Automation of Requirements-Based Testing (RQBT)

R. Gerlich[1], R. Gerlich[1], Maria Hernek[2],

Allan Pascoe[3], Glenn Johnson[3]

Data Systems in Aerospace DASIA 2019

June 4 - 6, 2019, Torremolinos, Spain

[1] Dr. Rainer Gerlich BSSE System and Software Engineering
Immenstaad, Germany
E-Mail: Rainer.Gerlich@bsse.biz
Ralf.Gerlich@bsse.biz

[2] European Space Agency (ESA/ESTEC)
Noordwijk, The Netherlands
E-Mail: Maria.Hernek@esa.int,
Jinesh.Ramachandran@esa.int

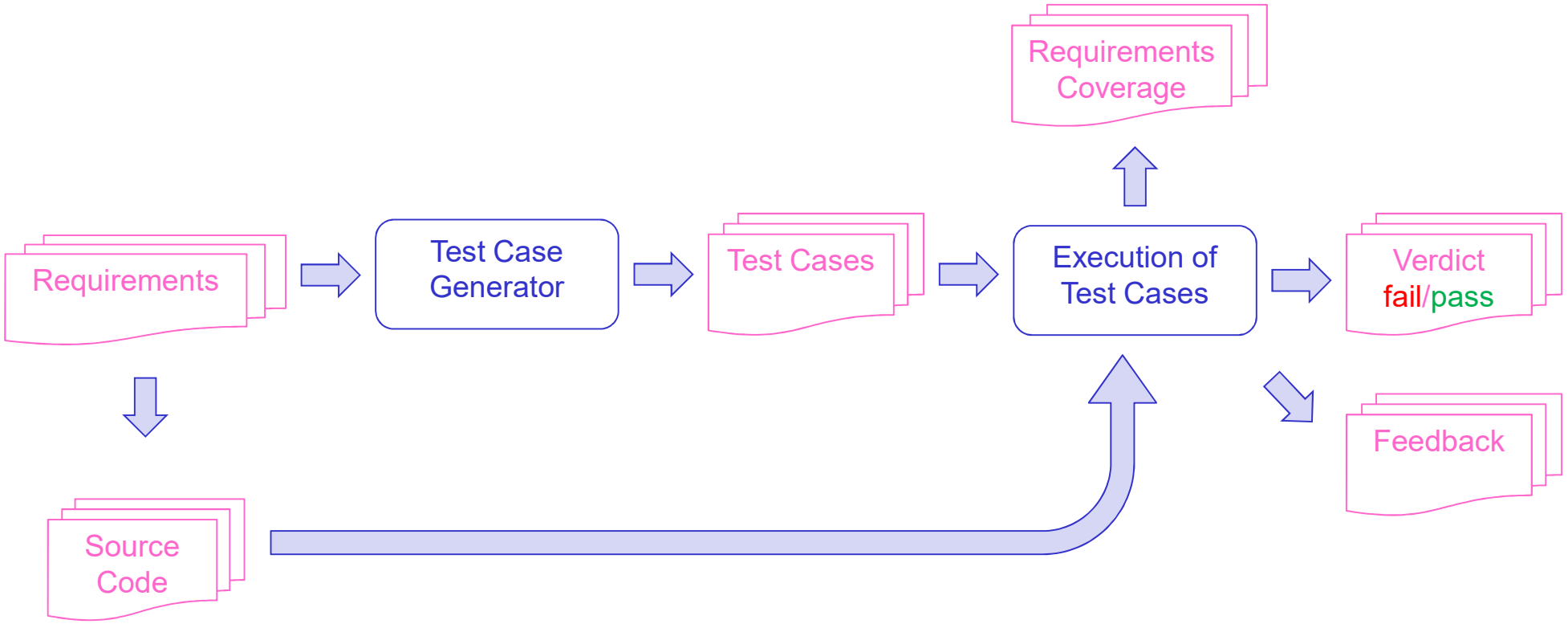
[3] SCISYS UK Ltd
Bristol, UK
E-Mail: Allan.Pascoe@scisys.co.uk
Glenn.Johnsom@scisys.co.uk

© Dr. Rainer Gerlich BSSE System and Software Engineering, 2019, DASIA'2019, Málaga, Spain: Automation of Requirements-Based Testing

