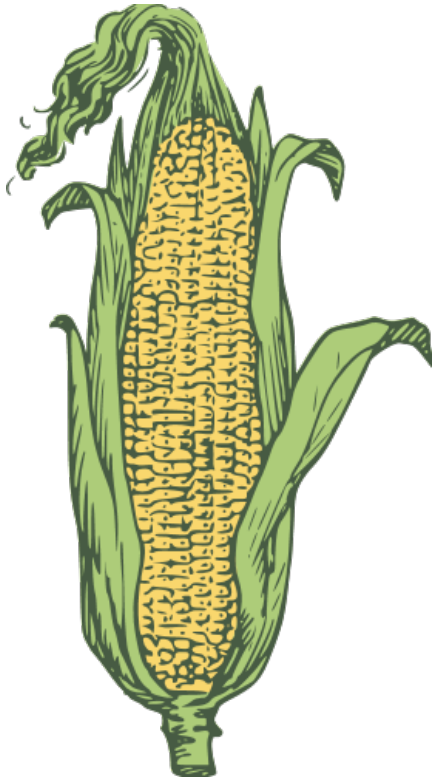
An industrial case study of TACO

Benjamin Lesage, Stephen Law, Iain Bate


Icons courtesy of https://icons8.com/

Rolls-Royce

Context



- Rolls-Royce VISIUMCORE platform
  - Integrated instruction tracing and timing
  - Instructions' execution is time-invariant
  - Limited time-relevant state
- WCET Timing analysis
  - Builds WCET from low-level block timings
  - Requires full coverage of executed code
- Full Authority Digital Engine Controller
  - Designed to DO-178C DAL A guidelines
  - Coverage through low-level tests
- Automatic test case generation technique [1]
  - Provide *better* coverage and timing data

[1] Stephen Law and Iain Bate, Achieving Appropriate Test Coverage for Reliable Measurement-Based Timing Analysis, ECRTS, 2016

Context

On-Target testing is expensive

Formal testing late in life cycle

Rolls-Royce VISIUMCORE platform

- Integrated instruction tracing and timing
- Instructions' execution is time-invariant
- Limited time-relevant state

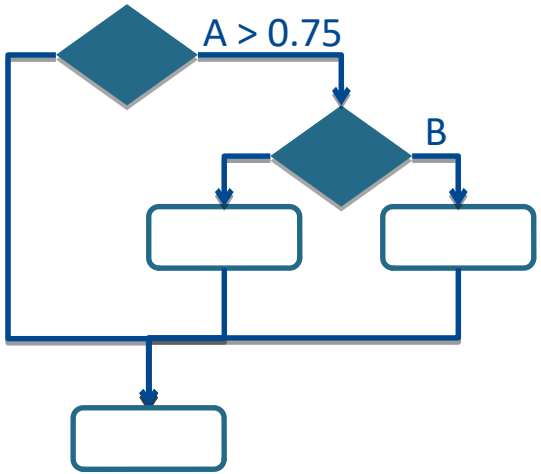
- WCET Timing analysis
  - Builds WCET from low-level block timings
  - Requires full coverage of executed code
- Full Authority Digital Engine Controller
  - Designed to DO-178C DAL A guidelines
  - **Coverage through low-level tests**
- Automatic test case generation technique [1]
  - Provide *better* coverage and timing data

[1] Stephen Law and Iain Bate, Achieving Appropriate Test Coverage for Reliable Measurement-Based Timing Analysis, ECRTS, 2016

Coverage Technique [1] - Example

- Function F has three *inputs*: B, C, and A
  - B and C are direct parameters
  - A is part of the internal system state
- F Inputs type and ranges are known:
  - $A \in \mathbb{R}, 0 \leq A \leq 1$
  - $B \in \{True, False\}$
  - $C \in \mathbb{Z}, 0 \leq C \leq 7$

function F (B: Boolean, C: Byte)



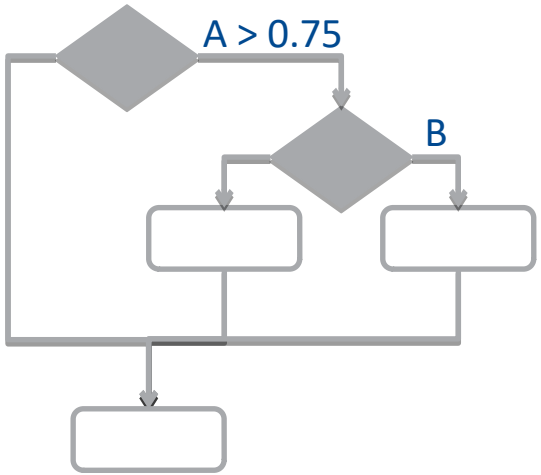
[1] Stephen Law and Iain Bate, Achieving Appropriate Test Coverage for Reliable Measurement-Based Timing Analysis, ECRTS, 2016

Coverage Technique [1] - Example

{A: 0.5, B: True, C: 7}

Generate a test vector

function F (B: Boolean, C: Byte)

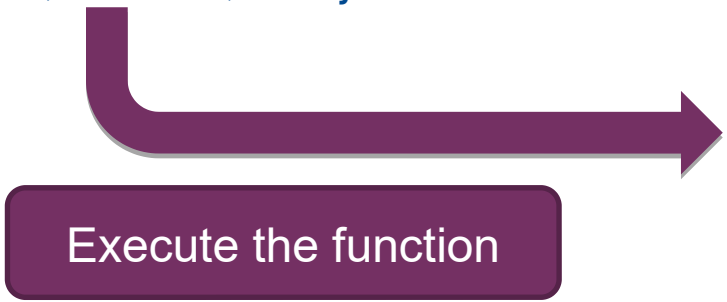


[1] Stephen Law and Iain Bate, Achieving Appropriate Test Coverage for Reliable Measurement-Based Timing Analysis, ECRTS, 2016

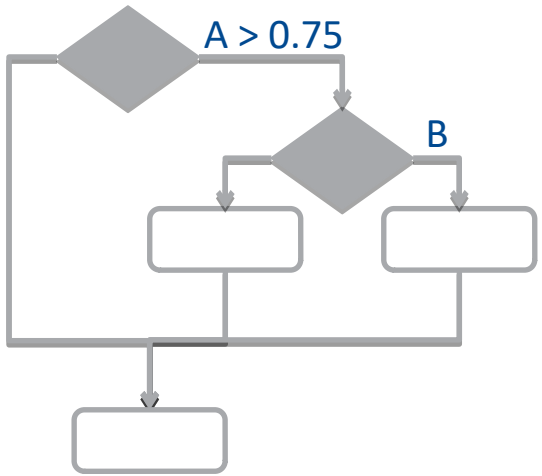
Coverage Technique [1] - Example

{A: 0.5, B: True, C: 7}

function F (B: Boolean, C: Byte)



Execute the function

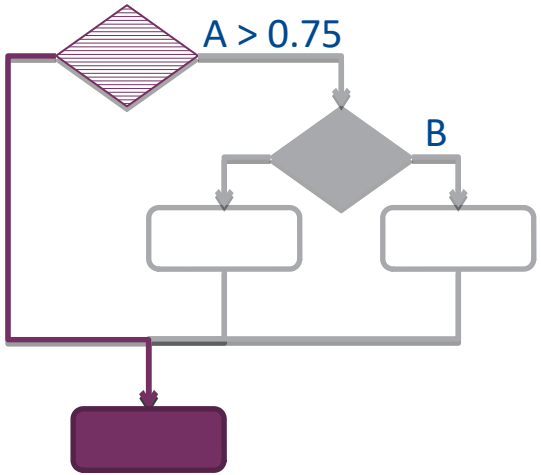


[1] Stephen Law and Iain Bate, Achieving Appropriate Test Coverage for Reliable Measurement-Based Timing Analysis, ECRTS, 2016

Coverage Technique [1] - Example

{A: 0.5, B: True, C: 7}

function F (B: Boolean, C: Byte)



Collect coverage metrics

[1] Stephen Law and Iain Bate, Achieving Appropriate Test Coverage for Reliable Measurement-Based Timing Analysis, ECRTS, 2016

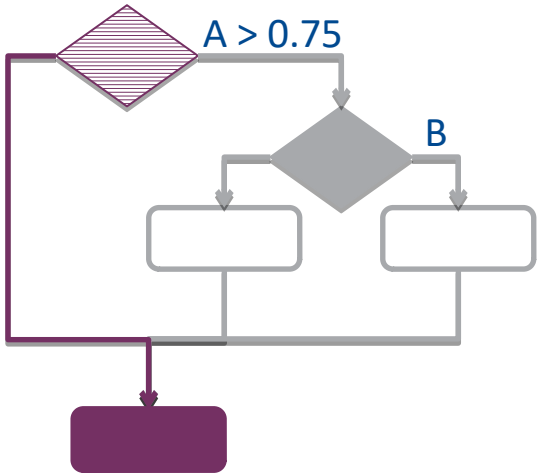
Coverage Technique [1] - Example

{A: 0.5, B: True, C: 7}

{A: **0.95**, B: True, C: 7}

Mutate the test vector

function F (B: Boolean, C: Byte)

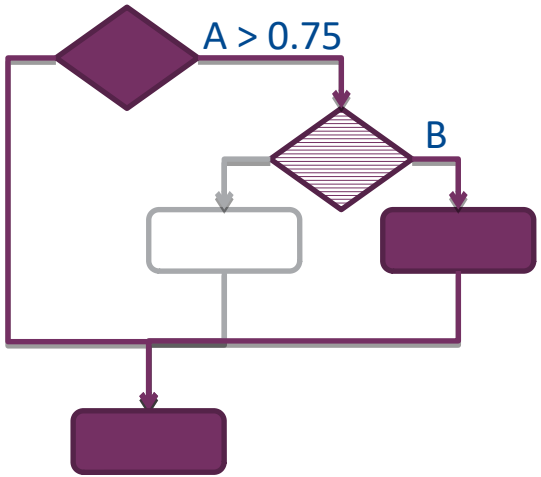


[1] Stephen Law and Iain Bate, Achieving Appropriate Test Coverage for Reliable Measurement-Based Timing Analysis, ECRTS, 2016

