore)
Andres Toom (Krates/IRIT)

First Tool Qualification Symposium

AdaCore – Open Source Tool Provider for High-Integrity Systems

Development Environment

Multi-language development suite (Ada, C/C++)

Validated compilers (1.4M test results / day)

Certified run-time

Src-to-obj code traceability study

Qualified verification tools (stack usage, coding standard, …)

QA & Testing for Embedded

Qualified structural coverage analysis

up to MC/DC (no instrumentation)

Advanced Static Analysis

High-security development (EAL 5-7)

Formal methods & verification

DO-178B/C, ECSS, EN-50128 Qual./Cert. Kits

http://www.adacore.com/customers

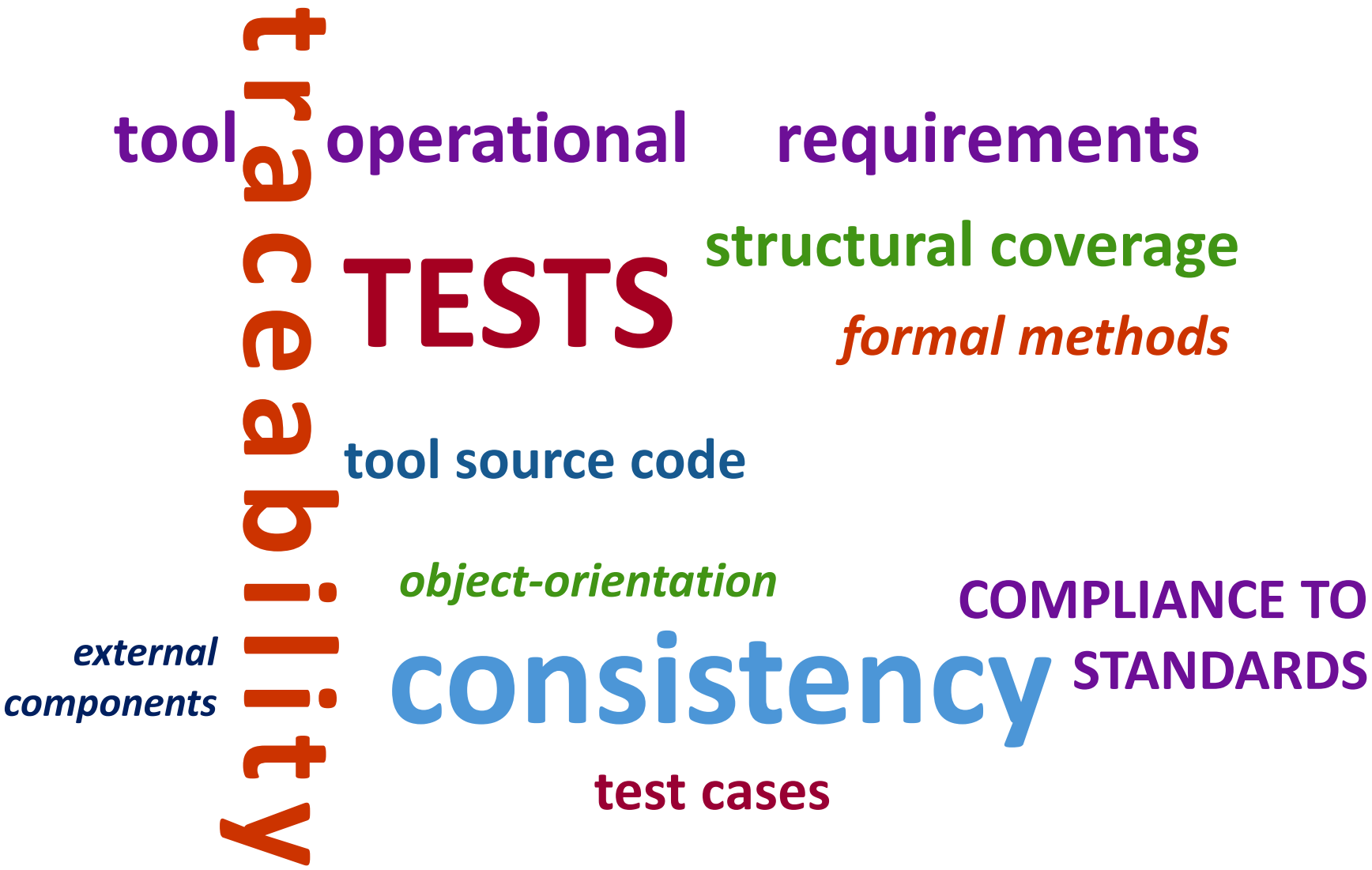
Warning! The meaning of "Tool Qualification" in this presentation!

