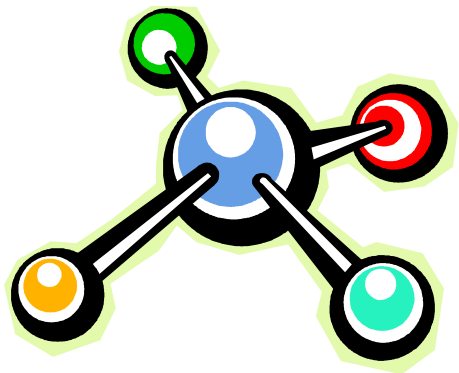
Model-Based Testing



Harry Robinson
Google
harryr@google.com

Goals for Today

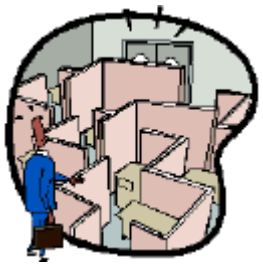
- Convey techniques for
  - How to model systems
  - How to generate tests
  - How to verify results
- Communicate a mindset
- Provide inspiration
- Foster discontent

Non-Goals for Today

- Specific tools
- Interfacing with an application

What are the Problems of Software Testing?

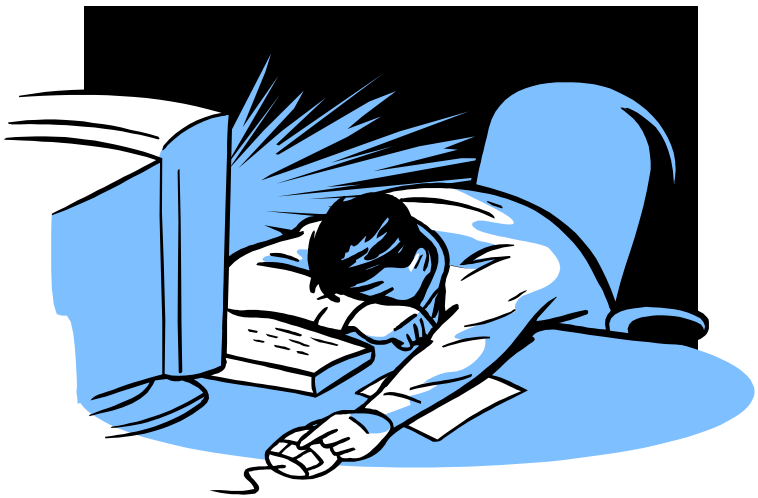
- Time is limited



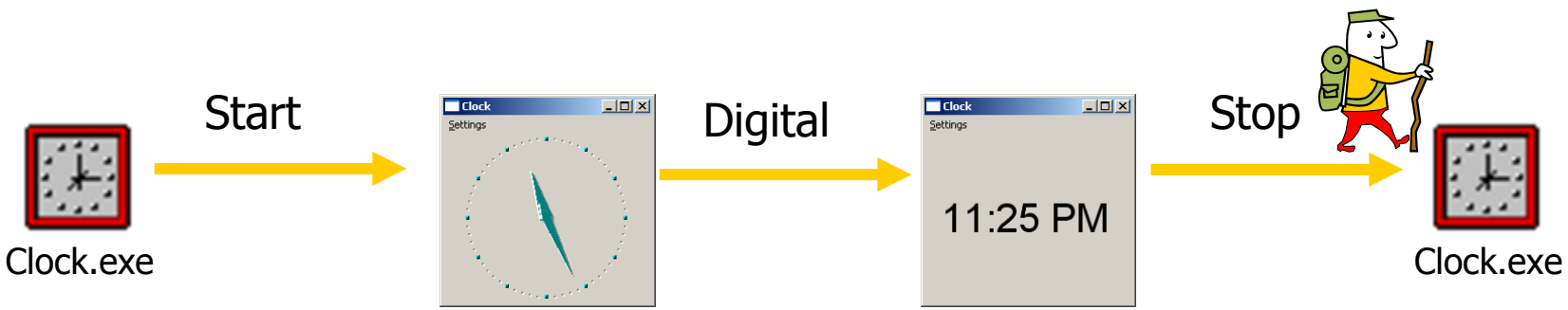
- Applications are complex

- Requirements are fluid

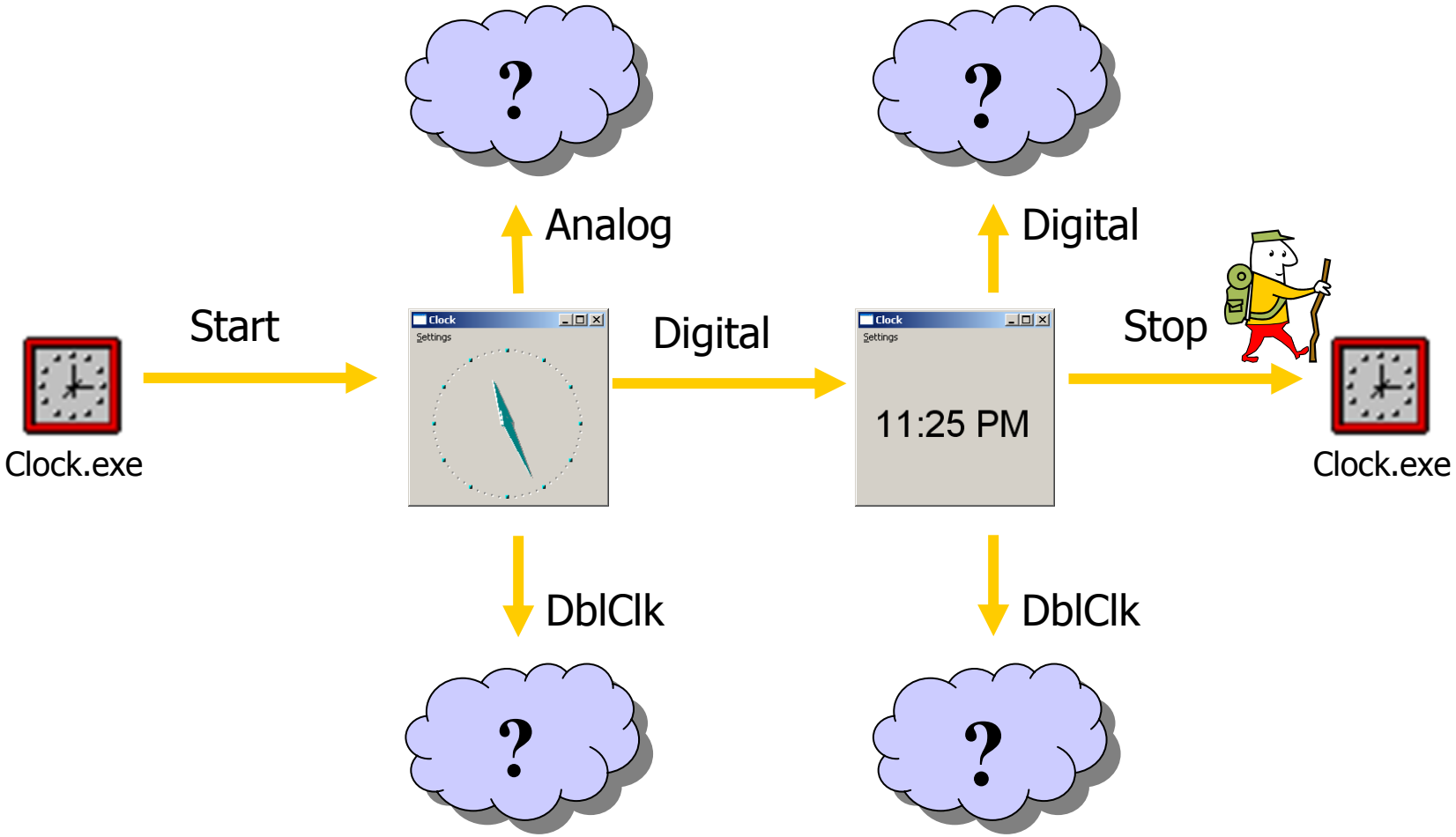
What's wrong with manual testing?



Manual testing is ok sometimes …



… but it can rarely go deep enough



What's wrong with scripting?

The Villain of this Piece

```
CurrentWindow = WFndWndC("Calculator", "SciCalc")
WSetWndPosSiz(CurrentWindow, 88, 116, 260, 260)
WMenuSelect("@2\@2")
CurrentWindow = WFndWndC("Calculator", "SciCalc")
WSetWndPosSiz(CurrentWindow, 88, 116, 480, 317)
WButtonClick("@32")
WButtonClick("@27")
WButtonClick("@38")
WButtonClick("@58")
WMenuSelect("@2\@1")
CurrentWindow = WFndWndC("Calculator", "SciCalc")
WSetWndPosSiz(CurrentWindow, 88, 116, 260, 260)
WMenuSelect("@2\@1")
Play "{Click 330, 131, Left}"
WToolbarButtonClk("@2", "@6")
WToolbarButtonClk("@2", "@7")
WToolbarButtonClk("@1", "@1")
```

