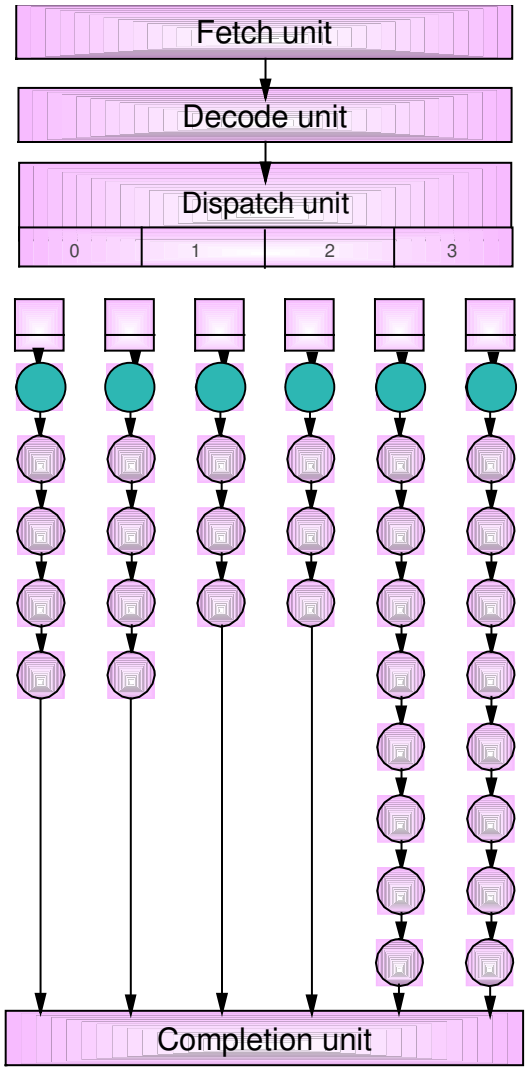
Deep Knowledge Test Generators & Functional Verification Methodology

IBM Verification Seminar

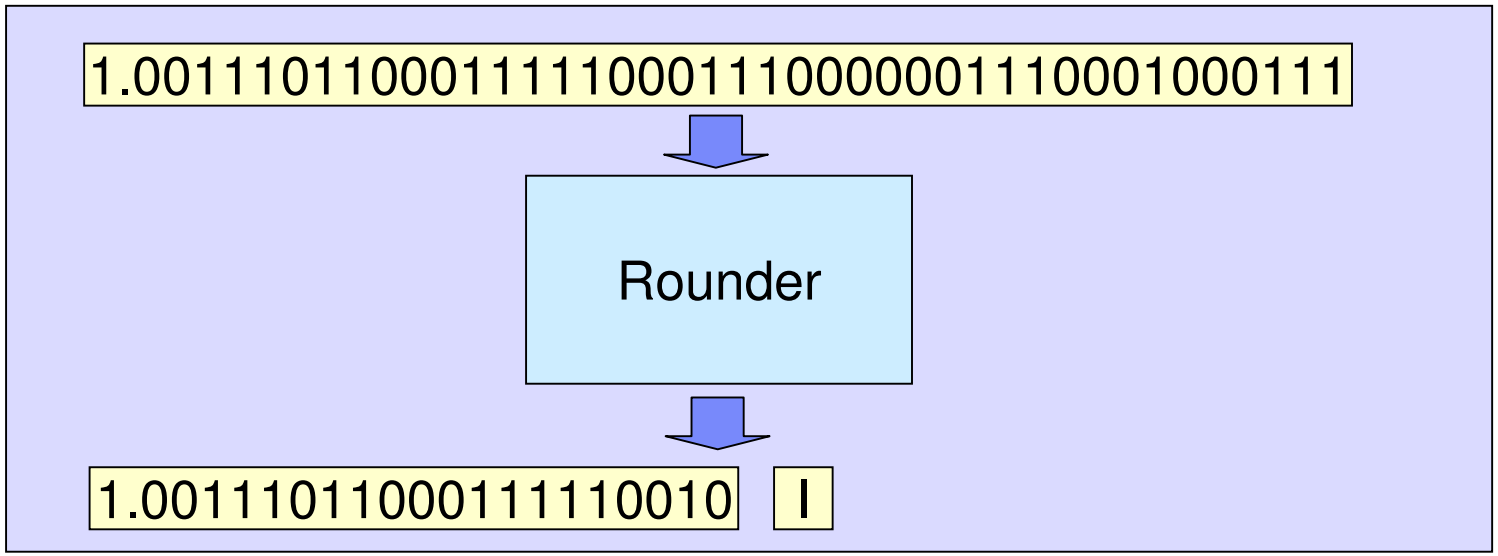
October 2003 - Laurent Fournier

IBM Labs in Haifa

All Starting Stages in Holding Position



Extreme Case of Influence on Sticky Bit



1.ccccccccccccccccccccccc0000000000000000000001

Motivation for Deep Knowledge Test Generation

Gap + Bug-Prone

Control of test generator

Interesting scenarios

Deep Knowledge Test Generator?

Definition

- Test generation focused on specific verification areas, e.g. FPU, Microarchitecture Flow, SCU

Problem

- Inefficiency of generic architecture tools in specific error-prone verification areas (bugs not found or found late)

Objectives

- To provide greater control to reach non-covered areas
- To enable a systematic and comprehensive verification approach to speed up the rate of coverage

The Players

FPgen

- Generic, quasi-optimal solution for a well-defined, albeit complex, field
- Complex mathematic algorithms

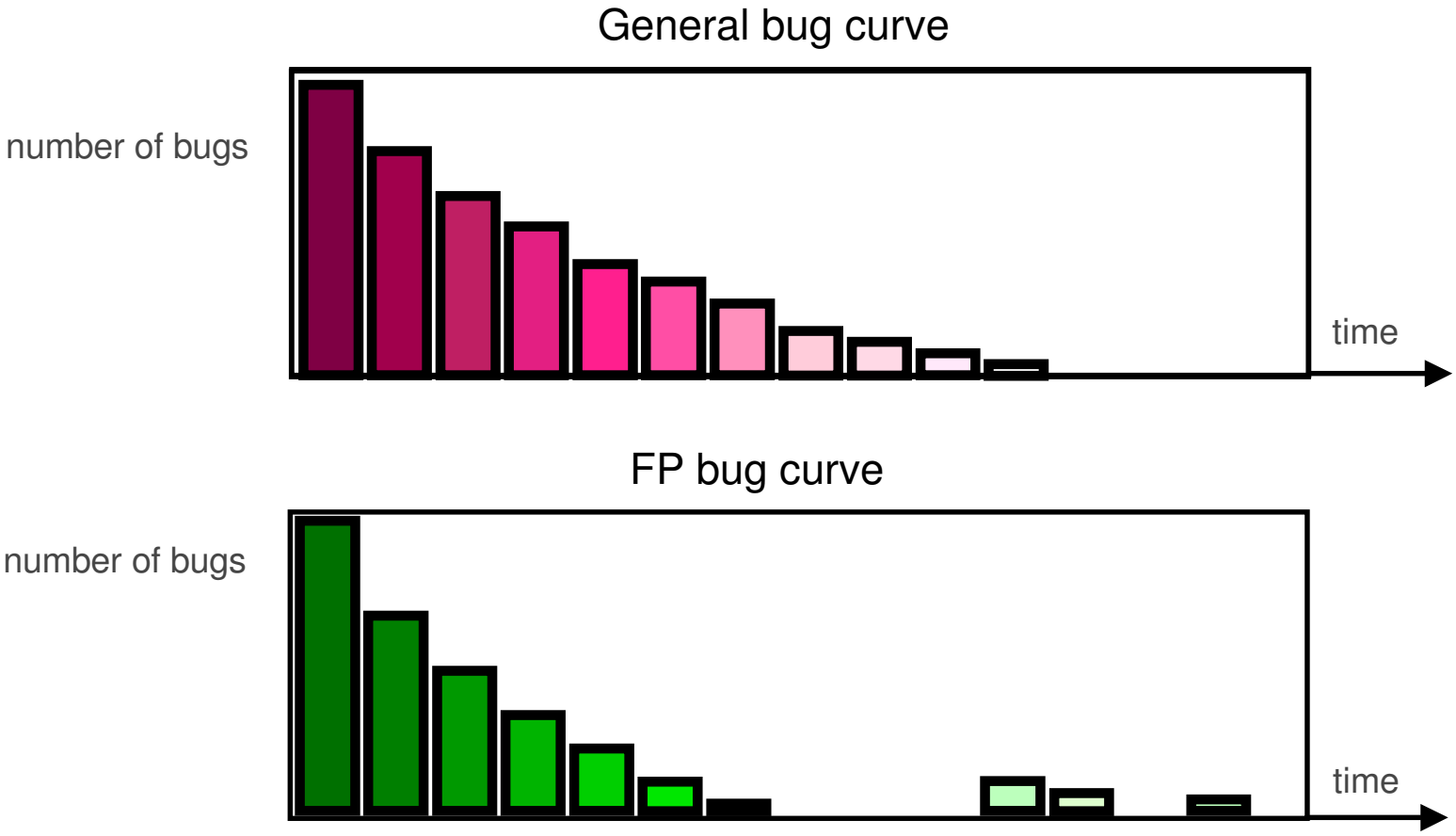
Piparazzi

- An evolving solution to cope with microarchitecture flow complexity
- Classic constraint solving engine (CSP)