- **Not a generic *"good enough"* stamp**
- **Always to avoid manual activities without manual review**
- **Always for a precise objective**
  - Coding standard verification
  - Structural coverage analysis
  - …
  - **In the case of a code generator**
    - *Manual review of source code (compliance/consistency)*
    - *Low-level requirement-driven testing*
    - *Structural coverage*

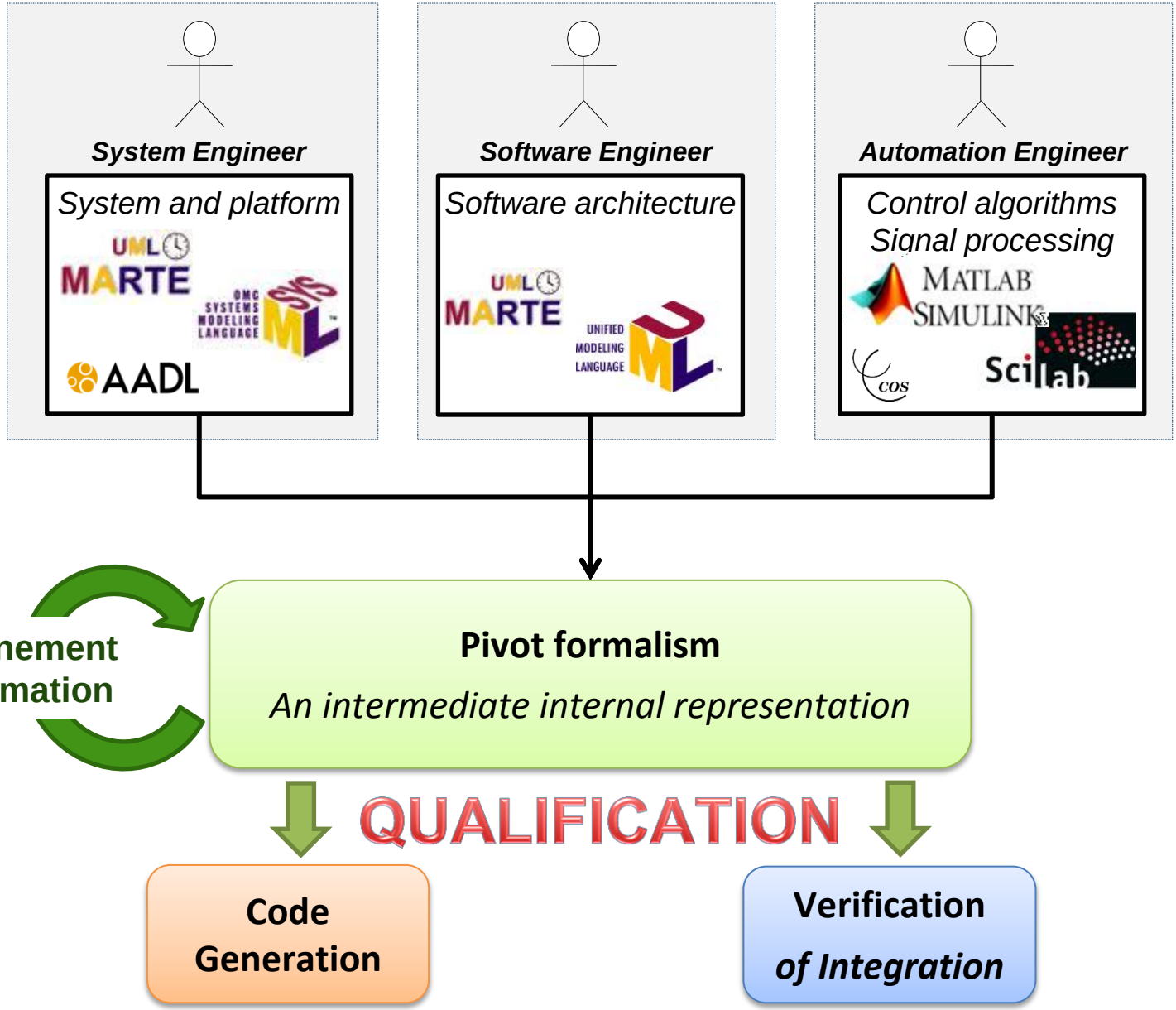
Is it really that difficult to qualify a code generator?