- Awe-inspiring
- Unchanging
- Indecipherable

Scripts are ok for some uses …

Test case 1: Start Digital Stop Start Analog Stop
Test case 2: Start DblClk Stop Start DblClk Stop

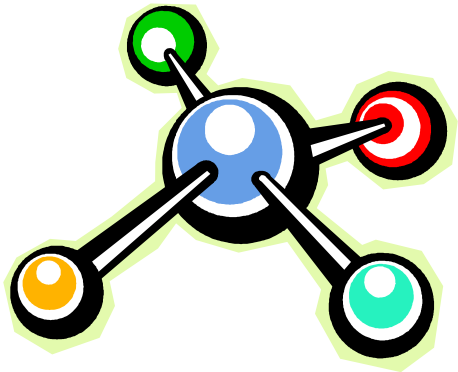
… but they pile up quickly …

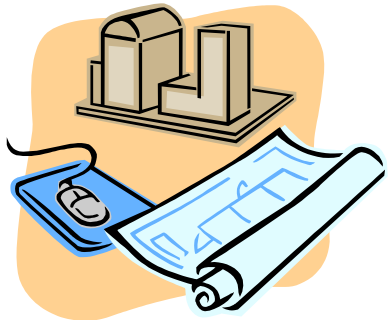
Test case 1: Start Digital Stop Start Analog Stop
Test case 2: Start DblClk Stop Start DblClk Stop
Test case 3: Start Digital DblClk Stop Start DblClk Analog Stop
Test case 4: Start DblClk DblClk Digital DblClk DblClk Stop Start Analog Stop
Test case 5: …

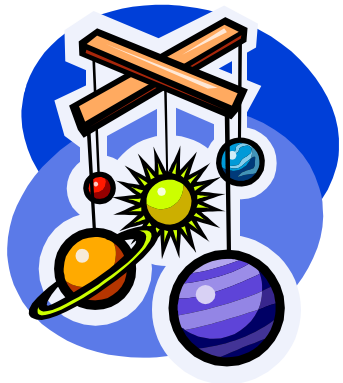
… and what are you left with?

Test case 1: Start Digital Stop Start Analog Stop
Test case 2: Start DblClk Stop Start DblClk Stop
Test case 3: Start Digital DblClk Stop Start DblClk Analog Stop
Test case 4: Start DblClk DblClk Digital DblClk DblClk Stop Start Analog Stop
Test case 5: Start Digital Digital Stop Start Analog Stop
Test case 6: Start DblClk Stop Start Analog DblClk Stop
Test case 7: Start Digital DblClk Stop Start Digital DblClk Analog Stop
Test case 8: Start DblClk DblClk Digital Analog DblClk DblClk Stop
Test case 9: …

What is a model?







- A model is a description of a system.
- Models are simpler than the systems they describe.
- Models help us understand and predict the system's behavior.