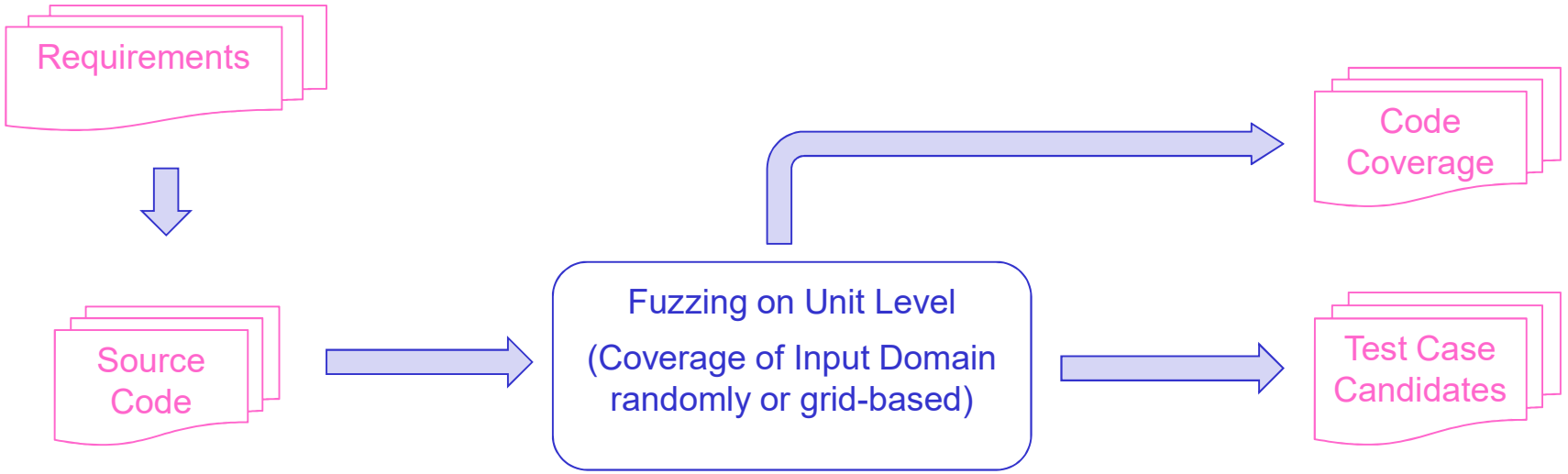
Agenda

- Introduction/Current Status
- Our Approach
- Examples
- Open Challenges
- Conclusions

Our Goal


Requirements
Coverage
Requirements
Test Case
Generator
Test Cases
Execution of
Test Cases
Verdict
fail/pass
Feedback
Source
Code