Project P - One core question

Is it really that difficult to qualify a code generator? Yes, it is



Some additional difficulties



DO-330 & other annexes

One of the first applications

Open

Role of the consortium and business model

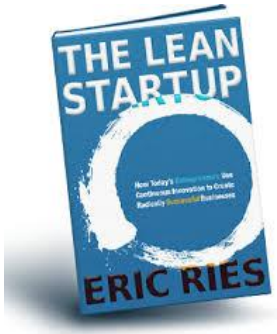
Minimize re-qualification effort

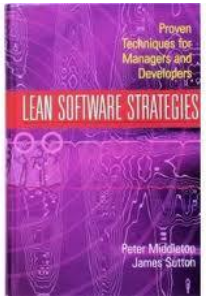
Add new input/output languages, bug fixes, …

Multi-domain

Avionics, space, automotive, …

From Fordism to Lean Manufacturing to the Lean Startup





agile

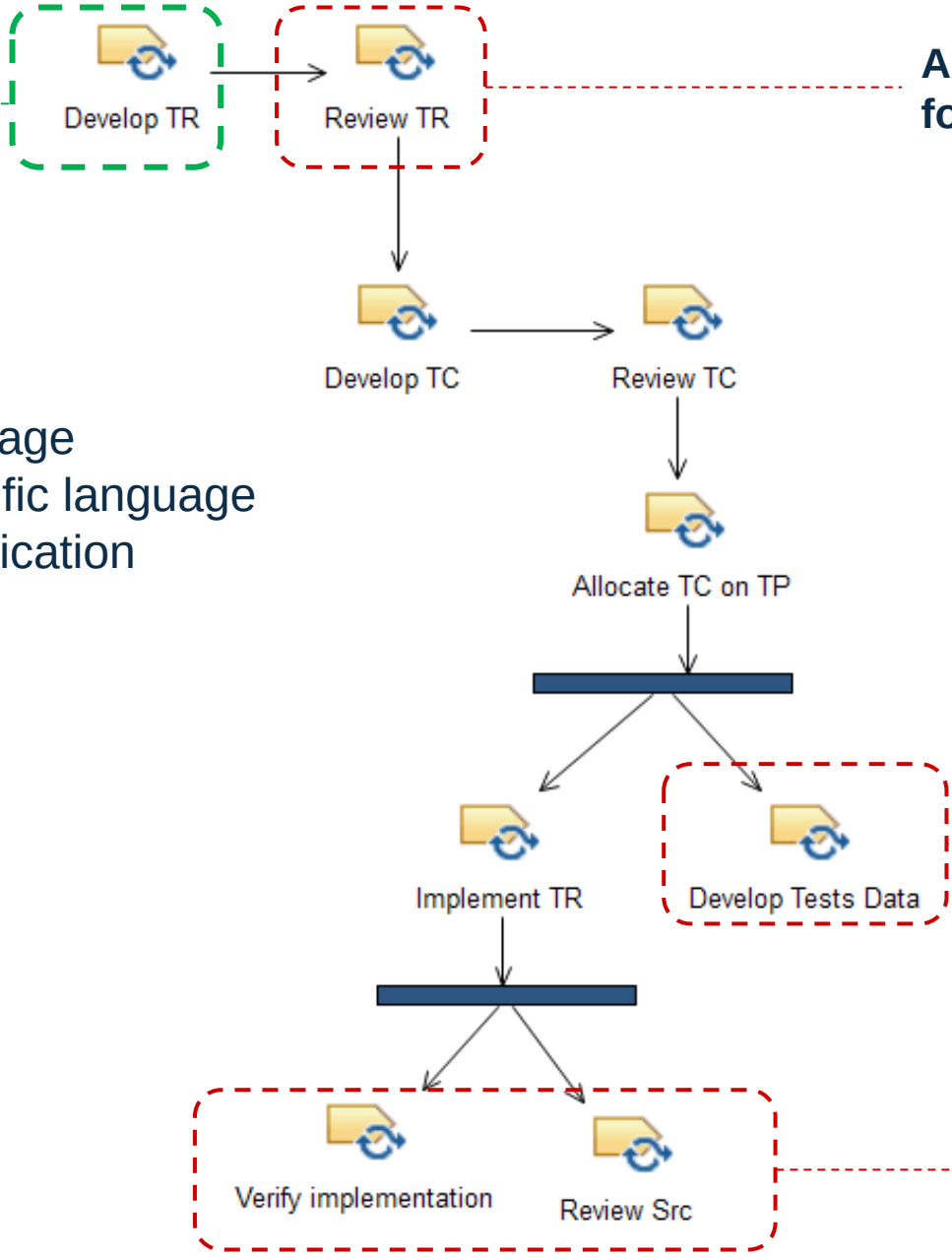
TOYOTA