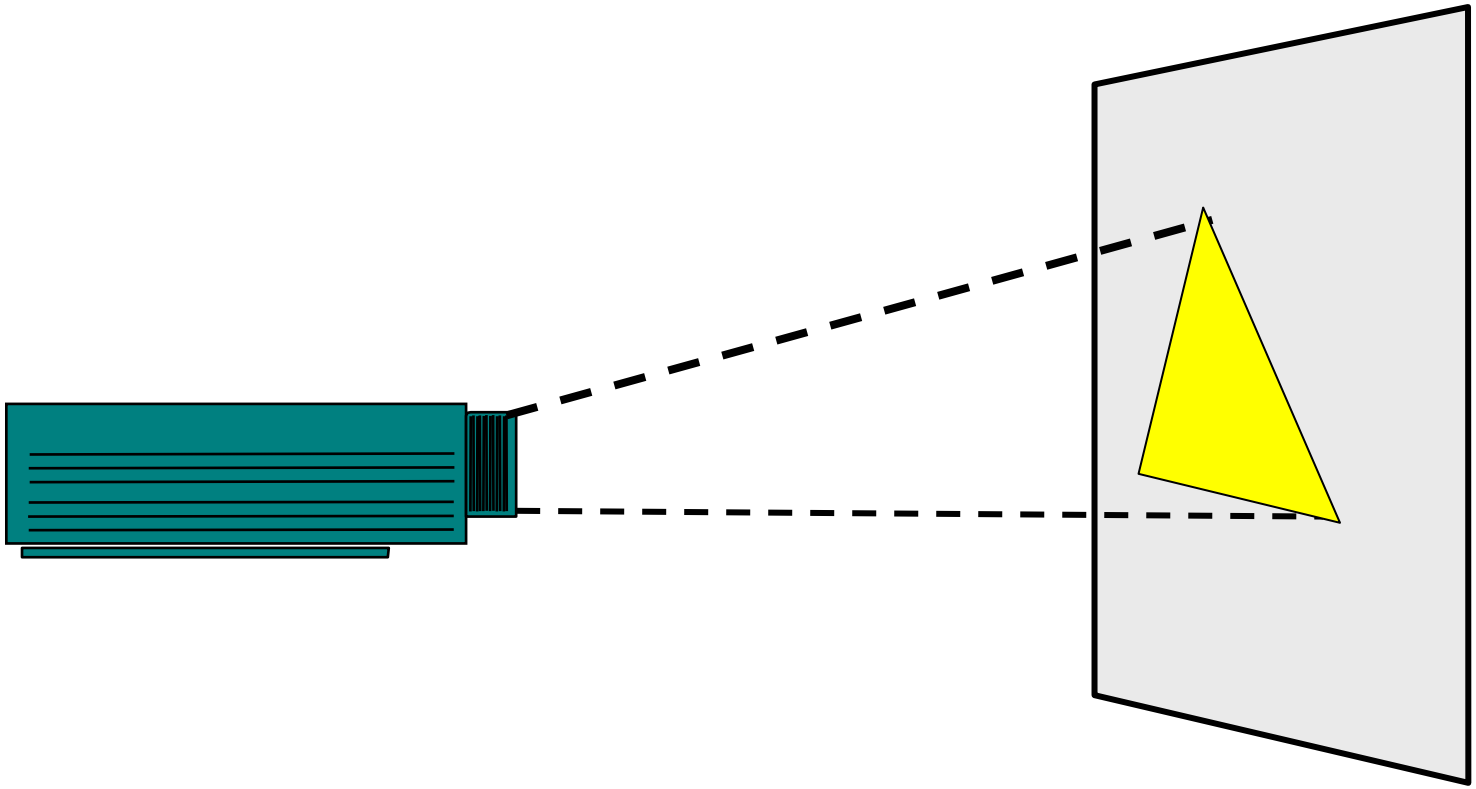
What is model-based testing?

"Model-based testing is a testing technique where the runtime behavior of an implementation under test is checked against predictions made by a formal specification, or *model*." - Colin Campbell, MSR

In other words

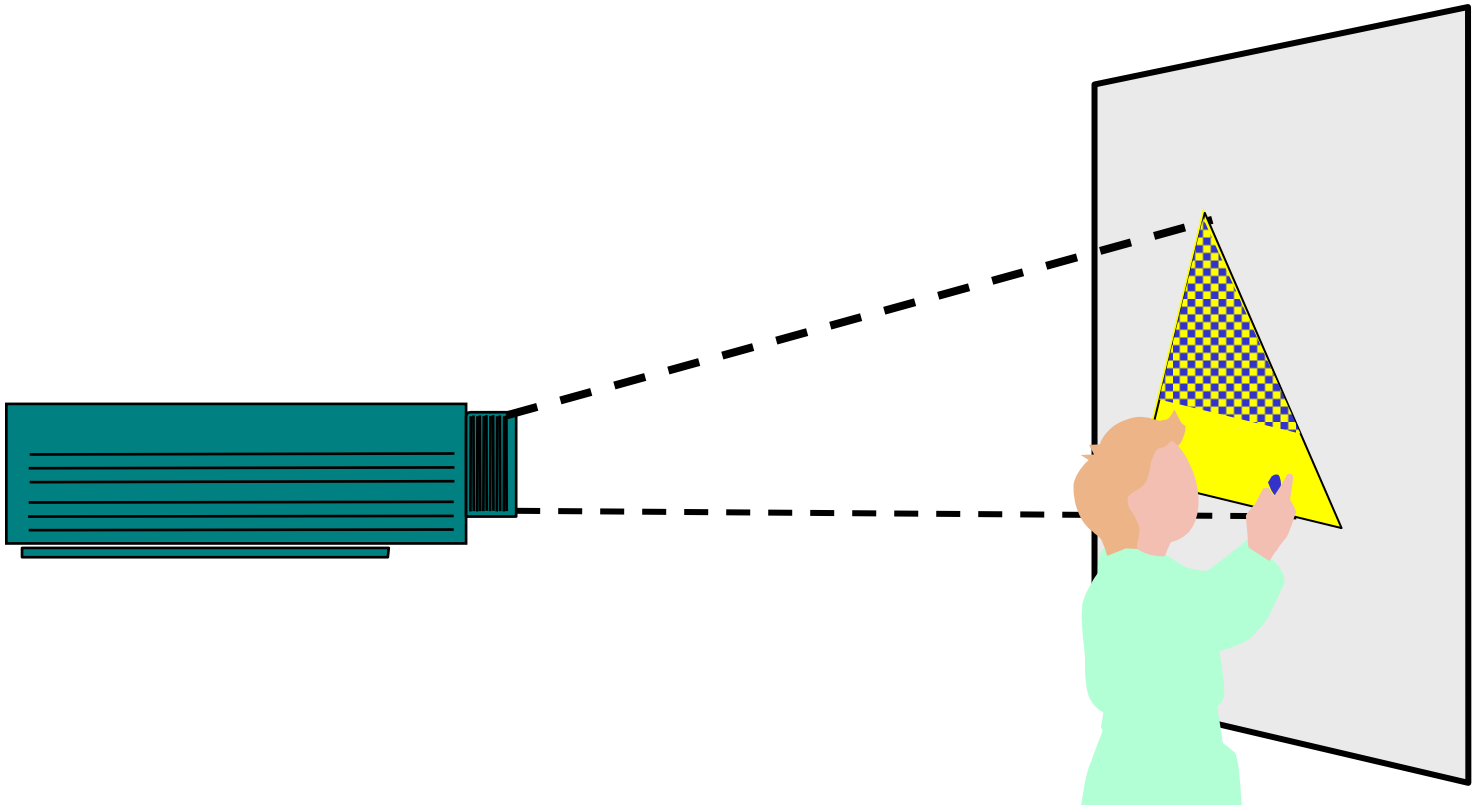
- A model describes how a system should behave in response to an action.
- Supply the action and see if the system responds as you expect.

