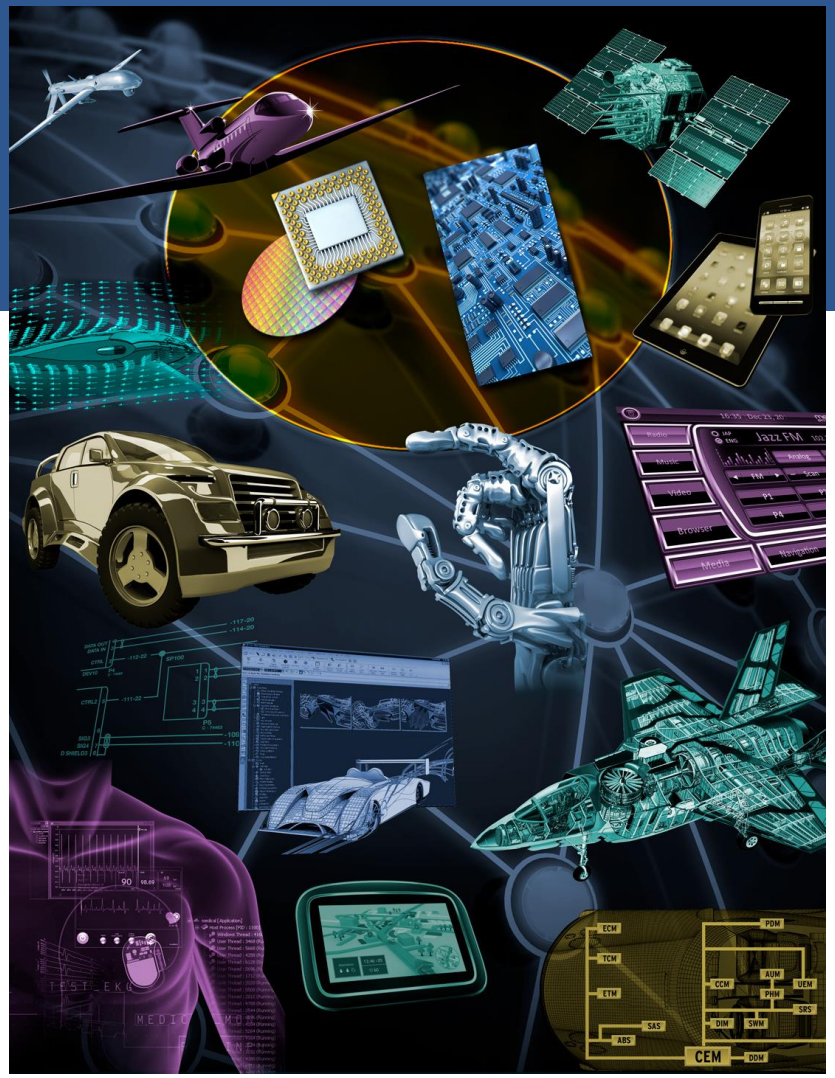


I'm Excited About Formal… My Journey From Skeptic To Believer

Neil Johnson

Product Engineering Manager

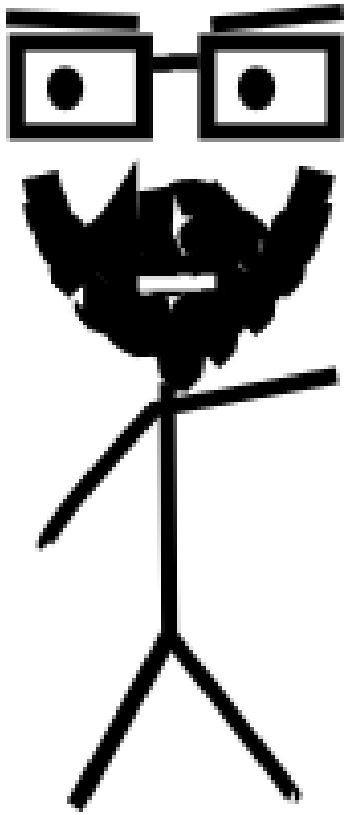
Questa DVT

Sept 2021

Mentor®
A Siemens Business

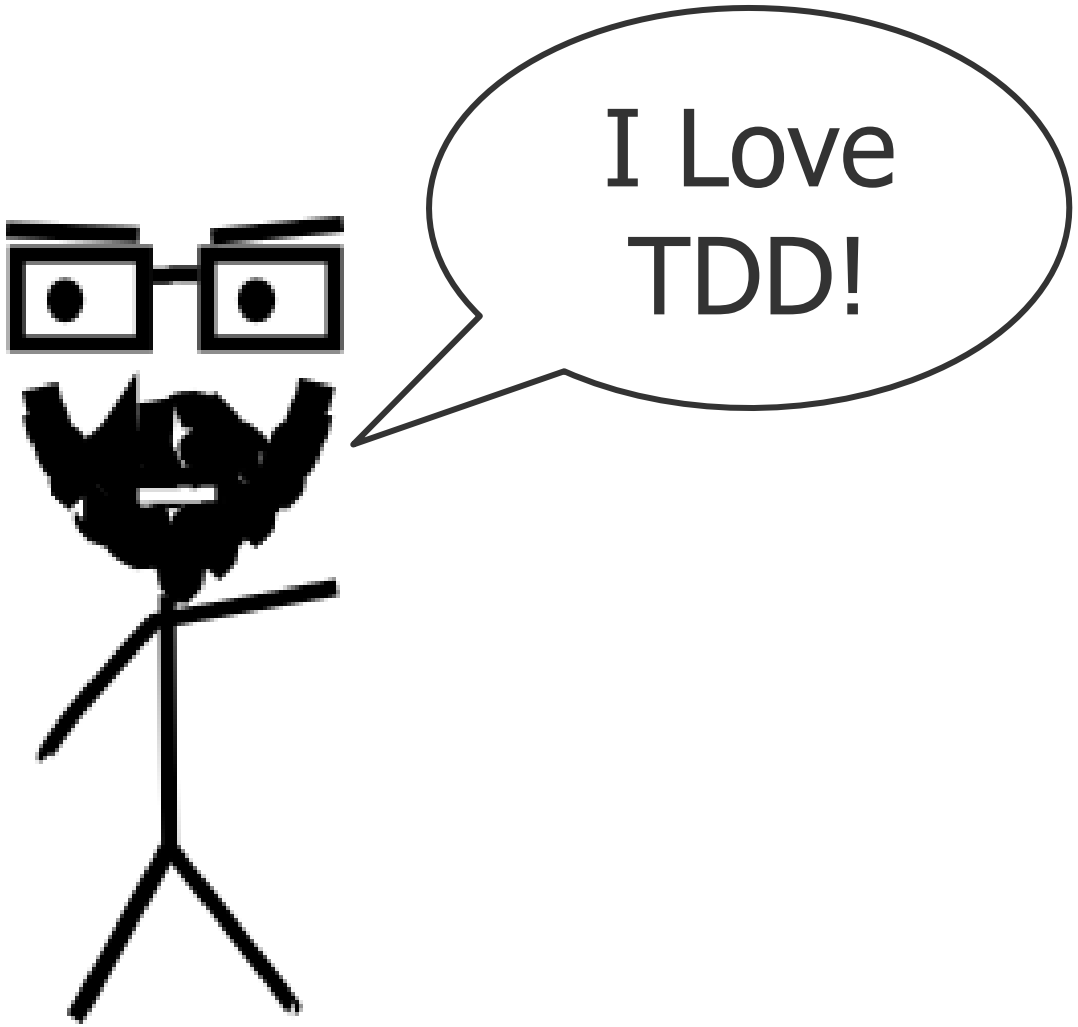
Meet Neil

- ~2 years RTL
- 16 years verification
- 2 years EDA
- 0 years formal

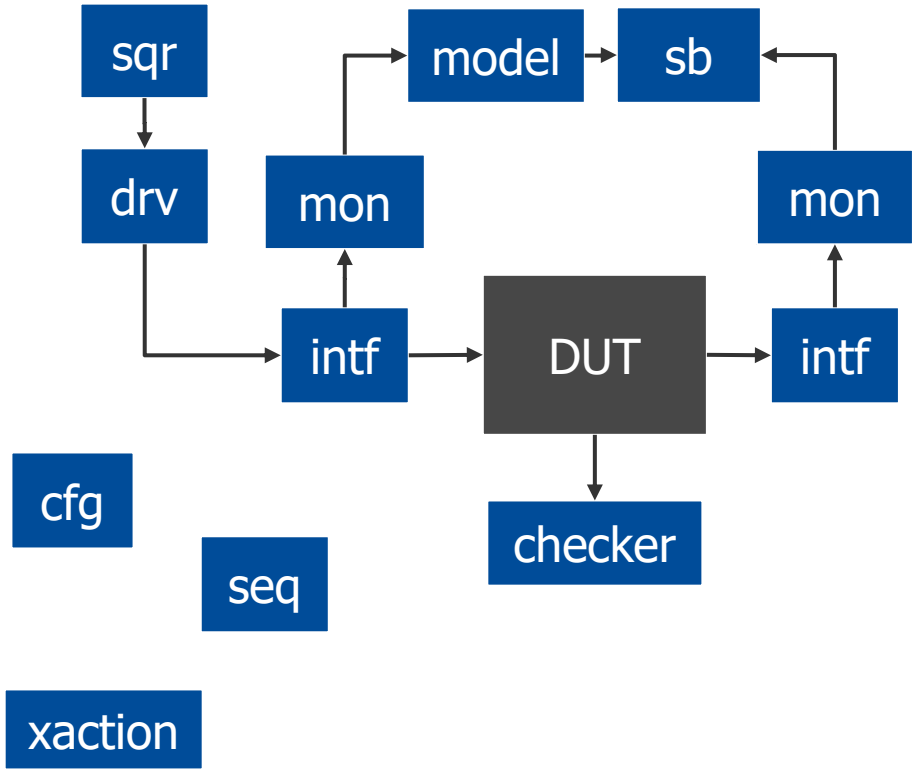


Meet Neil

- ~2 years RTL
- 18 years verification
- 2 years EDA