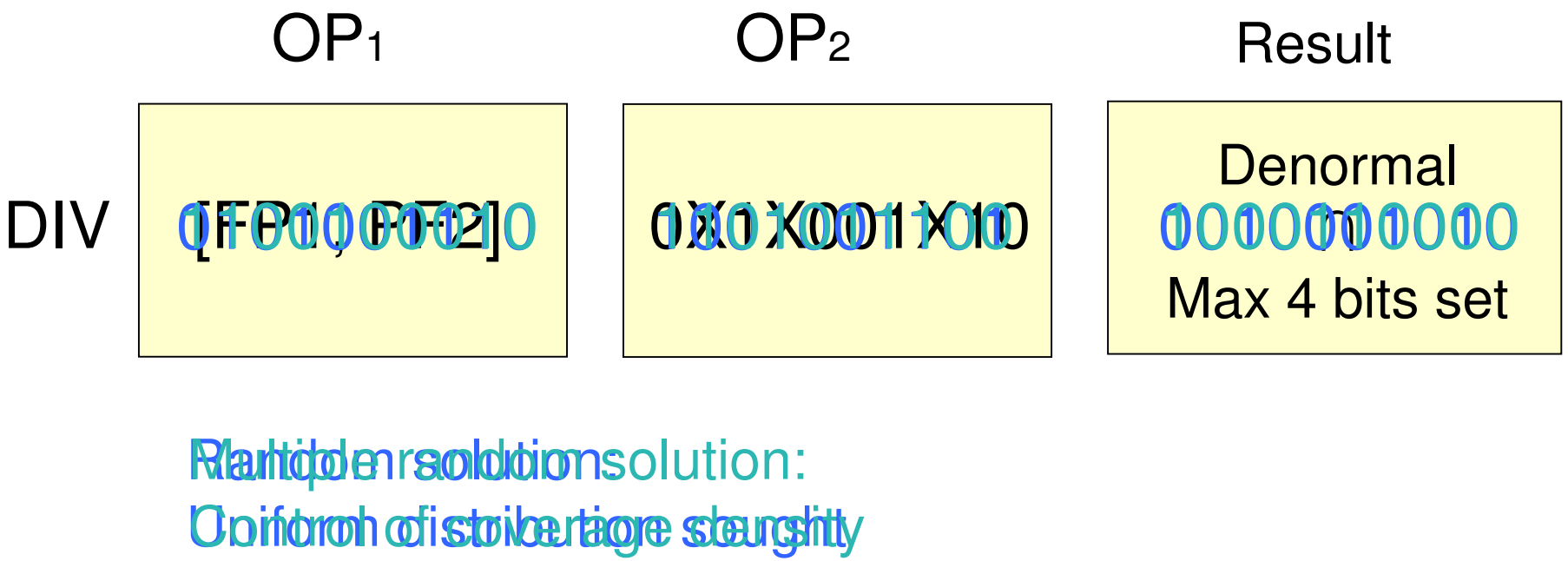
DeepTrans

- More a library of services for now
- Rounds off the model-based technology

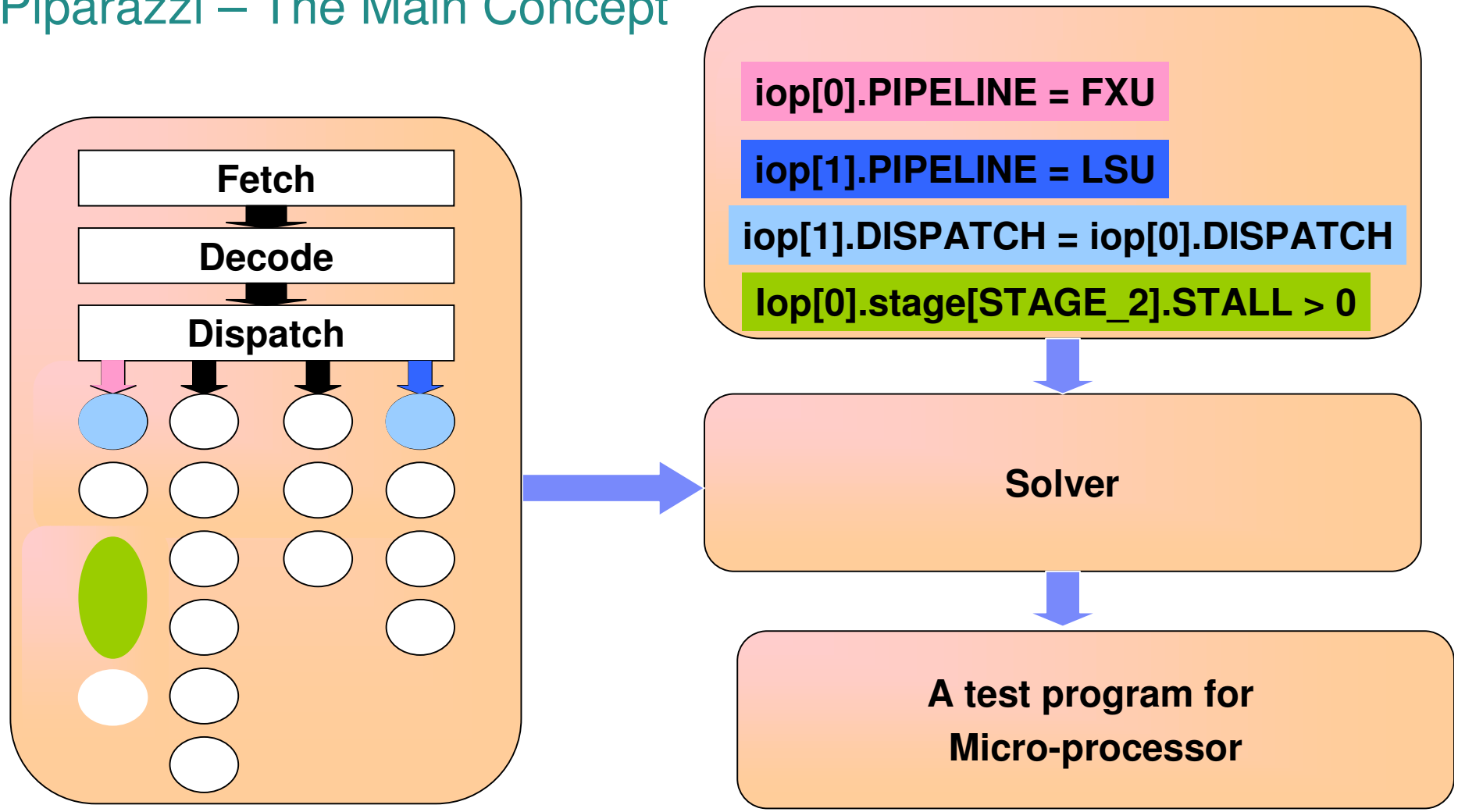
Motivation: Floating Point Bugs



Basic Functionality



Piparazzi – The Main Concept

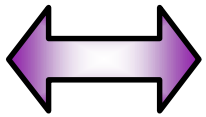


Input can be a Full Cross-Product

Example: All types model

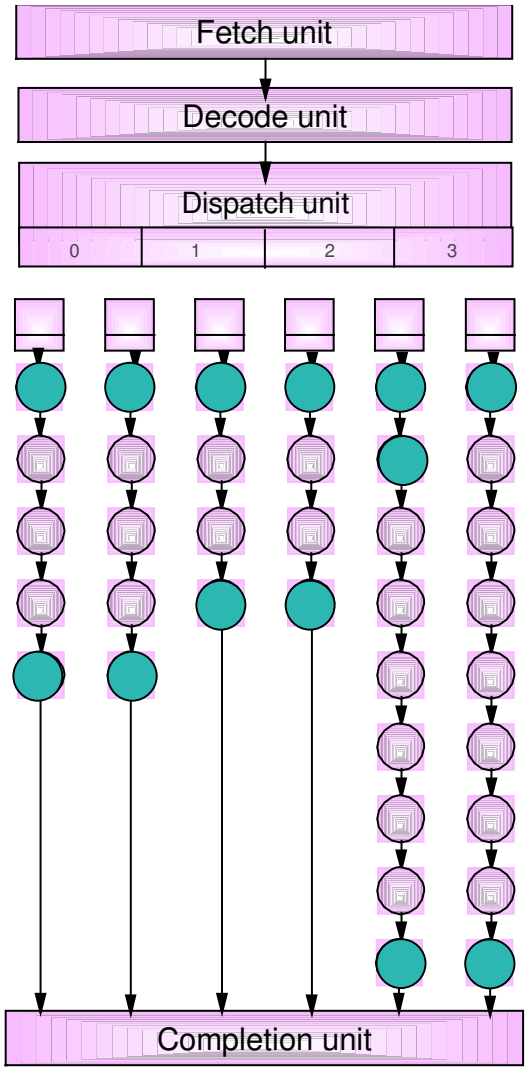
Operand1	Multiply	Operand2	=	Result
+/- Infinity +/- Zero +/- Norm +/- Denorm +/- Large number +/- Small number +/- Min