Coverage Technique [1] - Example

{A: 0.5, B: True, C: 7}
{A: **0.95**, B: True, C: 7}

function F (B: Boolean, C: Byte)



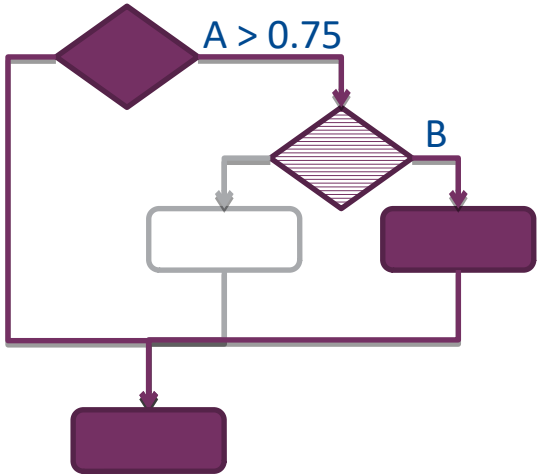
[1] Stephen Law and Iain Bate, Achieving Appropriate Test Coverage for Reliable Measurement-Based Timing Analysis, ECRTS, 2016

Coverage Technique [1] - Example

{A: 0.5, B: True, C: 7}
{A: **0.95**, B: True, C: 7}
{A: 0.95, B: True, C: **6**}

Reject poor solutions

function F (B: Boolean, C: Byte)

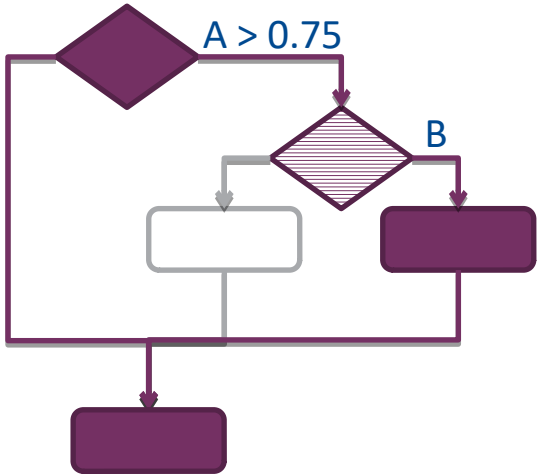


[1] Stephen Law and Iain Bate, Achieving Appropriate Test Coverage for Reliable Measurement-Based Timing Analysis, ECRTS, 2016

Coverage Technique [1] - Example

{A: 0.5, B: True, C: 7}
{A: **0.95**, B: True, C: 7}
{A: 0.95, B: True, C: **6**}
{A: 0.95, B: True, C: **7**}

function F (B: Boolean, C: Byte)

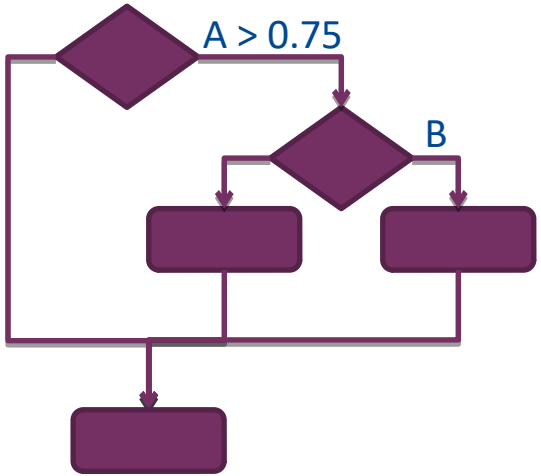


[1] Stephen Law and Iain Bate, Achieving Appropriate Test Coverage for Reliable Measurement-Based Timing Analysis, ECRTS, 2016

Coverage Technique [1] - Example

{A: 0.5, B: True, C: 7}
{A: **0.95**, B: True, C: 7}
{A: 0.95, B: True, C: **6**}
{A: 0.95, B: True, C: **7**}
{A: 0.95, B: **False**, C: 7}

function F (B: Boolean, C: Byte)

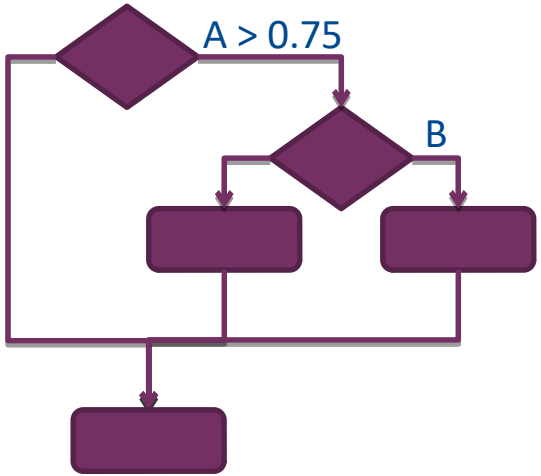


[1] Stephen Law and Iain Bate, Achieving Appropriate Test Coverage for Reliable Measurement-Based Timing Analysis, ECRTS, 2016

Coverage Technique [1] - Example

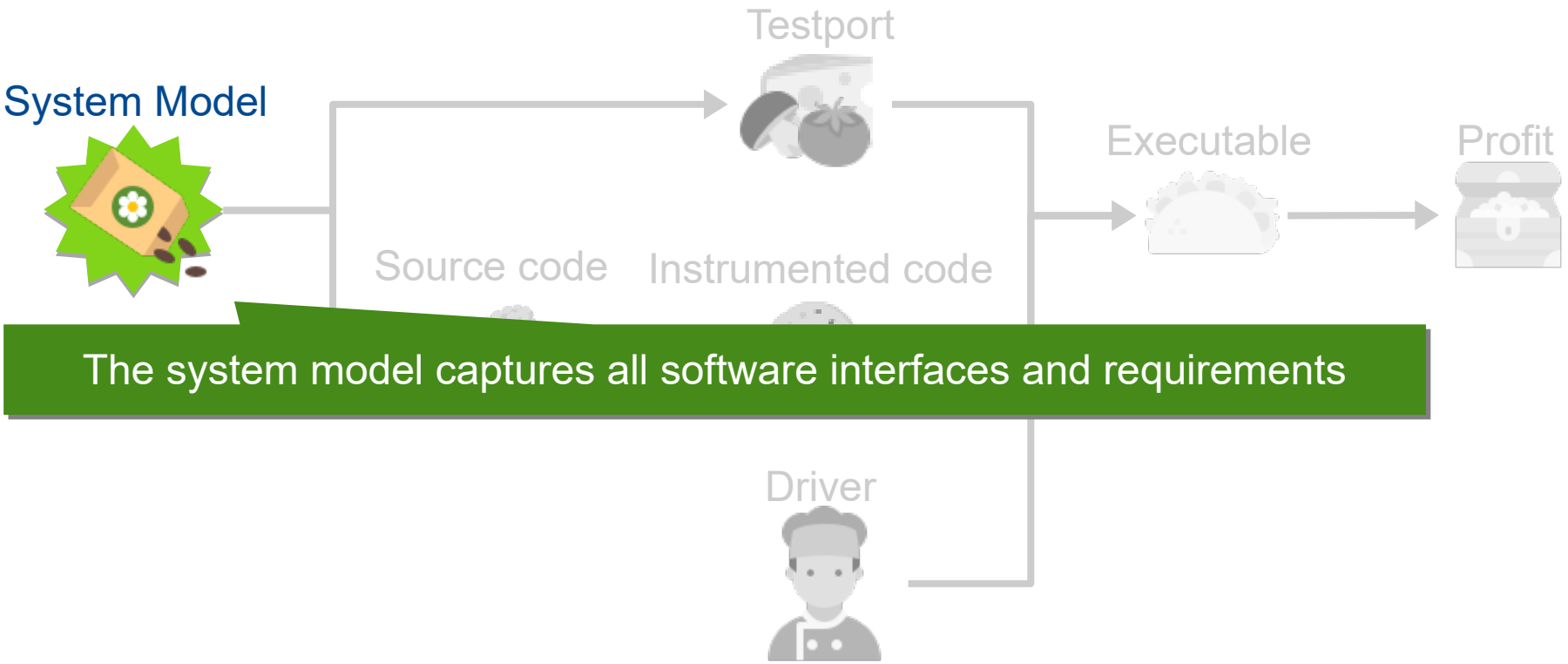
- Search-based timing analysis tool
  - Support measurement-based WCET analysis
  - Drive the execution of a tested function
  - Generate a sequence of *test vectors*
- Solutions evaluated on coverage metrics
  - Executed blocks of code, loops branches
  - Different heuristics target different objectives

function F (B: Boolean, C: Byte)