FORD MOTOR CO.
DETROIT, MICH.

Our practical approach

1. Process modeling & process lines
2. Architecting of qualification material
3. Component-based development
4. Extensive use of metamodeling techniques
5. Advanced programming techniques
6. Constant measuring

Step 1 – Process modeling & process lines

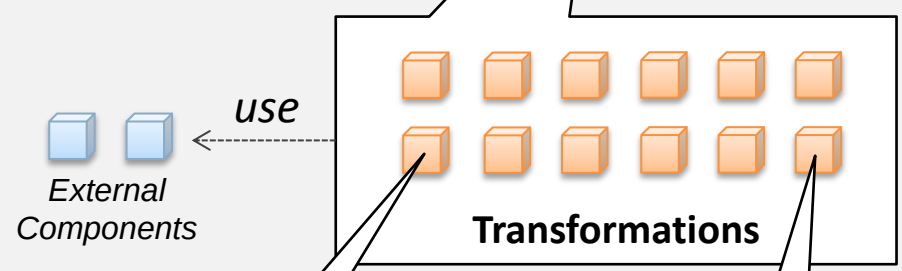


Alleviated if using formal specifications

Can use:

- Natural language
- Domain specific language
- Formal specification

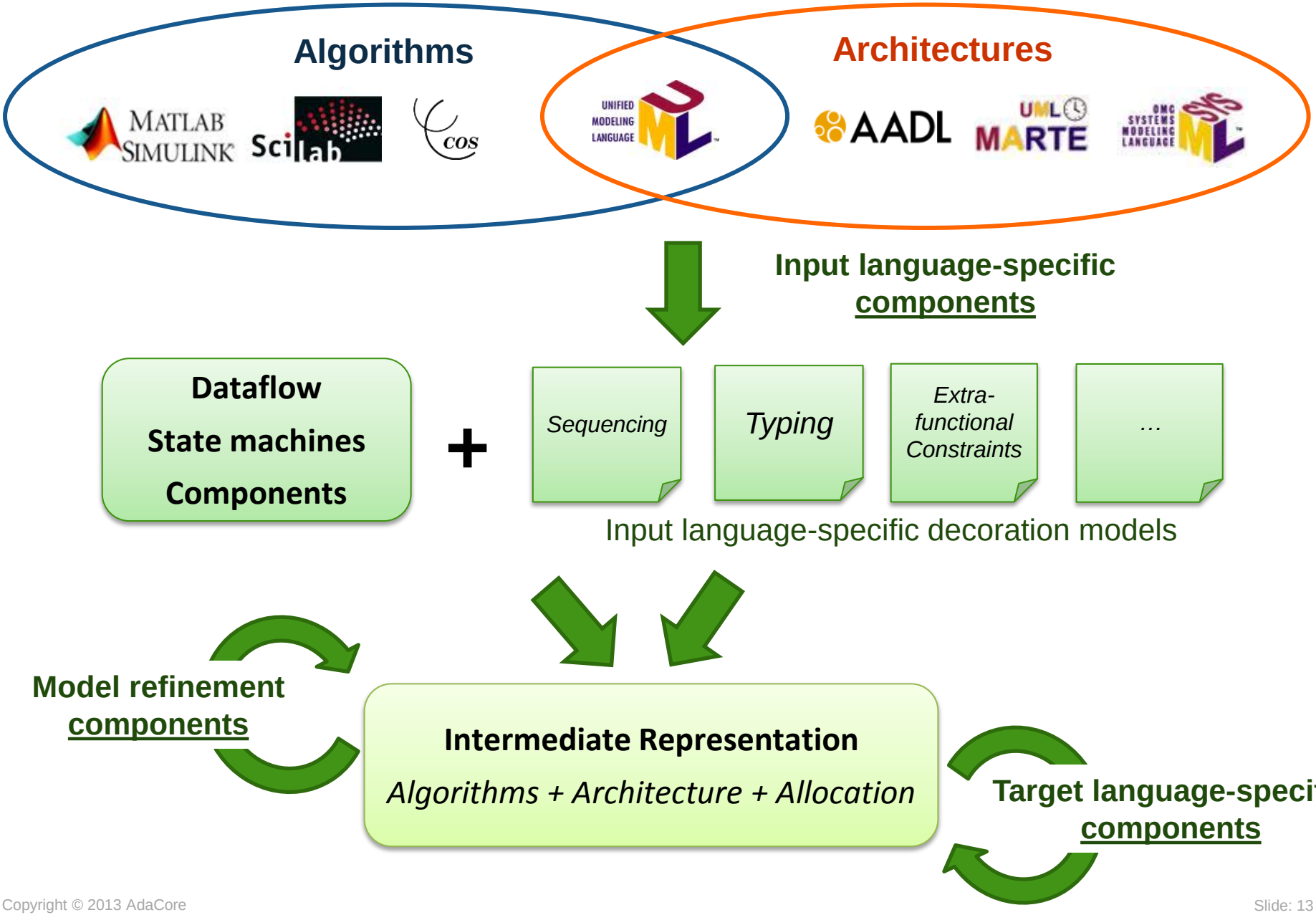
Step 2 – Architecting of qualification material