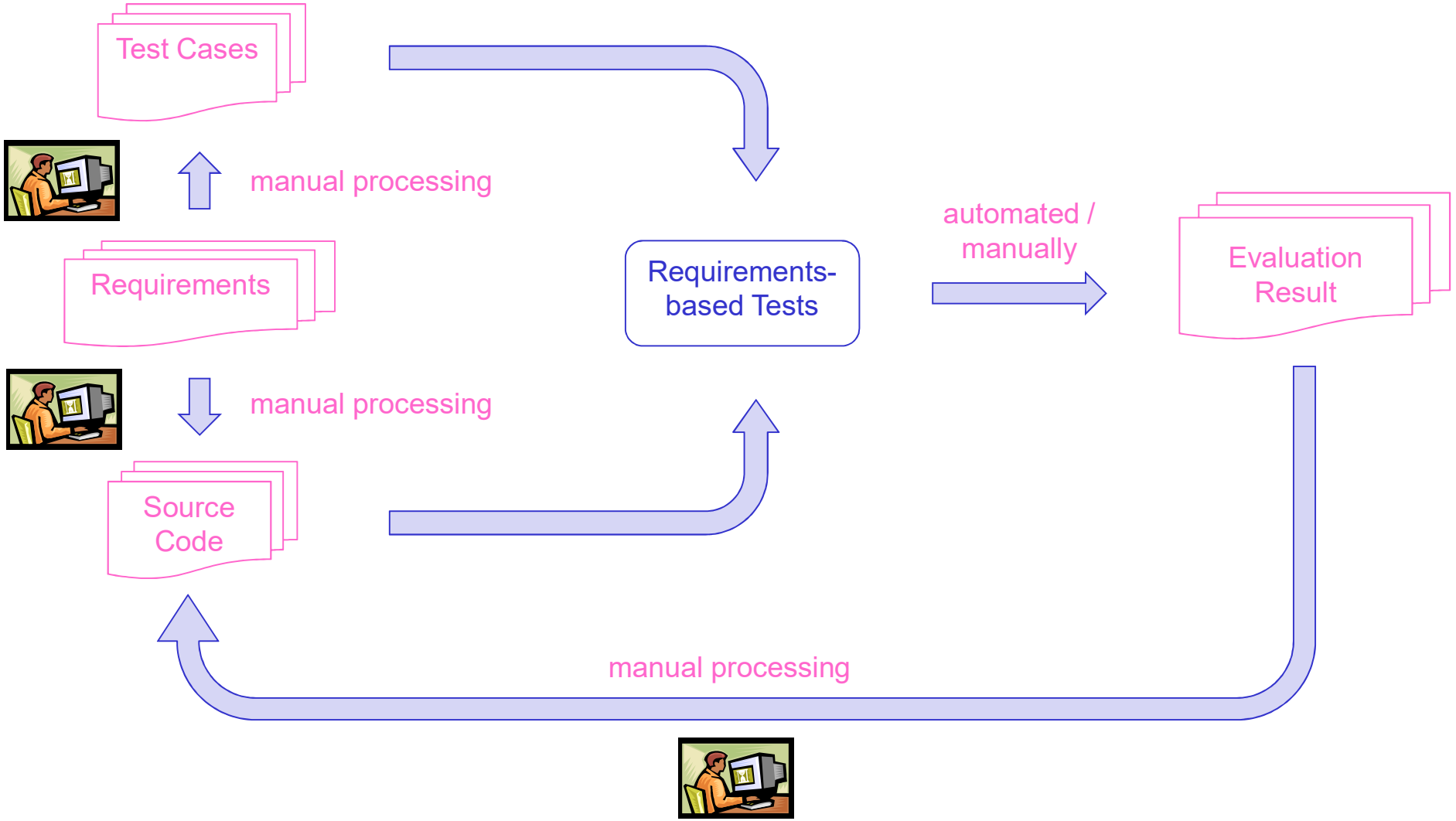
Current Status



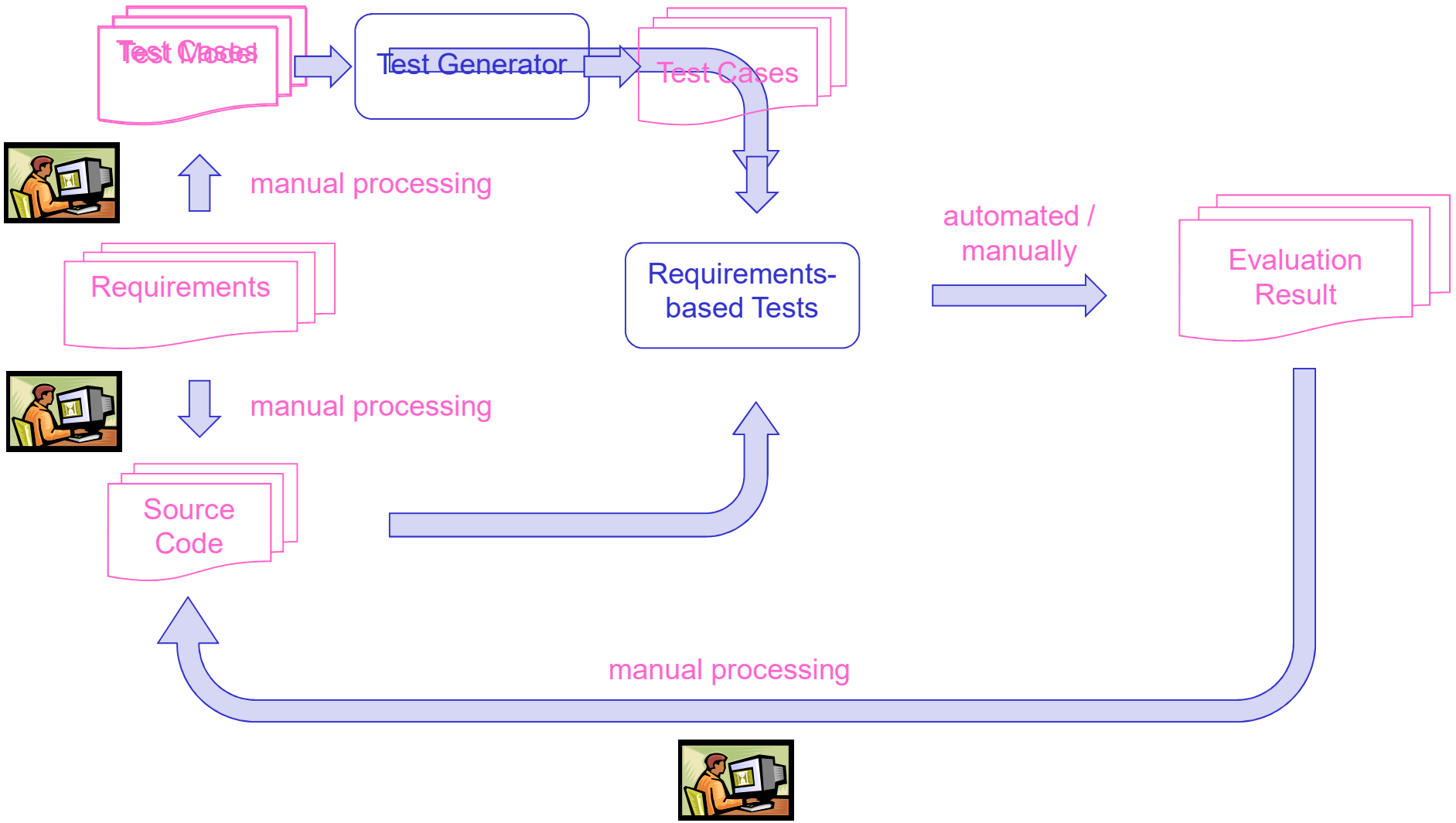
Goal for Automation of RQBT

- Benefitting from automated massive stimulation
- Automated correlation of unit tests with requirements
- Automated evaluation of test results w.r.t. requirements
- Automated propagation of verification results (pass/fail) bottom-up in the hierarchy of requirements