Traditional Automated Testing



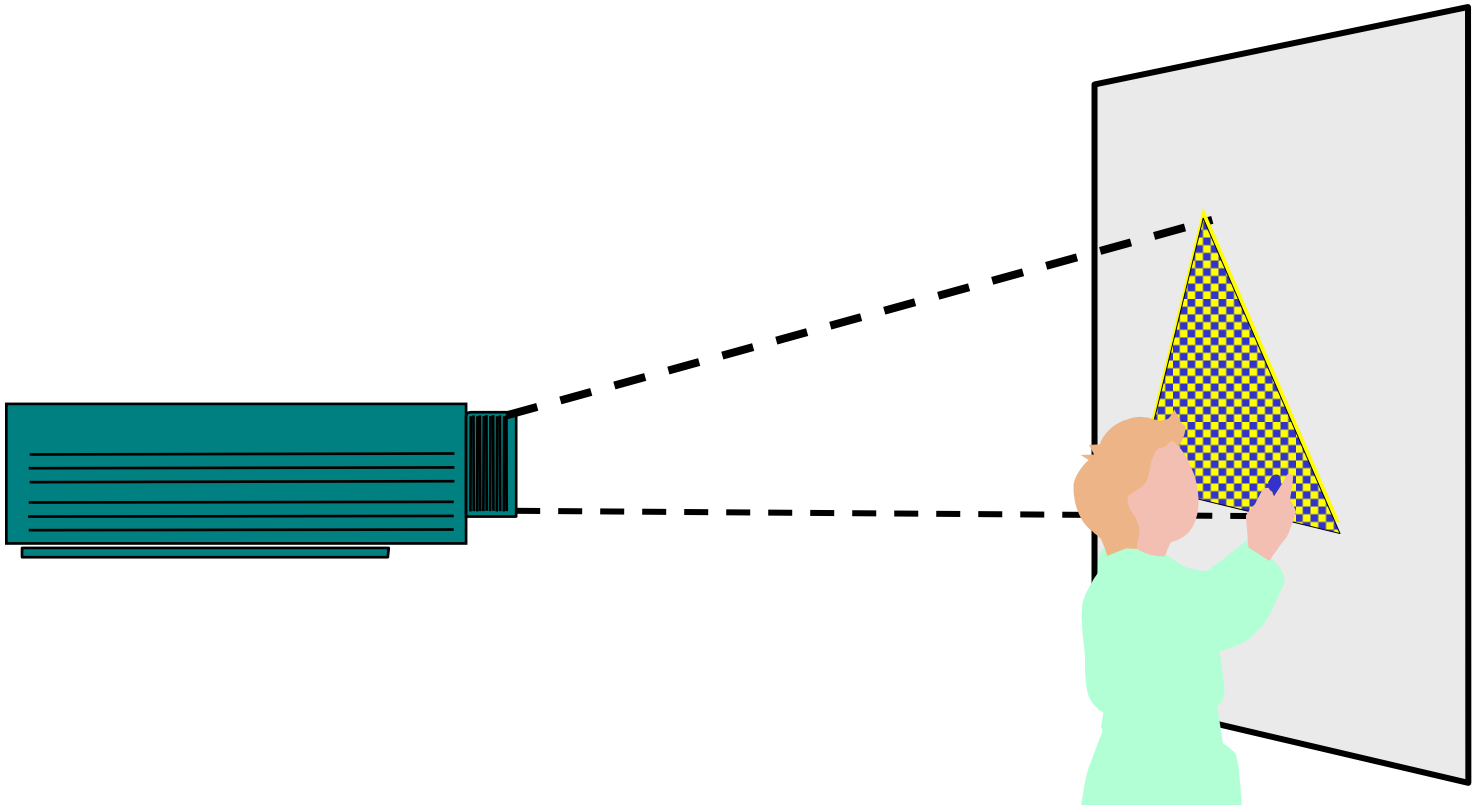
Imagine that this projector is the software you are testing.