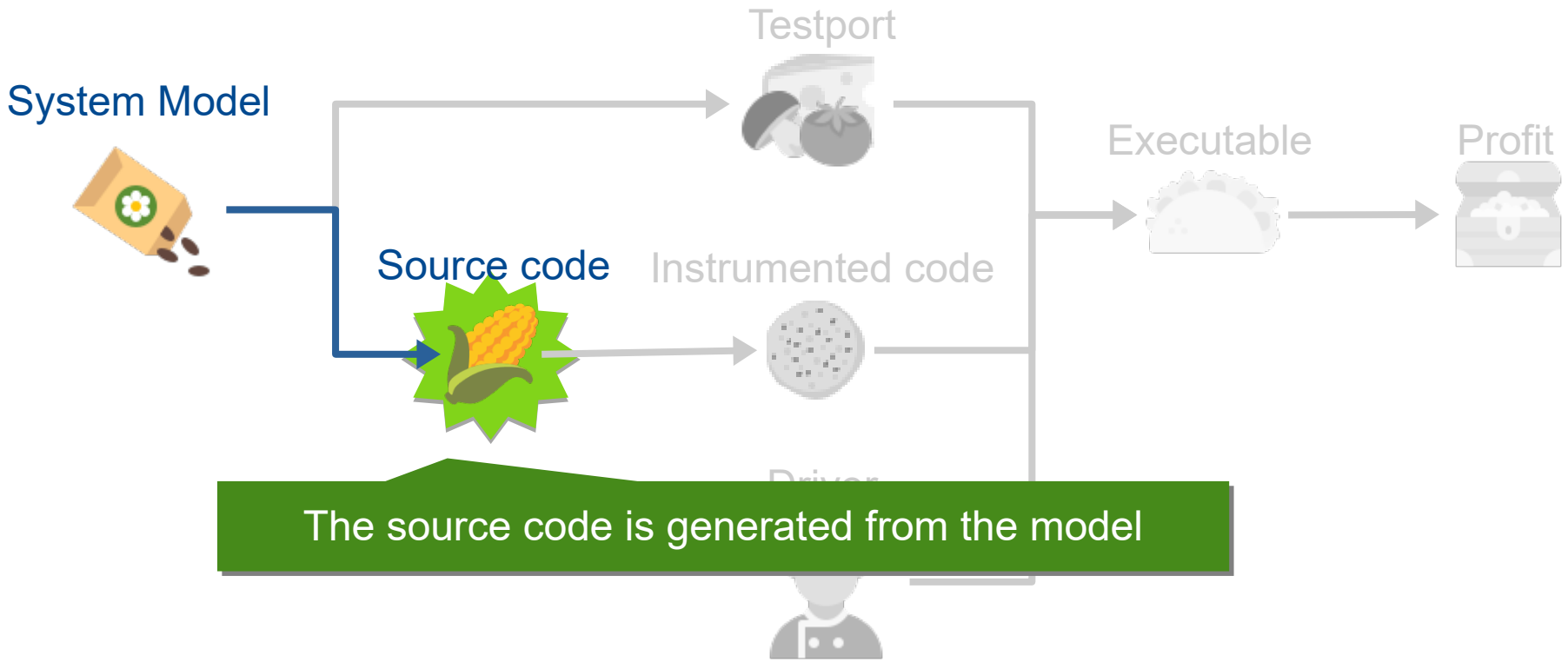


- Requires knowledge about functions' *inputs*, types and value ranges

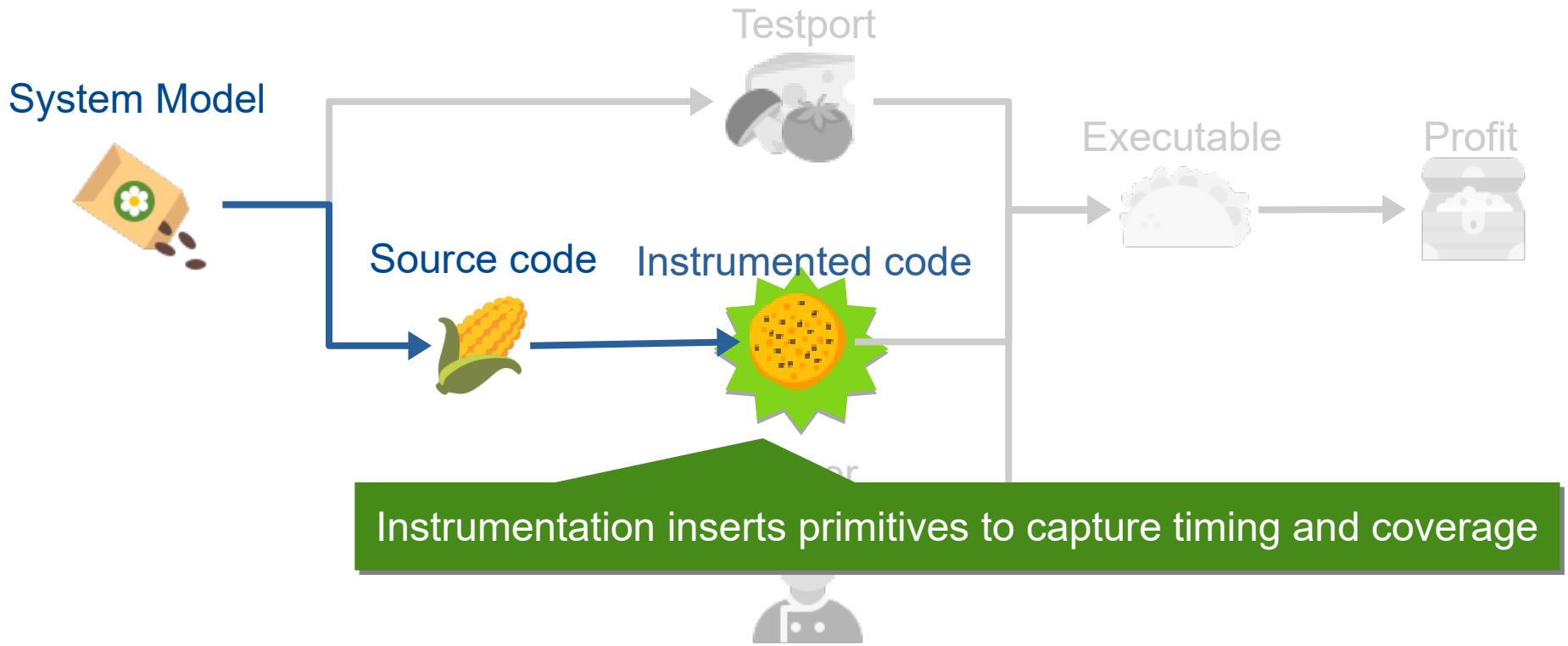
TACO - Overview



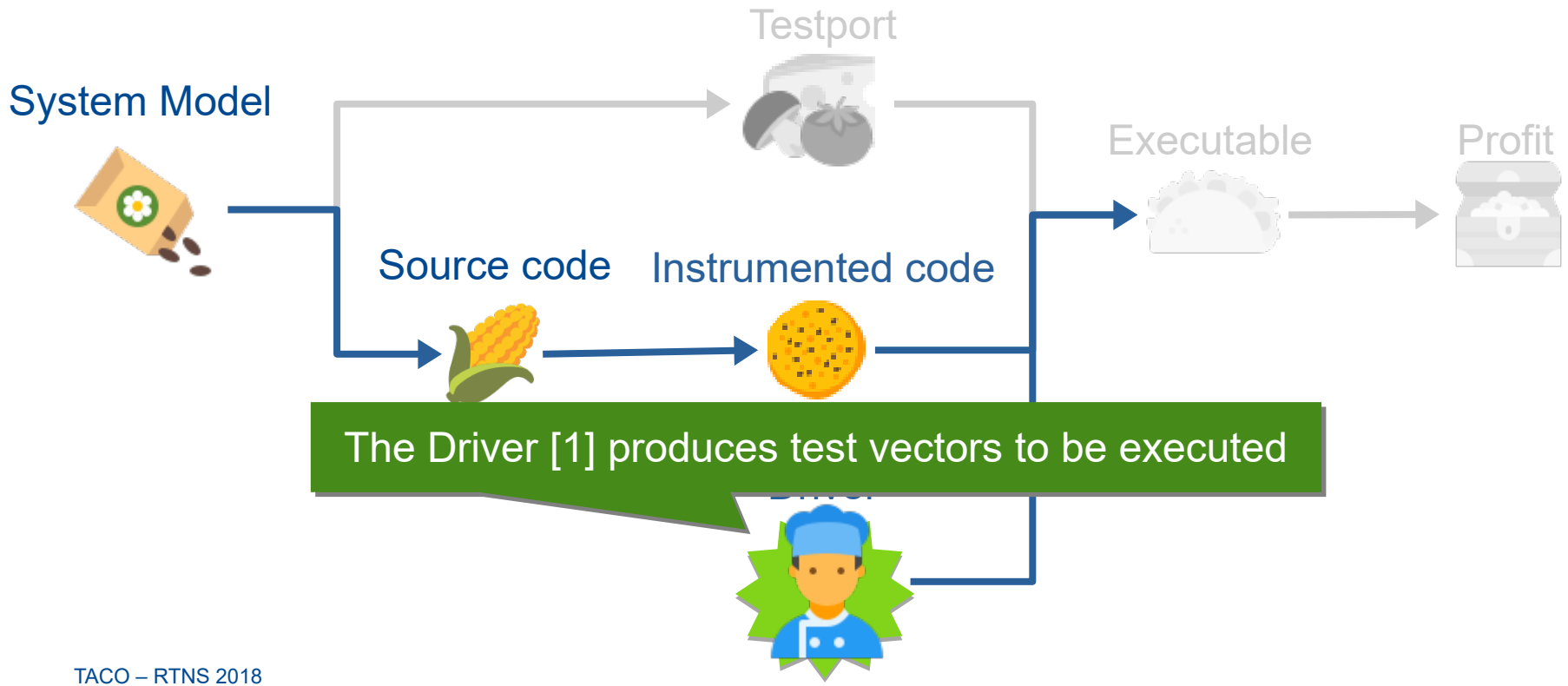
TACO - Overview



TACO - Overview

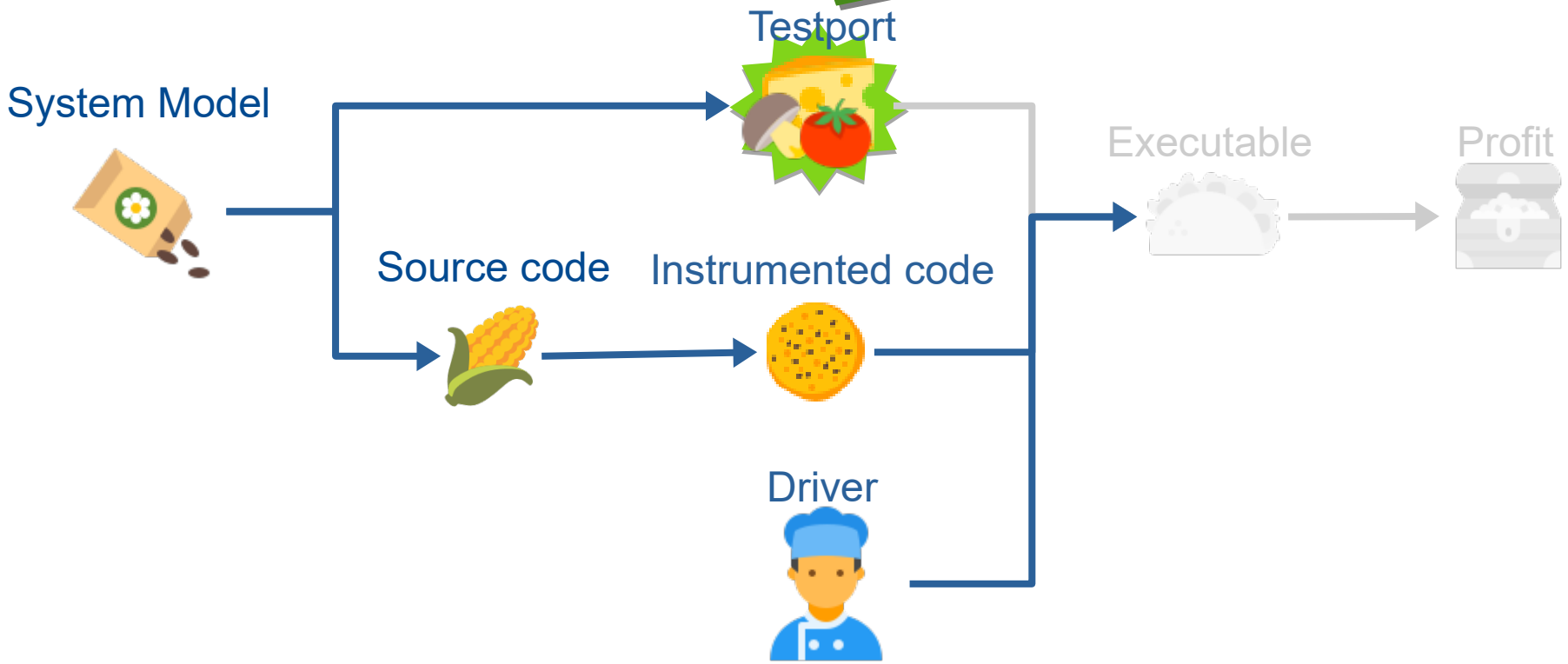


TACO - Overview



TACO - Overview

A testport is also generated from the system model



TACO – Testport

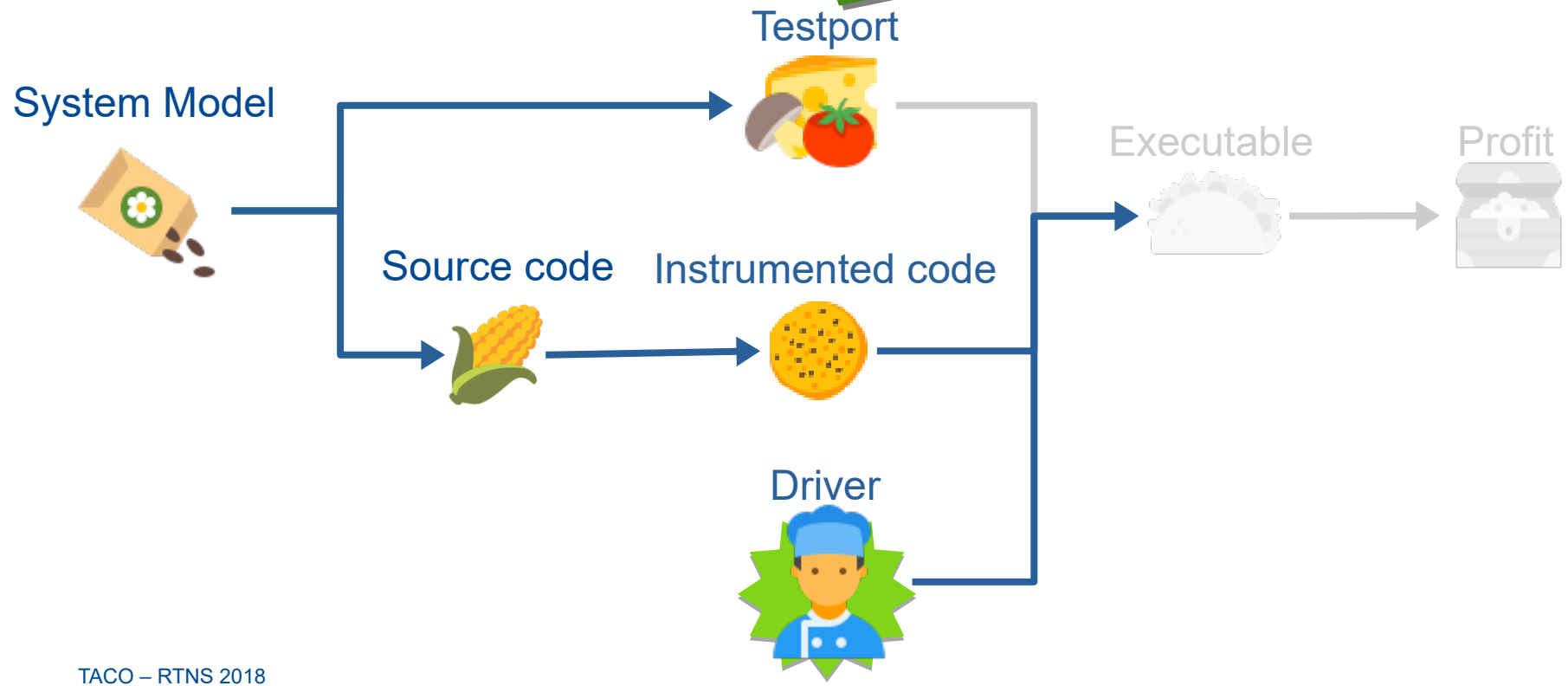