- 0 years formal

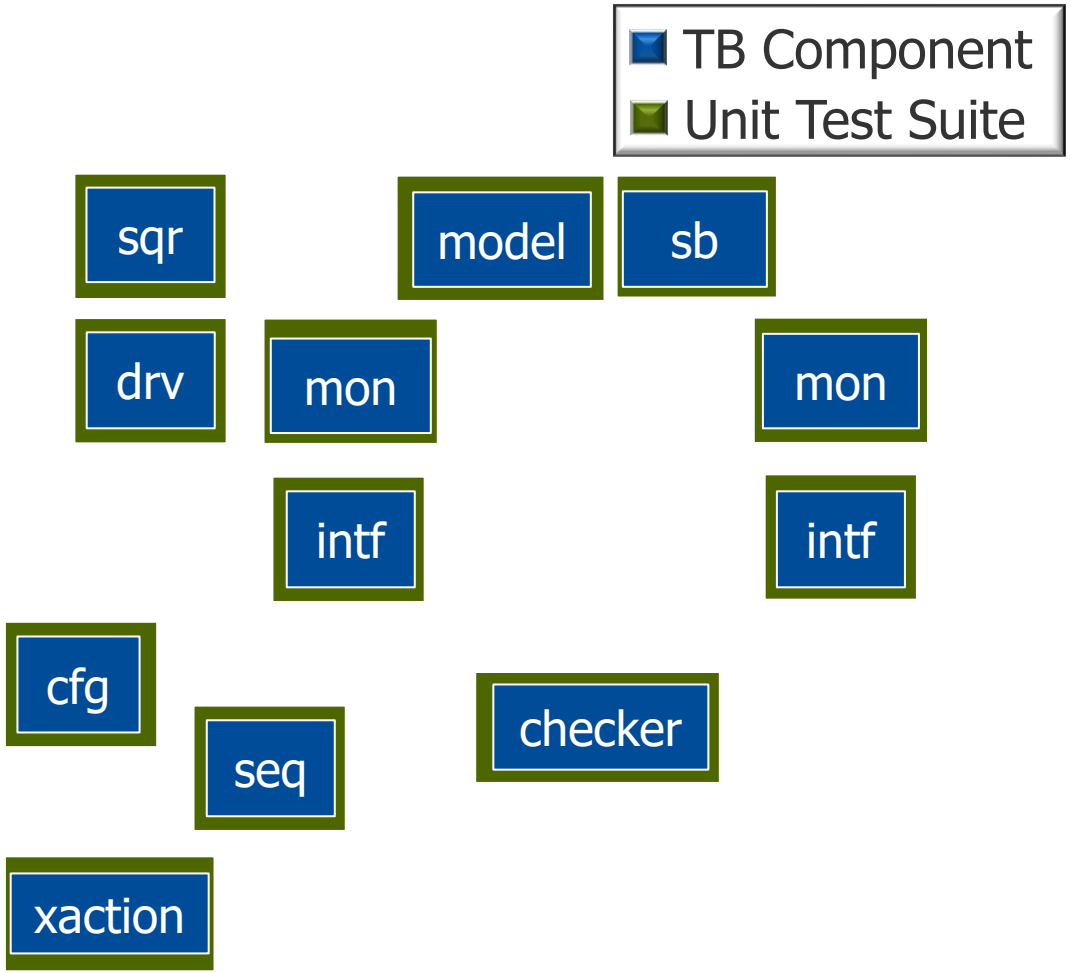


Introduction To Test-Driven Development



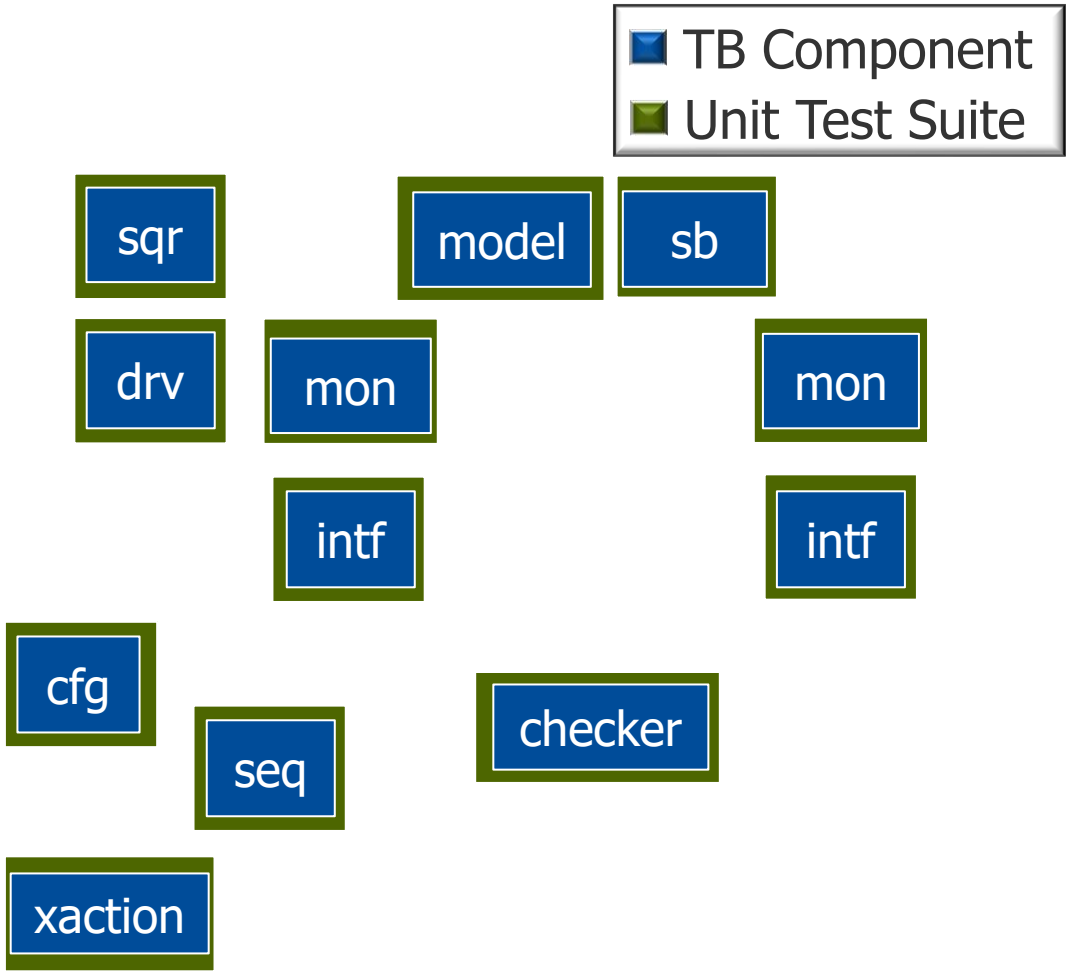
- Individual components are developed and tested in isolation
  - Unit test suites are exhaustive
  - Regressions are automated

Introduction To Test-Driven Development



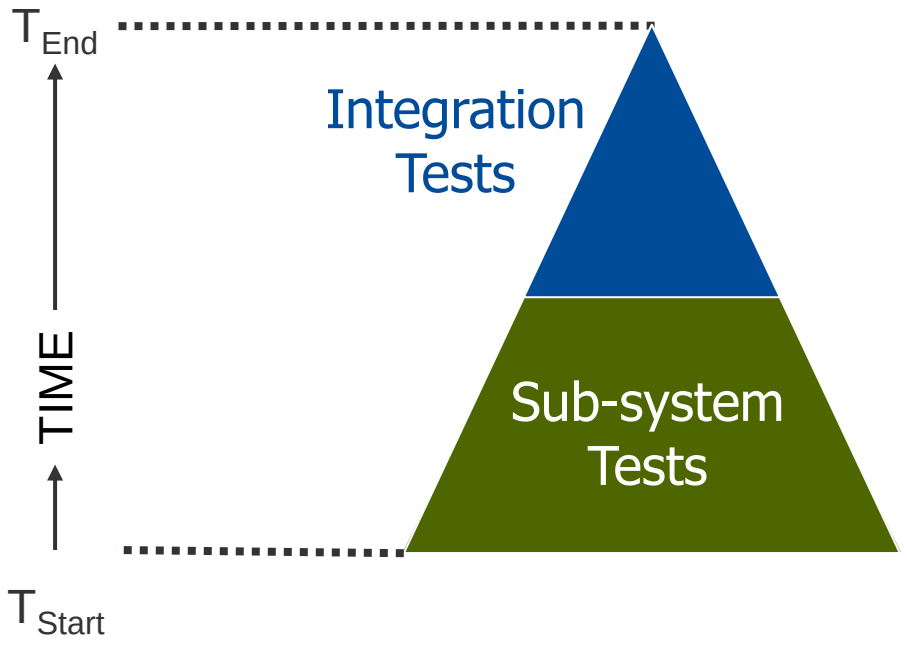
- Individual components are developed and tested in isolation
  - Unit test suites are exhaustive
  - Regressions are automated

