First
Tool Qualification
Symposium

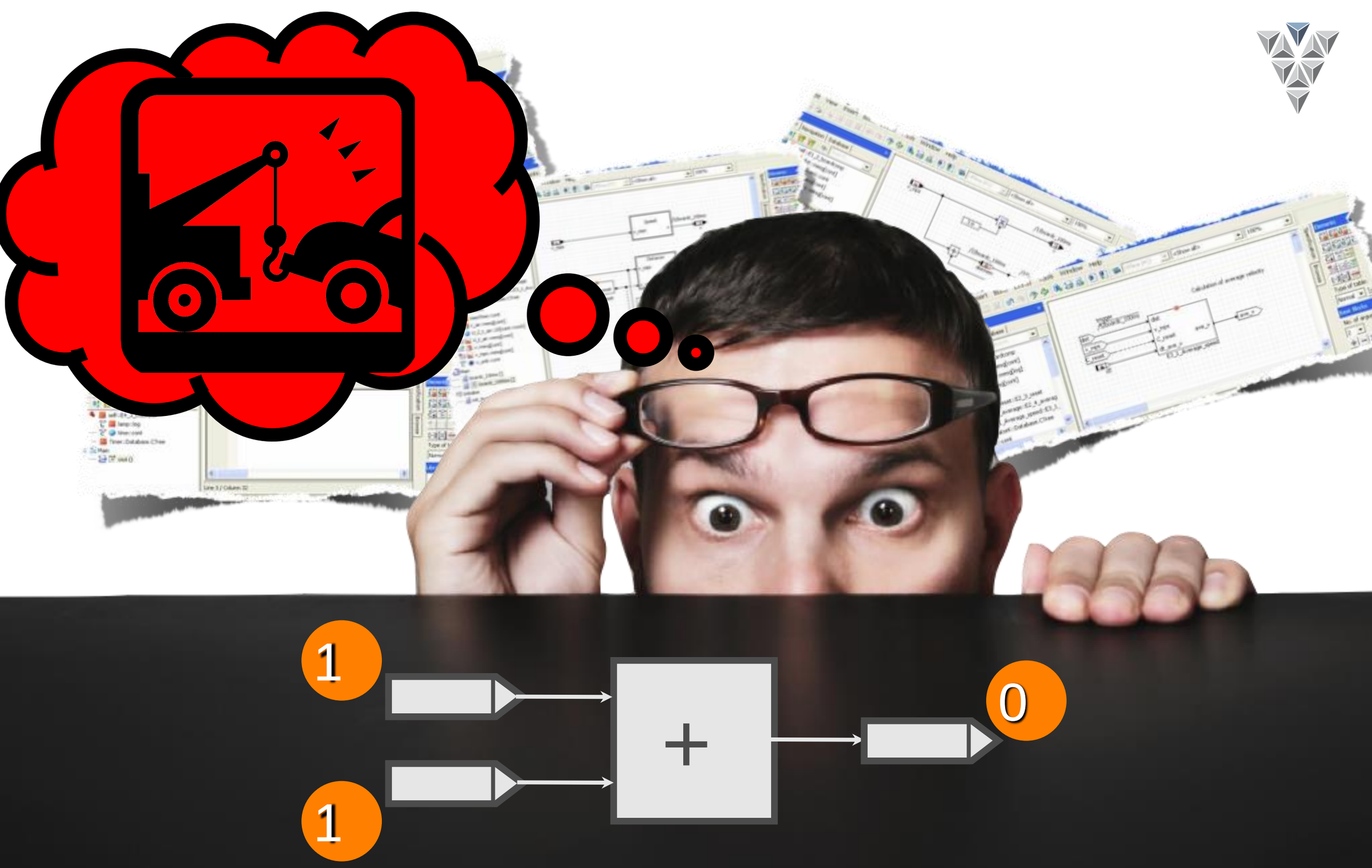
Munich, 10. April 2013

Andrea Osterloh, Validas AG

# Tool Chain Analysis Method

# Agenda

- **Tool Qualification in Standards**
- **Tool Qualifiction in ISO 26262: Activities & Documents**
- **Tool Evaluation in ISO 26262**
- **Risk Assessment of Functions**
- **Assessment of Risk from Tool Use**
- **Tool Evaluation Process**
  - Step1: List the Tools
  - Step2: Identify the Usecases
  - Step3: Determine Tool Impact
  - Step4: Identify potential Tool Failures
    - Terminology
    - Goals for Tool Failure Determination
  - Step5: Assign Detection & Prevention Measures
  - Step6: Compute TCL
- **Documentation**
- **What else can you do?**
- **Visions**
- **Conclusion**


1
1
+
0

# Tool Qualification Comparison

| | ISO 26262 | IEC 61508 | DO 178 + DO 330 |
|---|---|---|---|
| Tool Classification | TCL1 – TCL3<br>ASIL | T1 – T3<br>SIL | Criteria 1 – 3<br>Risk Class<br>TQL 1 – TQL 5 |
| | Proven In Use<br>Process Assessment<br>Validation<br>Tool Safety Standard | 3-7.4.4: 19<br>Specific<br>Requirements | TQL dependent<br>safety stanTool<br>Qualification<br>dard |

# Tool Chain Analysis in Standards: Classification

**IEC 61508**

7.4.4.5 An **assessment** shall be carried out for offline support tools in classes T2 und T3 to determine the level of reliance placed on the tools, and the **potential failure mechanisms** of the tools that may affect the executable software. Where such failure mechanisms are identified, **appropriate mitigation measures** shall be taken

**DO 330**

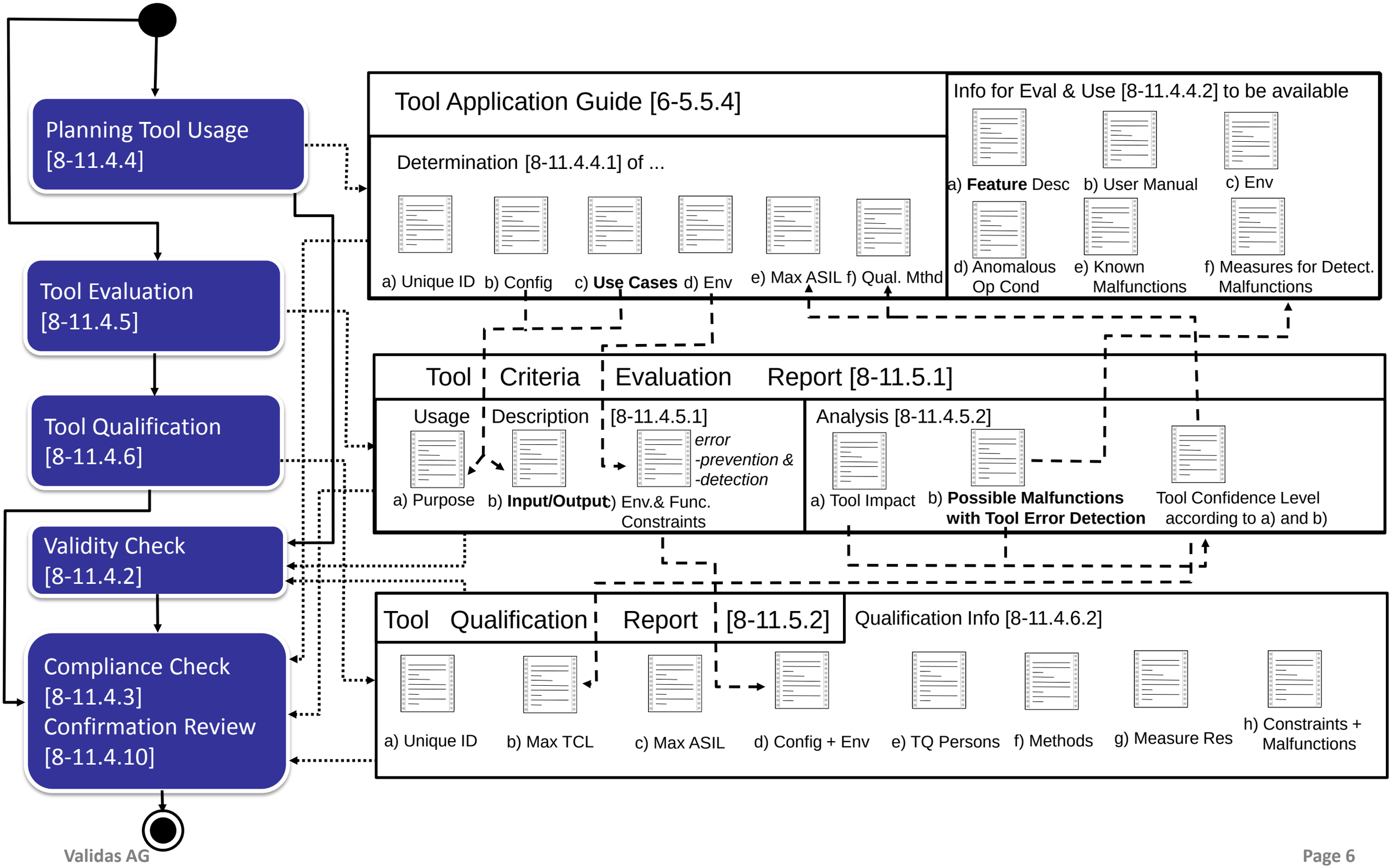
To reduce a tool`s qualification level, the reduction needs to be justified by performing a **tool use and impact analysis**. This analysis needs to evaluate the overall use of the tool in the development process and its impact on the software beeing produced.

**TOOL CHAIN ANALYSIS USEFUL IN MAIN SAFETY STANDARDS**

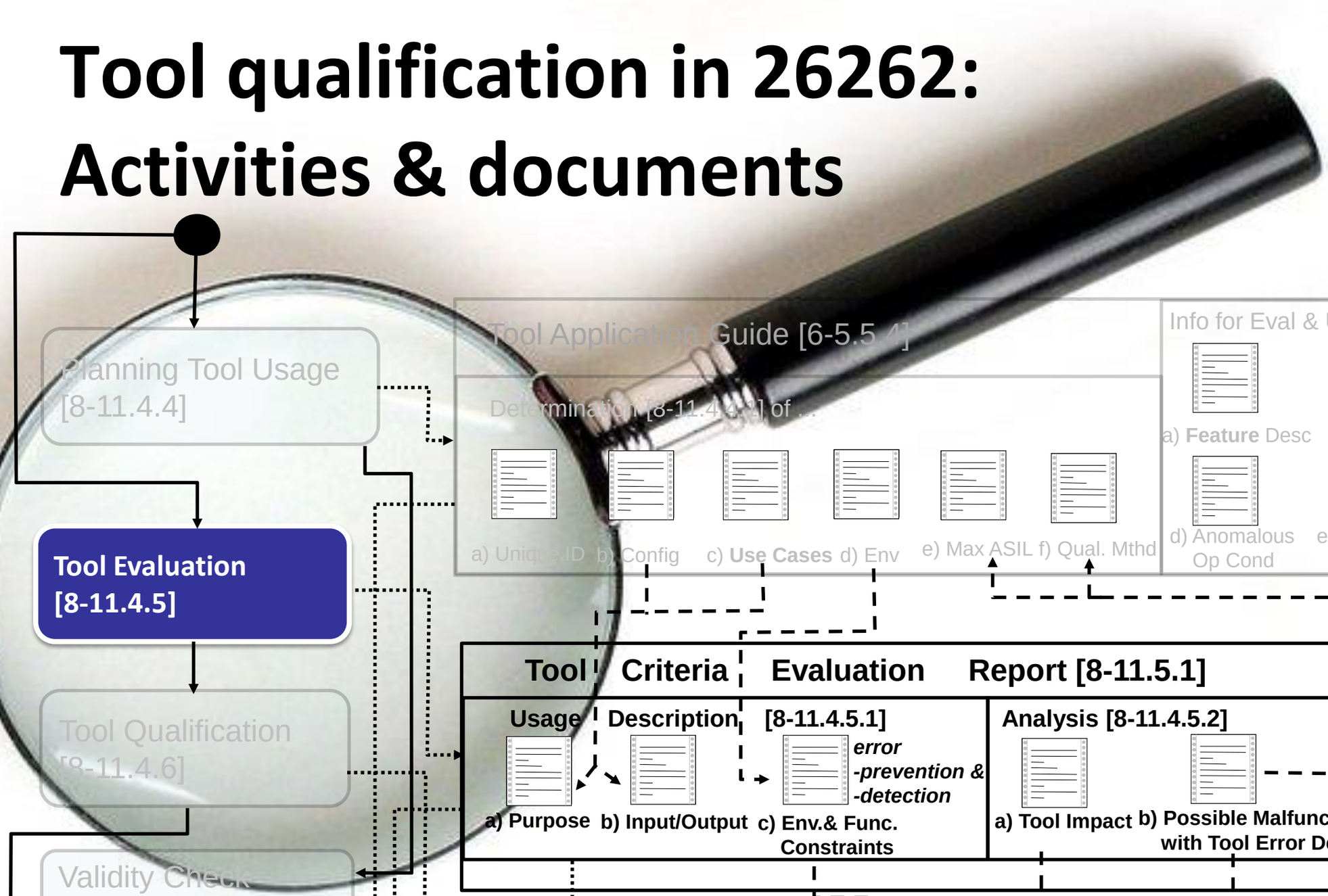
**ISO 26262**

| | | Tool error detection | | |
|---|---|---|---|---|
| | | TD1 | TD2 | TD3 |
| Tool impact | TI1 | TCL1 | TCL1 | TCL1 |
| | TI2 | TCL1 | TCL2 | TCL3 |

# Tool qualification in 26262: Activities & documents



# Tool qualification in 26262: Activities & documents



Planning Tool Usage [8-11.4.4]

**Tool Evaluation [8-11.4.5]**

Tool Qualification [8-11.4.6]

Validity Check [8-11.4.2]

Compliance Check [8-11.4.3] Confirmation Review [8-11.4.10]

Tool Application Guide [6-5.5.4]

Determination [8-11.4.4.1] of ...

a) Unique ID | b) Config | c) **Use Cases** | d) Env | e) Max ASIL | f) Qual. Mthd

Info for Eval & Use [8-11.4.4.2] to be available

a) **Feature** Desc | b) User Manual | c) Env | d) Anomalous Op Cond | e) Known Malfunctions | f) Measures for Detect. Malfunctions

**Tool Criteria Evaluation Report [8-11.5.1]**

**Usage Description [8-11.4.5.1]**

a) Purpose | b) Input/Output | c) Env.& Func. Constraints | *error -prevention & -detection*

**Analysis [8-11.4.5.2]**

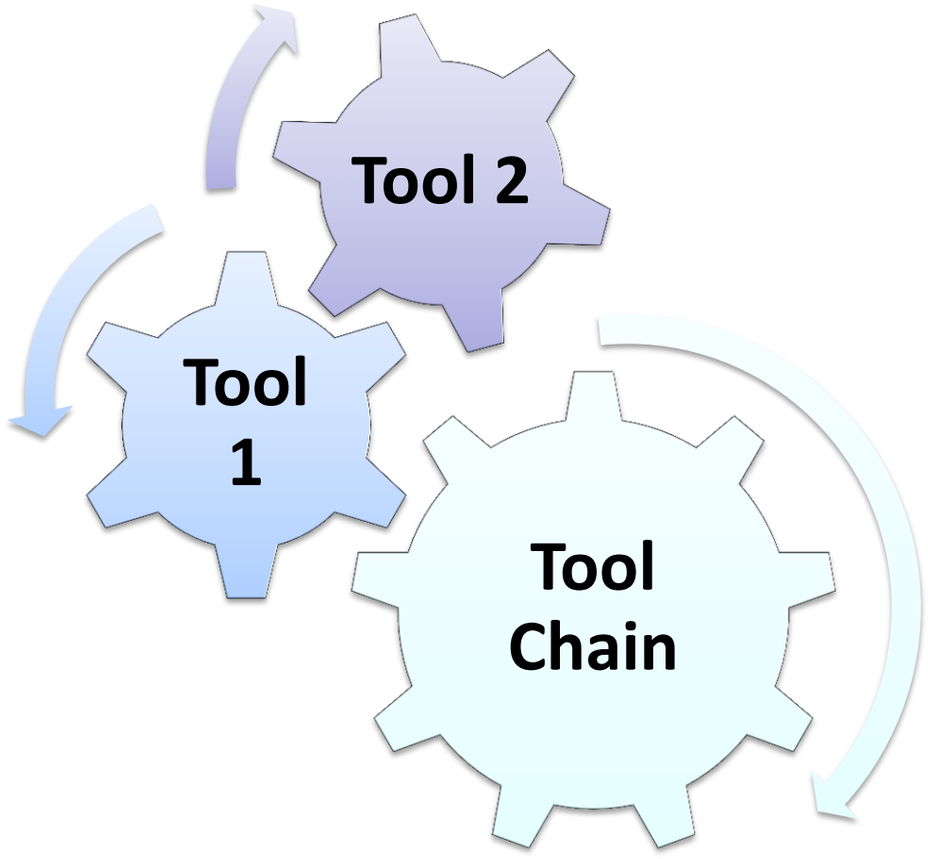
a) Tool Impact | b) Possible Malfunctions with Tool Error Detection | Tool Confidence Level according to a) and b)