- The testport is the interface between the test function and driver
  - Initialises inputs, runs and measures the function
  - Provides feedback to the driver
- A common interface means item and driver can be swapped

TACO – Testport

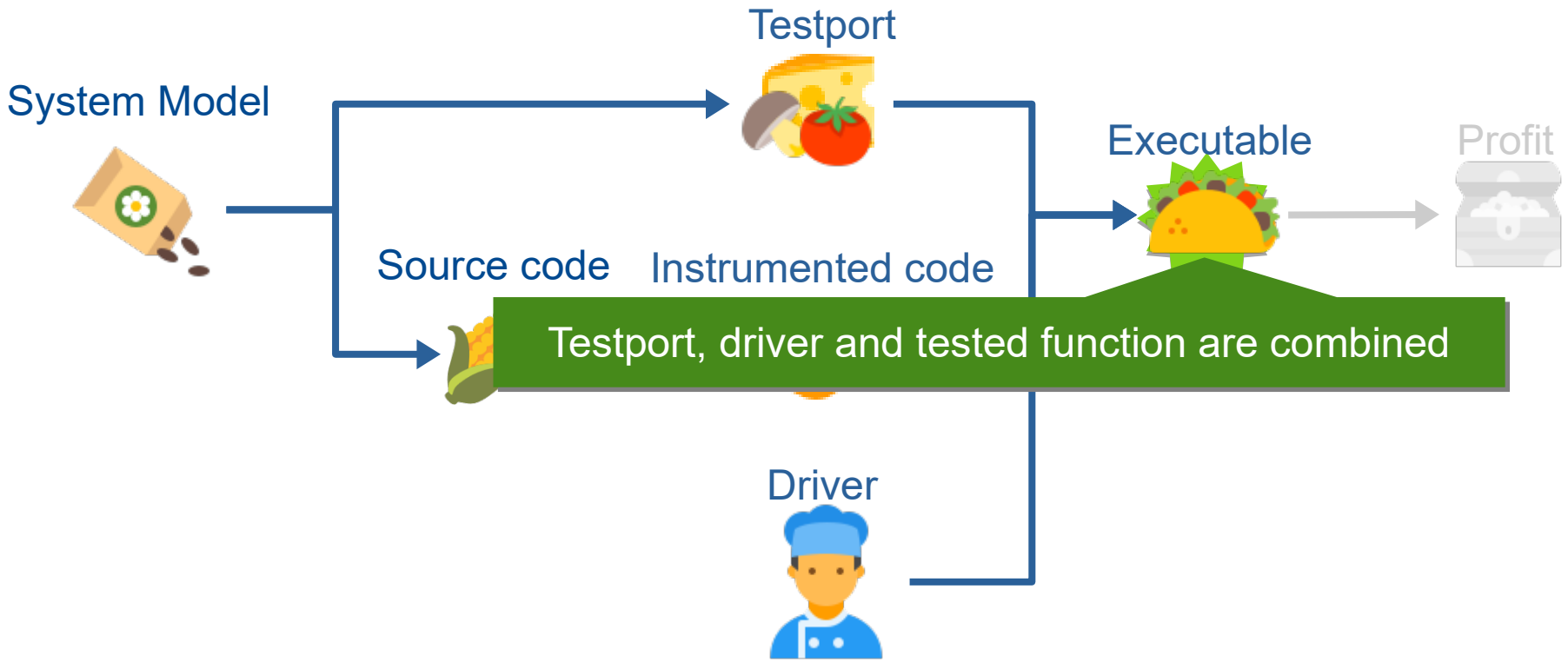
- The testport is the interface between the test function and driver
  - Initialises inputs, runs and measures the function
  - Provides feedback to the driver
- A common interface means item and driver can be swapped