Introduction To Test-Driven Development



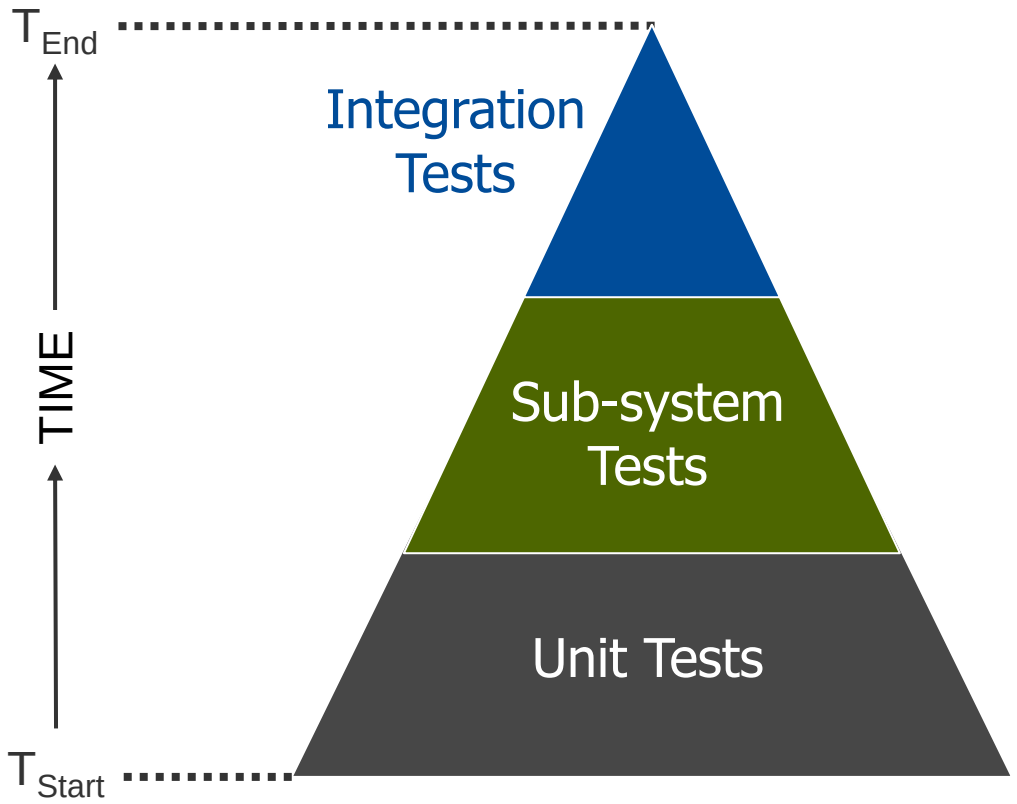
- Individual components are developed and tested in isolation
  - Unit test suites are exhaustive
  - Regressions are automated
- Unit tests are dependent on implementation
  - Planned and written during construction by the developer
  - Flexible planning and documentation requirements

Introduction To Test-Driven Development



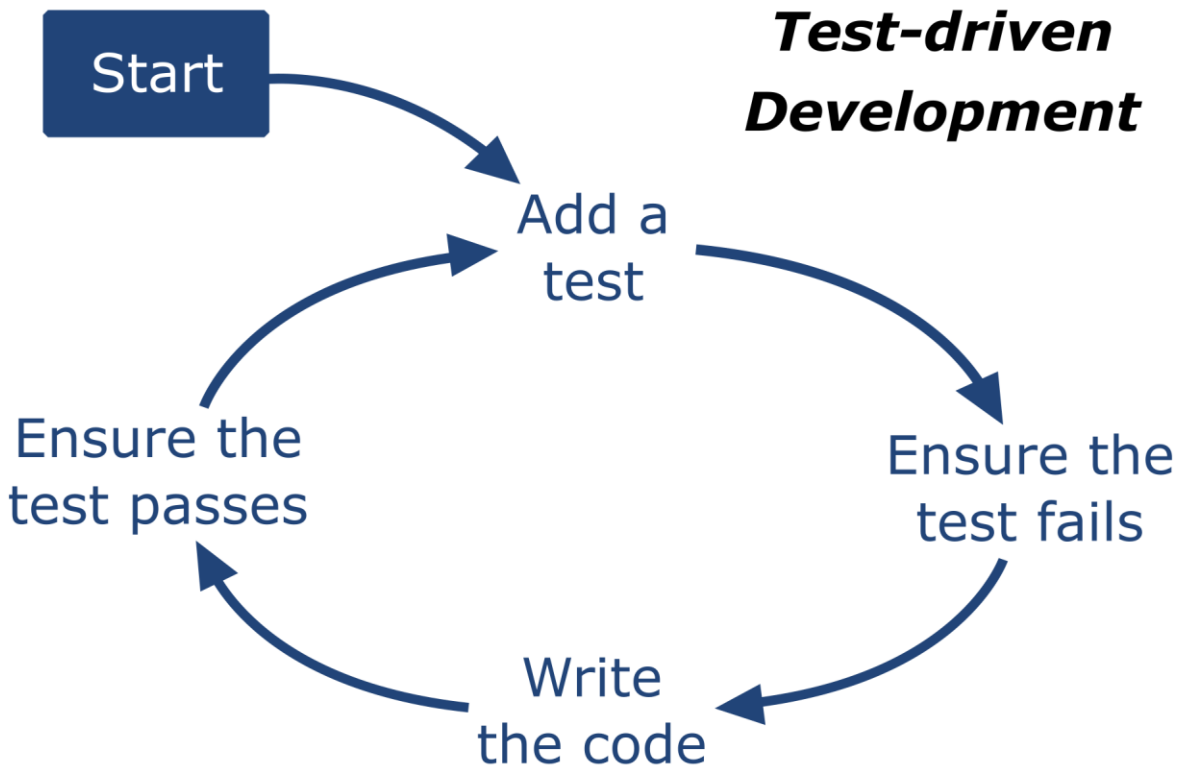
- Unit tests as a *quality foundation*
  - Unit tested features are then used in sub-systems or at top level
- Test-driven development is a complementary technique
  - Extra rigour means higher quality
  - Higher confidence means faster sub-system and integration testing

Introduction To Test-Driven Development



- Unit tests as a *quality foundation*
  - Unit tested features are then used in sub-systems or at top level
- Test-driven development is a complementary technique
  - Extra rigour means higher quality
  - Higher confidence means faster sub-system and integration testing

Introduction To Test-Driven Development

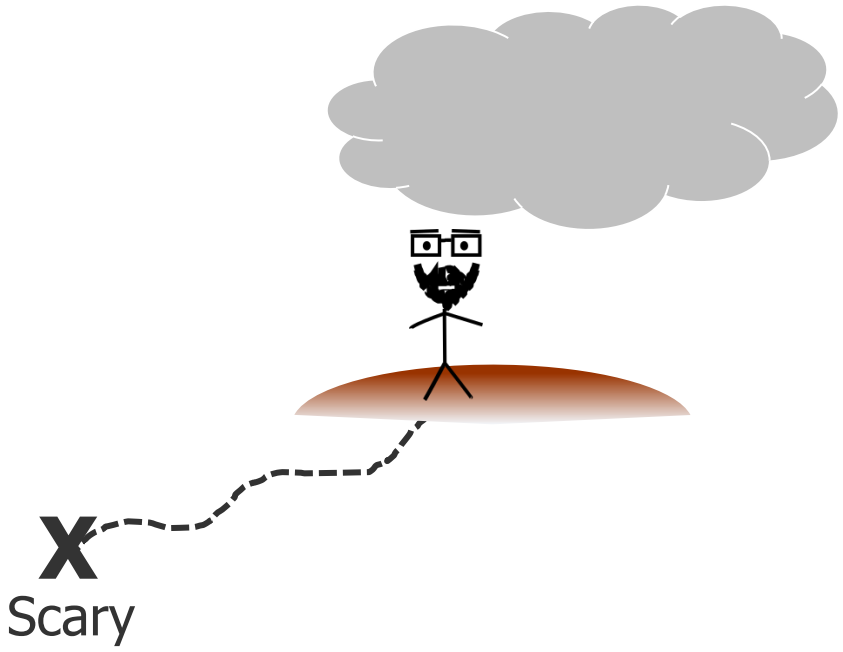


My Journey Into Formal…

Formal is for experts

Scary

My Journey Into Formal…



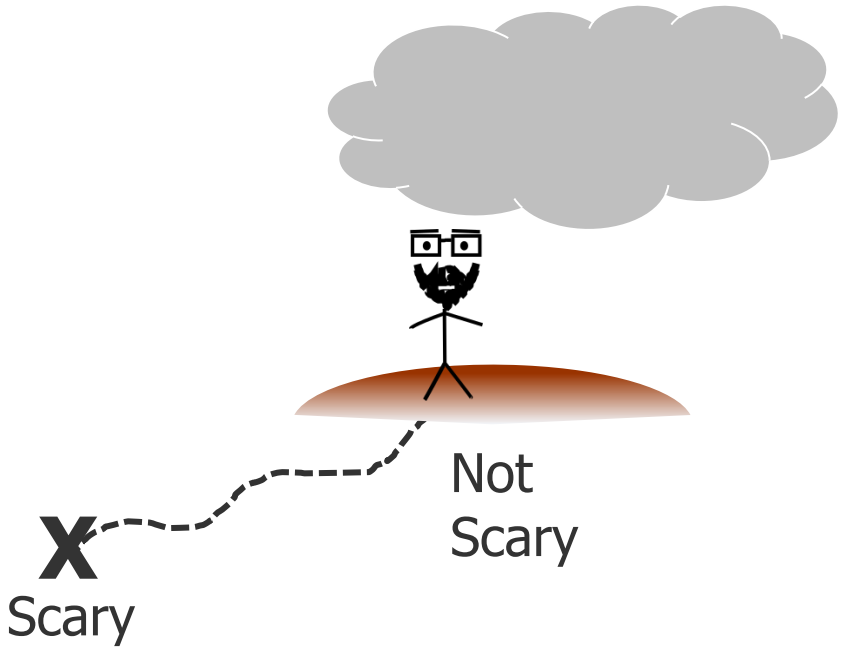
- Try CoverCheck…
  - CoverCheck to find dead logic in machine generated code
  - Results were definitive
  - Compile was familiar
  - Tool was push-button

```
# run
vlog ../L2CacheMetadataMemory.v -l vlog.log
qverify -c -do cc.do
```

```
# do cc.do
onerror {exit 1}
covercheck compile -d L2CacheMetadataMemory
covercheck verify -timeout 180s
covercheck generate ucdb cc_ex.ucdb -exclude
```