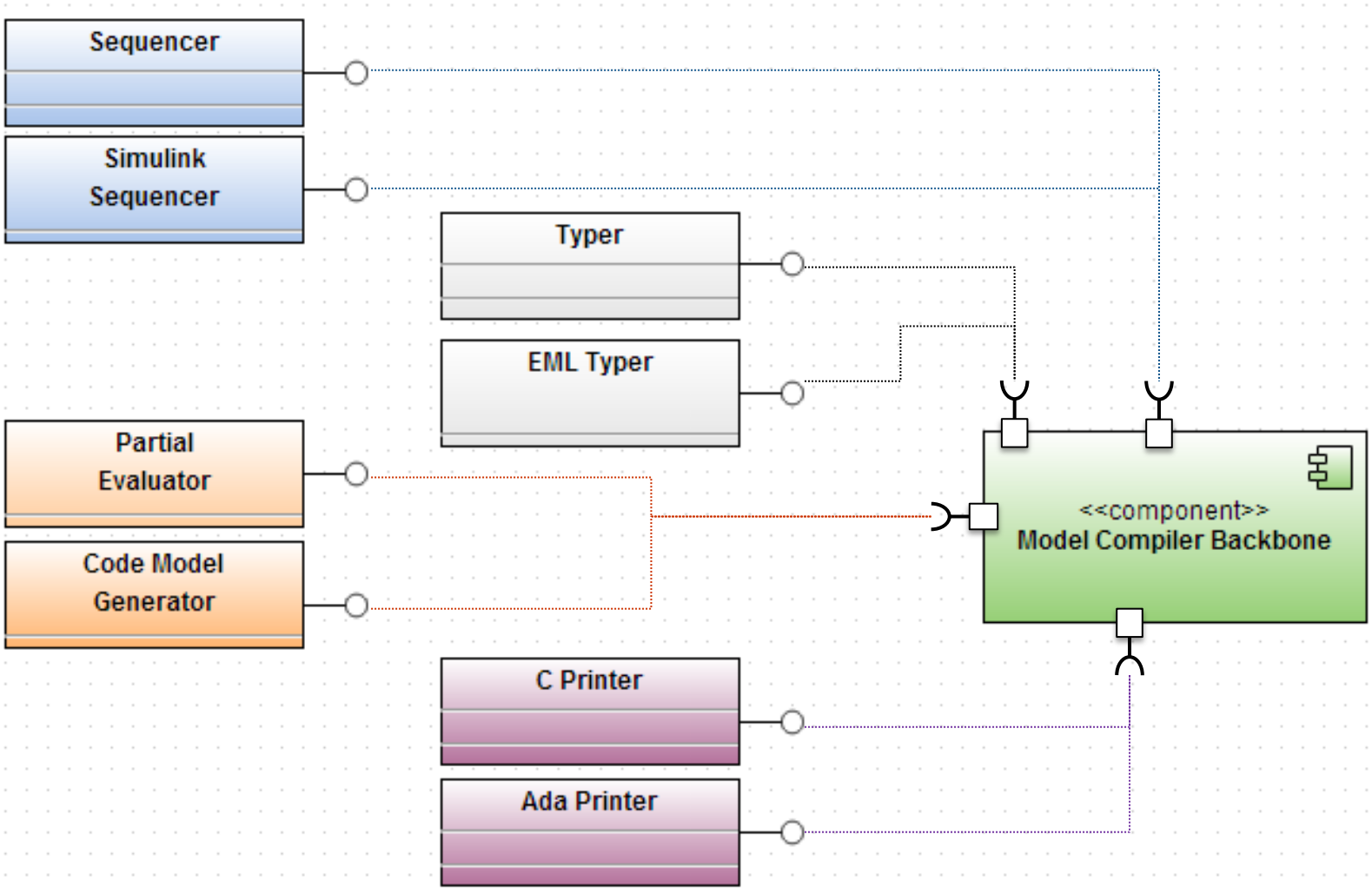
Tool Architecture
traceable to src subprograms