Denorm +/- Max Denorm +/- Min Norm +/- Max Norm	X	+/- Infinity +/- Zero +/- Norm +/- Denorm +/- Large number +/- Small number +/- Min Denorm +/- Max Denorm +/- Min Norm +/- Max Norm	X	+/- Infinity +/- Zero +/- Norm +/- Denorm +/- Large number +/- Small number +/- Min Denorm +/- Max Denorm +/- Min Norm +/- Max Norm

Relationship between Generator Input Language and Test-plan

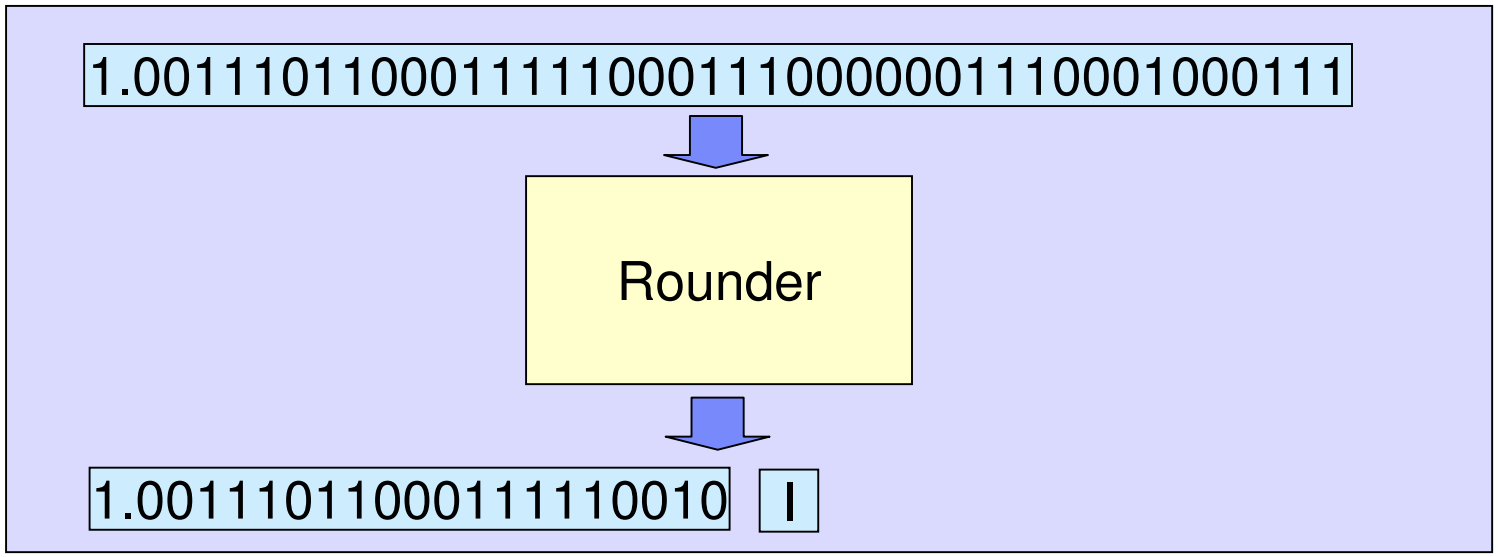
FPgen  **Test-Plan**

"Language shapes the way we think,
and determines what we can think about" - B.L.Whorf