My Journey Into Formal…

Formal Apps are (actually) easy to use

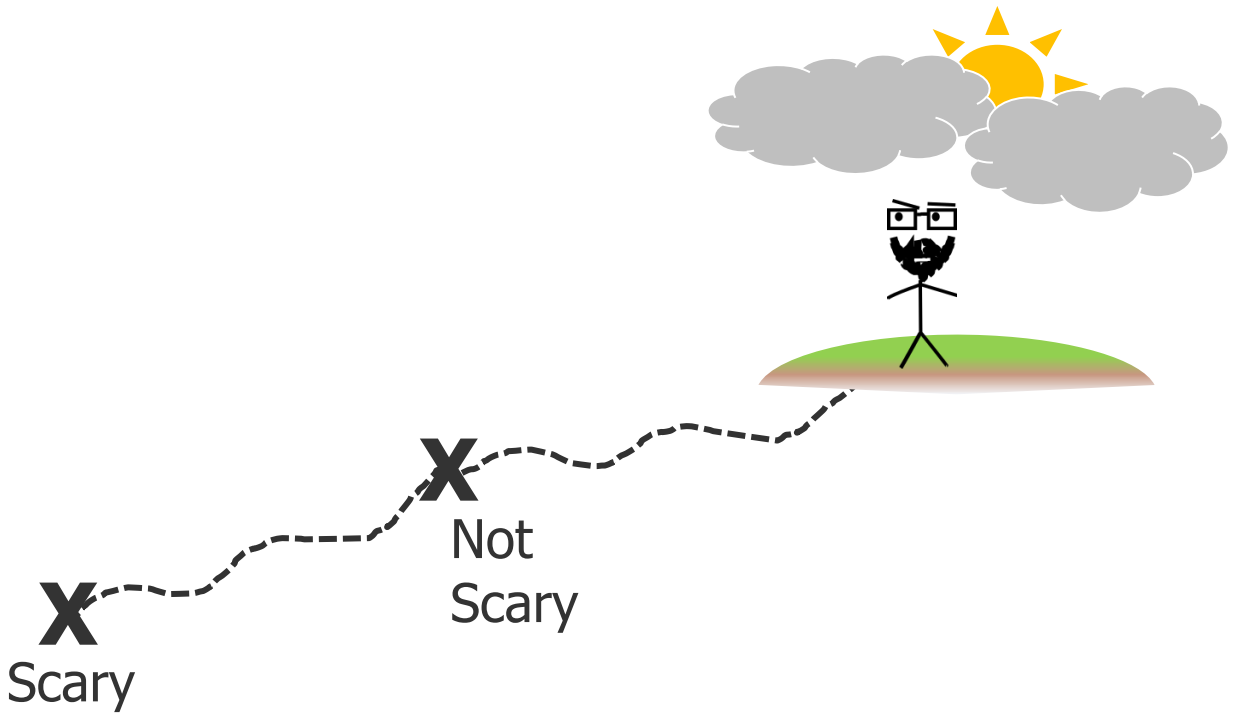


- Try CoverCheck…
  - CoverCheck to find dead logic in machine generated code
  - Results were definitive
  - Compile was familiar
  - Tool was push-button

```
# run
vlog ../L2CacheMetadataMemory.v -l vlog.log
qverify -c -do cc.do
```

```
# do cc.do
onerror {exit 1}
covercheck compile -d L2CacheMetadataMemory
covercheck verify -timeout 180s
covercheck generate ucdb cc_ex.ucdb -exclude
```

My Journey Into Formal…



- Try PropCheck…
  - Write a property

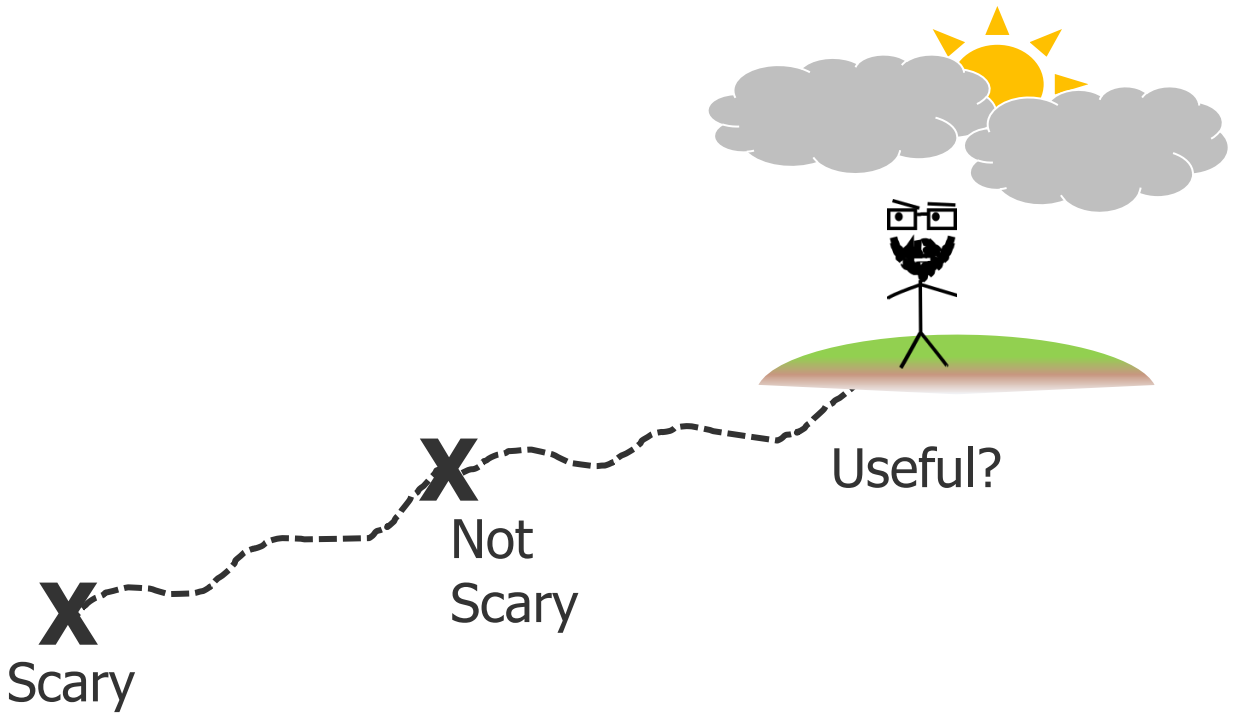
```
property writeActive;
  disable iff (!rst_n)
    @(posedge clk)
      iTVALID && oTREADY |-> (1) ##1 wr
endproperty
```

  - Prove it
    - Tool interface is the same as CoverCheck
  - Infrastructure was simple
    - Checker module w/bind

Mentor® A Siemens Business

My Journey Into Formal…

Property checking is easier than I expected



- Try PropCheck…
  - Write a property

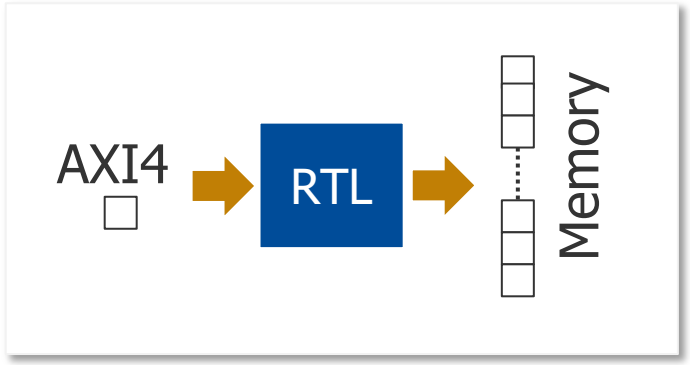
```
property writeActive;
  disable iff (!rst_n)
    @(posedge clk)
      iTVALID && oTREADY |-> (1) ##1 wr
endproperty
```