TACO - Overview

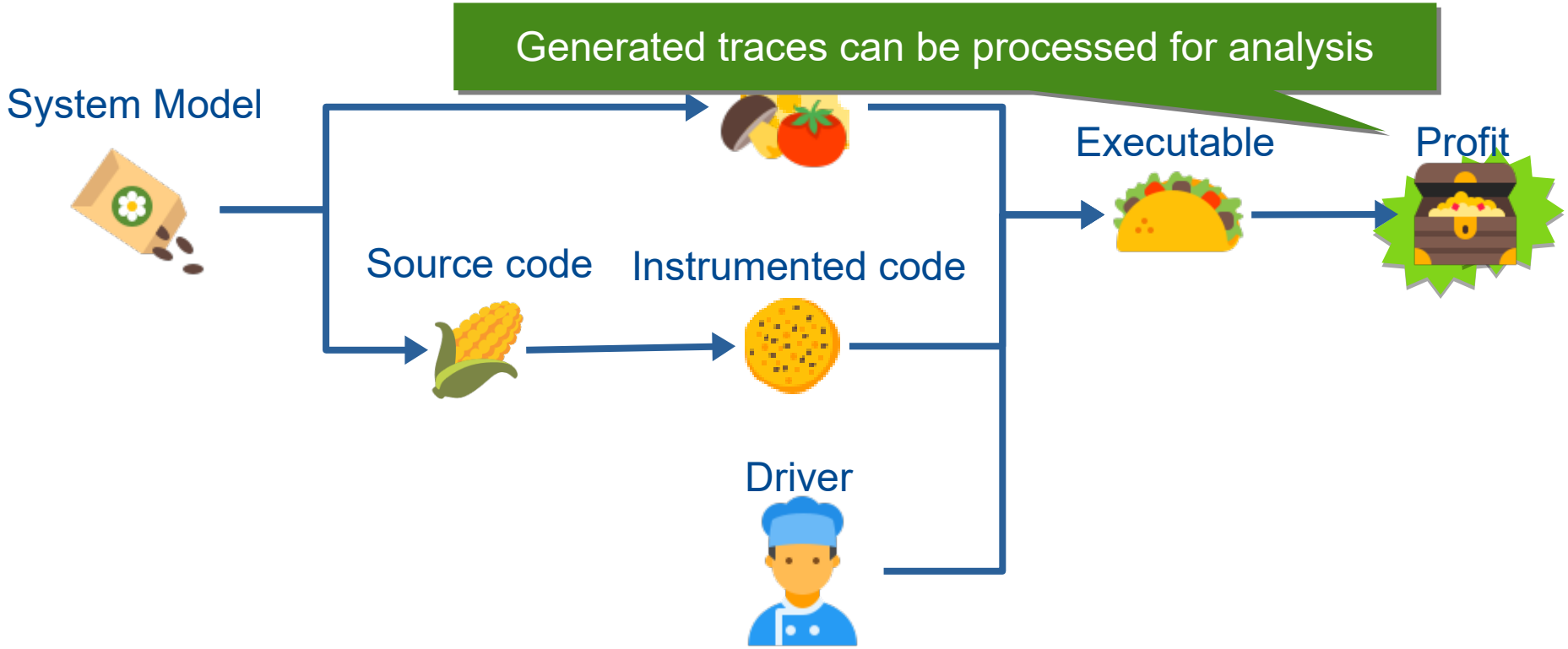
A testport is also generated from the system model