Generalized Piparazzi Example



Generalized FPgen Example



1.ccccccccccccccccccccc0000000000000000000001

1.ccccccccccccccccccccc0000000000000000000010

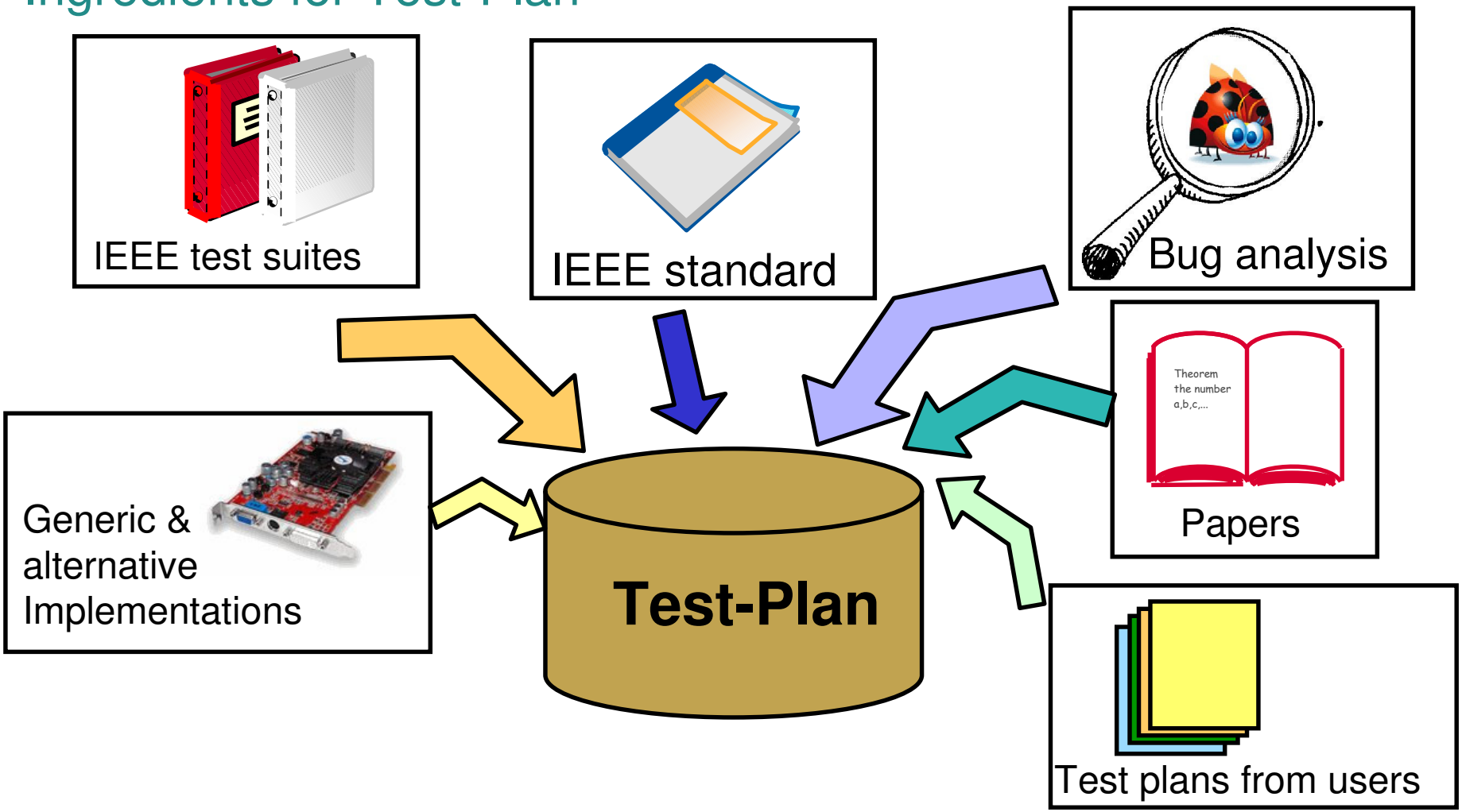
…

1.ccccccccccccccccccccc1000000000000000000000

Generic Test Plans

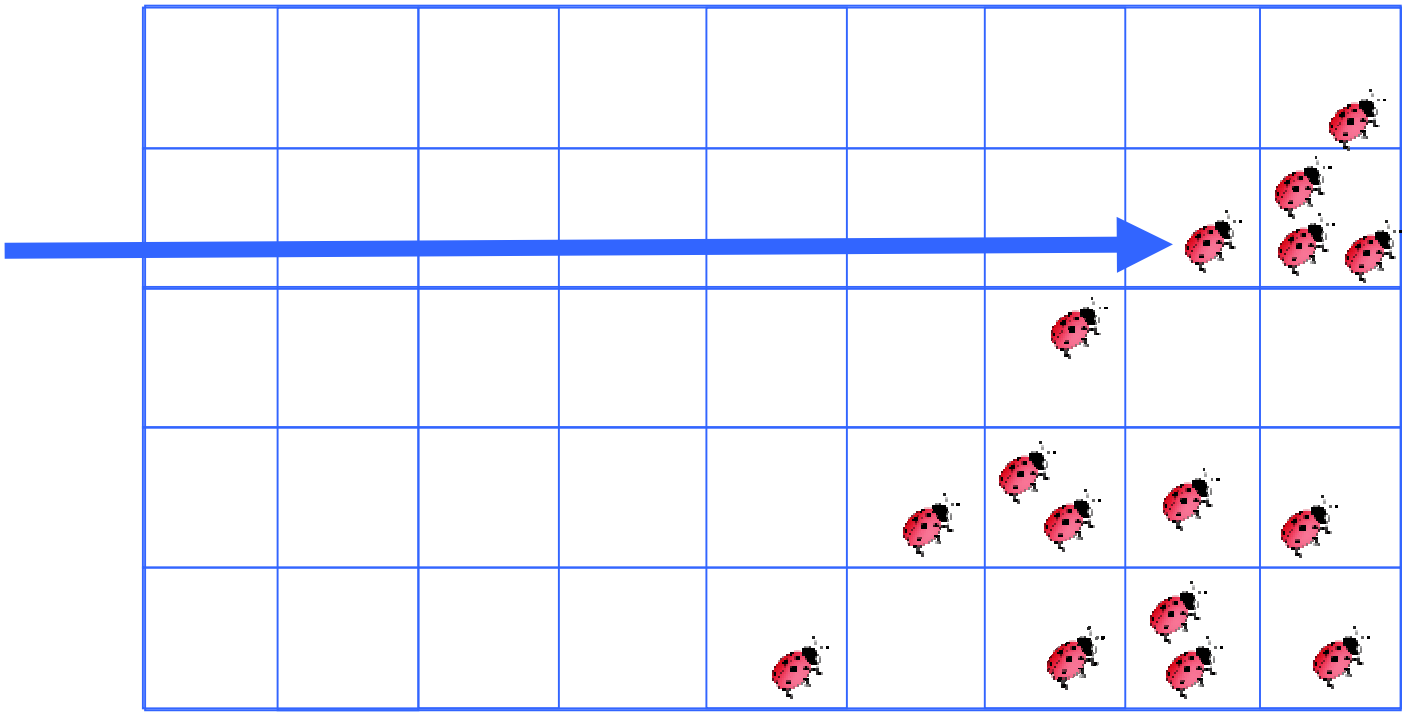
- Methodology – Resource Dependent
  - Density
  - Crossing
  - Reduction
  - Huge models

Ingredients for Test-Plan



The Evolution of Functional Verification Methodology

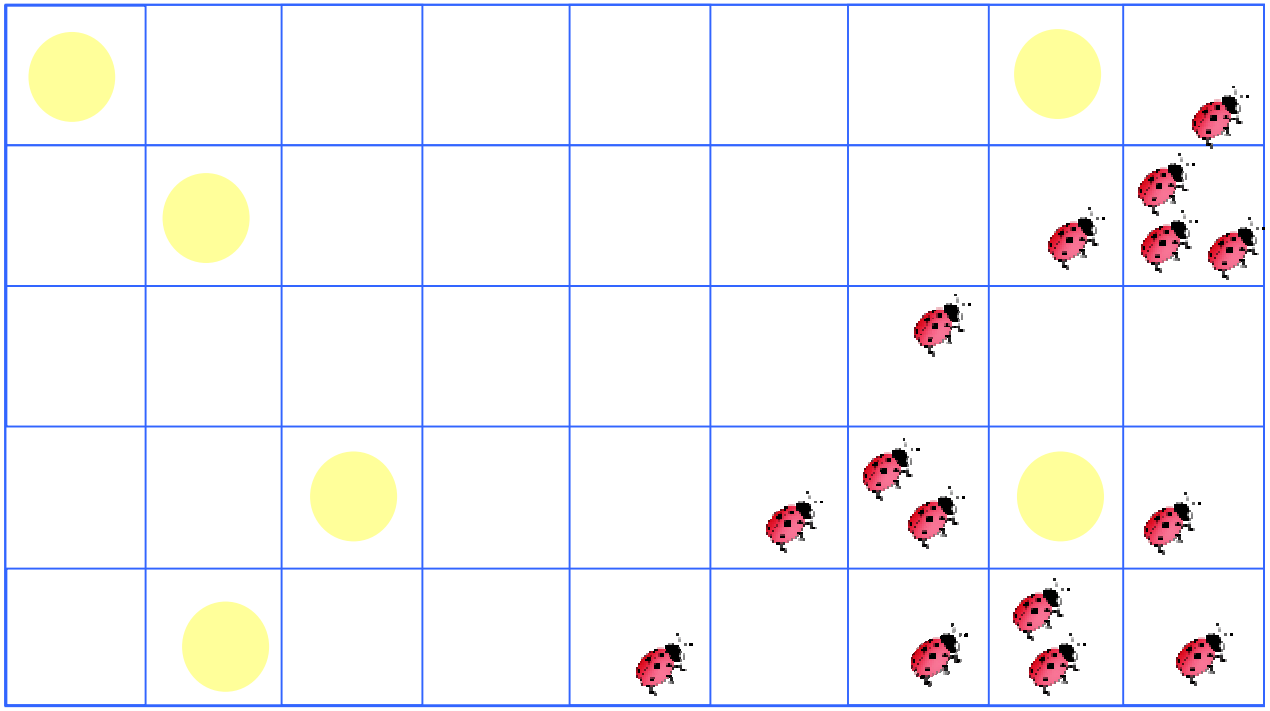
Small design: manual to reach all suspected cases