Tool Qualification Report [8-11.5.2]

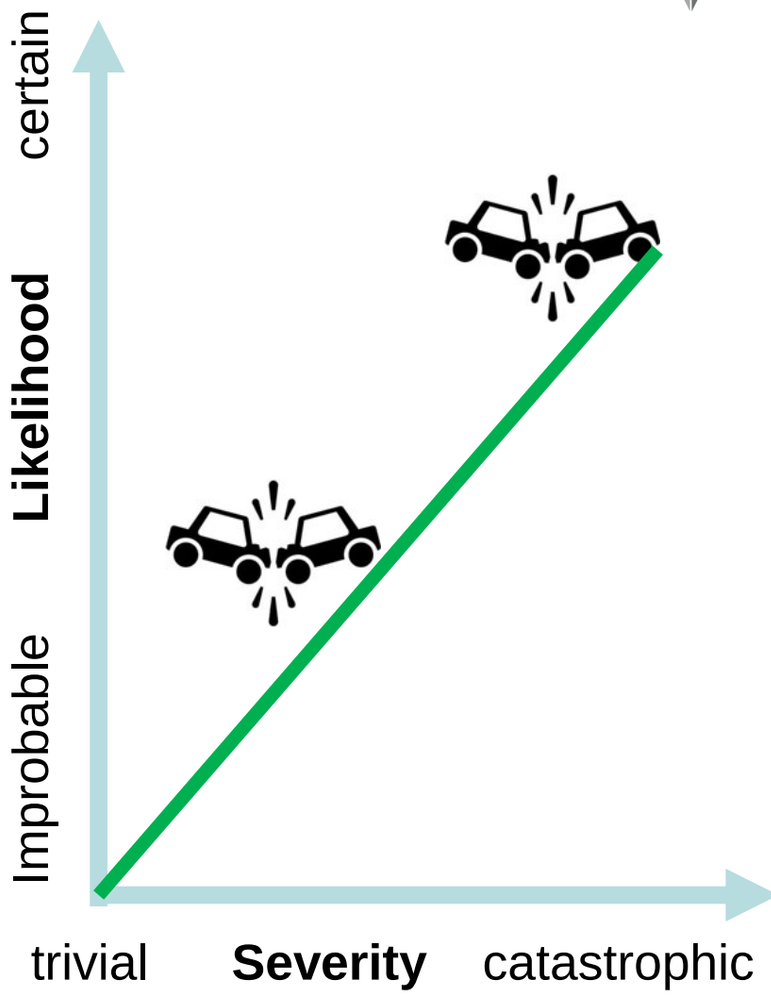
Qualification Info [8-11.4.6.2]

a) Unique ID | b) Max TCL | c) Max ASIL | d) Config + Env | e) TQ Persons | f) Methods | g) Measure Res | h) Constraints + Malfunctions

# Tool Evaluation




**RISK**




**Tool confidence Level (TCL)**

**ASIL Level**

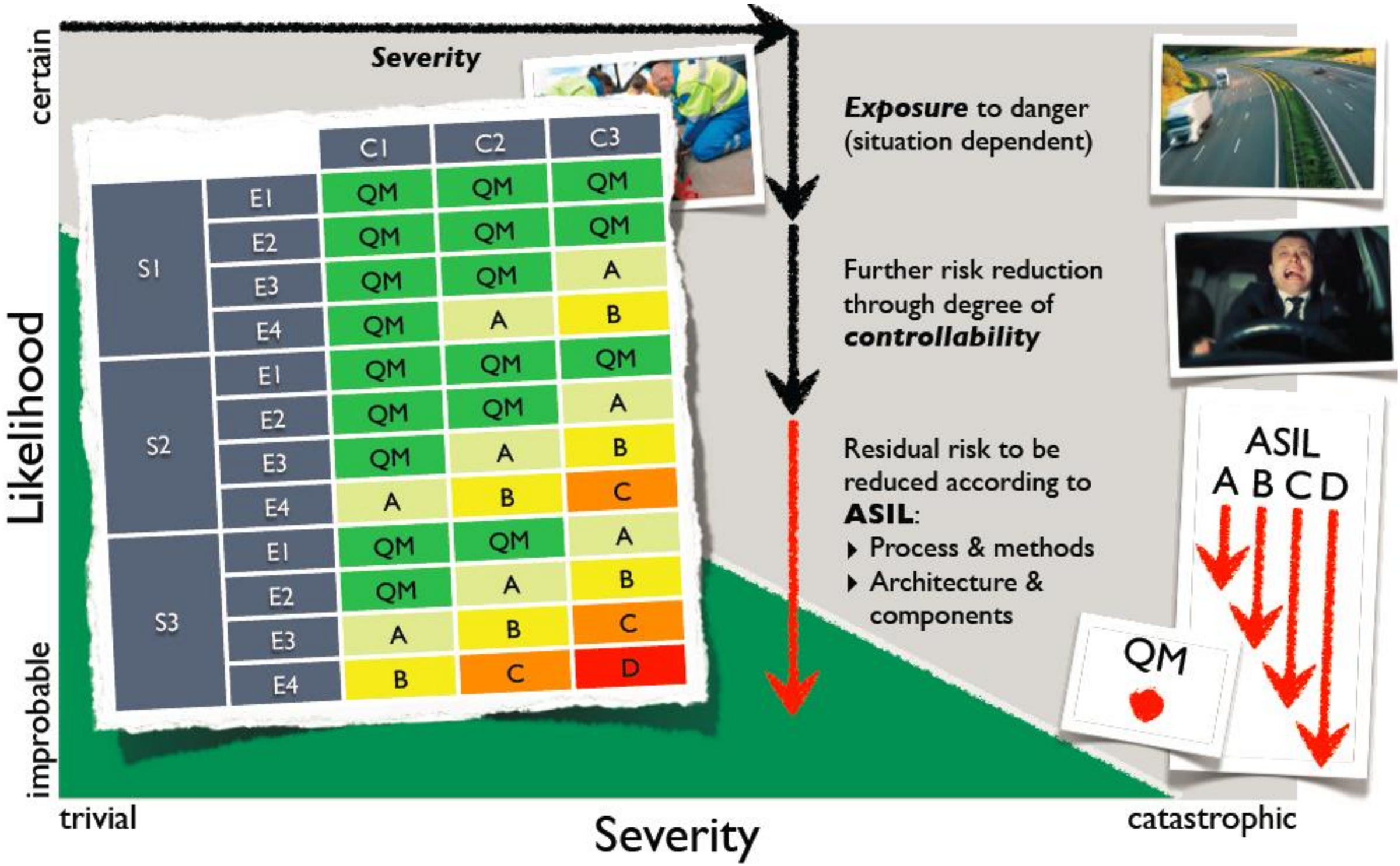
**ISO 26262 Keywords**

**Tools <-> Riskassessment**

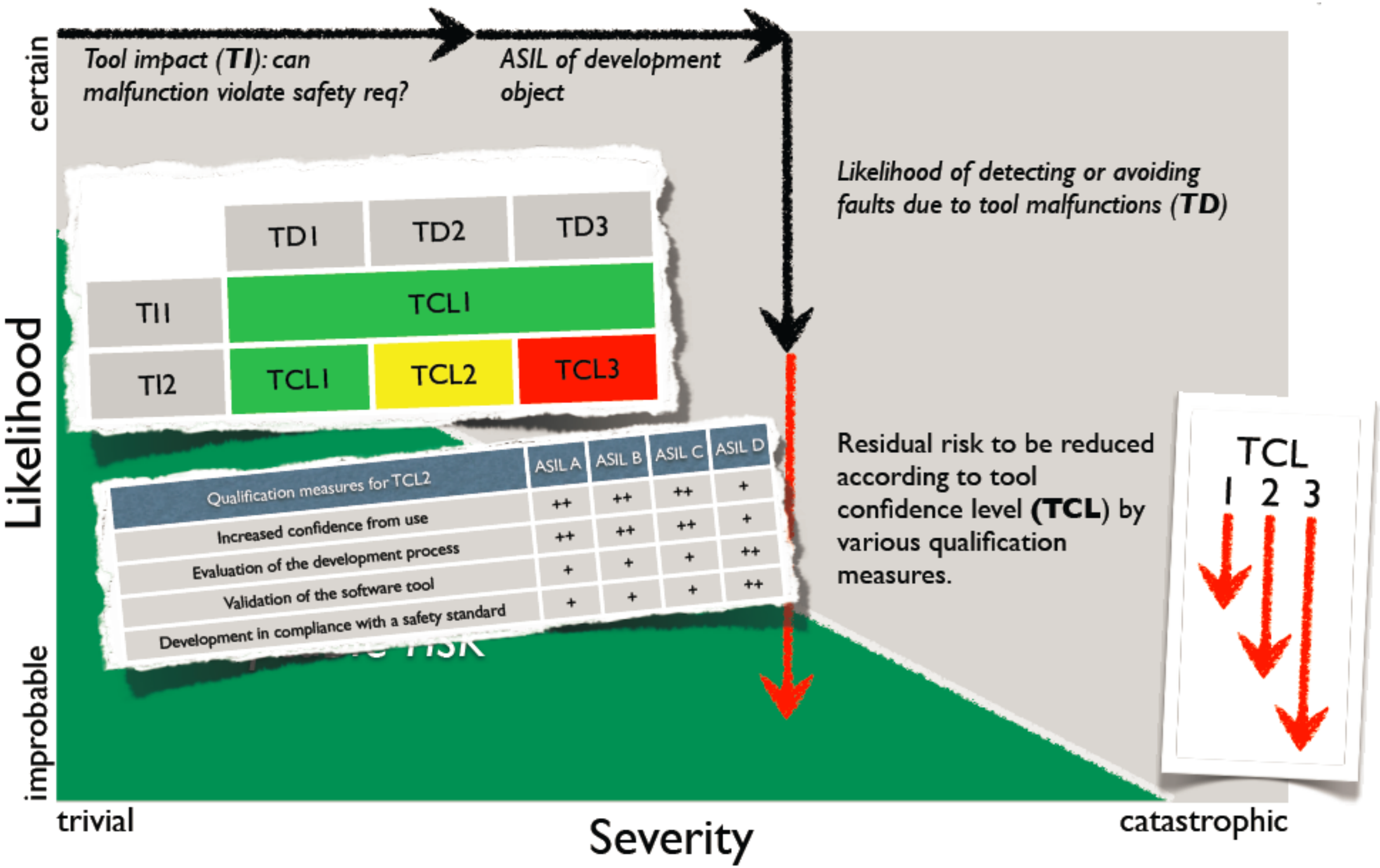
**Tool Impact (TI)**

**Tool Detection (TD)**

# Risk Assessment of Functions



# Assessment of Risk from Tool Use



# Classification & Qualification

## Classification

In ISO 26262, tool classification depends on tool *use*

**Tool Error Detection**

Can tool malfunction be detected or avoided?

**Tool Impact**

Is tool relevant to safety?

| | high | medium | otherwise |
|---|---|---|---|
| no | TI1 | TCL1 | |
| yes | TI2 | TCL1 | TCL2 | TCL3 |

**Tool Confidence Level**

How much confidence must we have that the tool functions correctly?

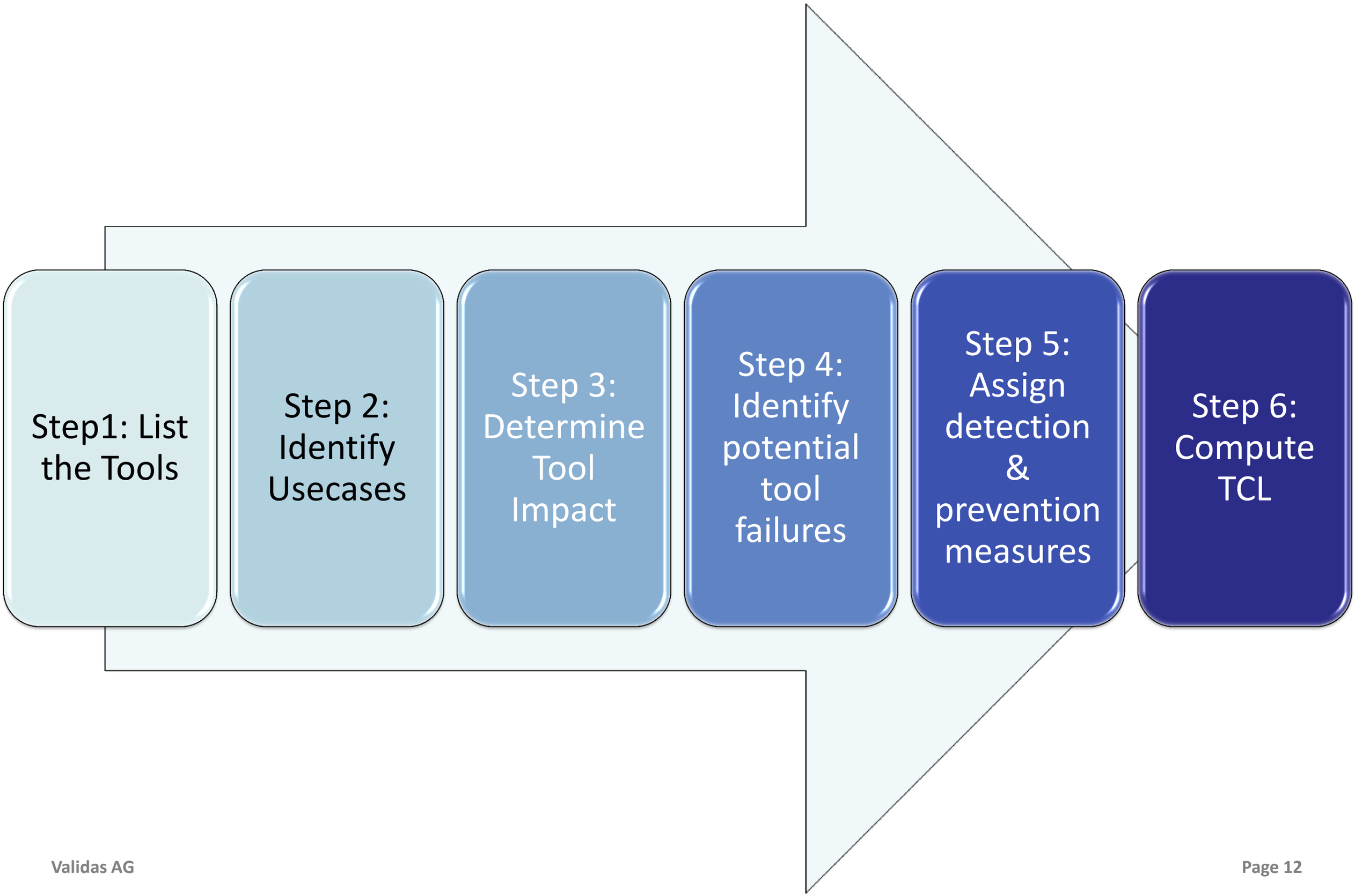
## Qualification

**Tool Qualification**

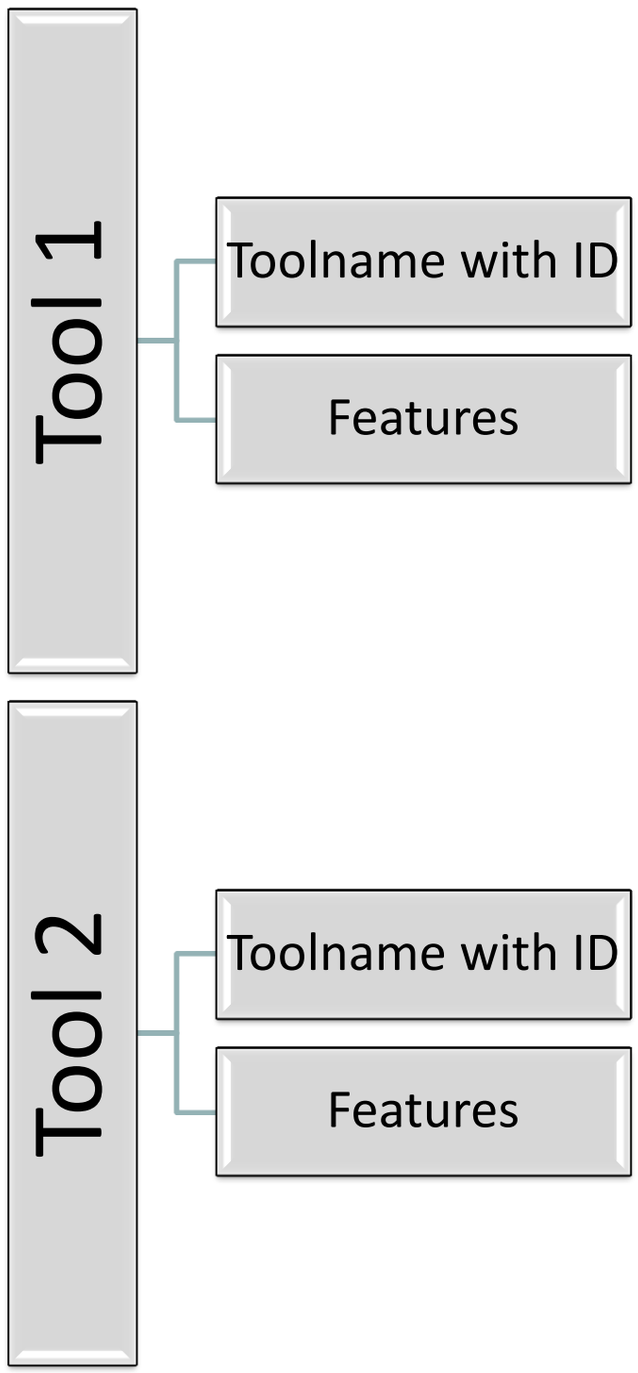
- Confidence from use
- Evaluation of development process
- Tool validation
- Development according to safety standard