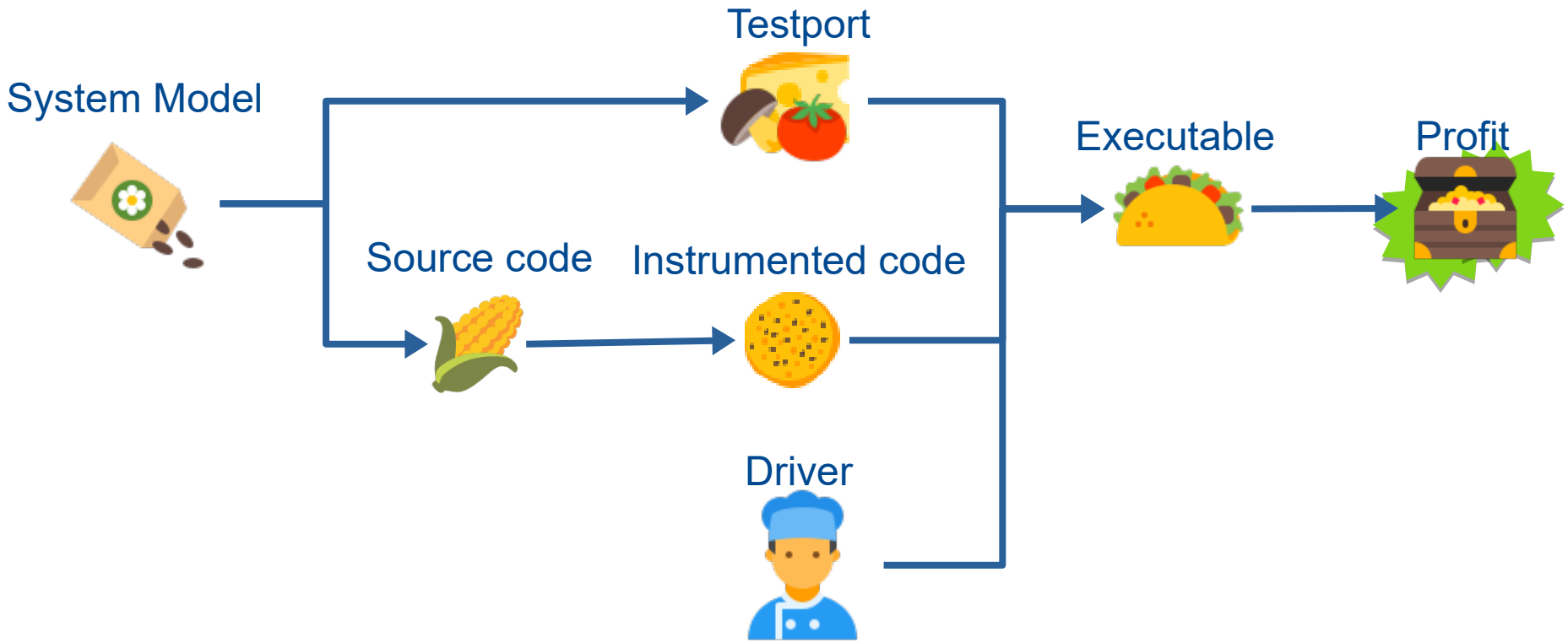
TACO - Overview



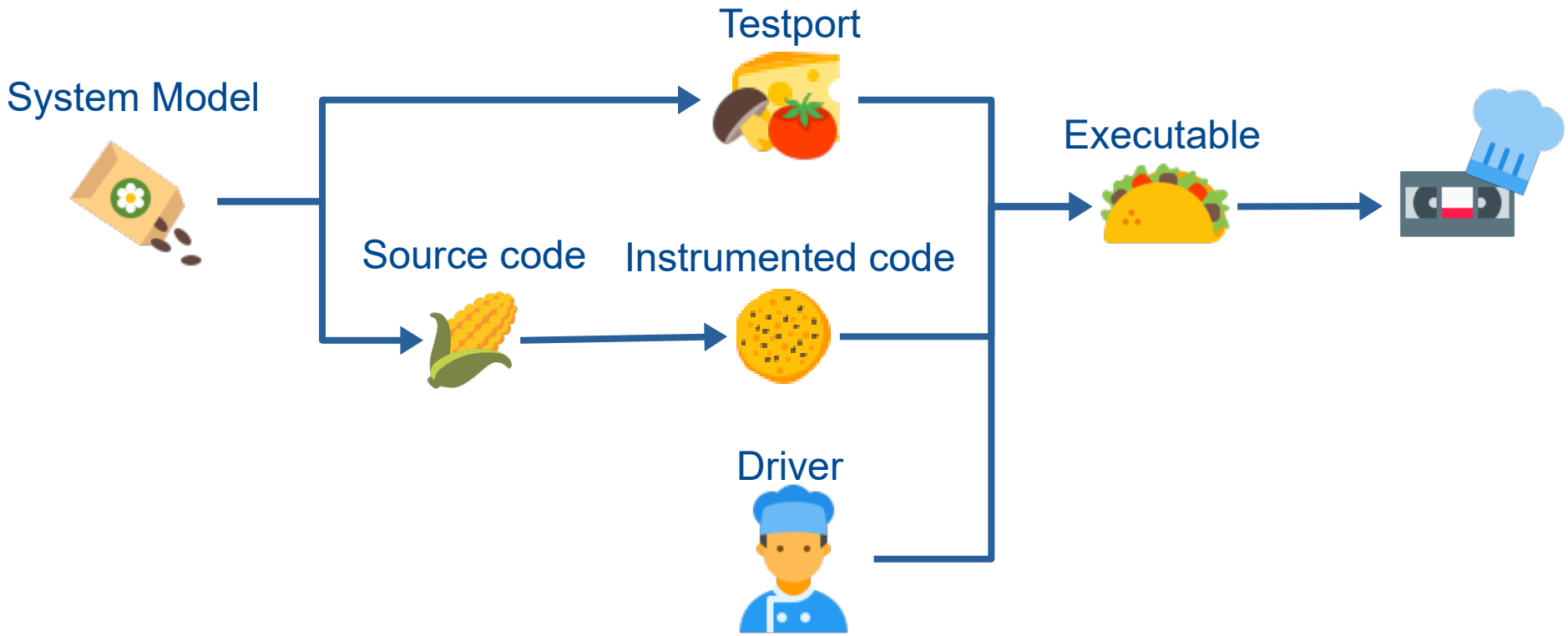
TACO - Overview



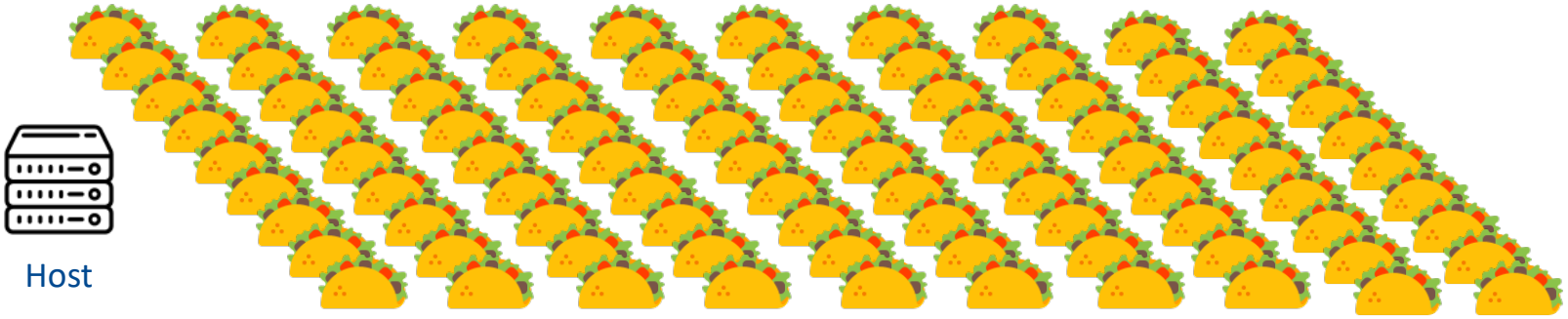
TACO - Overview



Reducing on-target testing

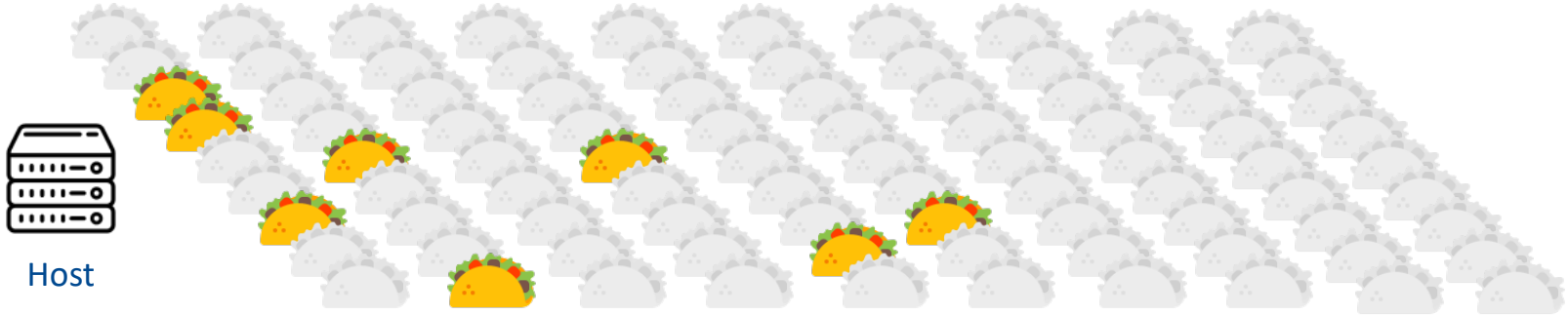


Reducing on-target testing



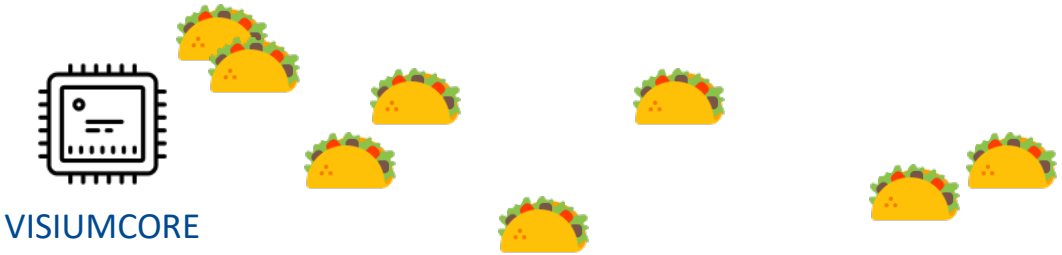
- The coverage technique [1] relies only on coverage information
  - Coverage is measured at the source level
  - **Coverage is platform-independent**
- Host-based testing can reduce the requirements on target
  - Collect coverage and inputs on host
  - Replay selection of tests on target

Reducing on-target testing



- The coverage technique [1] relies only on coverage information
  - Coverage is measured at the source level
  - **Coverage is platform-independent**
- Host-based testing can reduce the requirements on target
  - Collect coverage and inputs on host
  - Replay selection of tests on target

Reducing on-target testing



- The coverage technique [1] relies only on coverage information
  - Coverage is measured at the source level
  - **Coverage is platform-independent**
- Host-based testing can reduce the requirements on target
  - Collect coverage and inputs on host
  - Replay selection of tests on target

Evaluation - Objectives

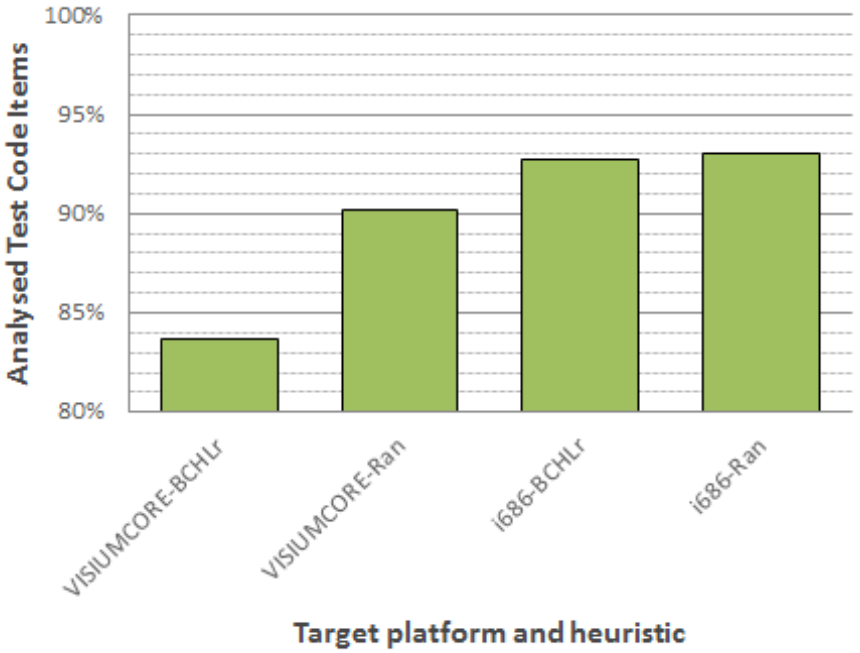
- **TACO scalability**
  - Can TACO be automatically applied to a full control system?
- **Driver portability**:
  - Can [1] be applied on different platforms?
- **Driver scalability**:
  - Can [1] be applied to a full control system?
  - What coverage can be achieved?
- **Reducing on-target testing**:
  - Can TACO reduce requirements on target?

Evaluation - Configuration

- Unmodified control system
  - Designed according to DO-178C as DAL-A
  - 1800 items for analysis, including 250+ scheduled tasks
- Two evaluation platforms:
  - Target: VISIUMCORE
  - Host: i686
- Two heuristics for driver:
  - Ran: Random search through input space
  - BCHLr: Search focused on unexplored branches and loops
- 100 runs of TACO per item, heuristic and platform

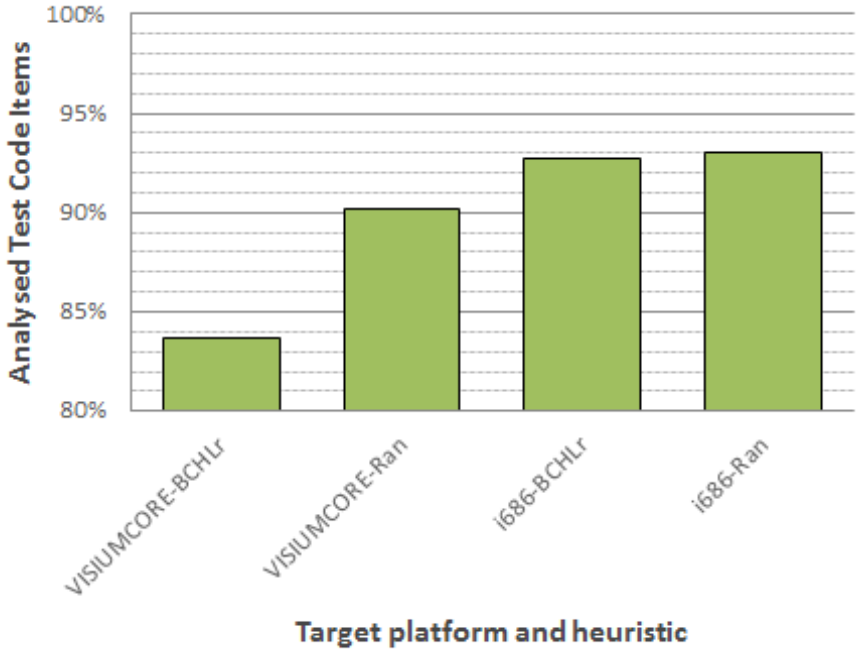
Evaluation - Scalability of TACO

- Count analysed items if:
  - TACO generates a testport
  - Compiles with drivers
  - Runs and generate traces
- Higher is better
  - Ore items processed y framework
  - Higher likelihood for coverage