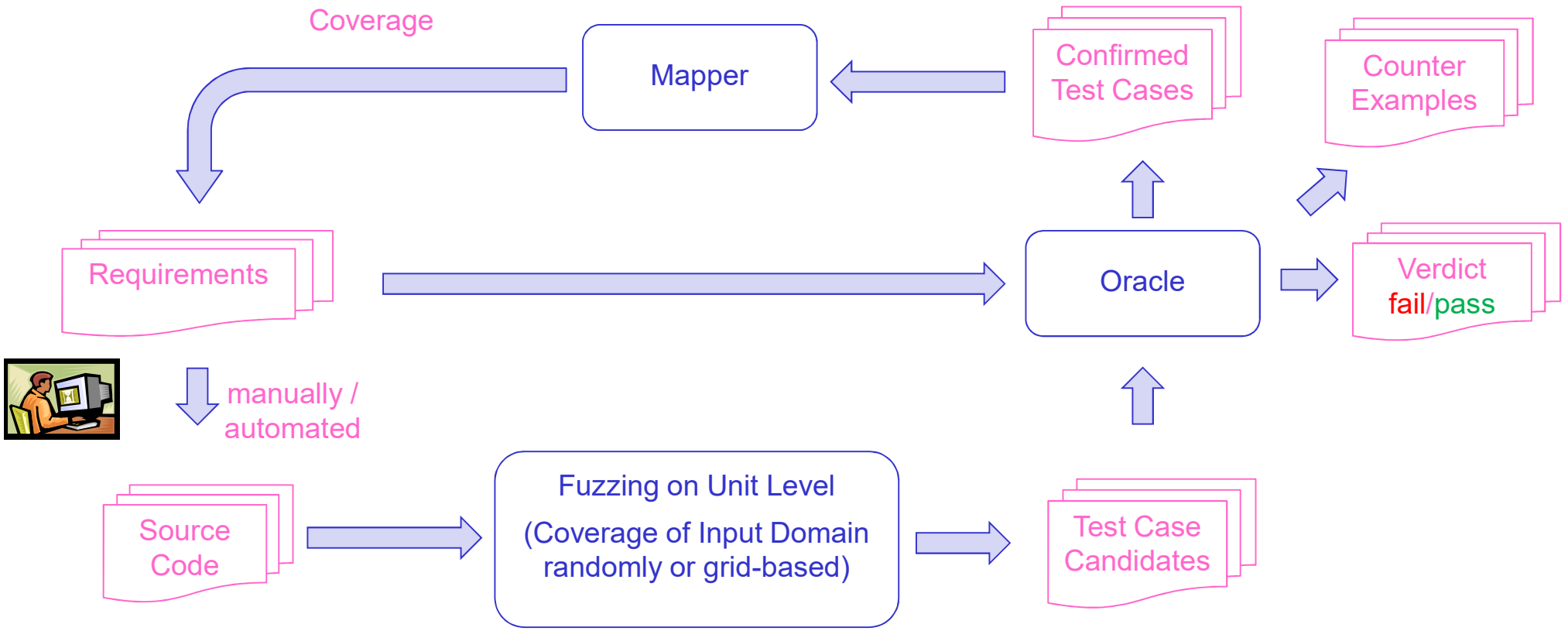
The Manual Procedure for RQBT



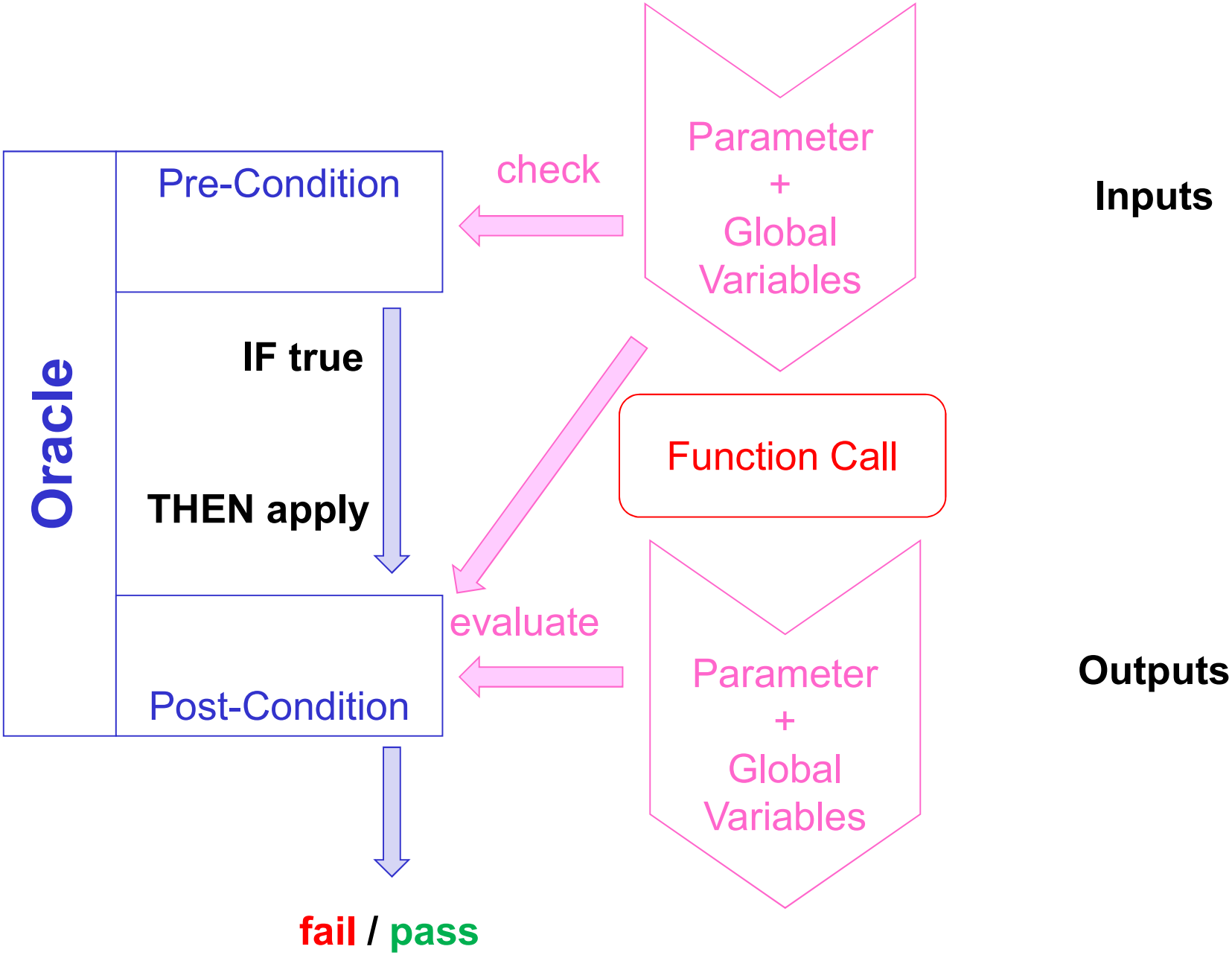
Model-Based Testing



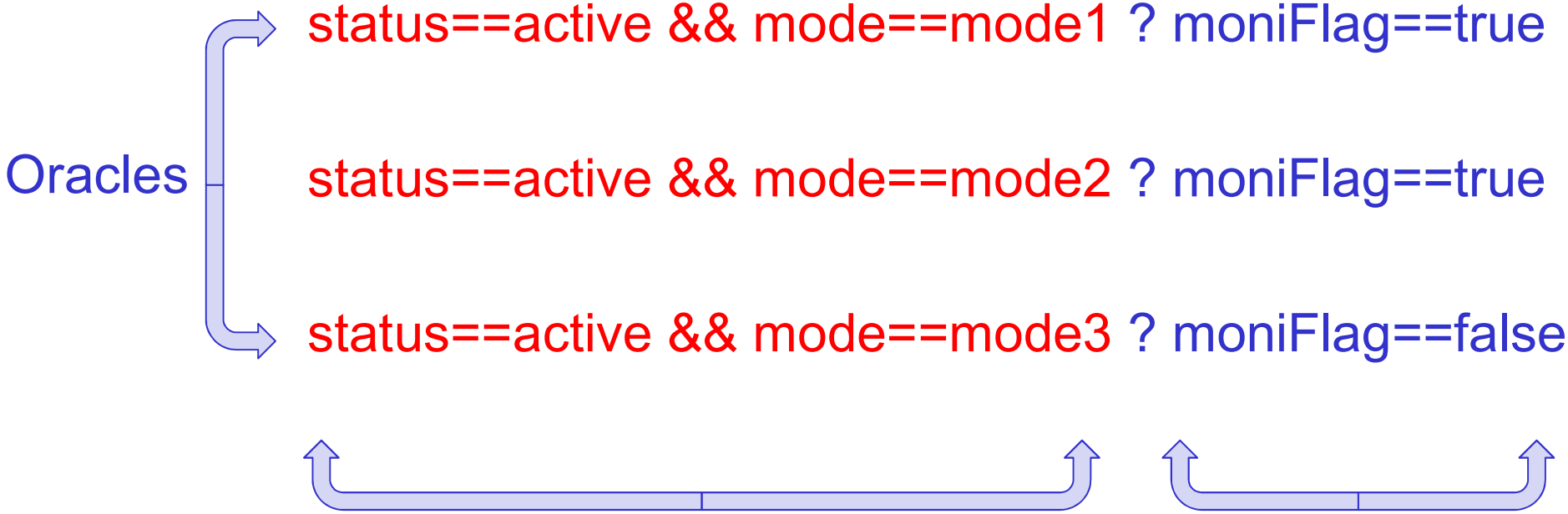
Our Approach



Oracle Structure



Example: Status Monitoring

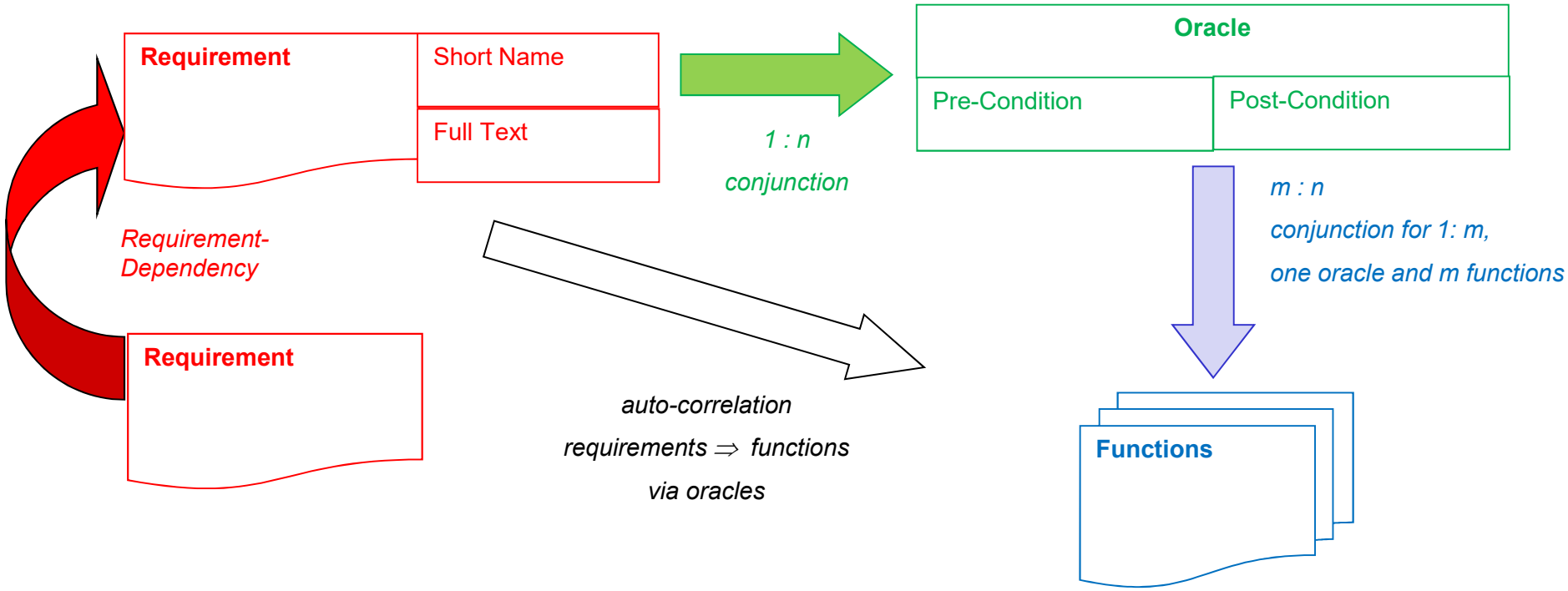