Tool qualification is ASIL-dependent and only needed for tools with high demands on confidence

# Tool Evaluation Process

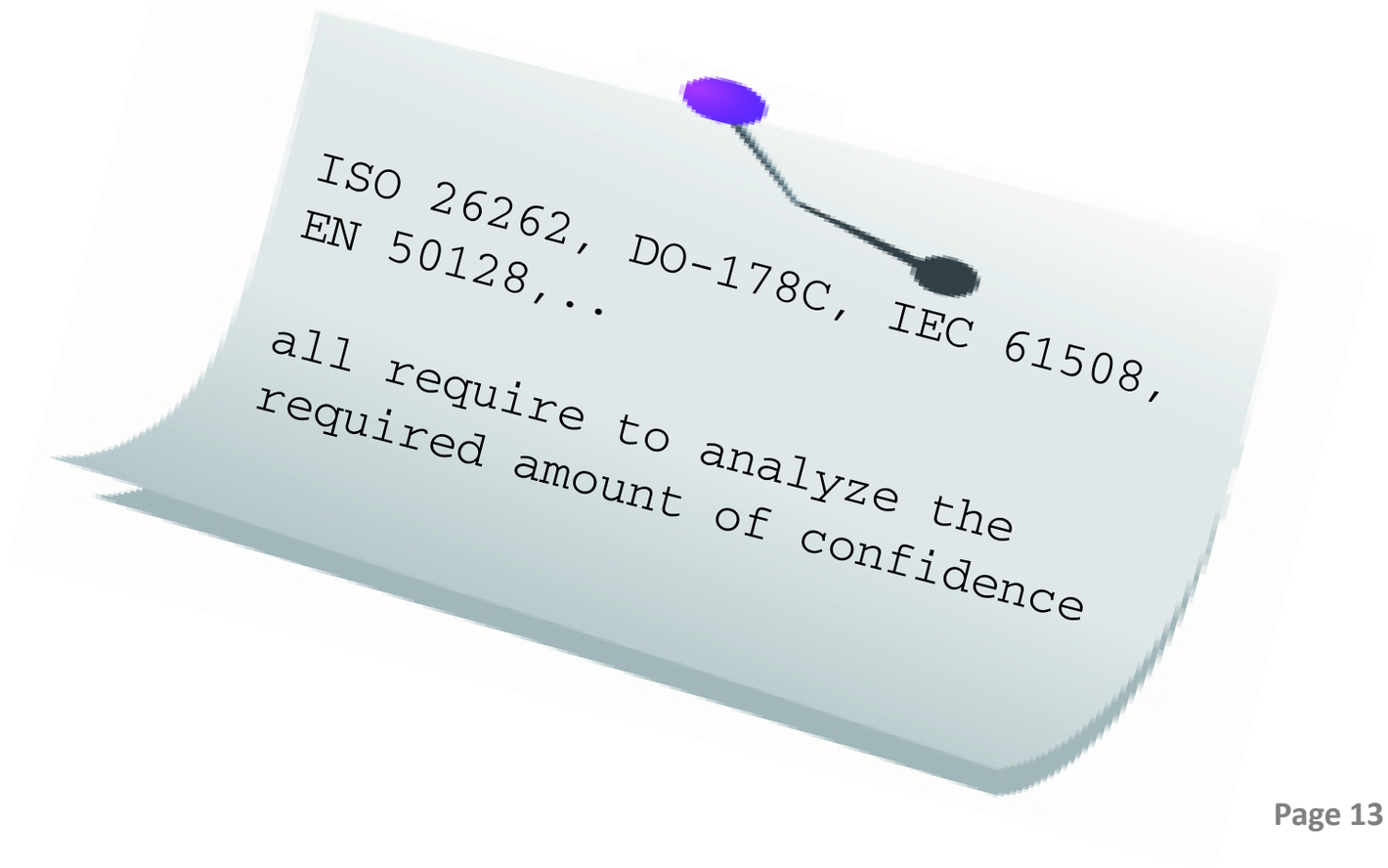


# Step 1: List the Tools




**WHICH TOOLS?**

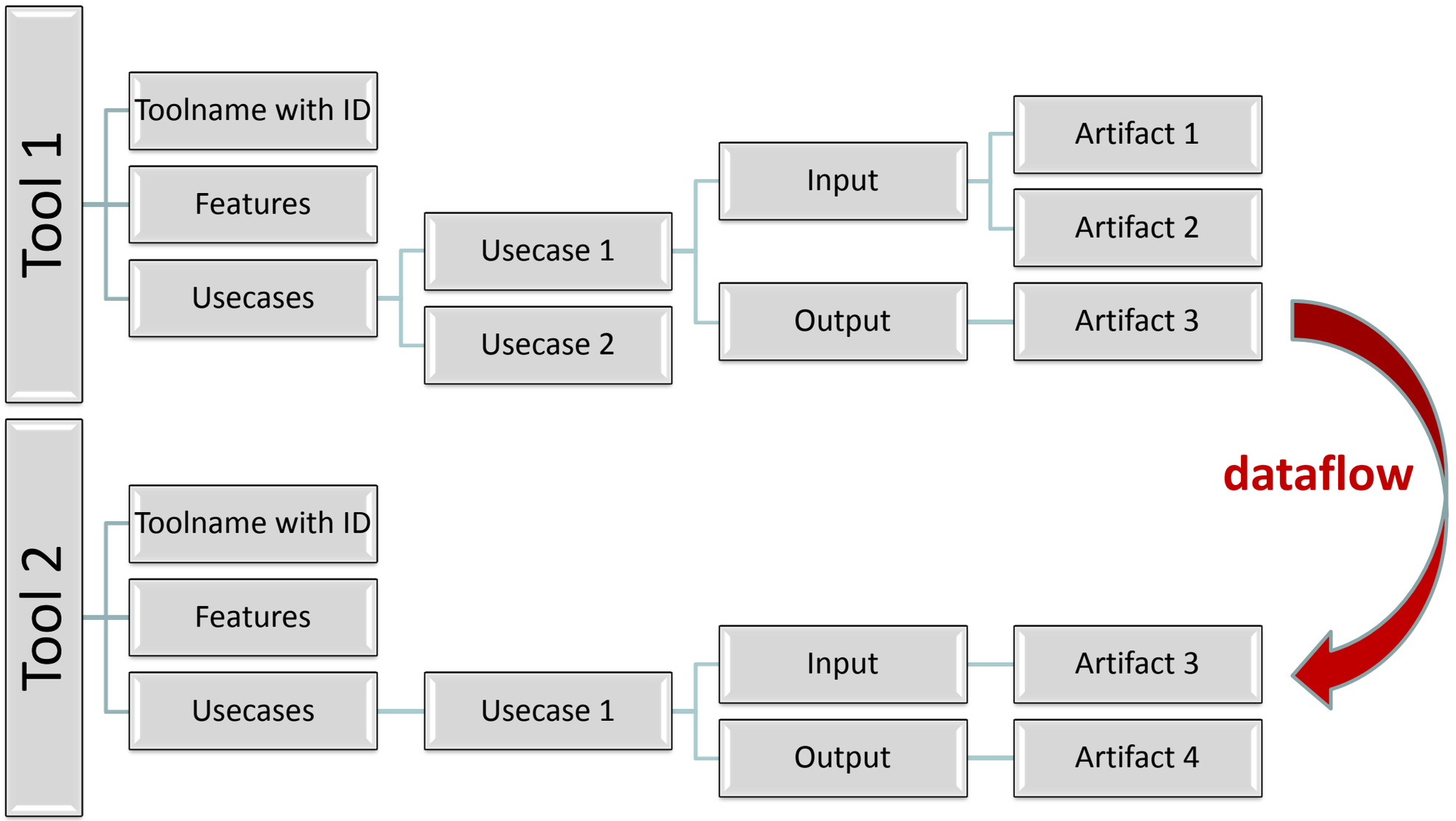
***All* tools *used* in development!**



# Step 2: Identify the Use-cases

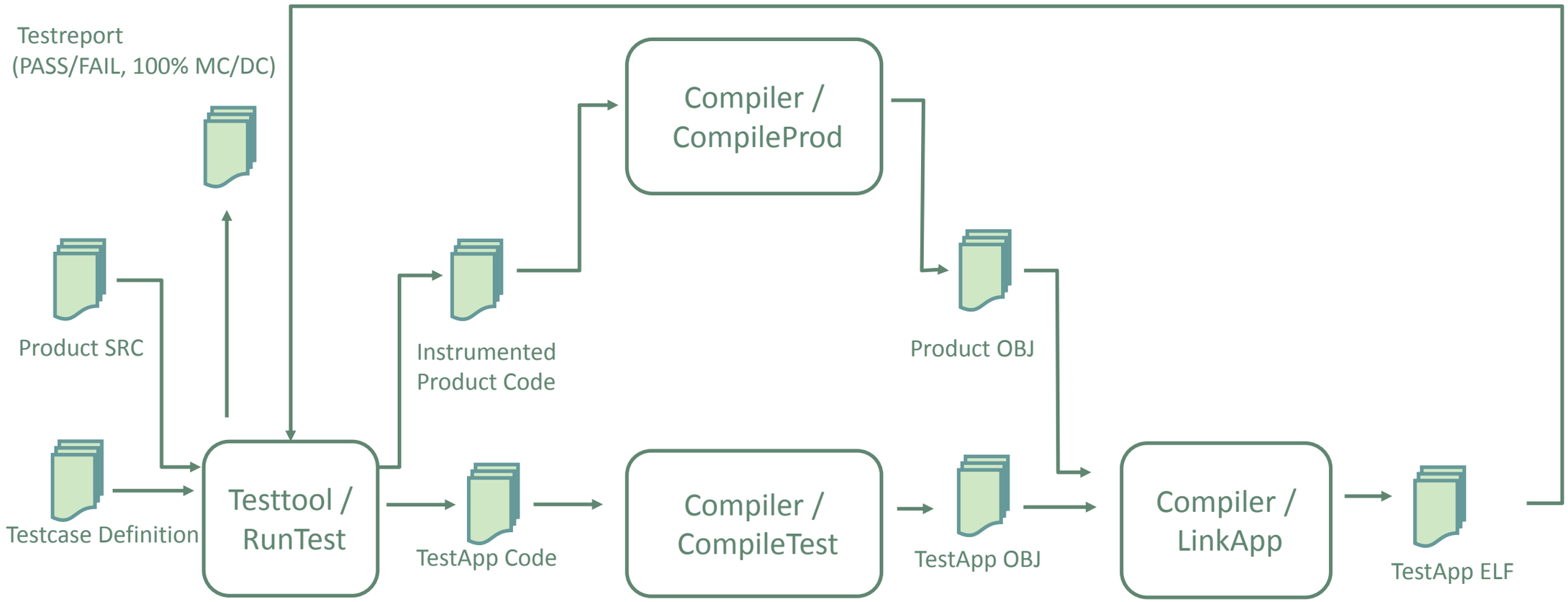
**WHY?**

- Eases Tool Impact analysis, as the usage of tool becomes clearer
- Defines the scope for the potential failure identification <= only used functions/features of the tool have to be classified



# Example: Testing Tool Chain

= define the dataflow
= define the tool chain




Tool Chain = by dataflow (inputs, outputs)
ordered list of tools

# Step 2: Identify the Use-cases

**HOW?**

Drawing tool chains is fun…

… but a lot of work,
… and error prone,
… and large chains don`t fit on a slide

…let‘s use a tool for that,
…let`s make a model!

TQS_Example.tca

Resource Set

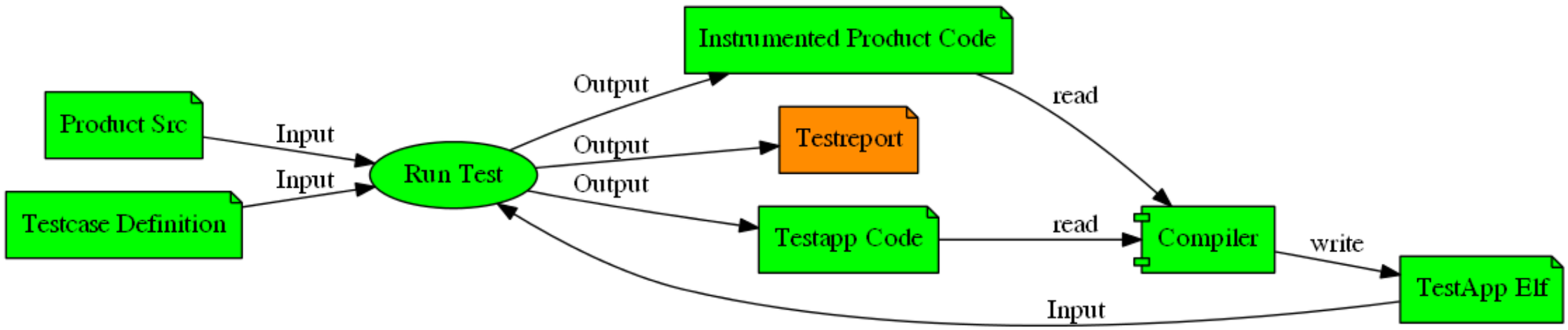
- file:/X:/ProjectData/Repositories/ISO26262_Toolqualifizie
  - Tool Chain Testing Tool Chain (TCL1)
    - Tool Compiler (TCL1)
      - Use Case Compiler:Compile Prod (TCL1)
      - Use Case Compiler:Compile Test (TCL1)
      - Use Case Compiler:Link App (TCL1)
    - Tool TestTool (TCL1)
      - Use Case TestTool:Run Test (TCL1)
  - Artifact Code
  - Artifact Code Object
  - Artifact Code Object:Product Obj
  - Artifact Code Object:Testapp Obj
  - Artifact Code:Product Src
  - Artifact Code:Testapp Code
  - Artifact Elf File
  - Artifact Elf File:TestApp Elf
  - Artifact Instrumented Product Code
  - Artifact Testcase Definition
  - Artifact Testreport

Properties

| Property | Value |
|---|---|
| Name | Link App |
| Description | Link SUT Object with Test Object |
| Impact | true |
| Inputs | Artifact Code Object:Product Obj, Artifact Code Object:Testapp Obj |
| Outputs | Artifact Elf File:TestApp Elf |
| Inputs Outputs | |
| Is Assumption | false |
| Virtual | false |
| Comment | |
| Required Features | |
| Long Description | |
| Called Use Cases | |
| Calling Use Cases | |
| Show Parts Errors | true |
| Show Part Of Errors | true |
| Deactivated | false |
| Variant | |
| Tests | |
| ID | |
| Number Of Executions | 1 |
| Internal Reference | |
| Number Of Setups | 1 |
| Attributes | |