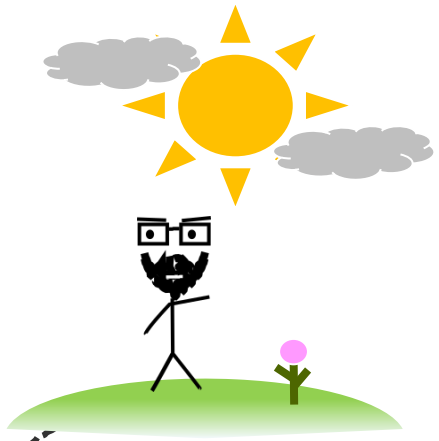
  - Prove it
    - Tool interface is the same as CoverCheck
  - Infrastructure was simple
    - Checker module w/bind

Mentor® A Siemens Business

My Journey Into Formal…

- Try PropCheck "unit tests"
  - Verify an RTL module
    - 22 properties
    - **1 day**





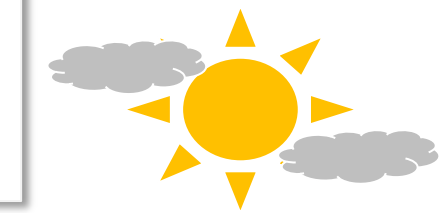
My Journey Into Formal…

- Try PropCheck "unit tests"
  - SVA features…

```
property writeAddr;
  logic [RAM_ADDR_WIDTH-1:0] expectedWaddr;

  disable iff (!rst_n)
    @(posedge clk)
      (ingress_pixel_seq and waddr < MEM_DEPTH-1, expectedWaddr = waddr + 1)
          |-> ##1 waddr == expectedWaddr
endproperty
```

Implication
whenever 'A' |-> prove 'B'



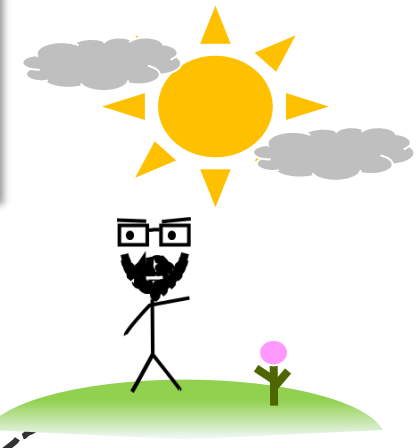
My Journey Into Formal…

- Try PropCheck “unit tests”
  - SVA features…

```
property writeAddr;
  logic [RAM_ADDR_WIDTH-1:0] expectedWaddr;

  disable iff (!rst_n)
    @(posedge clk)
      (ingress_pixel_sec and waddr < MEM_DEPTH-1, expectedWaddr = waddr + 1)
          |-> (1) ##1 waddr == expectedWaddr
endproperty
```

And
‘A’ and ‘B’ are both true



X Scary

X Not Scary

X Useful?

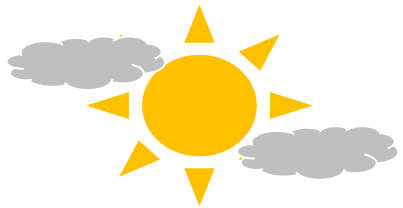
My Journey Into Formal…

- Try PropCheck “unit tests”
  - SVA features…

```
property writeAddr;
  logic [RAM_ADDR_WIDTH-1:0] expectedWaddr;

  disable iff (!rst_n)
    @(posedge clk)
      (ingress_pixel_seq_and_waddr < MEM_DEPTH-1, expectedWaddr = waddr + 1)
          |-> (1) ##1 waddr == expectedWaddr
endproperty
```

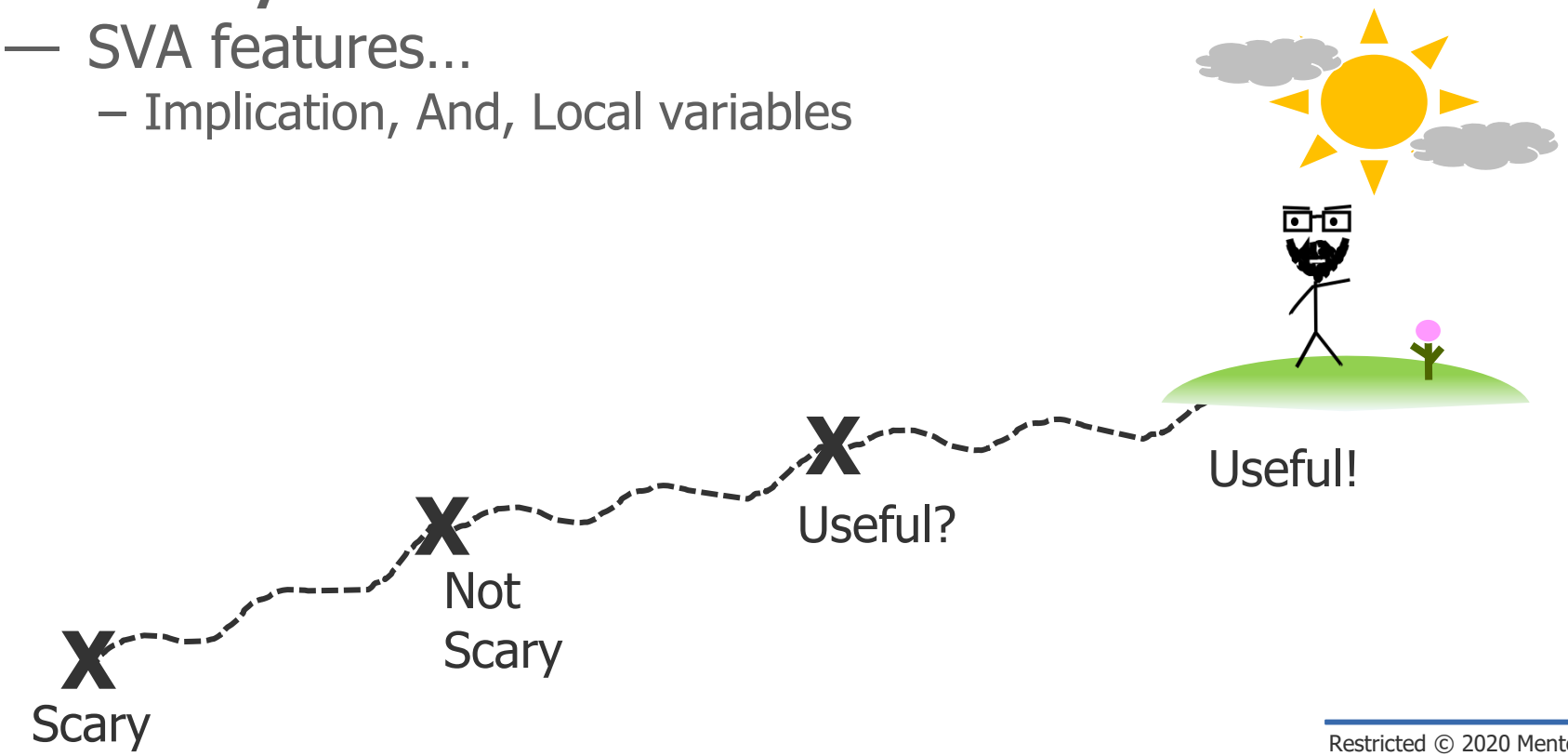
Local Variables
Save ‘A’ now for checking later



My Journey Into Formal…

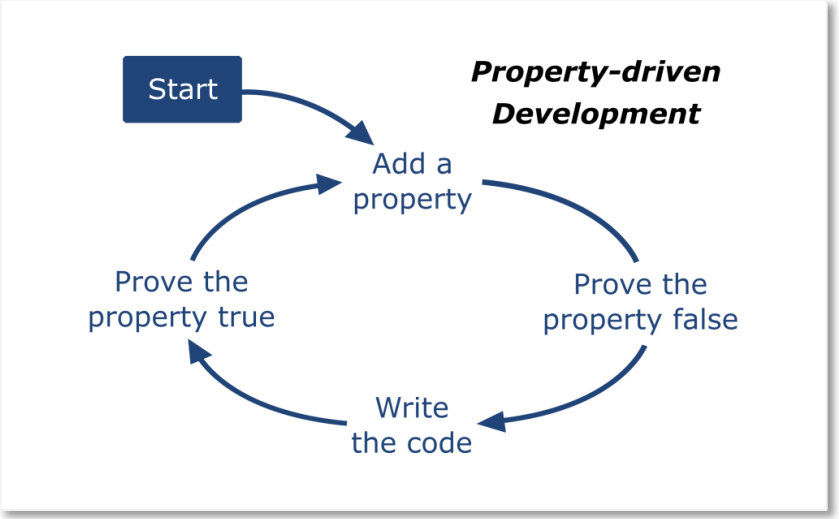
- Try PropCheck "unit tests"
  - Re-verify an RTL module
    - 22 properties
    - **1 day**
  - SVA features…
    - Implication, And, Local variables