Slow and tedious

The Evolution of Functional Verification Methodology

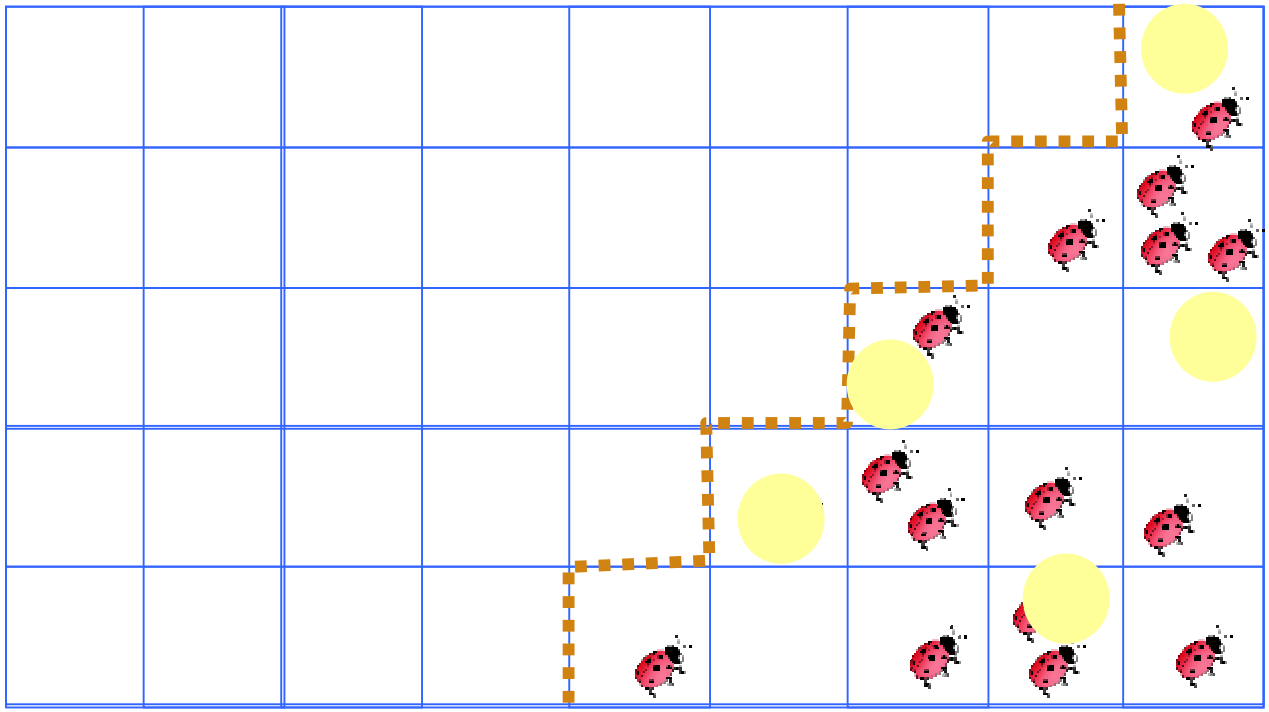
Increased complexity: Random generators



No uniformity in bug location probability

The Evolution of Functional Verification Methodology

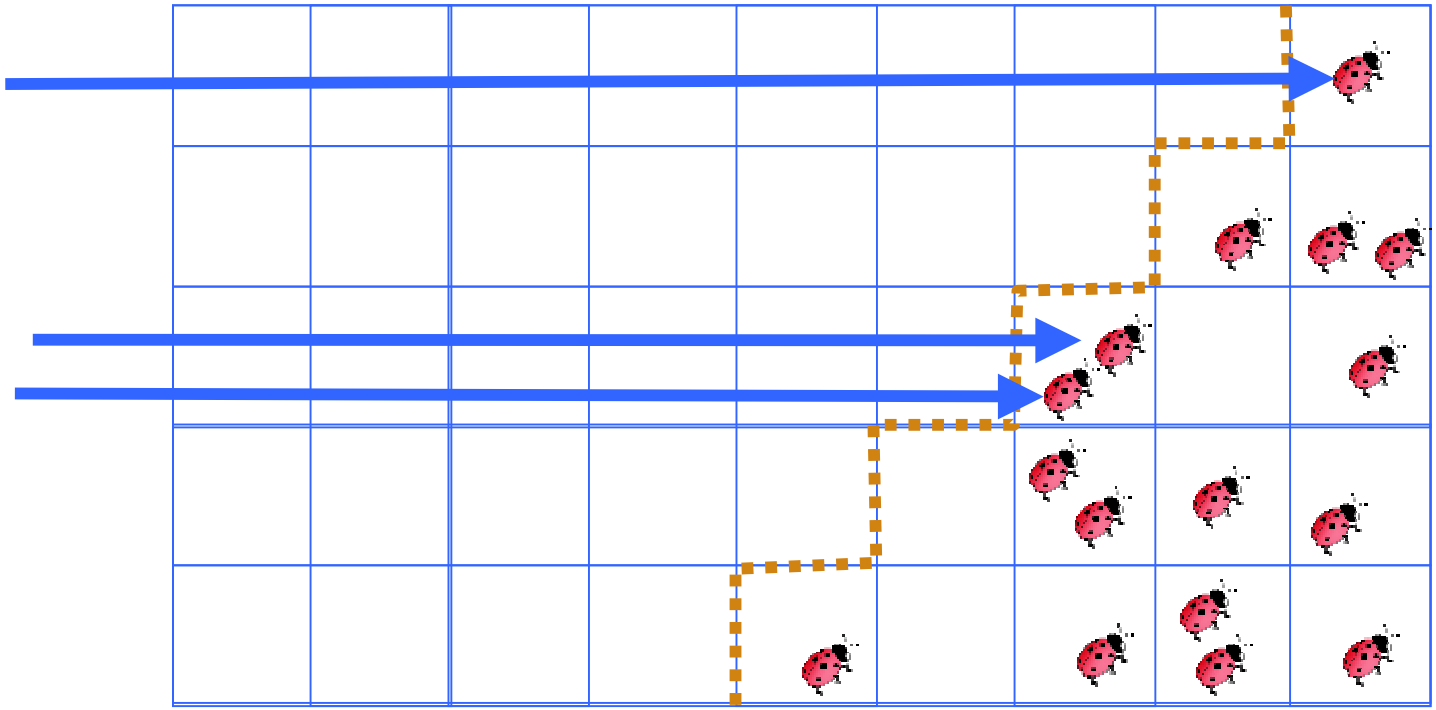
Biased Random generator to control towards suspected areas



Faster and more effective

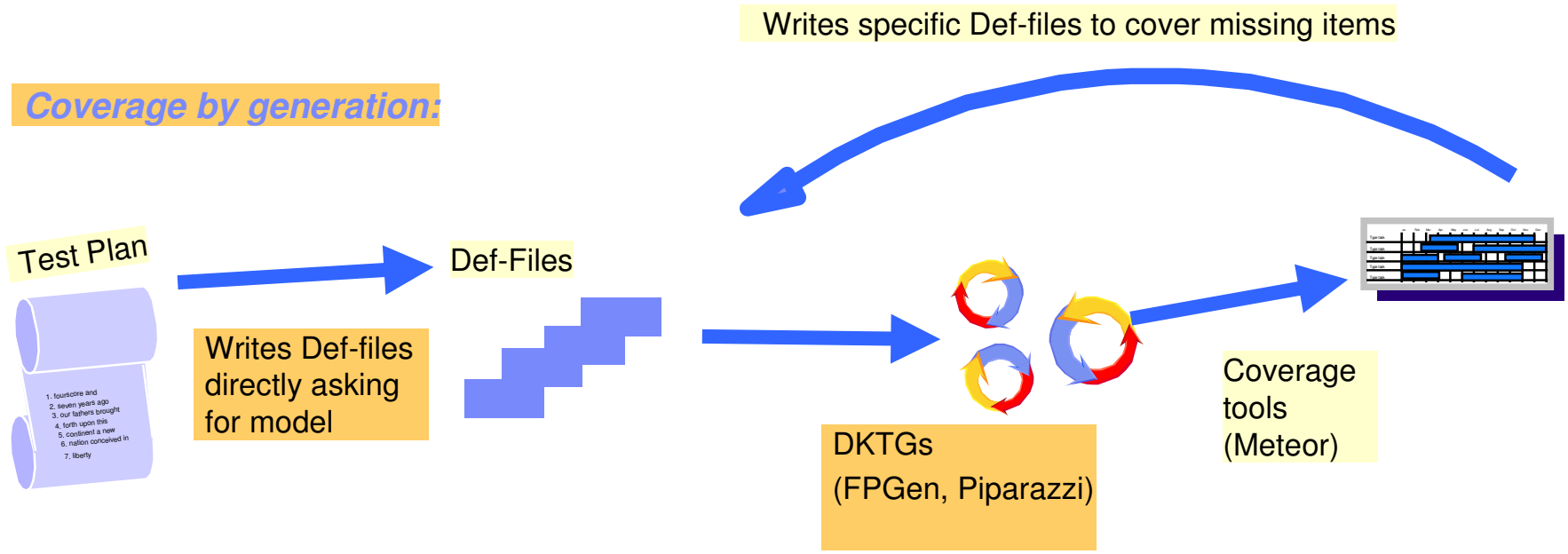
The Evolution of Functional Verification Methodology