# Step 2: Identify the Use-cases

From the model created graphs




Artifacts of Use Case: Run Test


Artifacts of Use Case: Link App

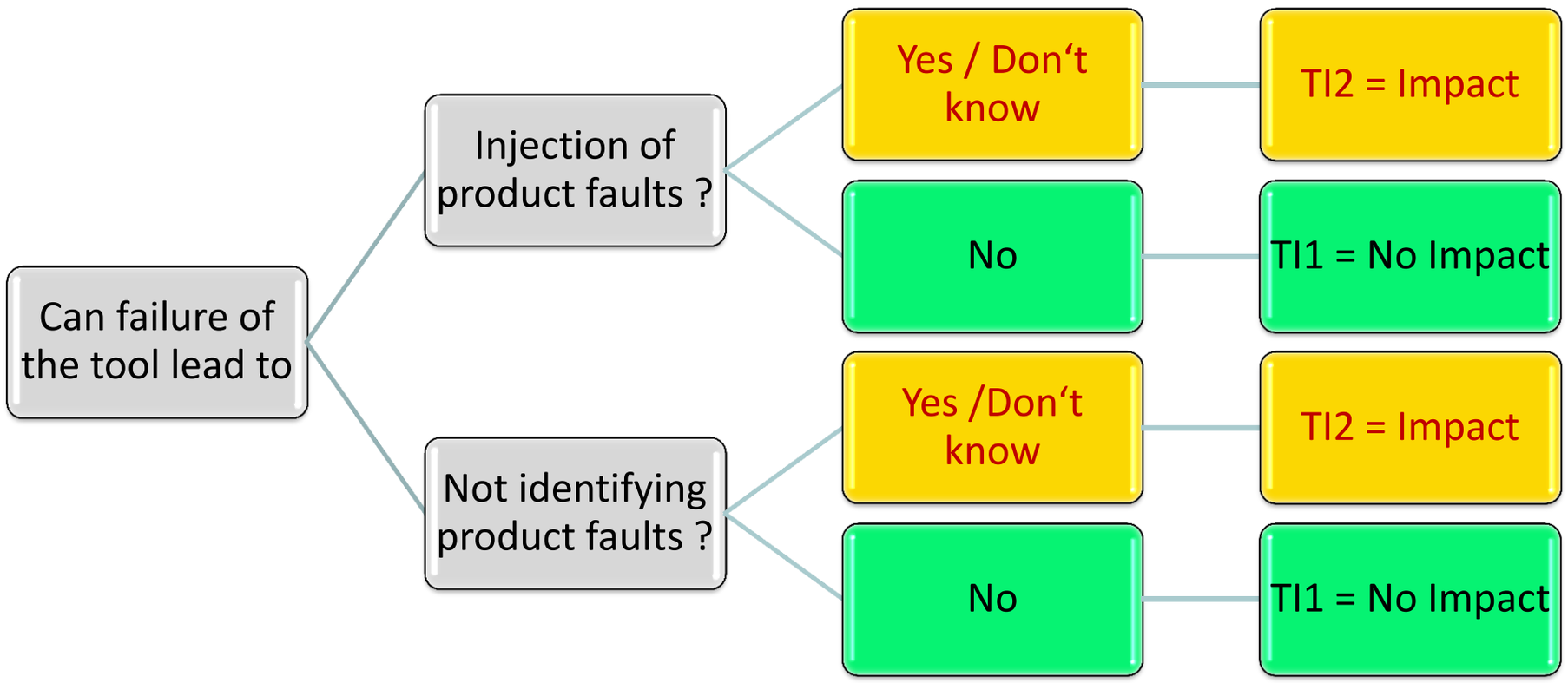
# Step 3: Determine Tool Impact

ISO 26262, DO-178C, IEC 61508, EN 50128,..

All require to determine tool impact on development of **all used** tools

HOW?

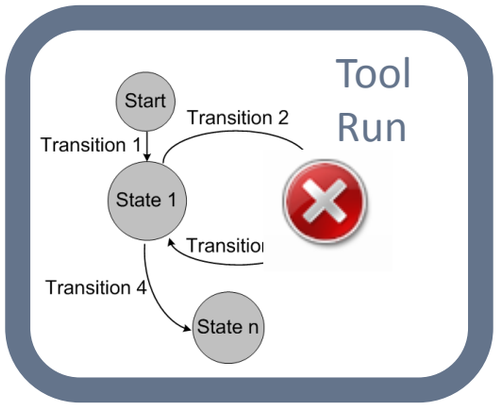
**Answer the following 2 questions for each tool:**



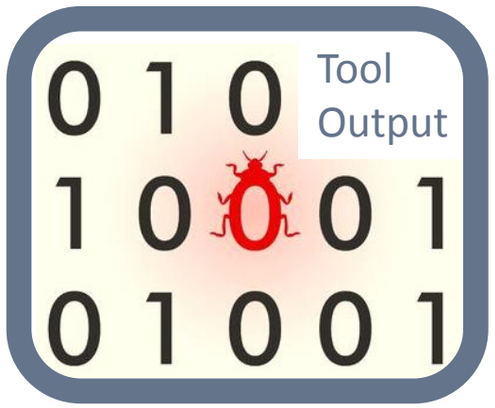
# Step 4: Identify potential Tool Failures

- **Terminology**


Tool Fault




Tool Error




Tool Failure

**Potential Tool Failure**

*fictive, abstract, wide, multiple instances*

Example: File Content Corruption

**Concrete Tool Failure**

*real, specific, narrow, single instance*

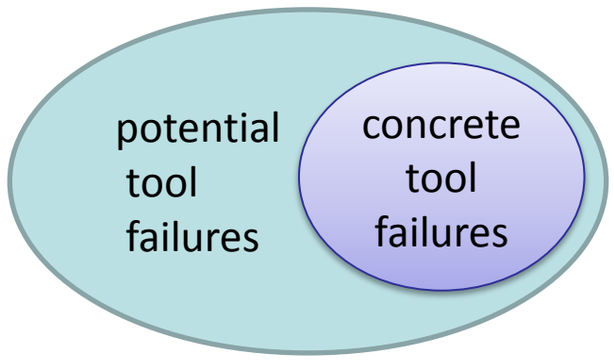
Example: Zip v7.3 corrupts files with more than 4gb size in compression method „Ultra“.

# Step 4: Identify potential Tool Failures

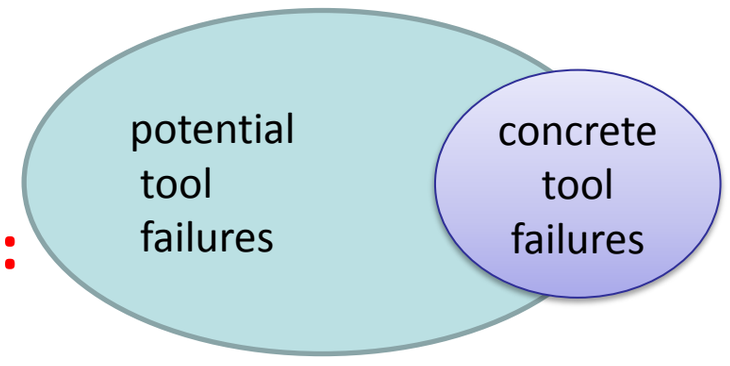
## Goals for Tool Failure Determination

- **Completeness:** **All concrete tool failures are subsumed by the determined potential tool failures.**

**safe case:**




**unsafe case:**

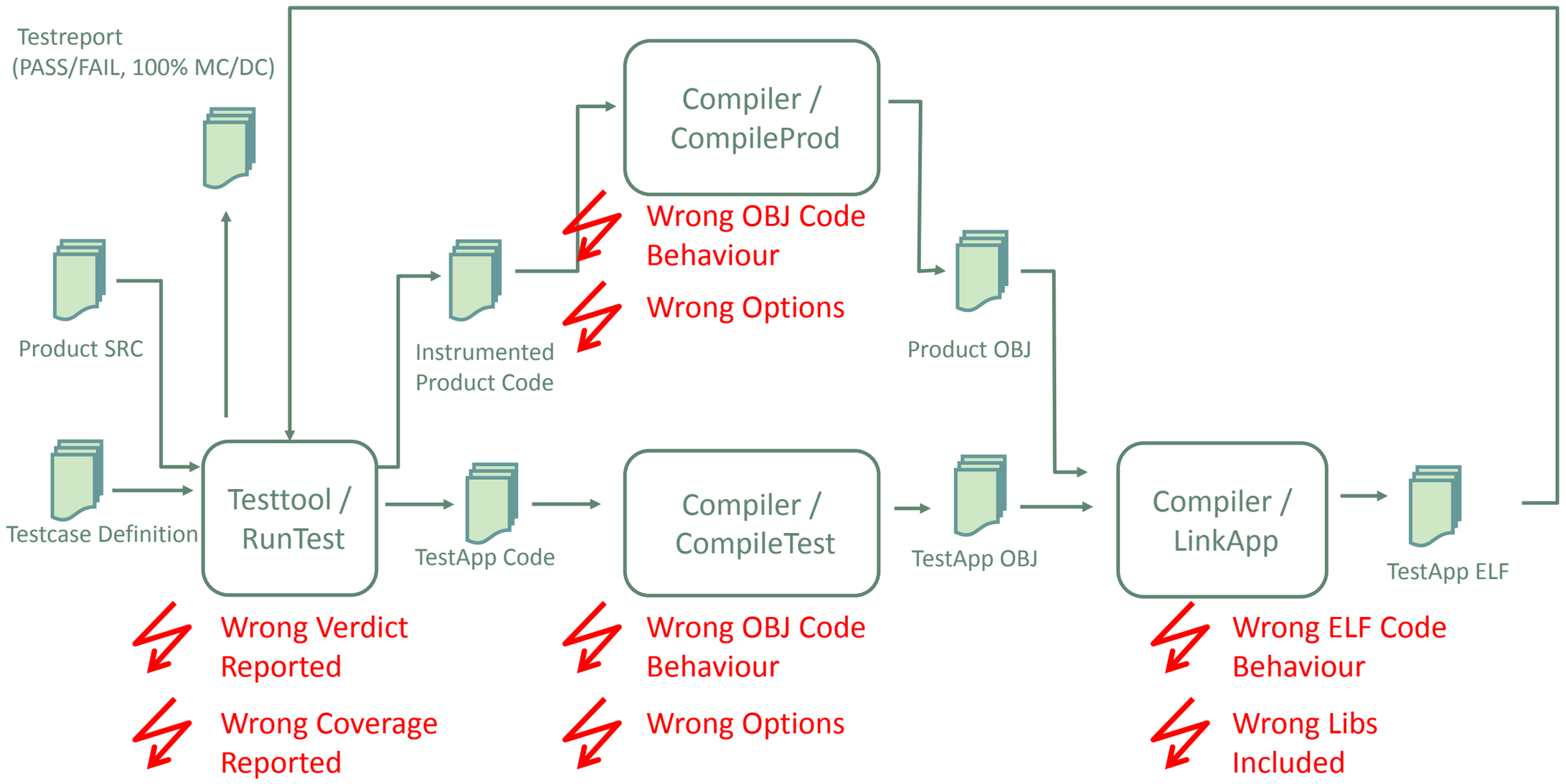



**Problem: How to measure? Number of concrete tool failures unknown!**

**Realistic Goals:**

- **Comprehensiveness:** **No blind spots.**
  **All potential tool failures can be found in principle.**
- **Uniformity:** **All tools analyzed, same method & intensity.**
- **Appropriate Abstraction:**
  **Failure descriptions neither too vague nor too detailed.**
- **Scalability:** **Effort is acceptable. Even for large tool chains.**