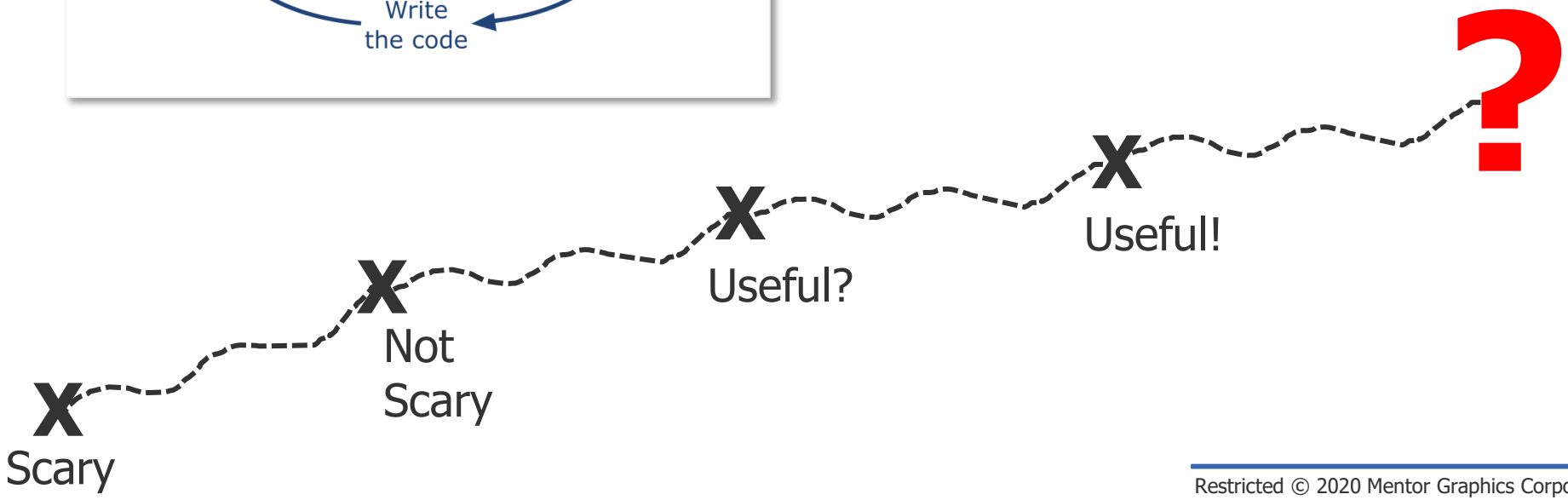
Property checking is a legit complement to simulation



My Journey Into Formal…

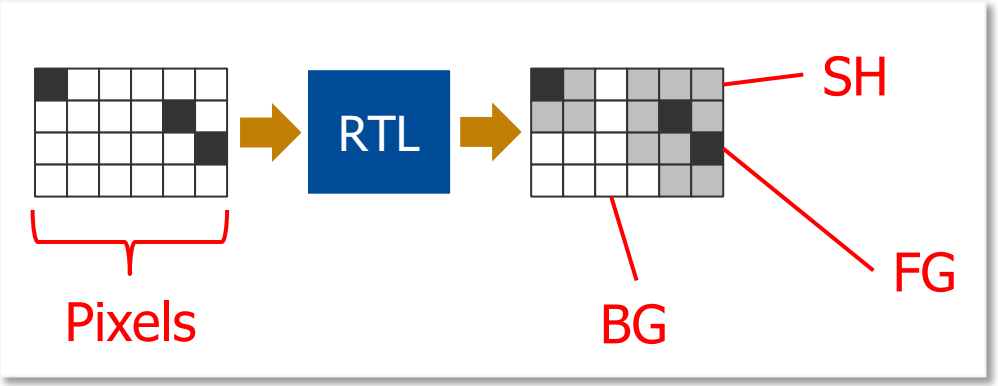
- Try Property-driven development

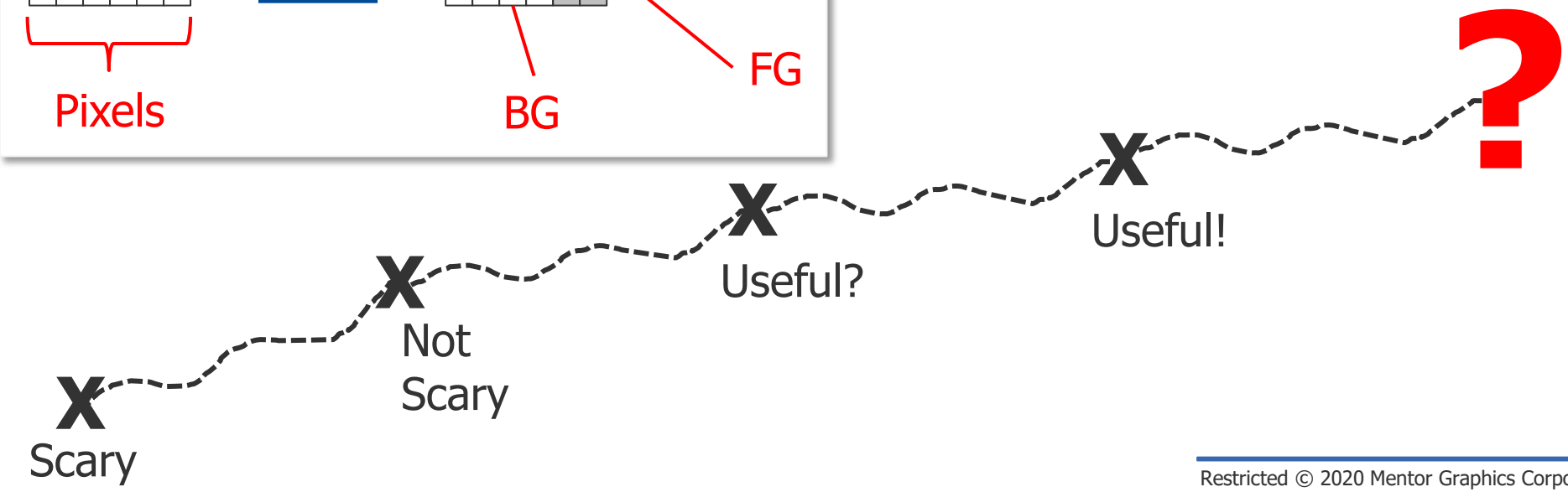




My Journey Into Formal…

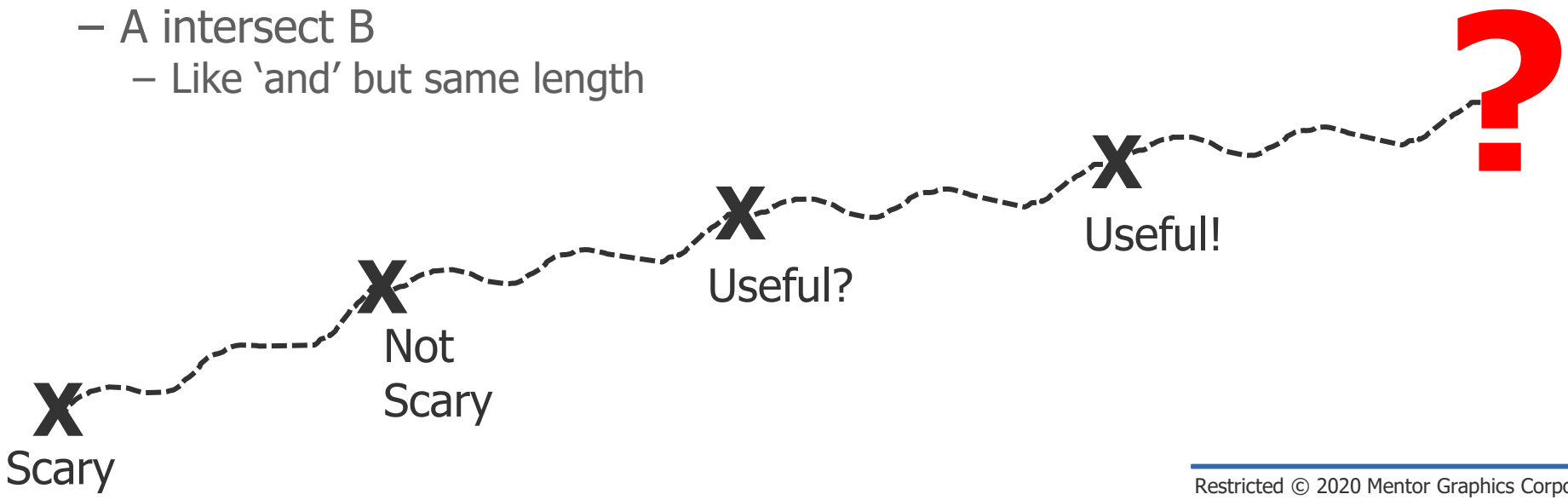
- Try Property-driven development
  - Re-develop an RTL module
    - 31 properties
    - 1 week