Deep Knowledge Test Generators

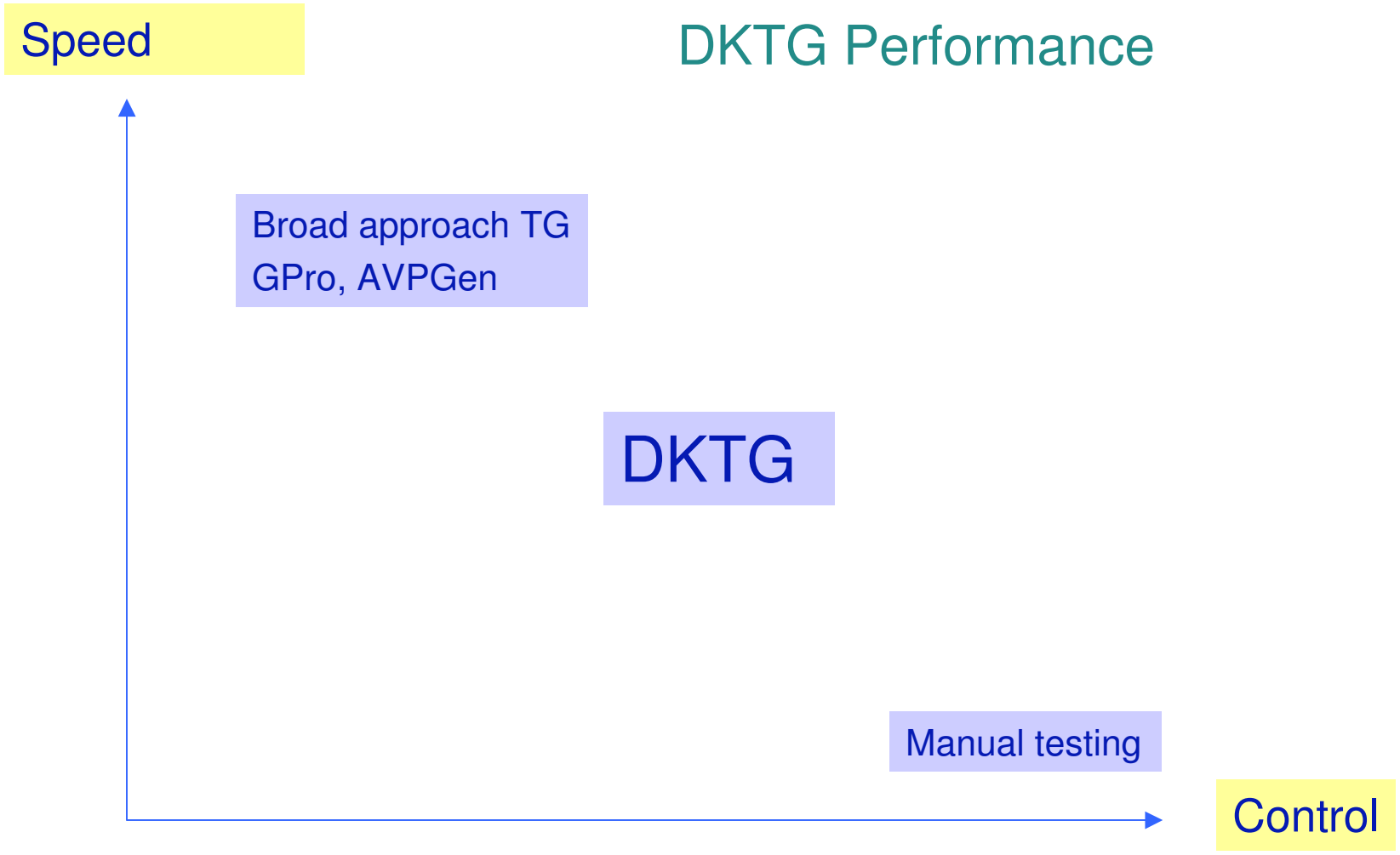


Many fast accesses to areas with a high probability of bugs

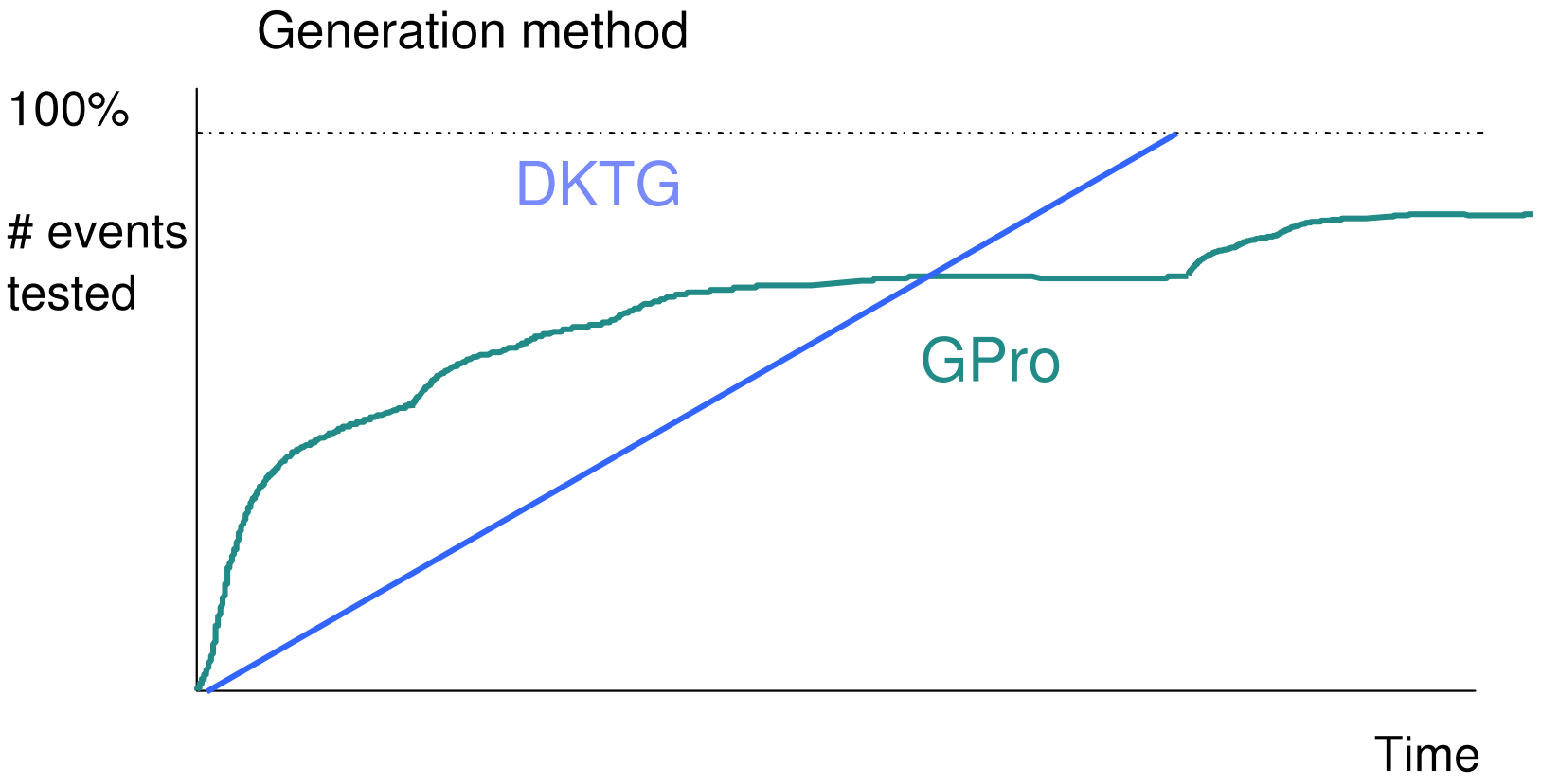
Coverage by Generation



DKTG Performance

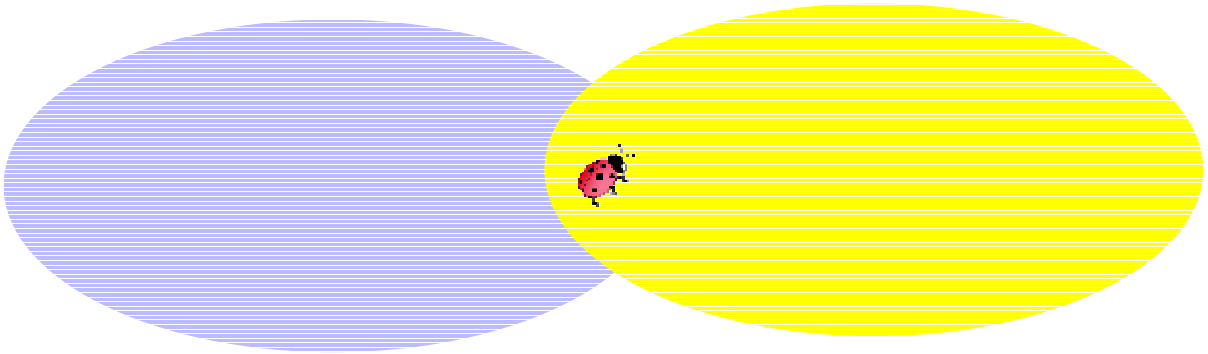


Coverage Graph



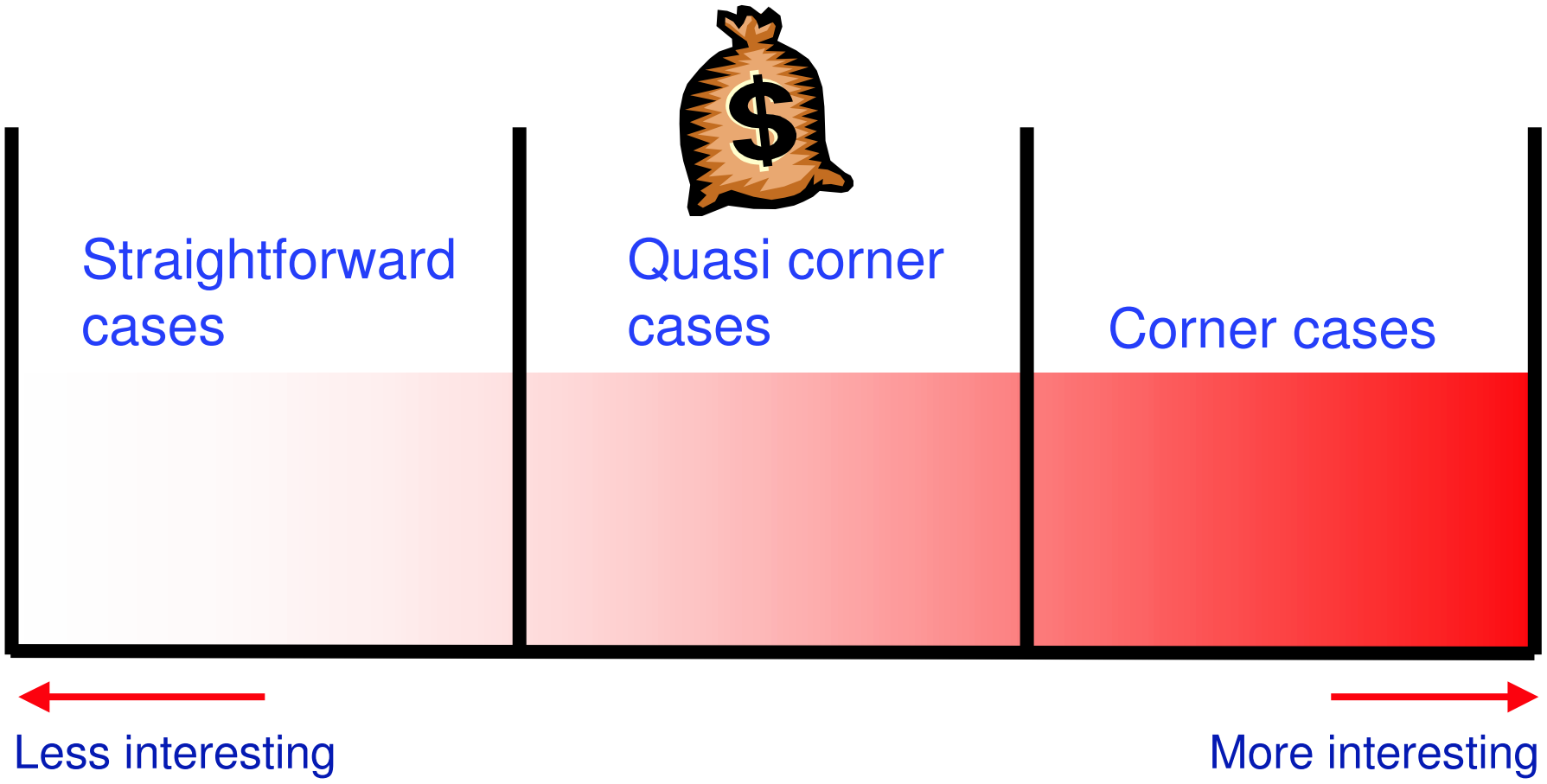
Cross-Product Approach