# Example: Testing Tool Chain

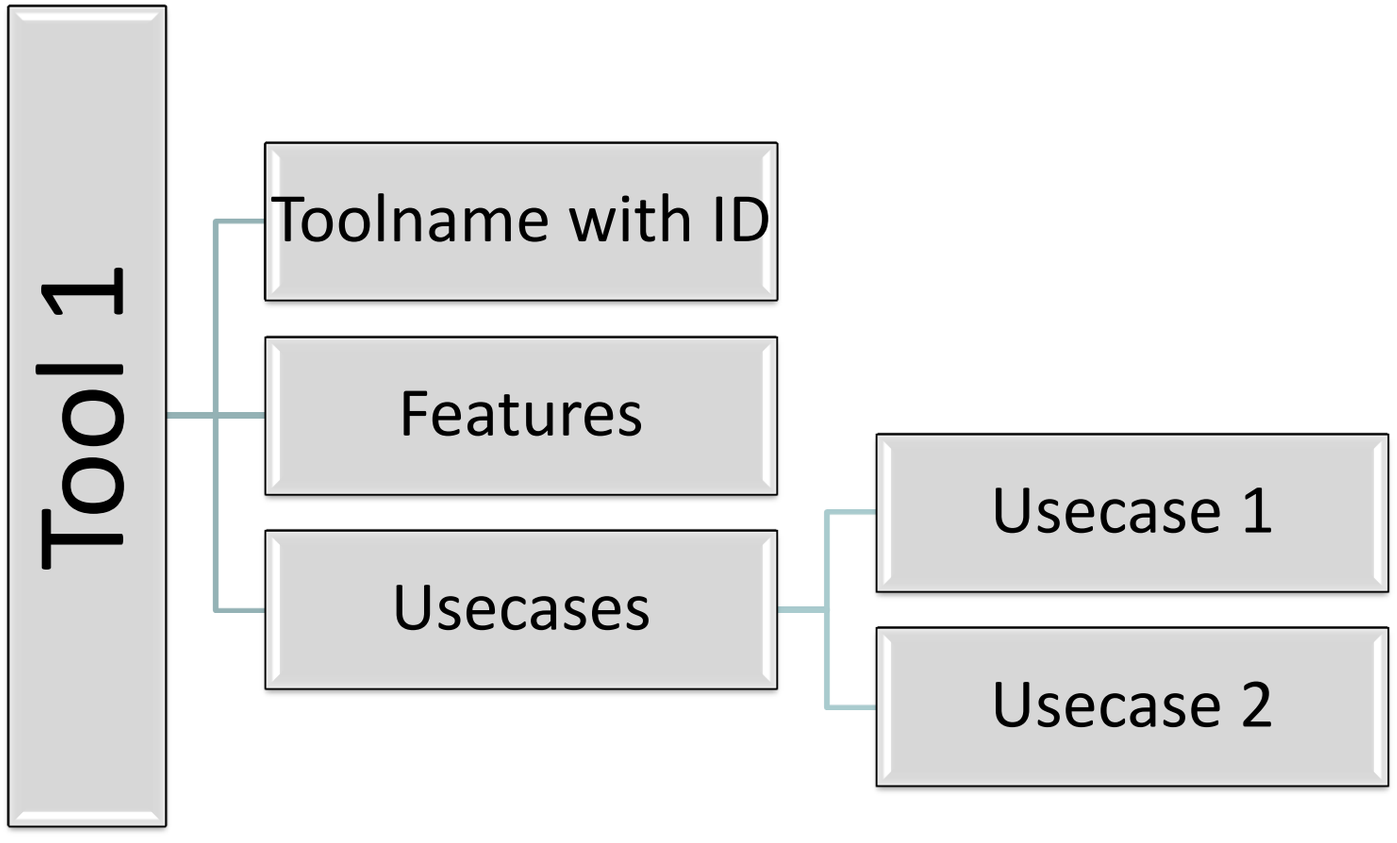


# Step 4: Identify potential Tool Failures

## HOW CAN I REACH THESE GOALS IN PRACTICE ?

**Scalability** **Uniformity**

**Our Solution: Modularity**

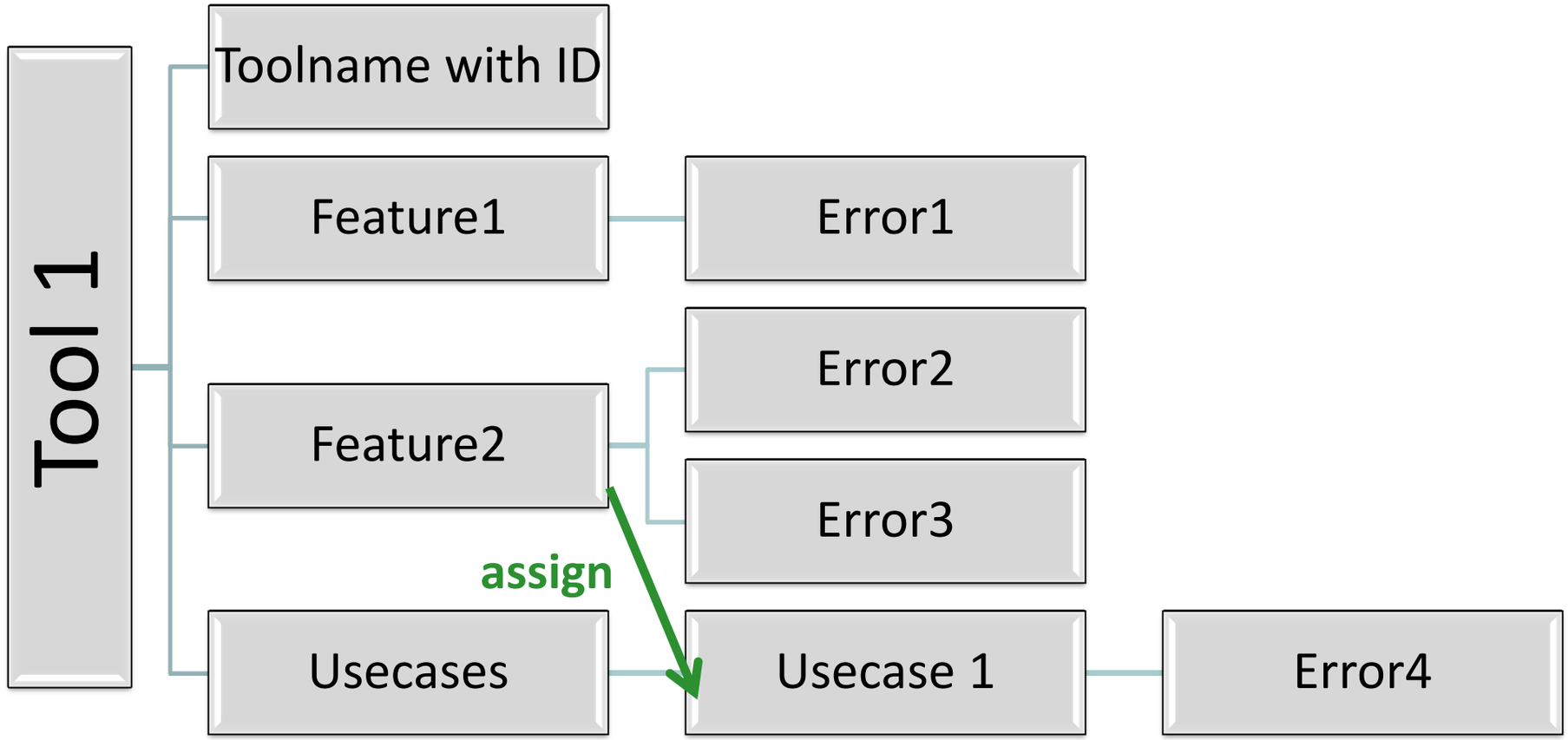


# Step 4: Identify potential Tool Failures

## HOW CAN I REACH THESE GOALS IN PRACTICE ?

**Scalability** **Uniformity**

**Our Solution: Modularity**

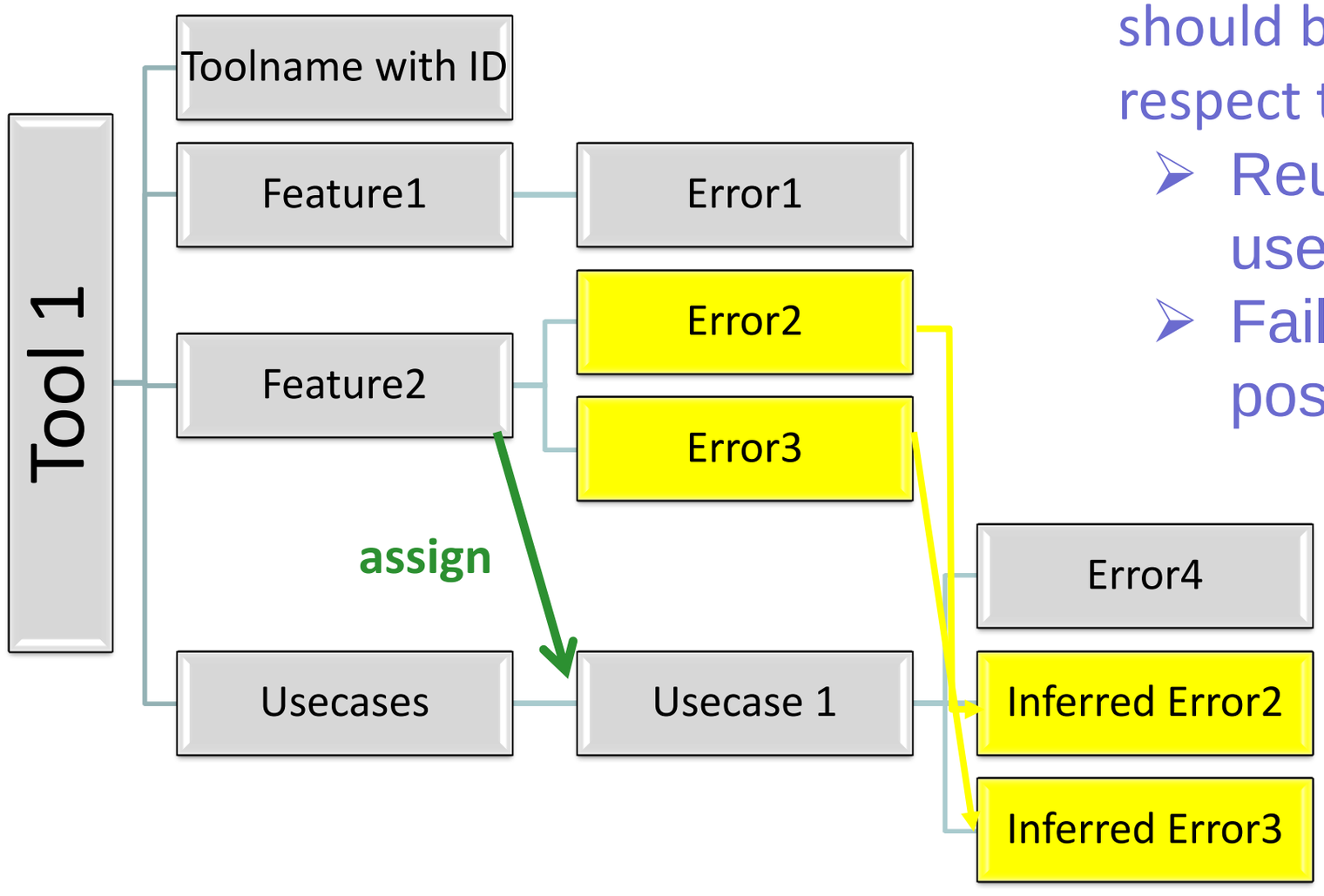


# Step 4: Identify potential Tool Failures

## HOW CAN I REACH THESE GOALS IN PRACTICE ?

**Scalability** **Uniformity**

**Our Solution: Modularity**




- Modularity of features should be defined in respect to
  - Reusability in usecases
  - Failure detection possibilities

# Step 4: Identify potential Tool Failures

## HOW CAN I REACH THESE GOALS IN PRACTICE ?

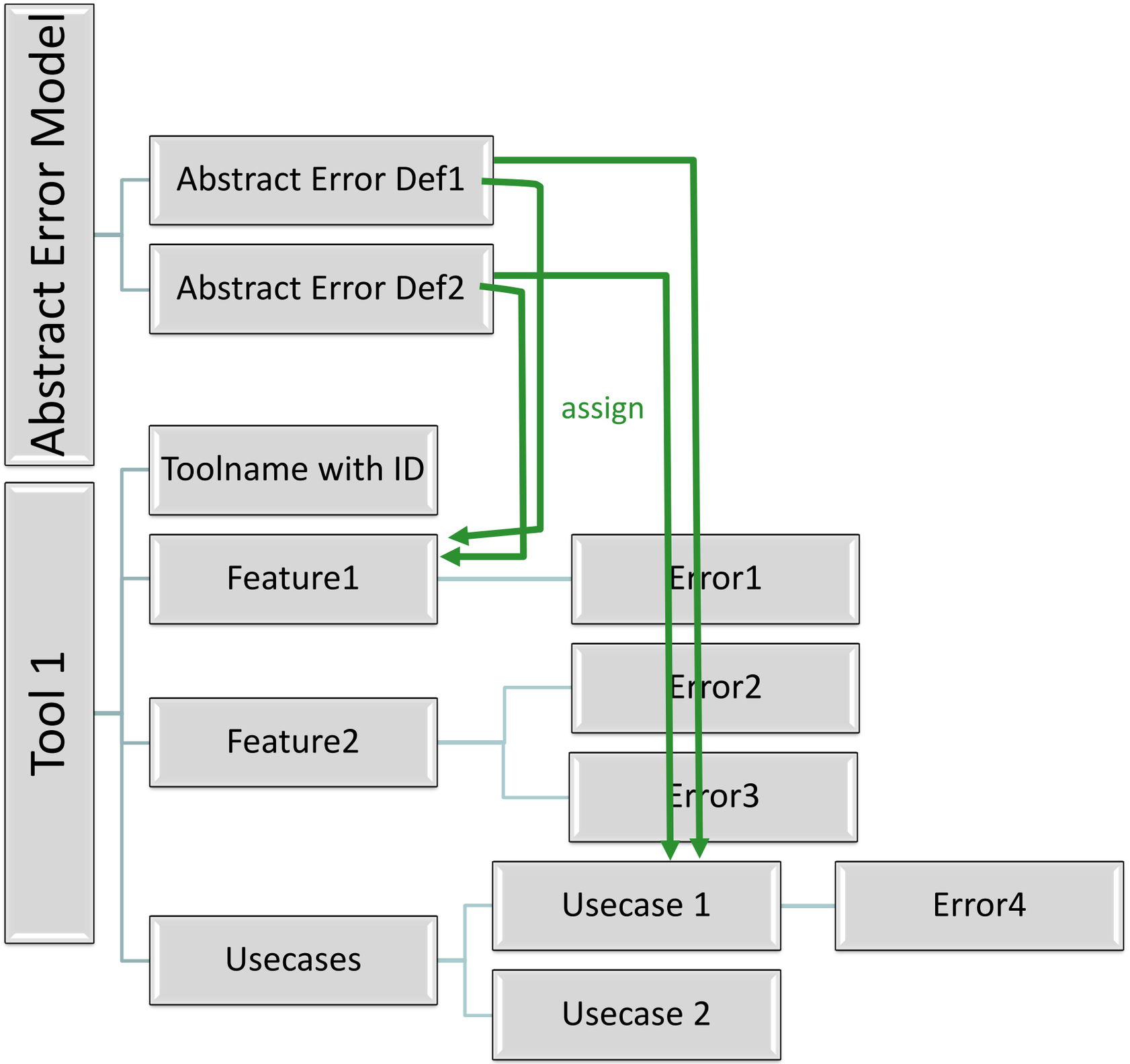
**Comprehensiveness** **Appropriate Abstraction**

**Our Solution: abstract error model -> Libraries of abstract errors**

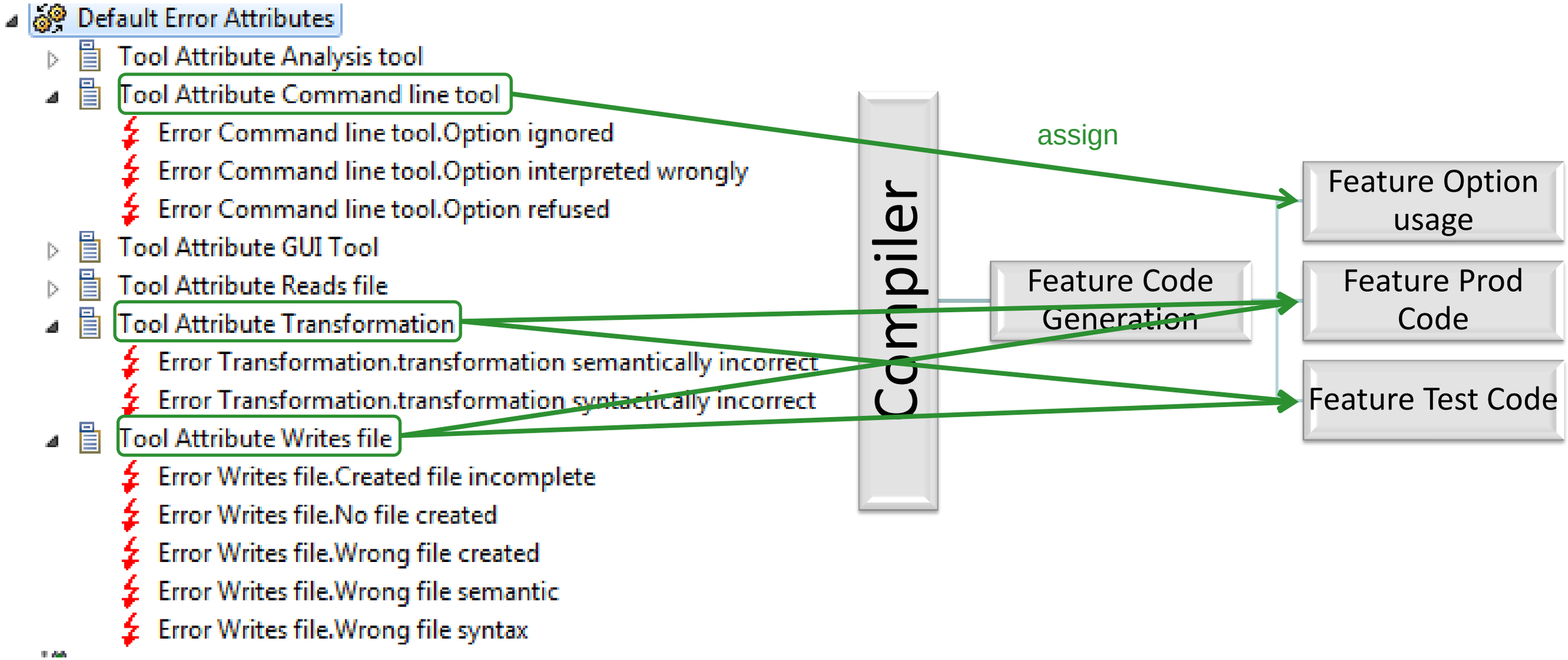
- Default Error Attributes
  - Tool Attribute Analysis tool
    - Error Analysis tool.Analysis result false-positiv
    - Error Analysis tool.Analysis result wrong
  - Tool Attribute Command line tool
    - Error Command line tool.Option ignored
    - Error Command line tool.Option interpreted wrongly
    - Error Command line tool.Option refused
  - Tool Attribute GUI Tool
    - Error GUI Tool.GUI wrong
  - Tool Attribute Reads file
    - Error Reads file.File uncorrectly read
    - Error Reads file.wrong file read
  - Tool Attribute Transformation
    - Error Transformation.transformation semantically incorrect
    - Error Transformation.transformation syntactically incorrect
  - Tool Attribute Writes file
    - Error Writes file.Created file incomplete
    - Error Writes file.No file created
    - Error Writes file.Wrong file created
    - Error Writes file.Wrong file semantic
    - Error Writes file.Wrong file syntax

- *One* error model describes possible tool errors in *every* tool
- Error model should be extensible (for new kinds of tools)
- Error model should cover all potential errors
- Error model should be acceptable (agreeable and simple to apply)

# Step 4: Identify potential Tool Failures



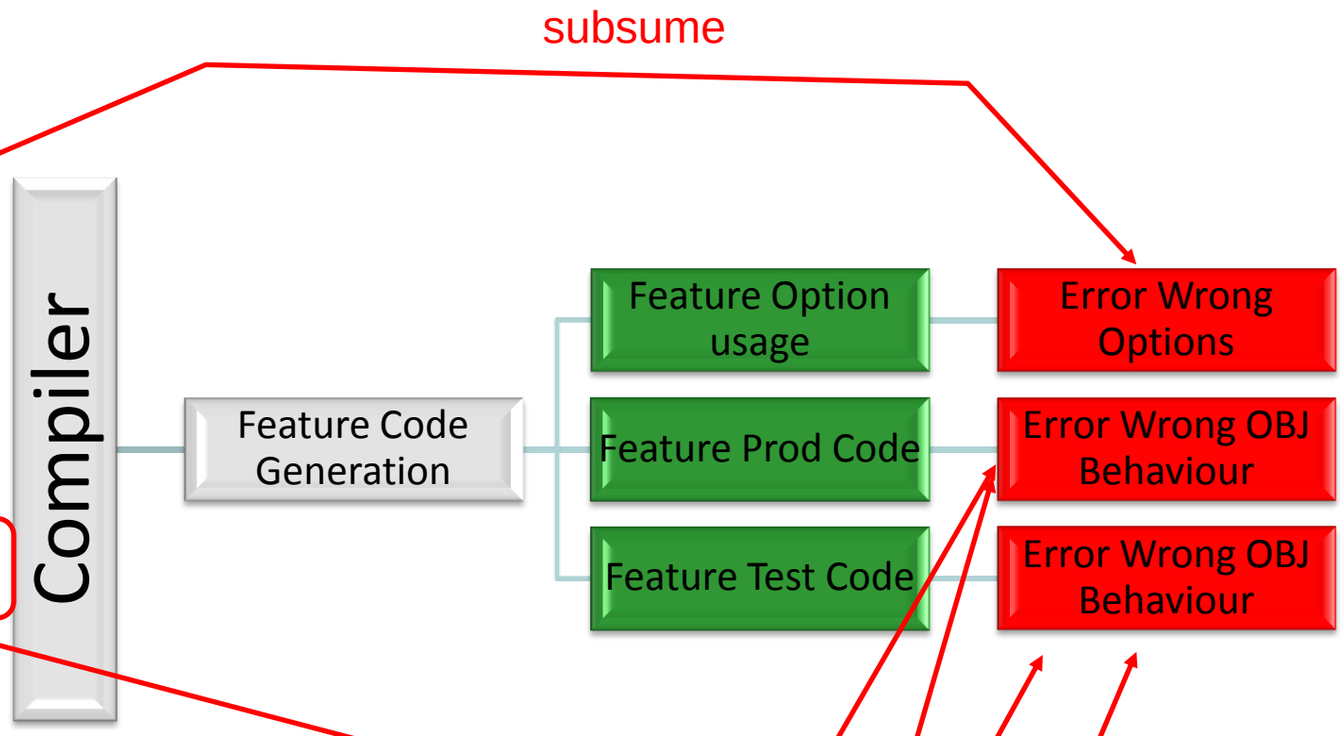
# Step 4: Identify potential Tool Failures