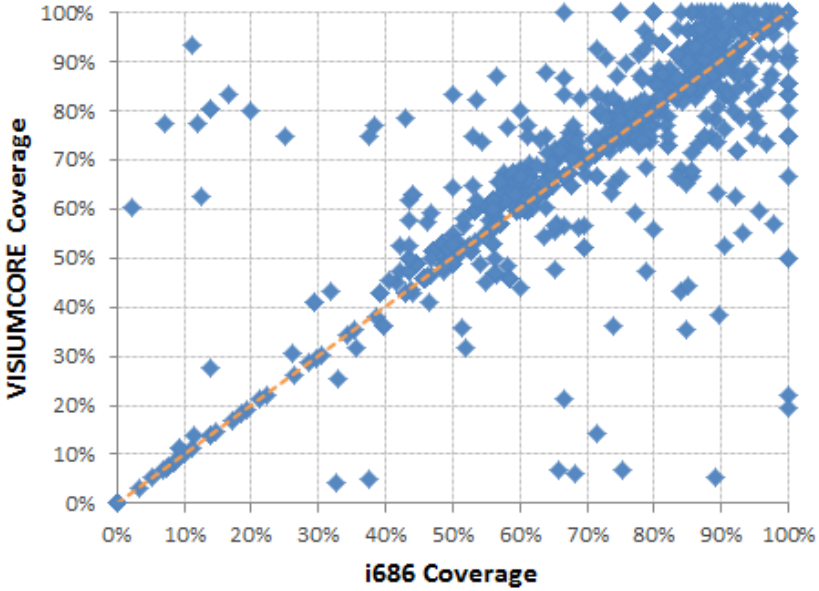
Evaluation - Scalability of TACO

- More than 90% items analysed
  - Contextual information missing from model
- Less items analysed on VISIUMCORE
  - Driver is platform independent
  - but lower resources on target
- Less items analysed using BCHLr
  - Higher-entry memory requirements



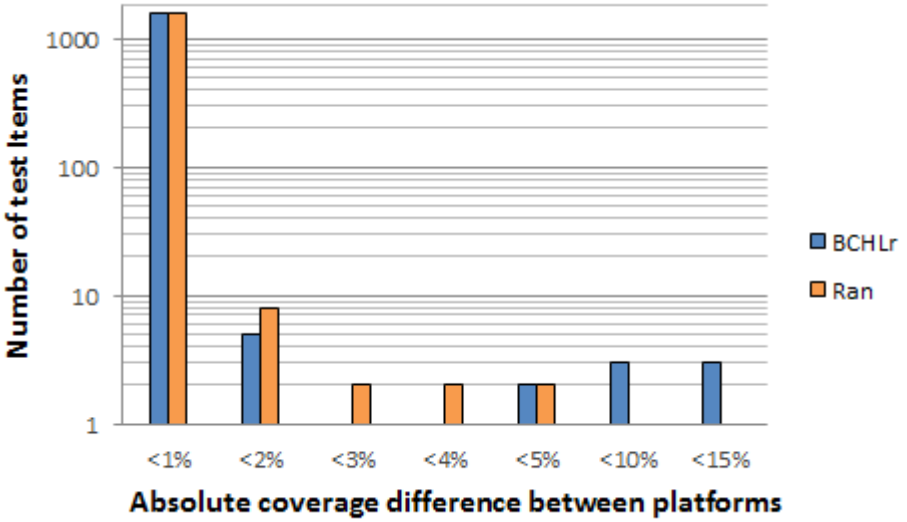
Evaluation - Portability of Driver

- Compare coverage
  - Across platforms
  - For a same item
  - For a same heuristic
- Closer to **y=x** is better
  - Same performance across platforms
  - Coverage collectable on Host

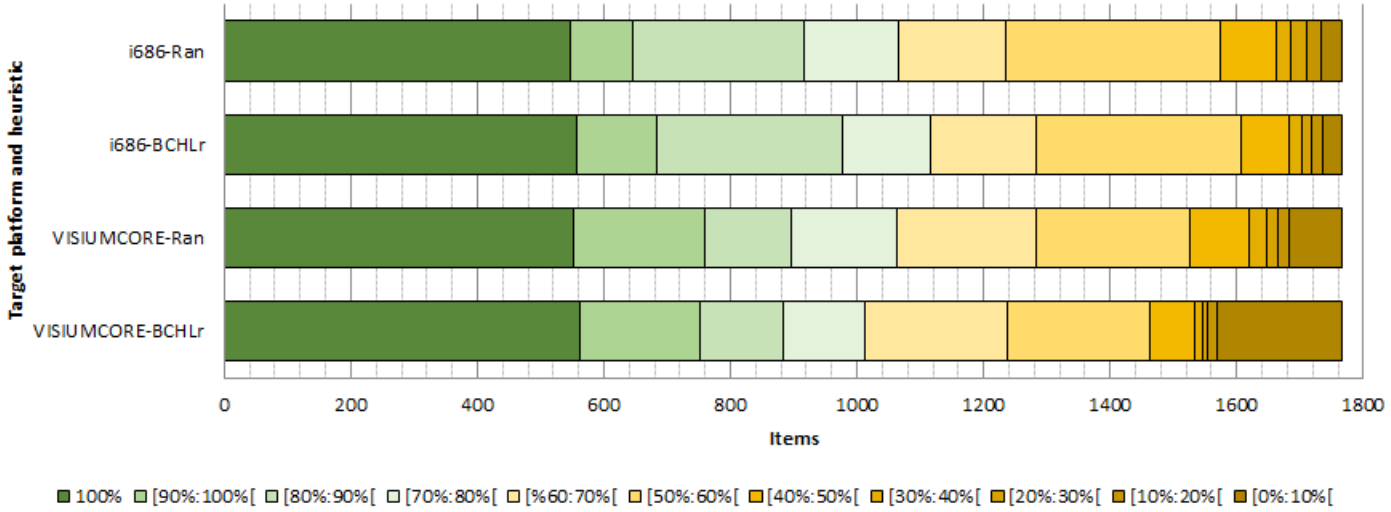


Evaluation - Portability of Driver

- Compare absolute coverage difference
  - Across platforms
  - For a same item
  - For a same heuristic
- Lower is better
  - Same performance across platforms
  - Coverage collectable on Host
- Small differences across platforms
  - Less than 15 functions > 1% difference
  - Variations due to PRNG