- Bugs often lie in the interaction of several factors.

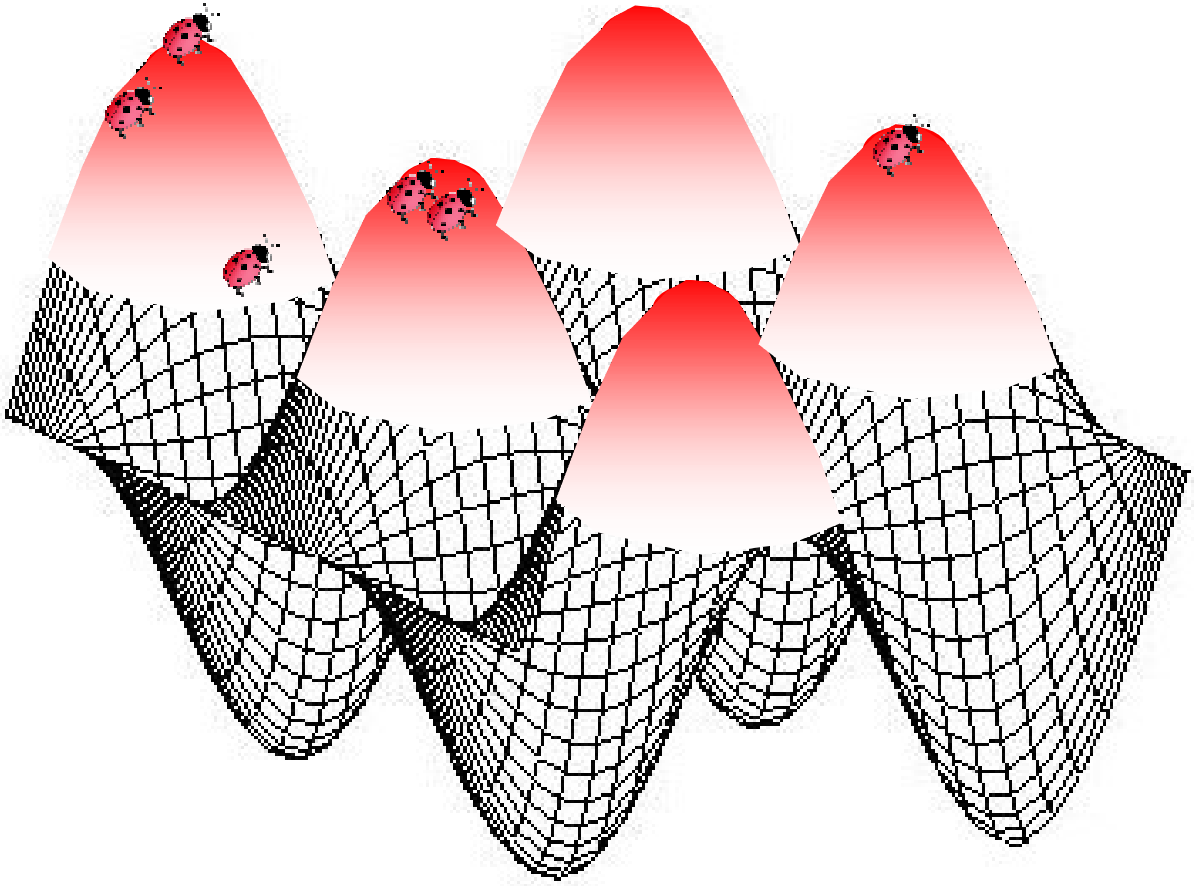


- This approach is more than a list of disparate tasks. It may include, often inadvertently, many “quasi corner cases”.
- All types model.
- Some cases are clearly corner cases, but others, while interesting, might have been overlooked.