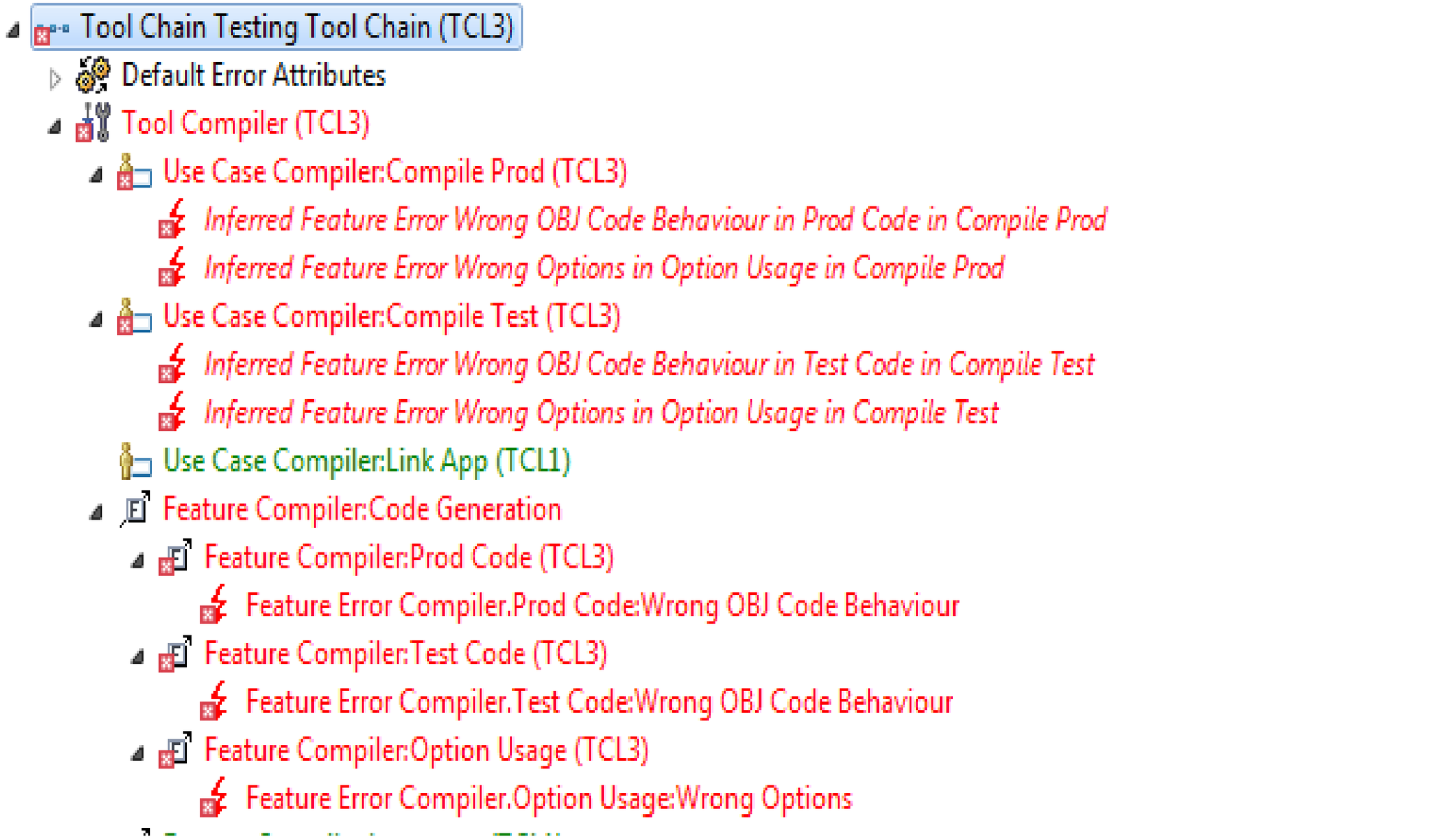
# Step 4: Identify potential Tool Failures

Subsume specific errors with abstract errors

- Default Error Attributes
  - Tool Attribute Analysis tool
  - Tool Attribute Command line tool
    - Error Command line tool.Option ignored
    - Error Command line tool.Option interpreted wrongly
    - Error Command line tool.Option refused
  - Tool Attribute GUI Tool
  - Tool Attribute Reads file
  - Tool Attribute Transformation
    - Error Transformation.transformation semantically incorrect
    - Error Transformation.transformation syntactically incorrect
  - Tool Attribute Writes file
    - Error Writes file.Created file incomplete
    - Error Writes file.No file created
    - Error Writes file.Wrong file created
    - Error Writes file.Wrong file semantic
    - Error Writes file.Wrong file syntax



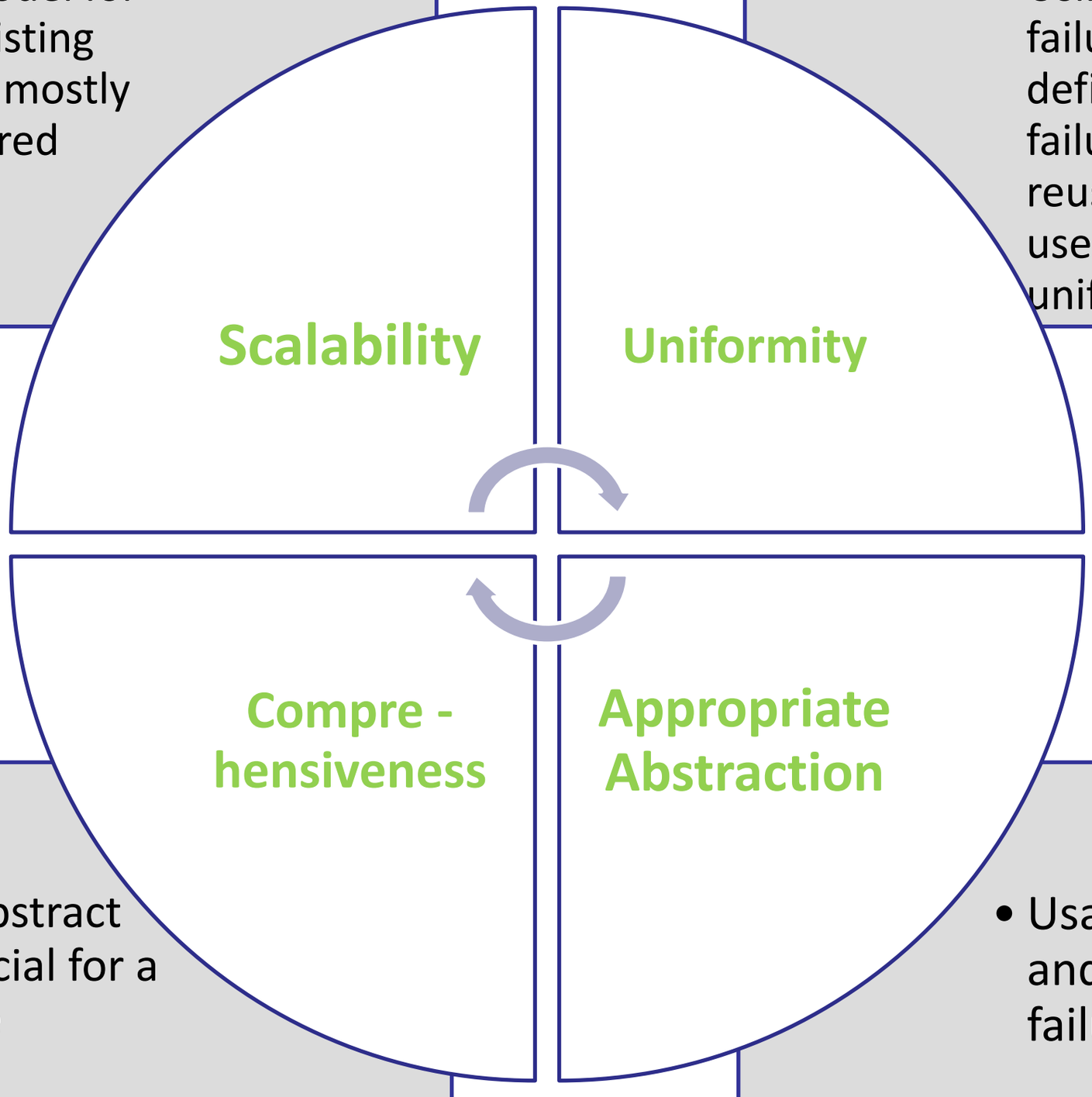
# Step 4: Identify potential Tool Failures



# Step 4: Identify potential Tool Failures

- Creating a TCA model for use cases with existing feature models is mostly selection of required features

- Using the abstract failure model for the definition of feature failures, which are reused in several usecases garantees uniformity

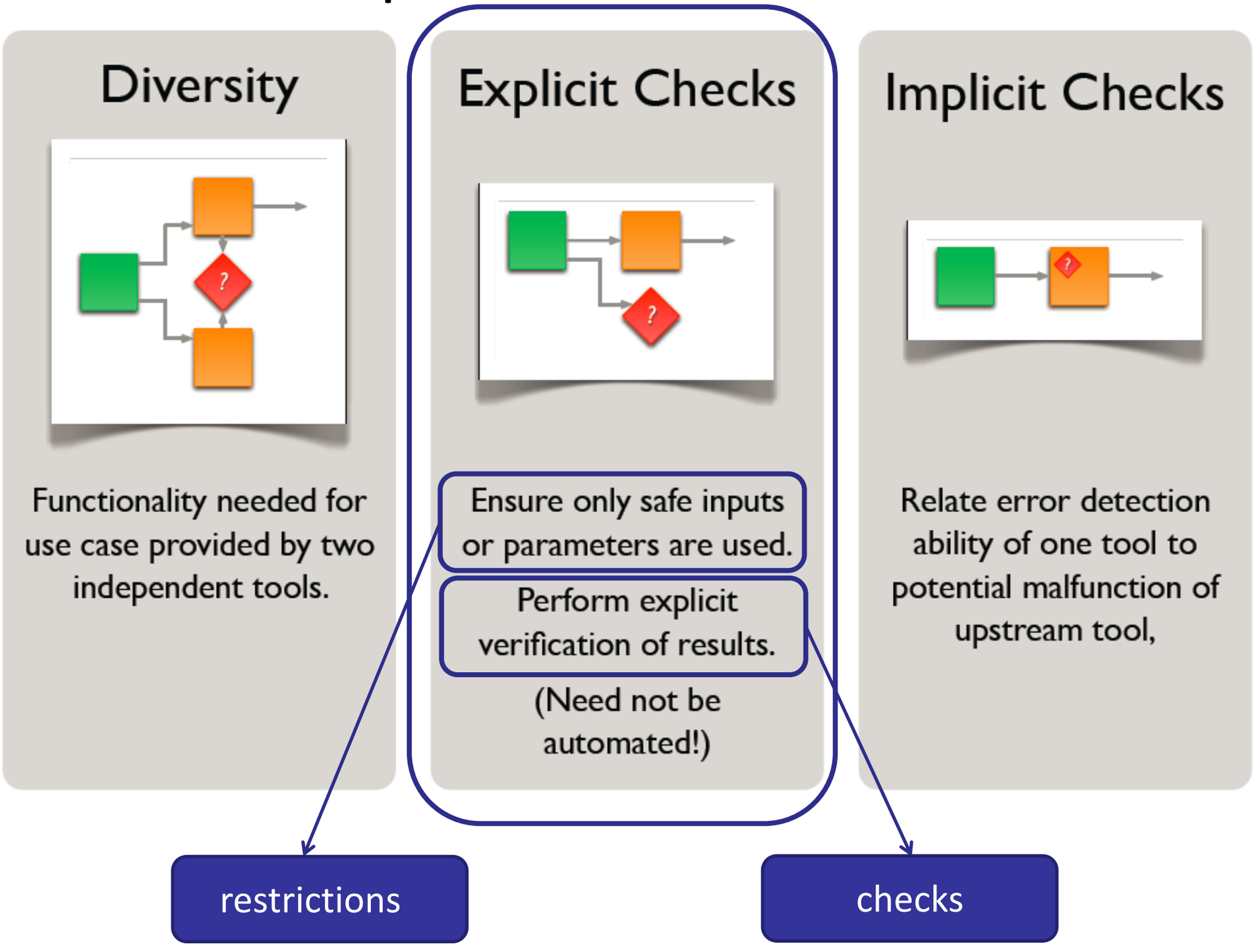



- Quality of the abstract error model crucial for a complete failure identification

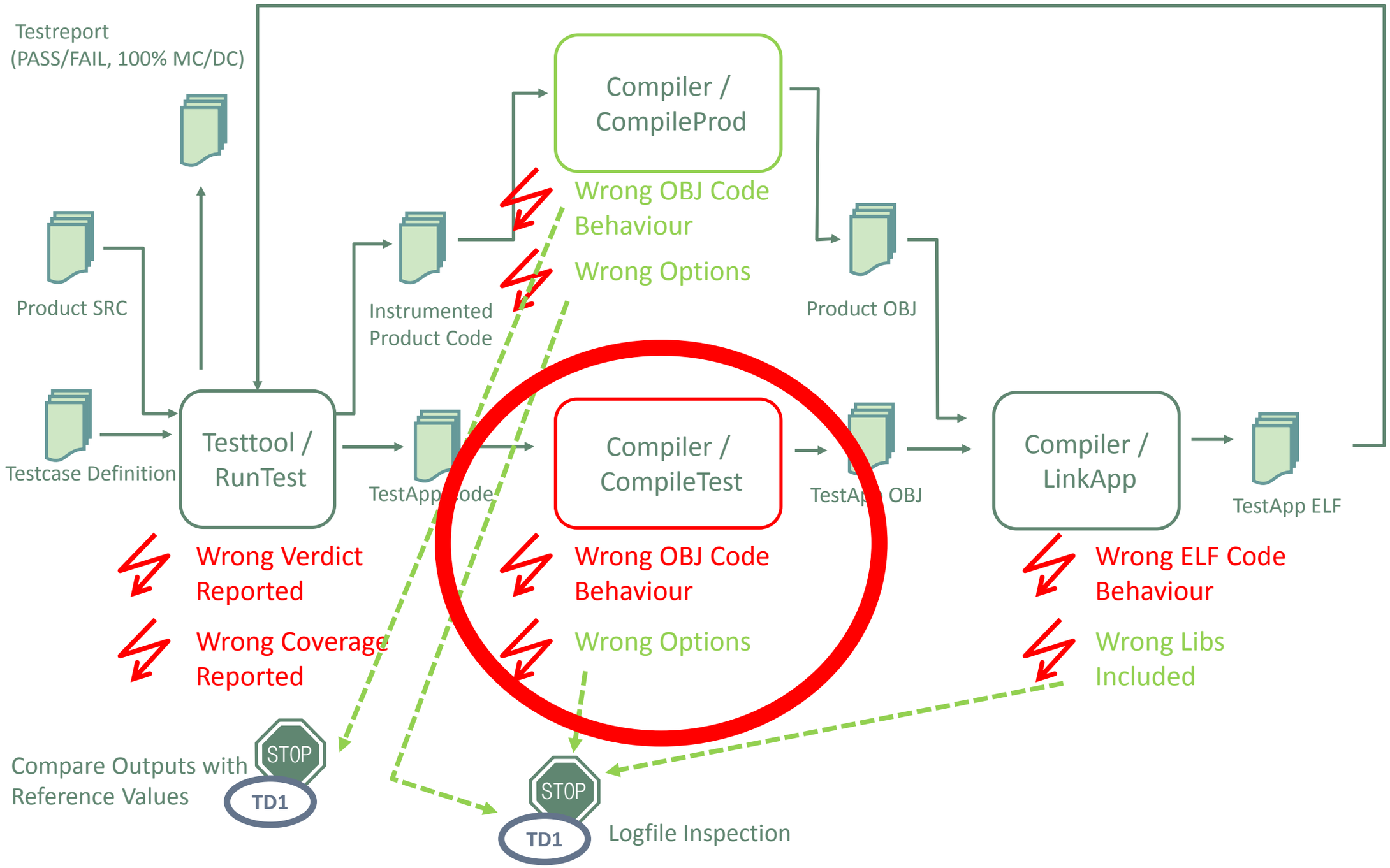
- Usage of an abstract and an concrete failure definition