Some Mathematical Examples and Results of Implementation

BSSE System and Software Engineering

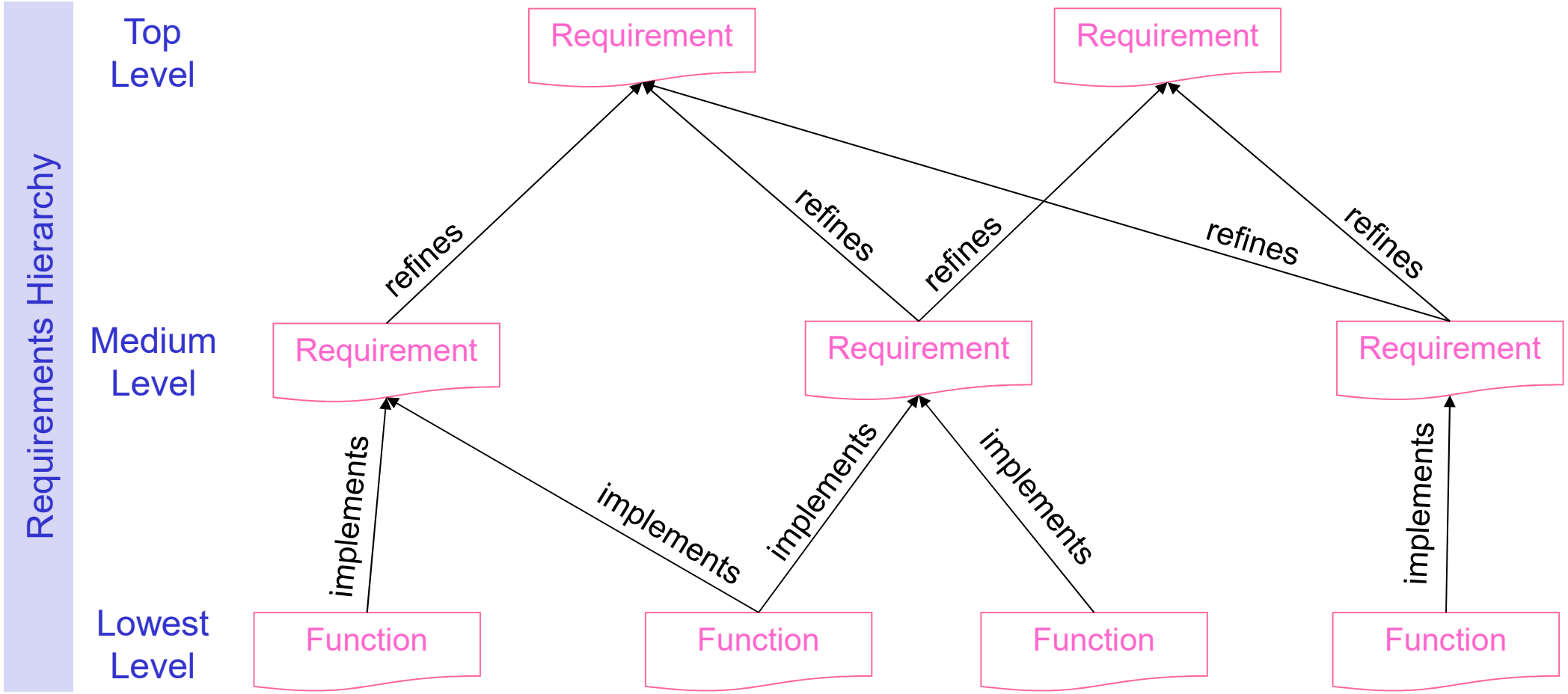
Function	Requirement	Oracle		Number of Tests	Oracle Output			RQ fully covered	RQ verified
		Pre-Condition	Post-Condition		Coverage	true	false		
x*x	$\forall x \in \{double\}\ \sqrt{x^2}$ shall not differ from $\|x\|$ more than ε	x normalised	`(fabs(fabs(x)-sqrt(retVal))/x)<eps`	302	299	225	74	yes	no
		x denormalised	`fabs(fabs(x)-sqrt(retVal))<eps`		3	3	0		
abs(x)	$\forall x \in \{sint\}$ abs(x) shall be ≥ 0	`true`	retVal>=0	302	302	301	1	yes	no