Traditional Automated Testing



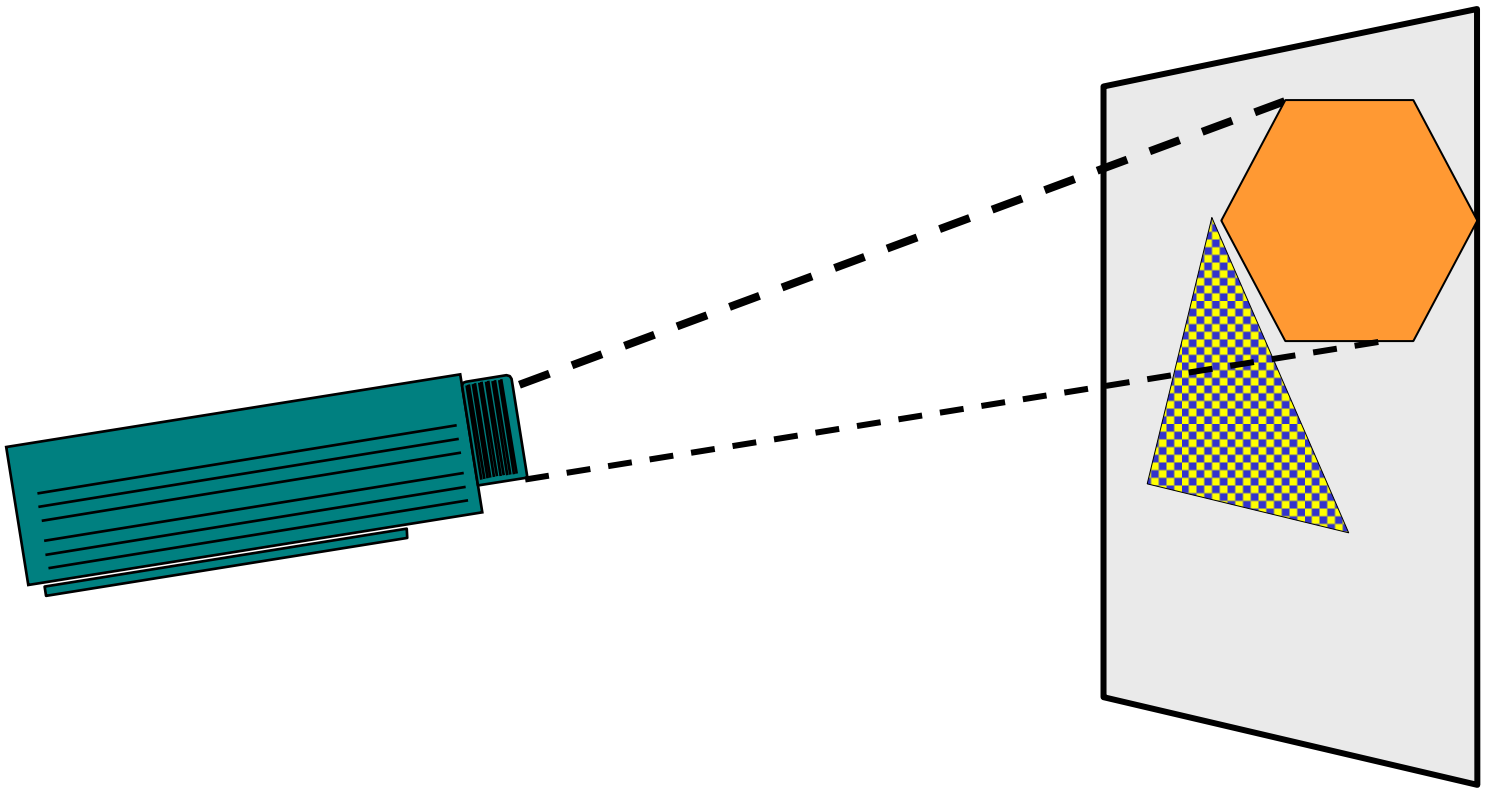
Typically, testers automate by creating static scripts.