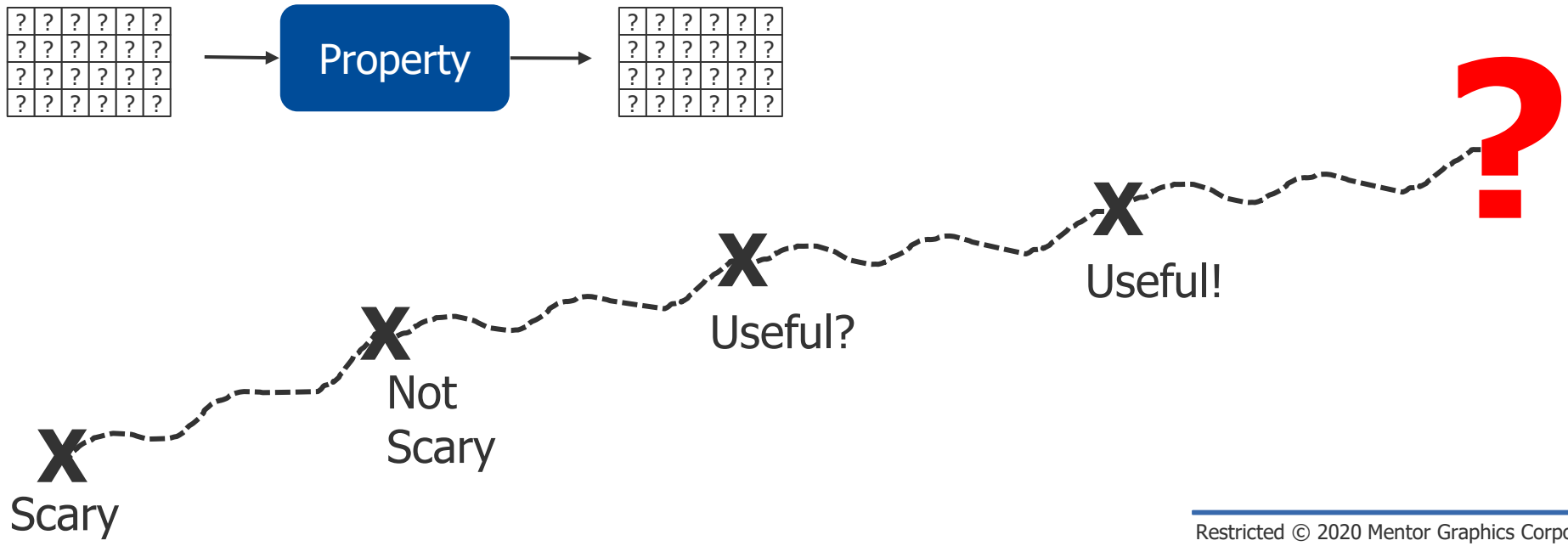
My Journey Into Formal…

- Try Property-driven development
  - Re-develop an RTL module
    - 31 properties
    - 1 week
  - SVA features…
    - A [*3] // repetition
      - ‘A’ 3 clocks in a row
    - A intersect B
      - Like ‘and’ but same length



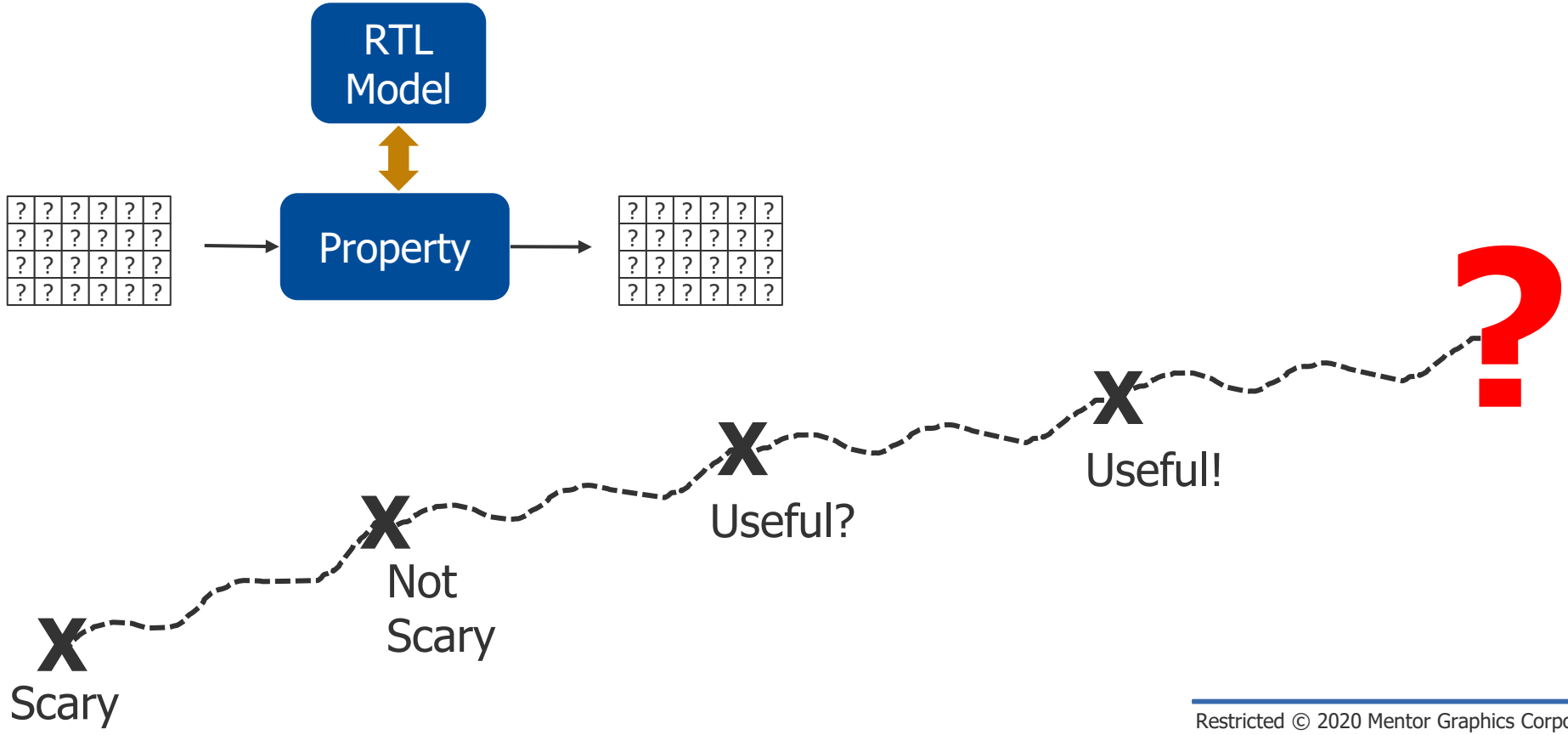
My Journey Into Formal…

- Try Property-driven development
  - Modeling?



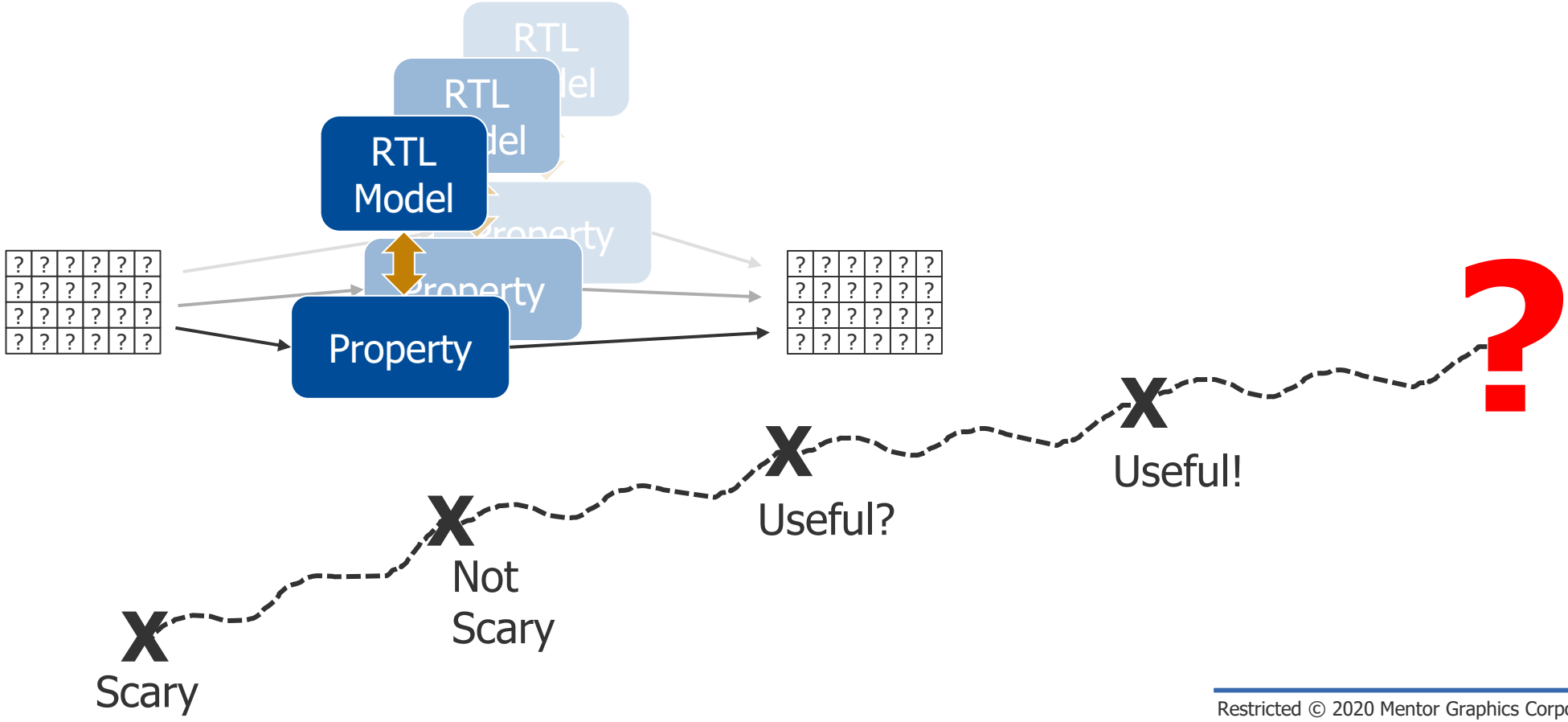
My Journey Into Formal…

- Try Property-driven development
  - Modeling?



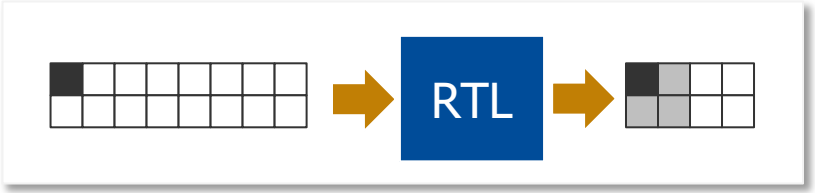
My Journey Into Formal…

- Try Property-driven development
  - Modeling?