Mapping Scheme

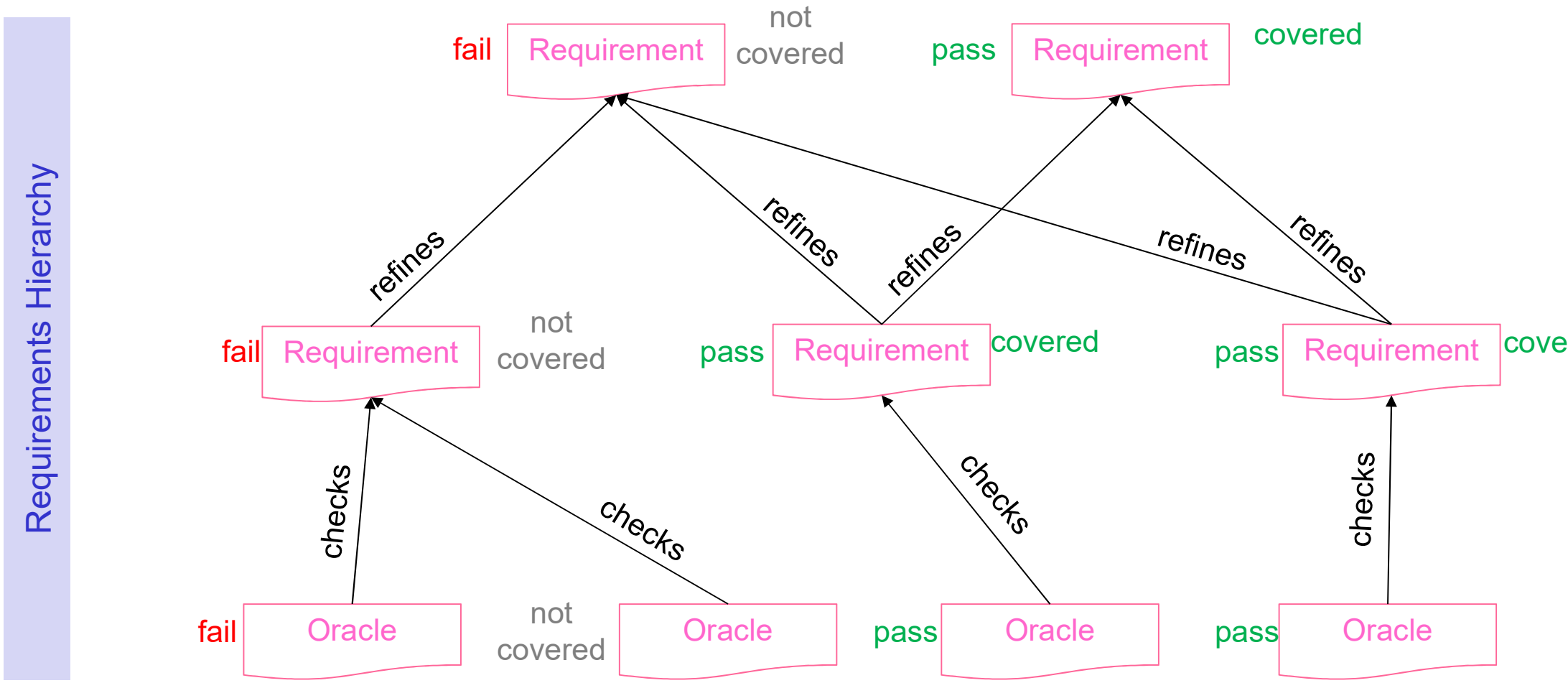


Goal / Challenge:
automated derivation
supported by some
formalised (an)notation
of requirements