# Step 5: Assign Detection & Prevention Measures

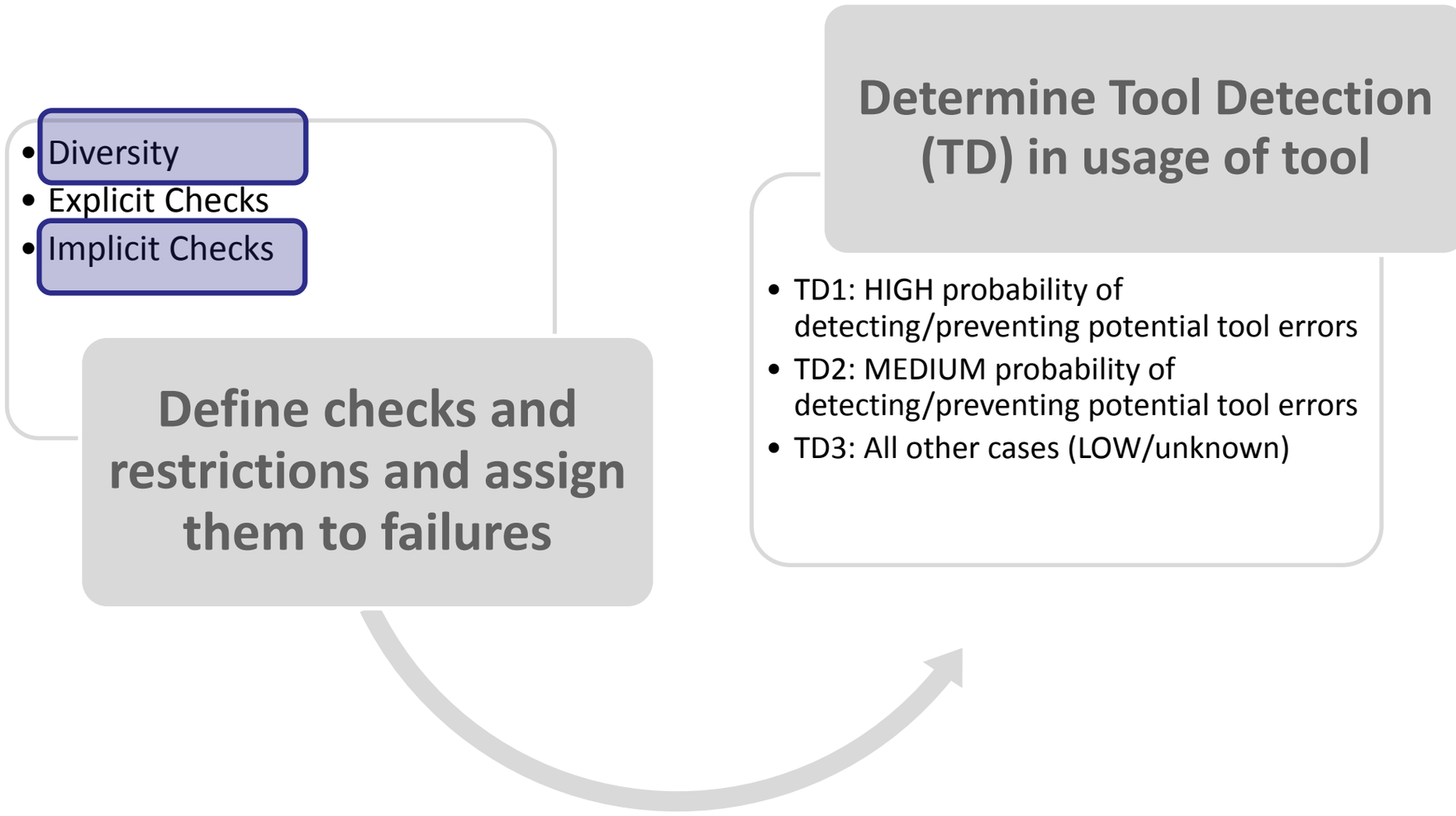
- **Detection & avoidance patterns**



# Example: Testing Tool Chain

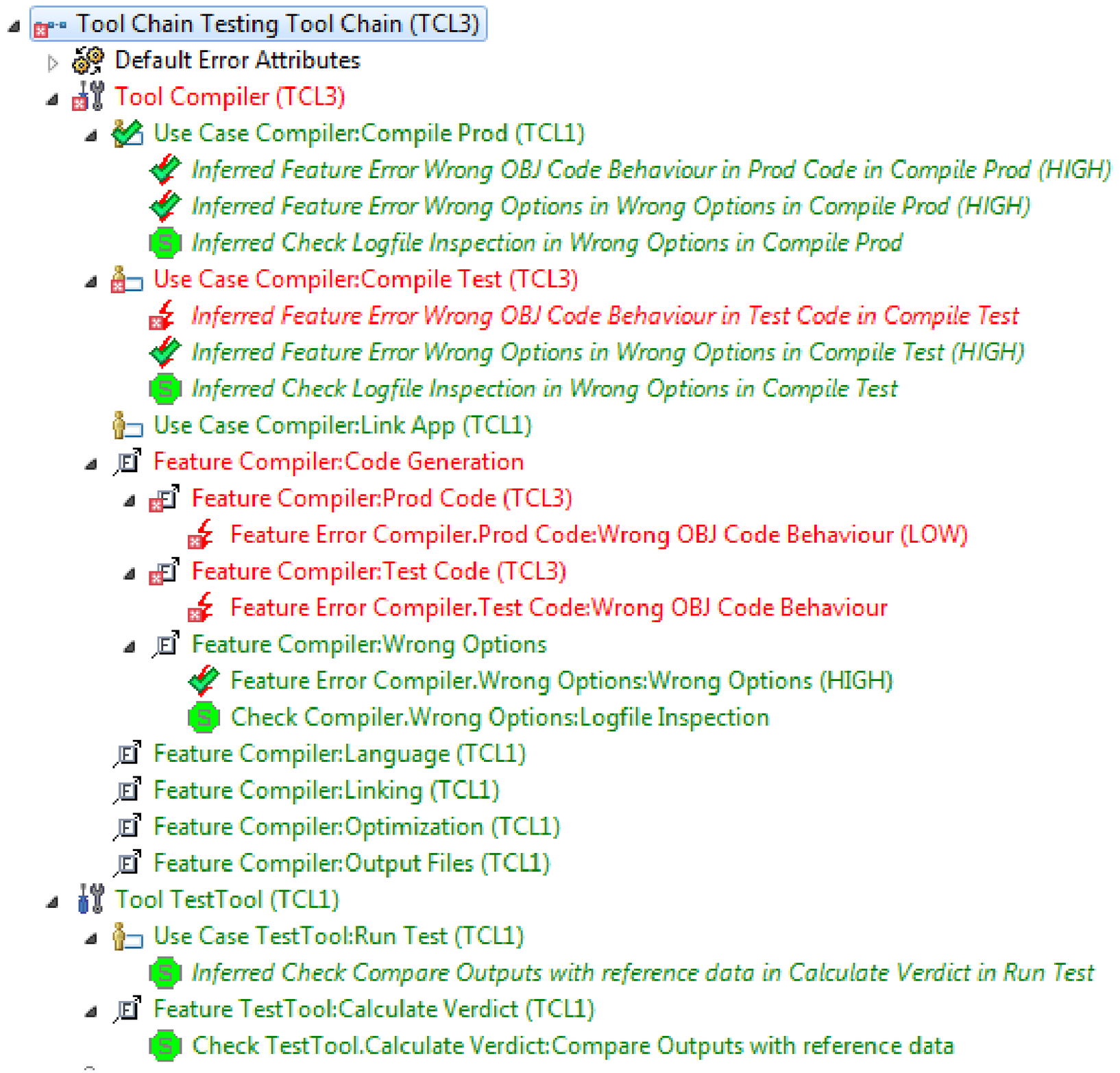


# Step 5: Assign Detection & Prevention Measures

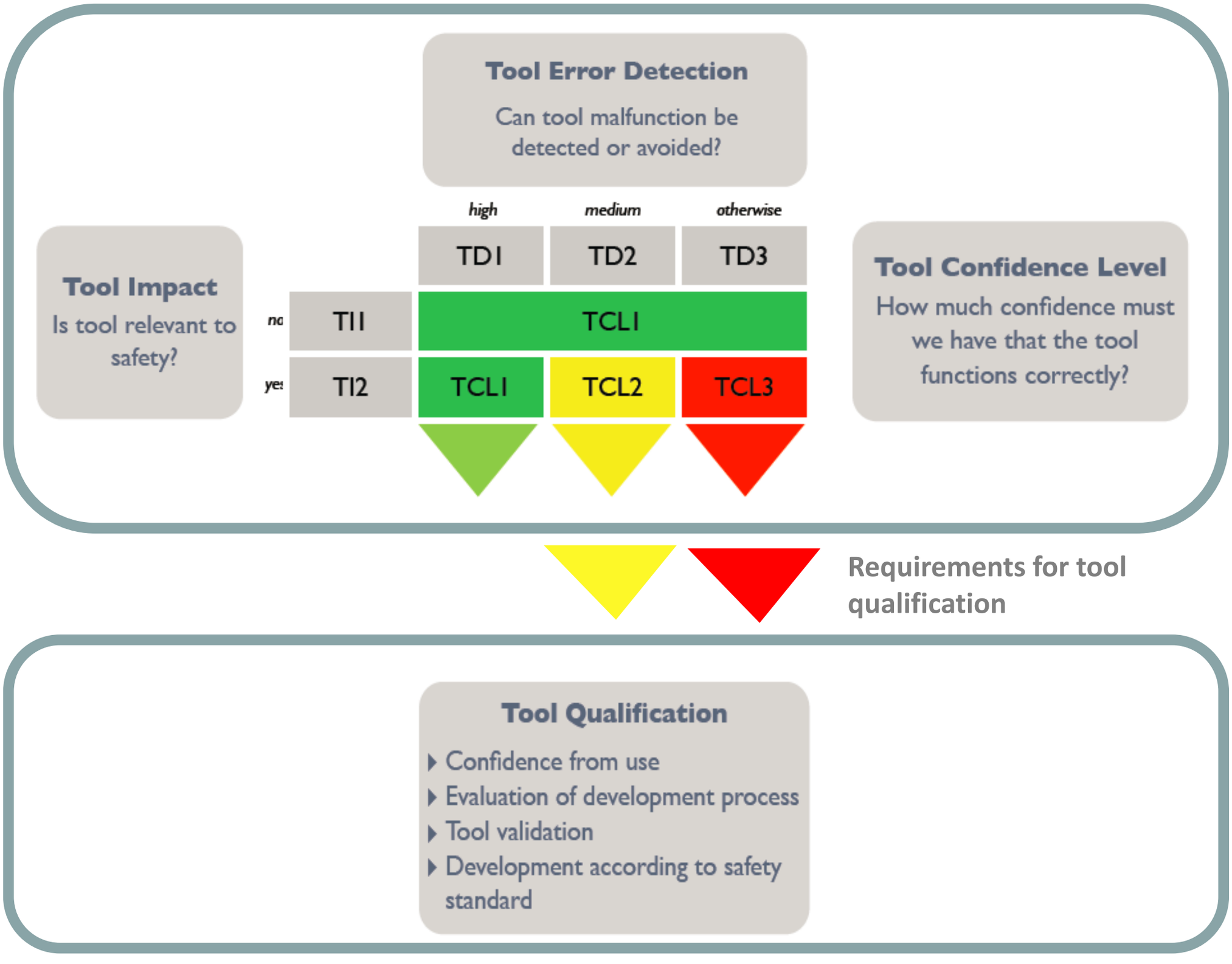



- **Depend on usage of other tools**
- **Tool chain analysis helps to reduce the TCL of single tools**
- **Tool chain analysis helps to prevent qualification**

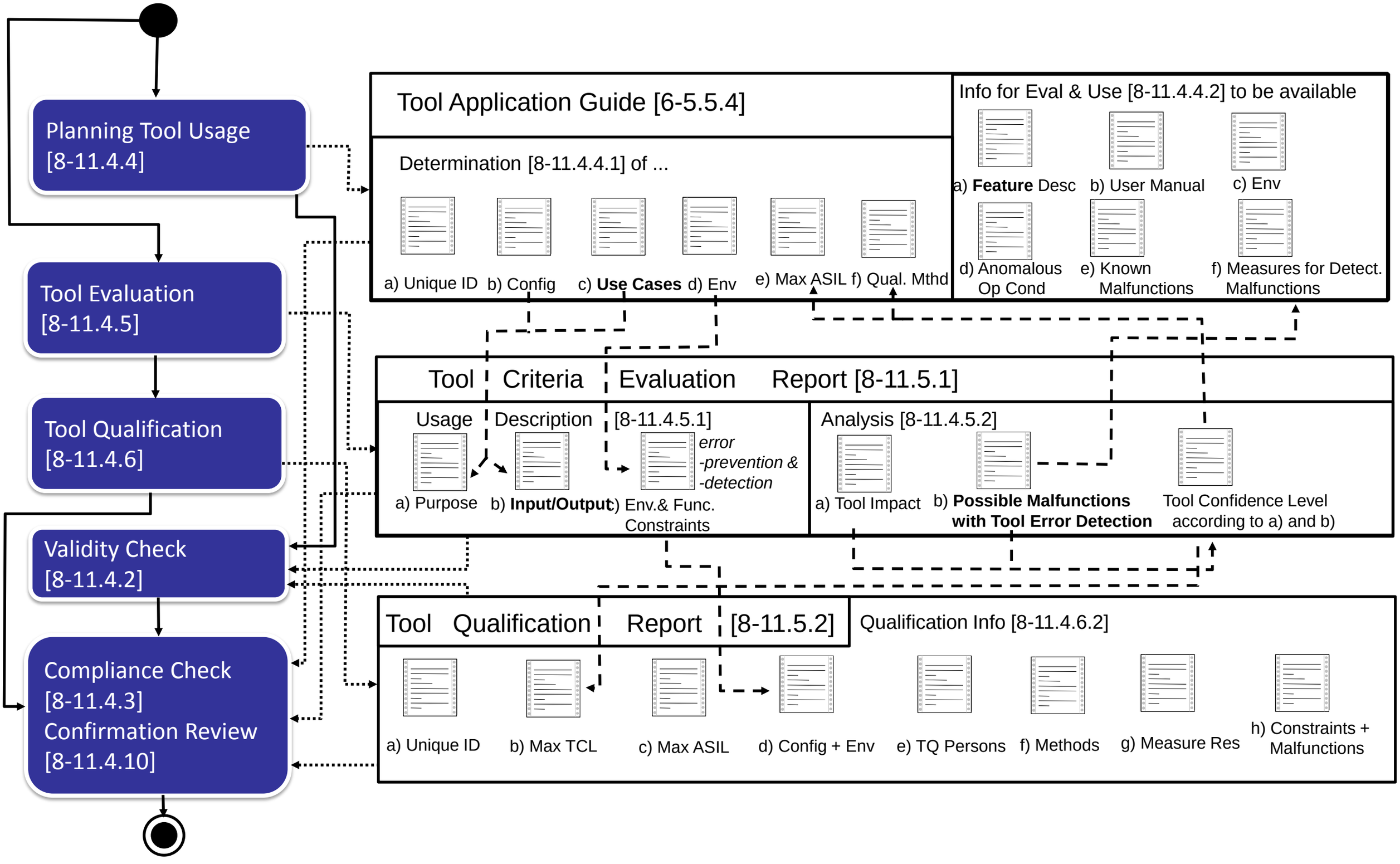
# Step 5: Assign Detection & Prevention Measures