My Journey Into Formal…

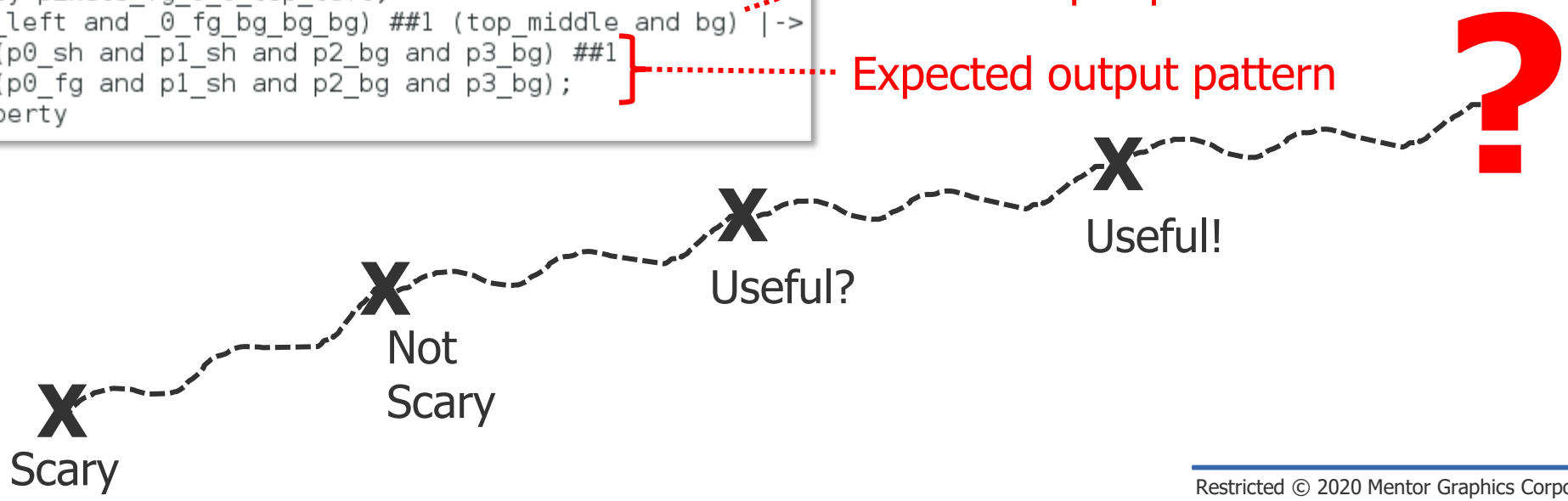
- Try Property-driven development
  - **Directed Properties > Modeling**




```
// ROW 0 FG
property pixels_fg_0_0_top_left;
  (top_left and _0_fg_bg_bg_bg) ##1 (top_middle and bg) |->
      (p0_sh and p1_sh and p2_bg and p3_bg) ##1
      (p0_fg and p1_sh and p2_bg and p3_bg);
endproperty
```

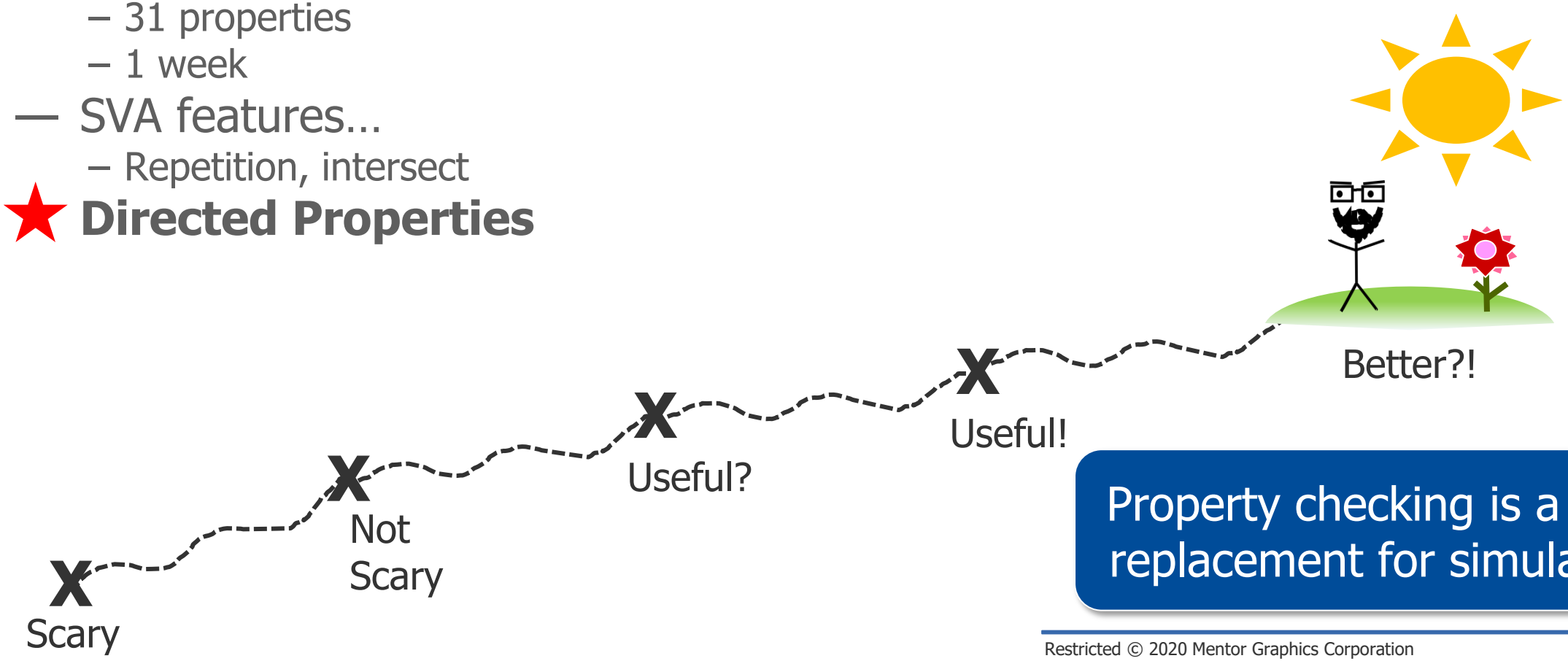
Directed input pattern

Expected output pattern



My Journey Into Formal…

- Try Property-driven development
  - Re-develop an RTL module
    - 31 properties
    - 1 week
  - SVA features…
    - Repetition, intersect
  - ★ **Directed Properties**



Property checking is a legit replacement for simulation

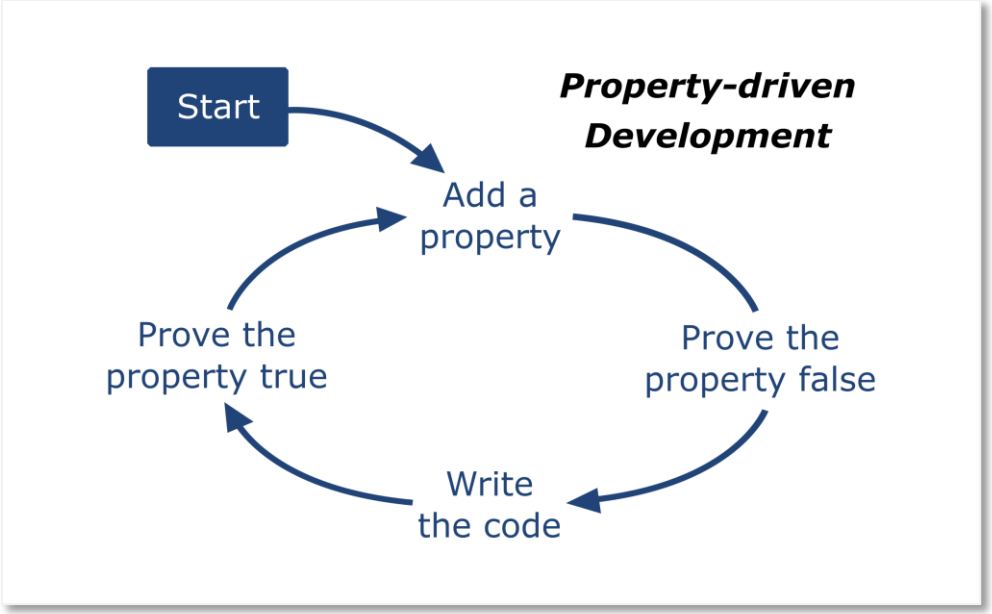
Mentor® A Siemens Business

Your Journey Begins…

- ~~Formal expert~~
- Start with formal apps
  - CoverCheck
  - AutoCheck
  - X checking
  - Connectivity checking
- Try property checking
  - Dedicate 1 day
  - Keep it small – module level
- Integrate property checking
  - Complementary to simulation

Your Journey Begins…

- ~~Formal expert~~
- Start with formal apps
  - CoverCheck
  - AutoCheck
  - X checking
  - Connectivity checking
- Try property checking
  - Dedicate 1 day
  - Keep it small – module level
- Integrate property checking
  - Complementary to simulation
- For design engineers



Questions?

I’m Excited About Formal… My Journey From Skeptic To Believer

neil_johnson@mentor.com