Traditional Automated Testing



Given enough time, these scripts will cover the behavior.

Traditional Automated Testing

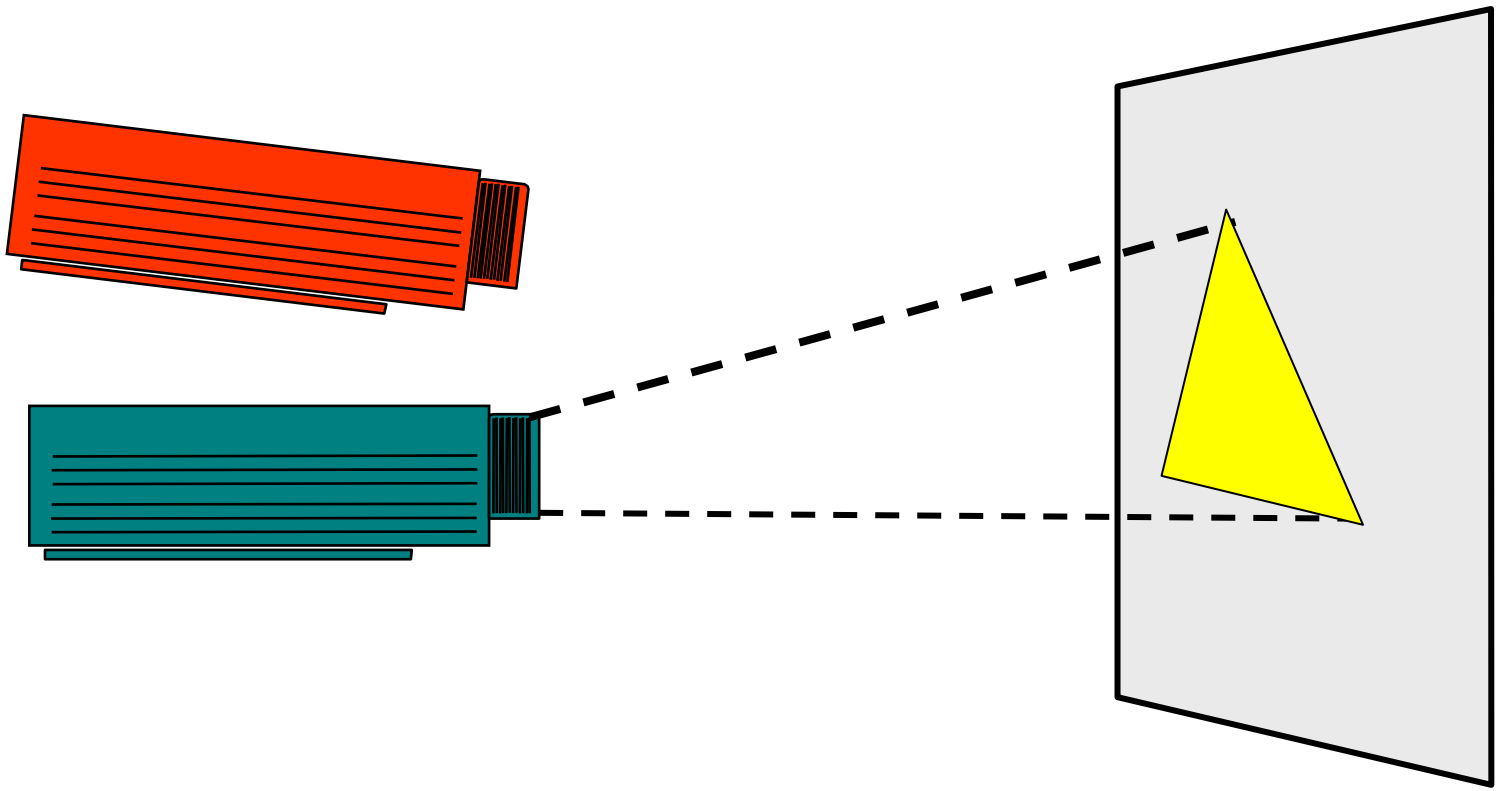


But what happens when the software's behavior changes?