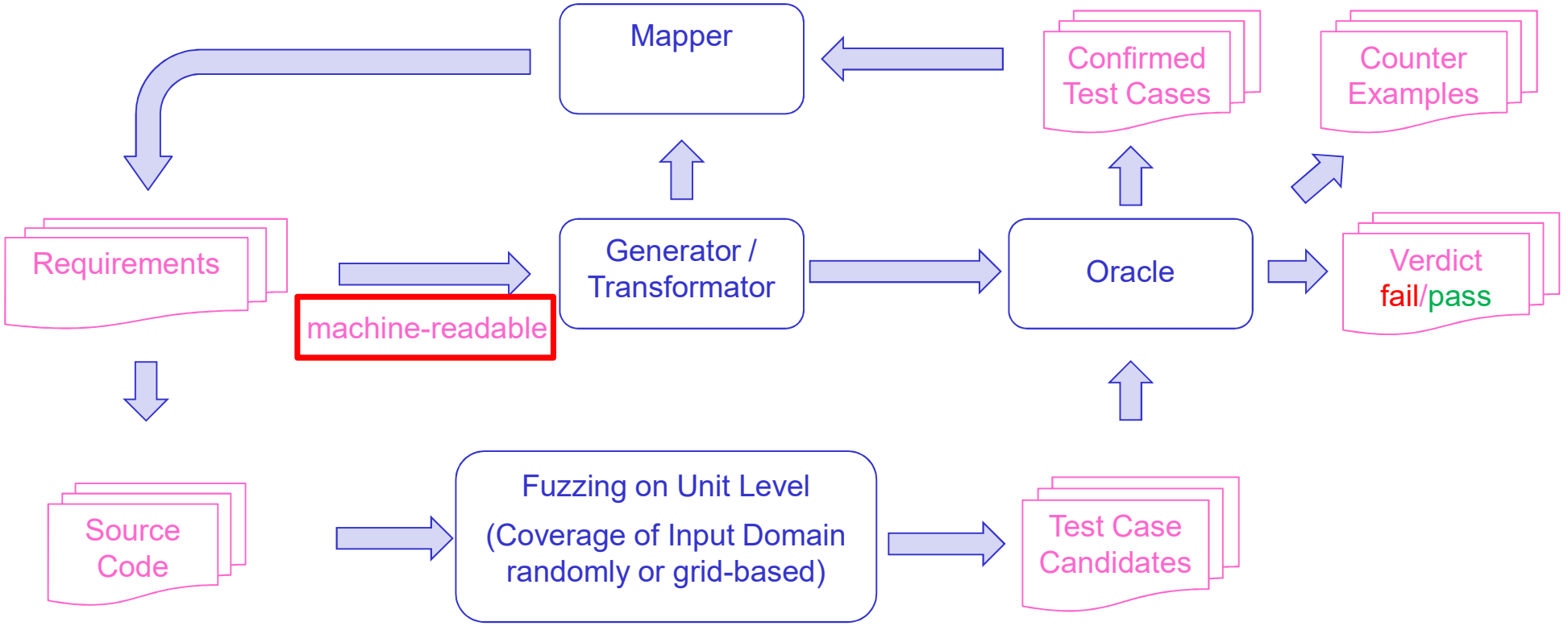
Functions vs. Requirements



Oracles vs. Requirements and pass / fail Inheritance



THE Challenge: machine-readable requirements (an)notation



Currently:
Requirements as pure text (Word, DOORS)
Automated extraction of oracles is difficult/impractical

Challenge:
Suitable and acceptable notation for requirements / oracles

Suggestions from users required!

Aspects of Quality Assurance

- The same measures for quality assurance as for the source code to be verified also apply to the oracles.
- This is similar to measures for manual verification.
- In addition, suitable automated checks can be applied in case of automated derivation of oracles.
- In contrast to manual processing the measures are consistently applied for every verification step.
- Improvement in the quality of requirements also expected.

Status

- Oracle notation defined
- Implementation available for integration of oracles with fuzzing / massive stimulation
- Reporting available with bottom-up propagation of pass/fail results
- Identification of counter examples confirmed
- Support for identification of source of failing oracles

Outlook

- Identification of user-acceptable machine-readable notations
- Analysis of current requirements
- Classification of requirements
  - top-level / low-level
  - functional
  - non-functional
  - quantifiable
- Proof-of-concept

Acknowledgement

The presented approach is an outcome of the ESA-project "Automated Source-code-based Testing, Continued" (FAST II) for the European Space Agency (ESA) (ESA Contract No. 4000116014) as part of the General Support Technology Programme (GSTP).

Funded by the German Ministery of Economics and Energy (Bundesministeriums für Wirtschaft und Energie, BMWi) and initiated by the German Space Agency (DLR Raumfahrtmanagement).