Quasi Corner Cases



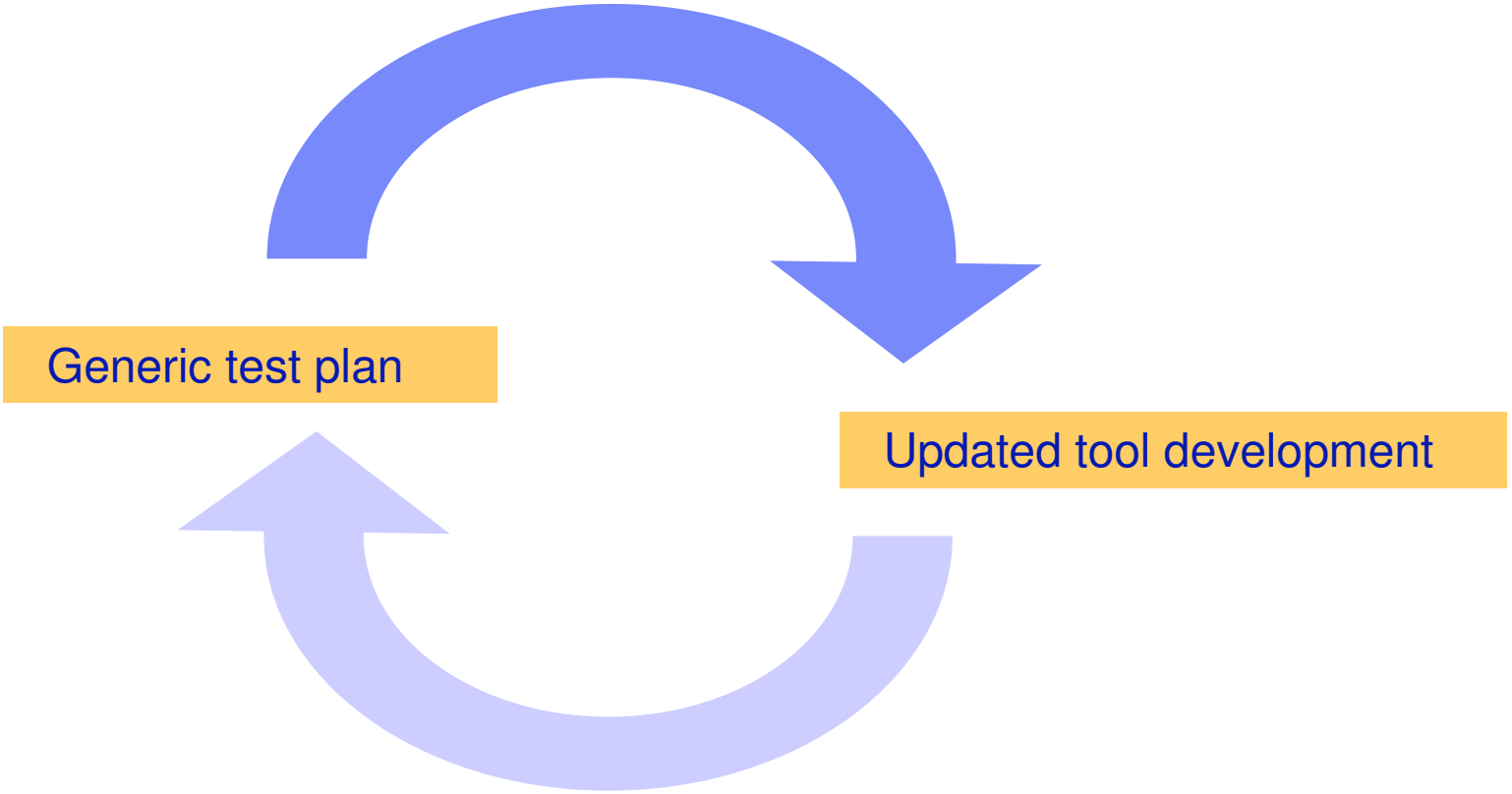
The Non-Uniform View



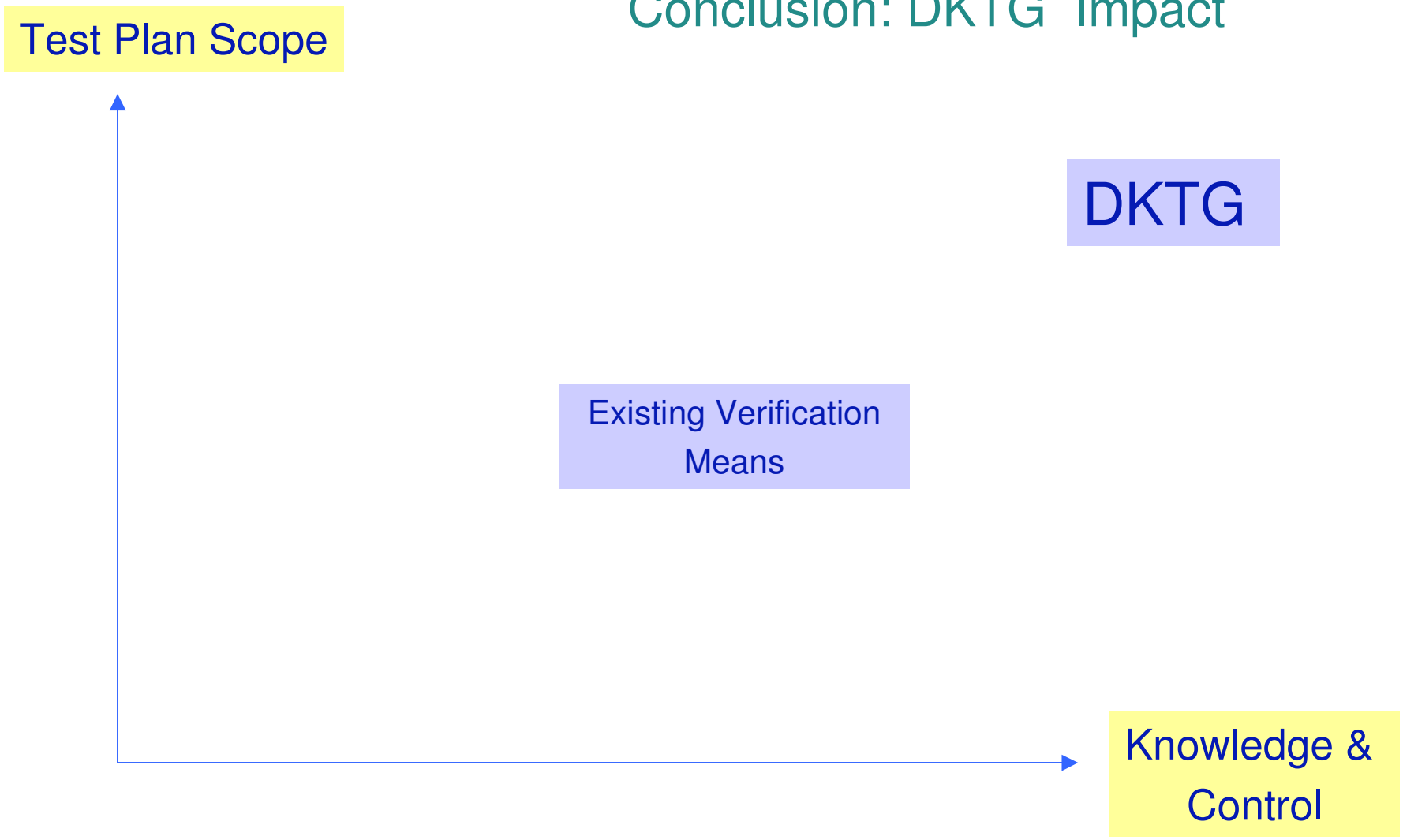
Pitfalls

- Quantity at the price of quality
  - Easy to create large, not meaningful event spaces
    - More events than can be covered (waste of verification bandwidth)
  - Leads people to be “thought-lazy” because easy to generate impressive test-plan (quantity-wise)
  - Blindly rely on “nice” coverage numbers
- Includes many events that require significant effort in knowing whether or not they are reachable (double-edged sword)

Conclusion: The Test-Plan Feedback Loop



Conclusion: DKTG Impact



Technology Perspective