Tool Requirements
traceable to src subprograms

Step 3 – A Component-based approach for qualification-by-composition



Step 3 – A Component-based Approach for qualification-by-composition (I)



Multiple configuration

Composition of different components

Evolution of qualification material

To follow tool releases & emergence of new requirements

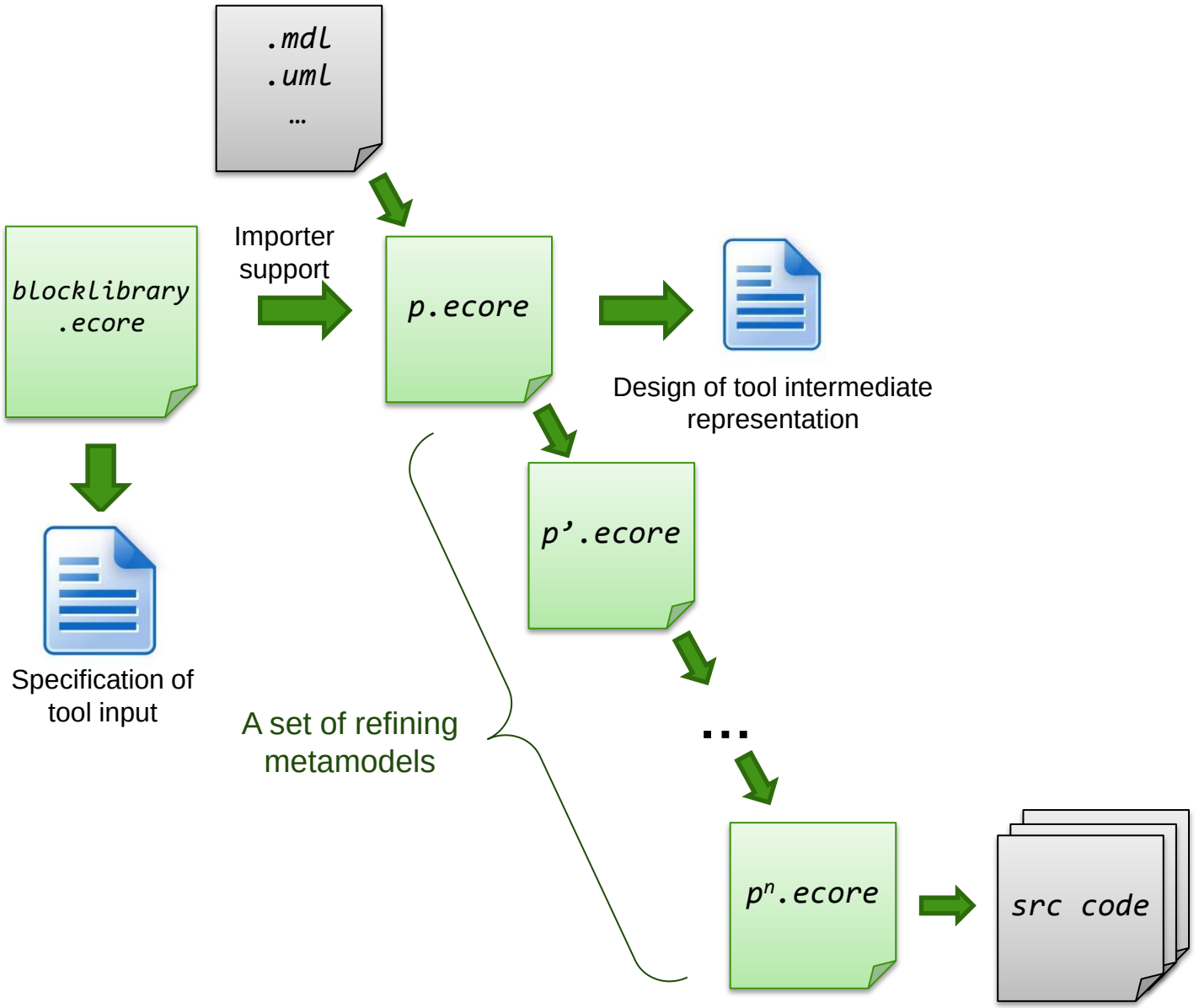
Continuous qualification

Qualification as part of common development activities

OPENCOSS, Safecer

FP/Artemis collaborations

Step 4 – Extensive use of metamodeling techniques



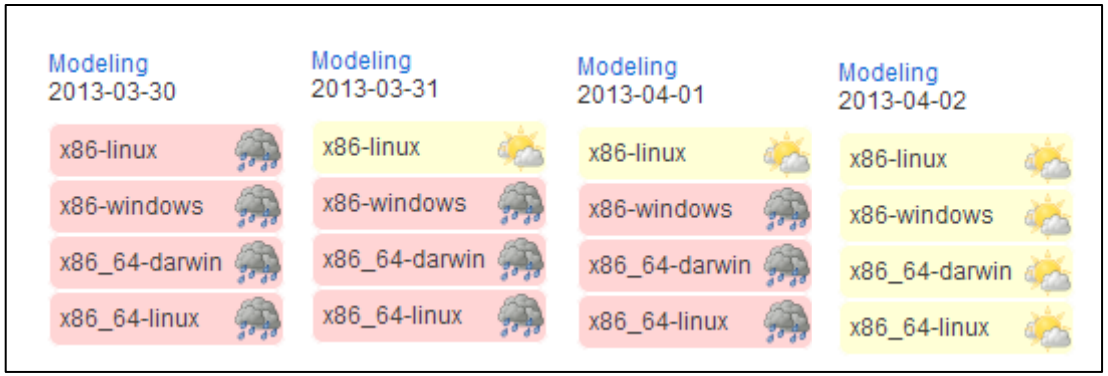
Note: EMF XMI is used as an interchange format, the P toolset does not depend on the EMF framework.

Tool Requirements ⇔ Pre/Post conditions

Unambiguous, Verifiable, Traceable

```
procedure Calculate_Total_Order
   (Self   : in out Sequencer;
    Blocks : Block_List)
with Post =>
   --  @tr TR_SEQ_1
   --  The executionOrder of each block shall be bigger than the
   --  one of the predecessors unless the block has internal memory
   --  or is a controlled block.
   (for all Elem of Blocks =>
       (if not Elem.Get_inControlPorts.Is_Empty then
           Elem.Get_executionOrder = -1
        else
           (if Elem.Is_DirectFeedThrough then
               (for all Pre of Get_All_Predecessors (Elem) =>
                   (if Pre.Is_DirectFeedThrough then
                       Elem.Get_executionOrder > Pre.Get_executionOrder))
            else Elem.Get_executionOrder = -1)));
```

Step 6 – Constant quality measurement



Continuous building &

Nightly test results

Qualification-specific metrics

Conclusion

- **A Lean path to the qualification of code generators**
  1. Process modeling & process lines
  2. Architecting of qualification material
  3. Extensive use of metamodeling techniques
  4. Component-based development
  5. Advanced programming techniques
  6. Constant measuring

- **Soon available to the community**
  - A first step towards collaborative tool qualification

www.open-do.org/projects/p