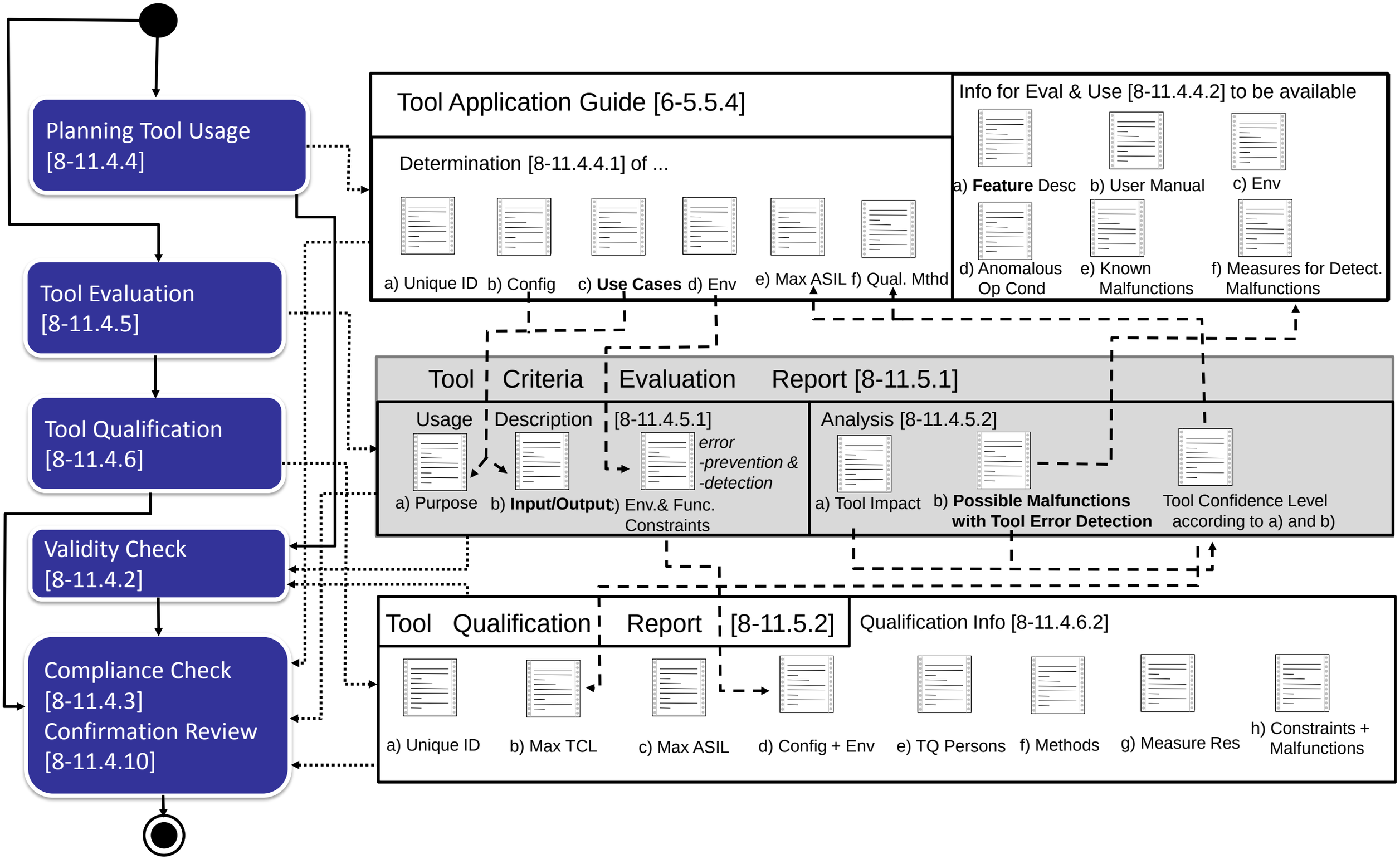
# Step 6: Compute TCL



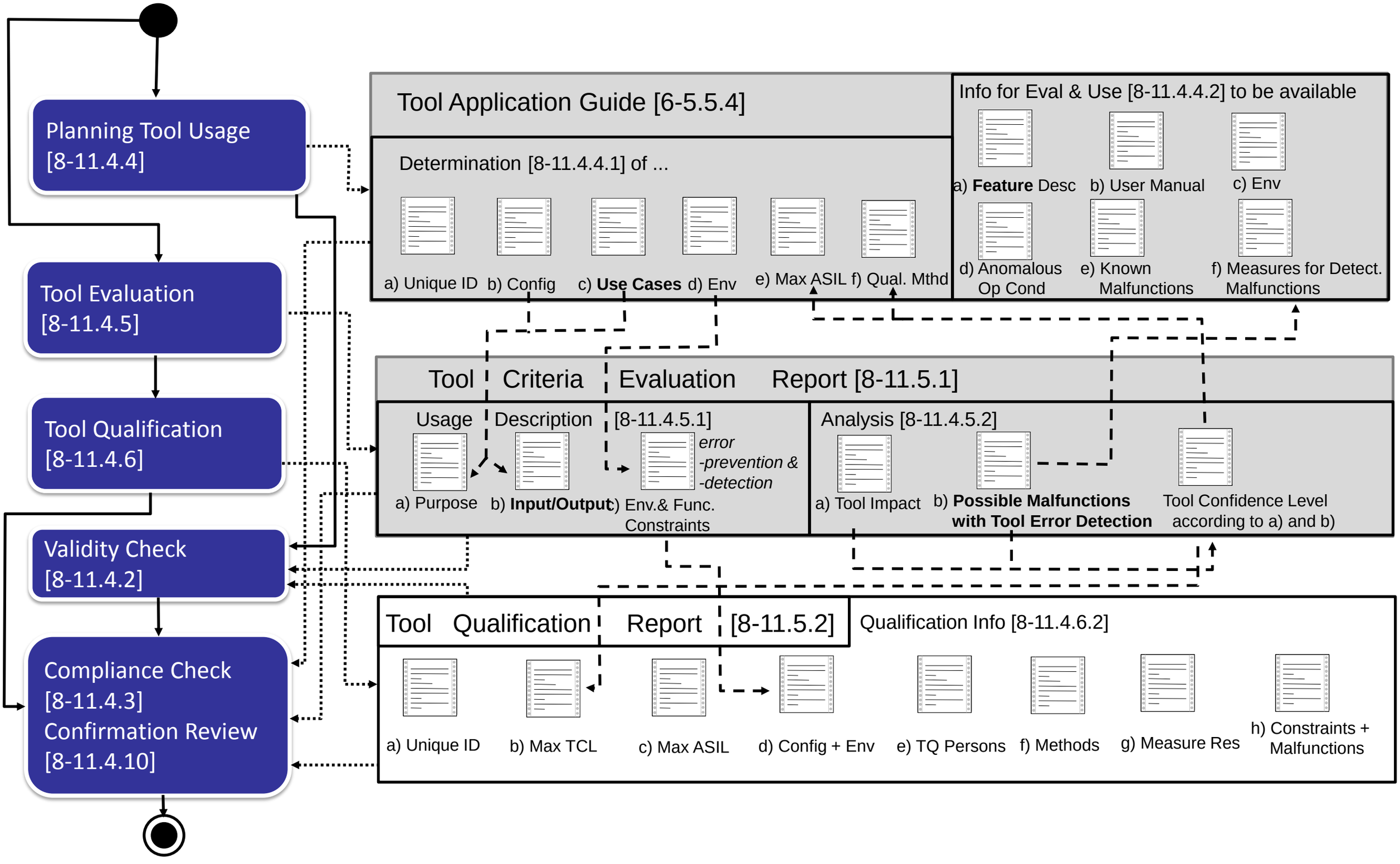
# What is left to do?



# What is left to do?



# What is left to do?



# What else can you do?

## Usage of Assumptions

- interface between model generator and the project
- company wide general model <=> projects enable their concrete models
- Only enabled elements considered for TCL computation

## Usage of Variants

- In the tool chain one or more variants can be active
- Only elements, that are assigned to the selected variants are active
- Only elements of active variantes considered for TCL computation

## How does it help?

- A tool chain is a „living creature“
- Instant TCL computation shows impact of changing elements immediately
- Report generation finalizes tool chain changes easily

# Visions

- **Reduce effort => reduce costs**

## Is State

**Definition of data flow**

- Use-Cases
- Artifacts

**Definition of detection model**

- Failures
- Detections / Mitigations

## To Be

**Definition of data flow**

- Process owner
  - Provide process information
  - Review model
- Process modeler
  - Model process information

**Definition of detection model**

- Process modeler
  - Design feature model with errors and mitigations
- Process owner
  - Review model

# Visions

- **Reduce effort => reduce costs**

## Is State

**Definition of data flow**

- Process owner
  - Provide process information
  - Review model
- Process modeler
  - Model process information

**Definition of detection model**

- Process modeler
  - Design feature model with errors and mitigations
- Process owner
  - Review model

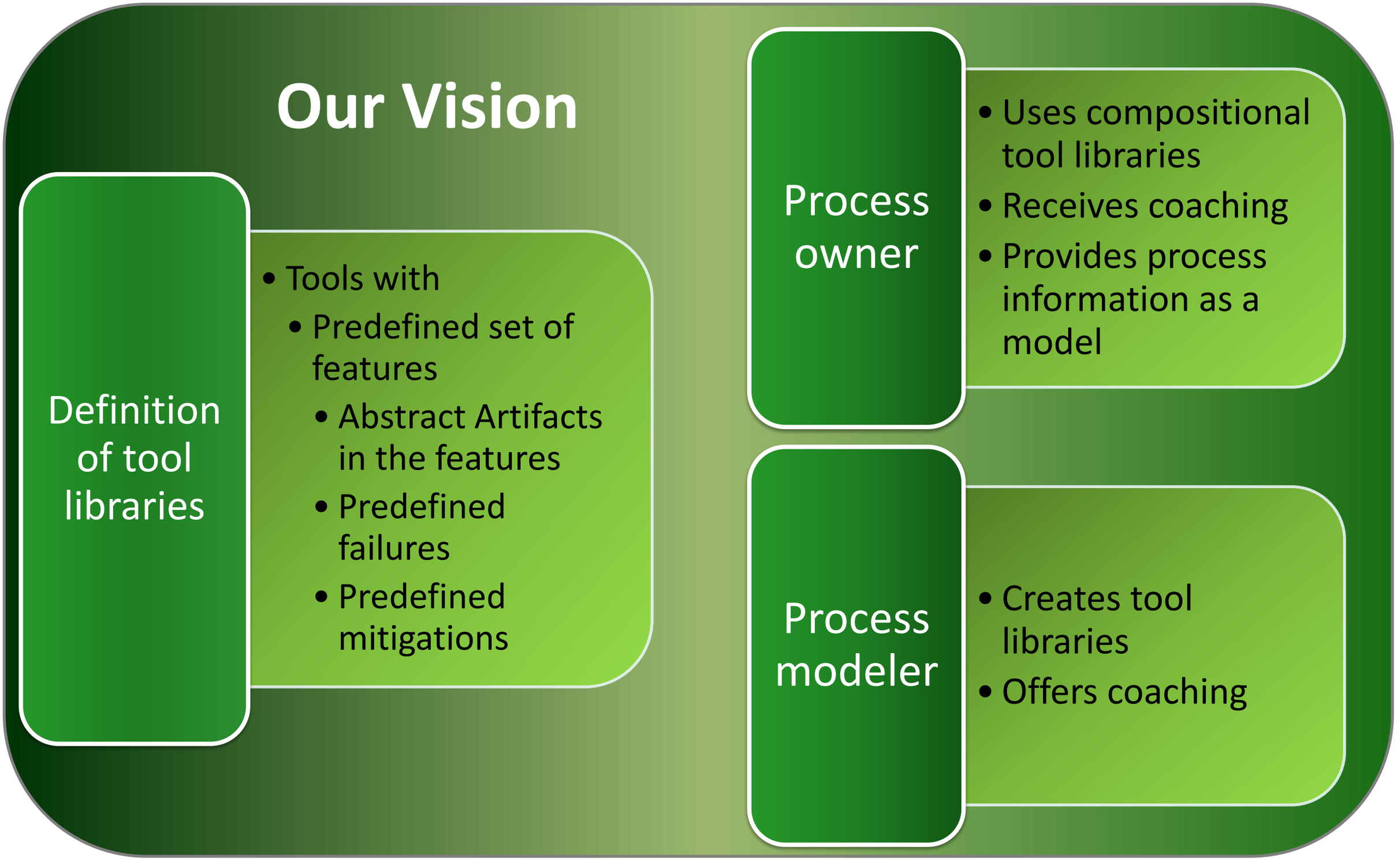
## Our Vision

Definition of tool libraries

- Tools with
  - Predefined set of features
    - Abstract Artifacts in the features
    - Predefined failures
    - Predefined mitigations

# Visions

- **Reduce effort => reduce costs**



# Visions

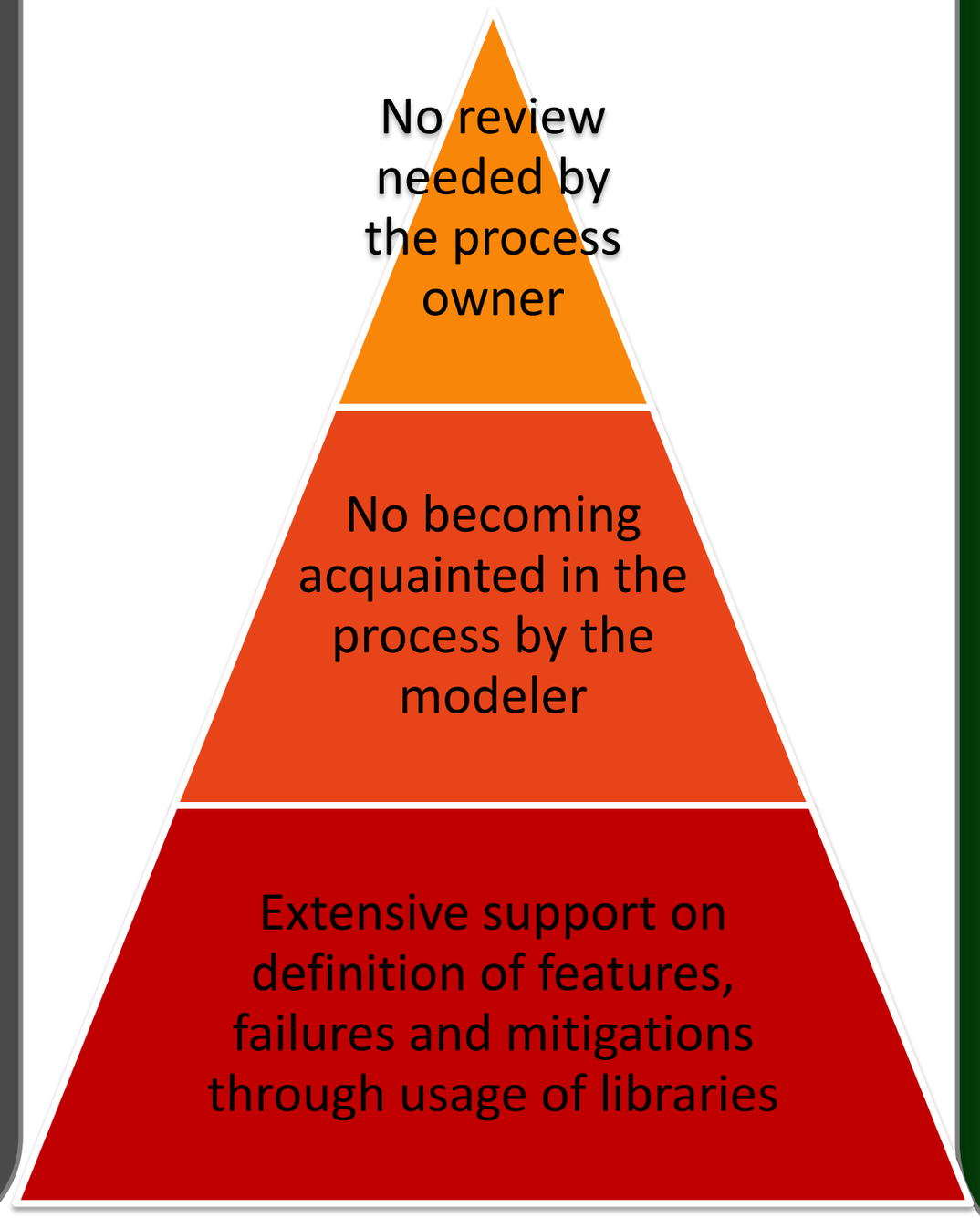
**Definition of data flow**

- Process owner
  - Provide process information
  - Review model
- Process modeler
  - Model process information

**Definition of detection model**

- Process modeler
  - Design feature model with errors and mitigations
- Process owner
  - Review model

## Significant reduction of effort




**Process owner**

- Uses compositional tool libraries
- Receives coaching
- Provides process information as a model

**Process modeler**

- Creates tool libraries
- Offers coaching

# Conclusion

**Tool Chain Analysis**

- **Deduced from ISO 26262 (use cases, features, inputs, outputs, errors)**
- **Automatically determines tool confidence level (TCL) and validates impact (TI)**
- **Provides transparency by formal model and**
  - Tool evaluation report (required from ISO 26262)
  - Assumptions for tool users
  - Qualification need (TCL) for each tool
  - **Qualification requirements for tool validation**
    - **Critical (required) features**
    - **Relevant errors**

This is mostly a small subset of the complete tool correctness and can save much effort compared to a „complete“ tool qualification

**Tool Chain Analyzer can be evaluated from [www.validas.de/TCA.htm](www.validas.de/TCA.htm)**

**Validas provides support for**

- **Tool users (i.e. help to avoid tool qualification)**
- **Tool providers (i.e. develop classification and qualification kits)**