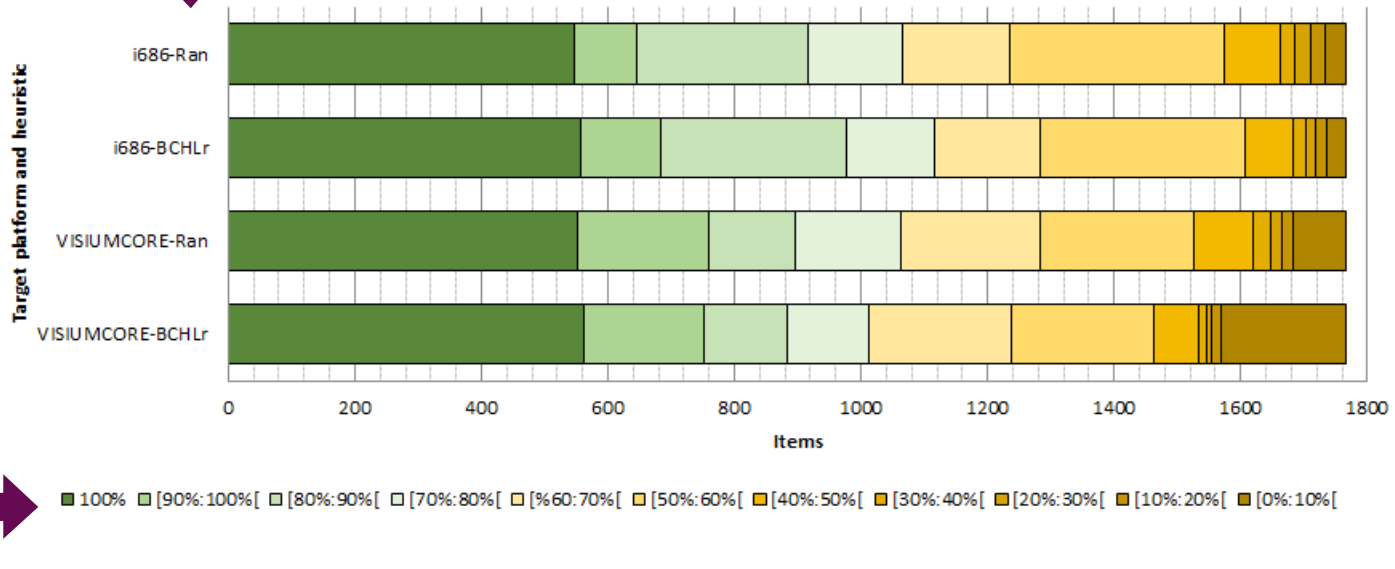


Evaluation - Coverage



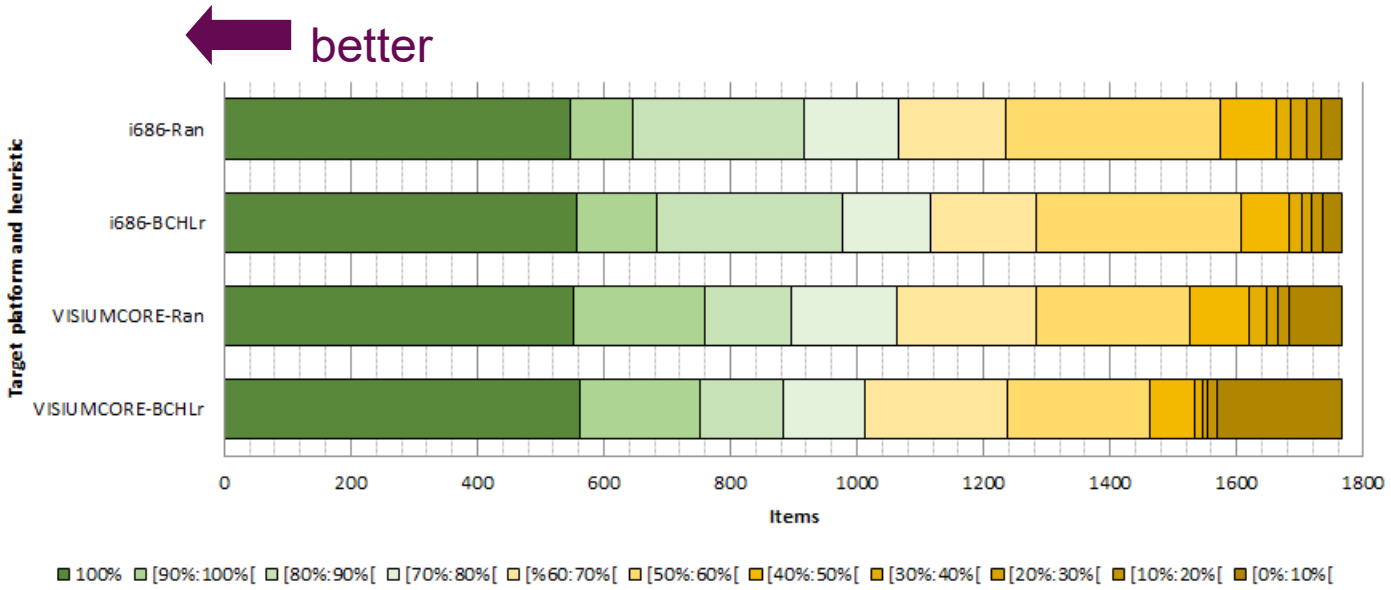
Evaluation - Coverage

Platform and driver

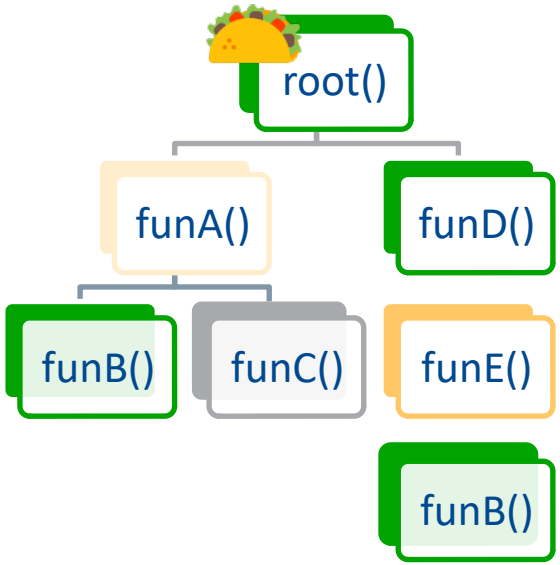


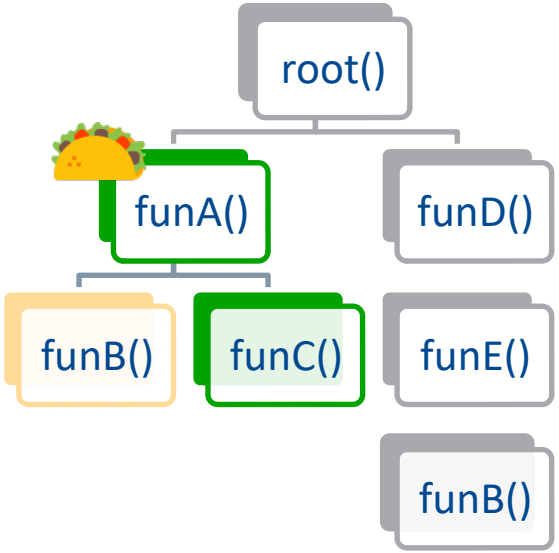
Achieved coverage level

Evaluation - Coverage

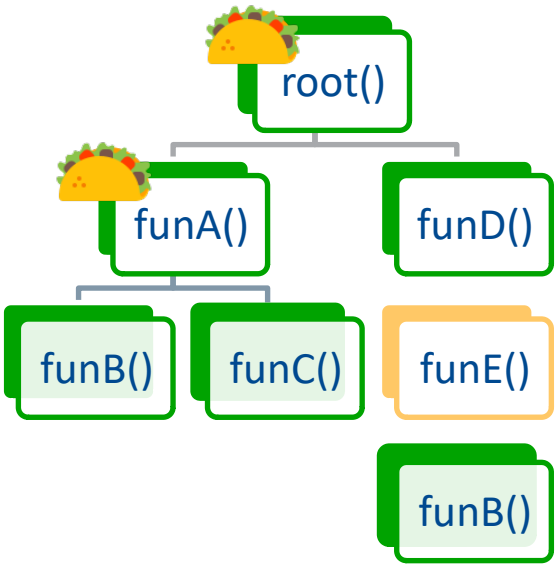


Evaluation - System-wide coverage

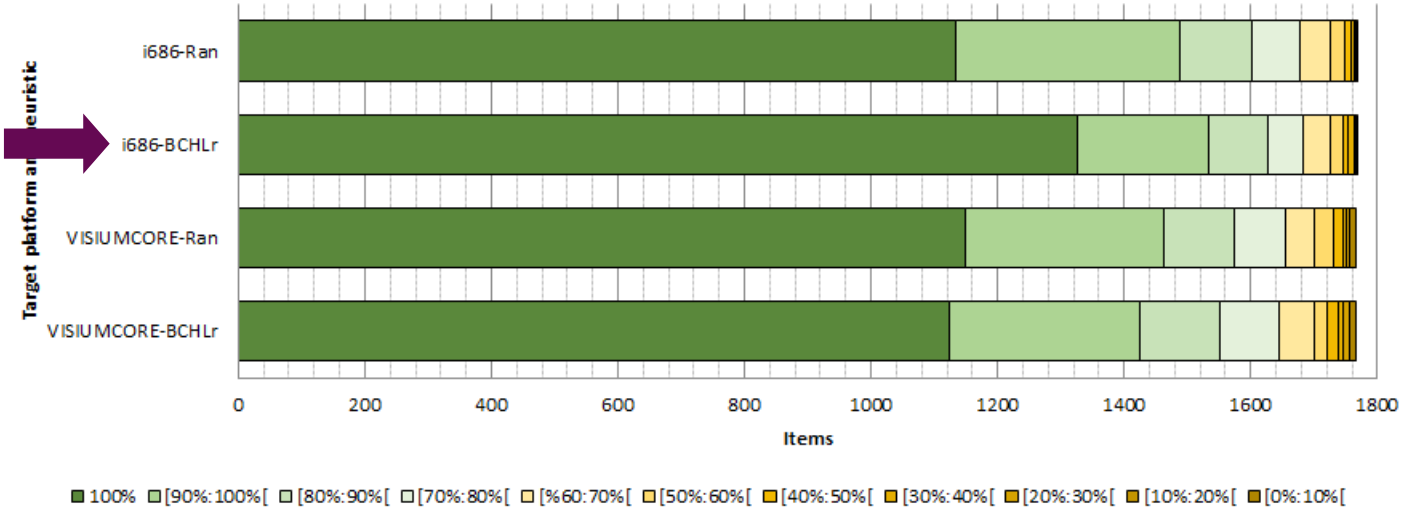




Evaluation - System-wide coverage

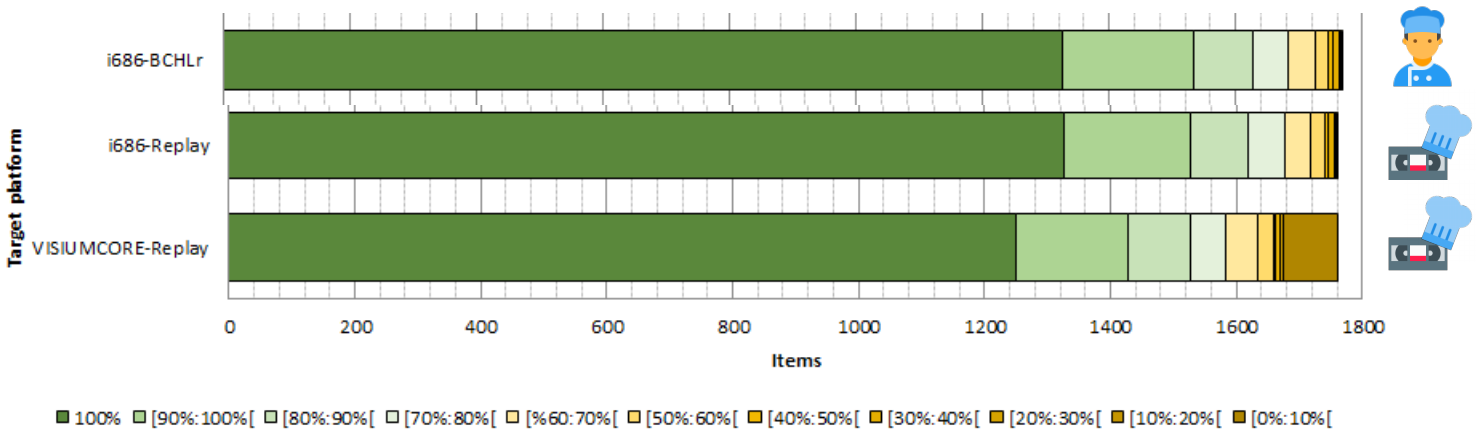


Evaluation - System-wide coverage



- More functions supported on i686-BCHLr
- Less items compiled with VISIUMCORE-BCHLr

Evaluation - Reducing on-target testing



Same coverage achieved on Host
22 Millions test vectors, 48 Hours

6131 test vectors, 658 items, 4 Hours on target

Conclusion

- Test Automation for COverage
  - Exercise all inputs of a test item
  - Drive the code through different paths
  - Drive search for specific targets
- Scales to full system analysis
- Achieves reliable test coverage
- Reduces on-target testing
- Offers a collection of interacting tools
  - Easy to parse, reuse, and extend

Future work

- Provide timing estimates early in the application life cycle
  - Rely on low-cost platforms
  - Correlate timings on Target and Host
- Quantify confidence and certainty in achieved coverage
  - Define stopping criteria for search algorithm
  - Understand returns of additional computational effort
- Refine test vector selection from host results
  - Target longest execution paths
  - Assess changes to the software