So What's a Model?

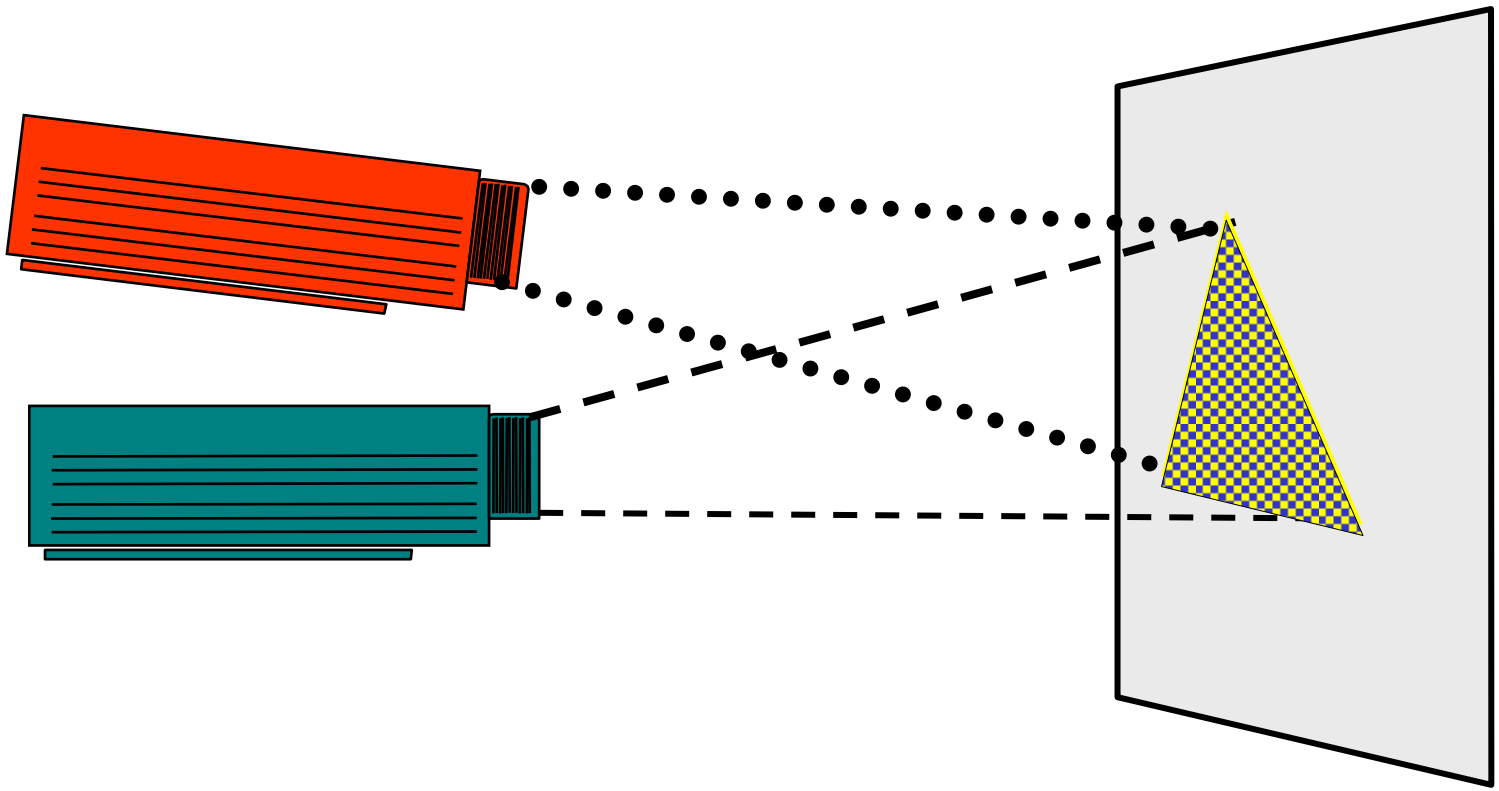
- A model is a description of a system's behavior.
- Models are simpler than the systems they describe.
- Models help us understand and predict the system's behavior.

Model-Based Testing



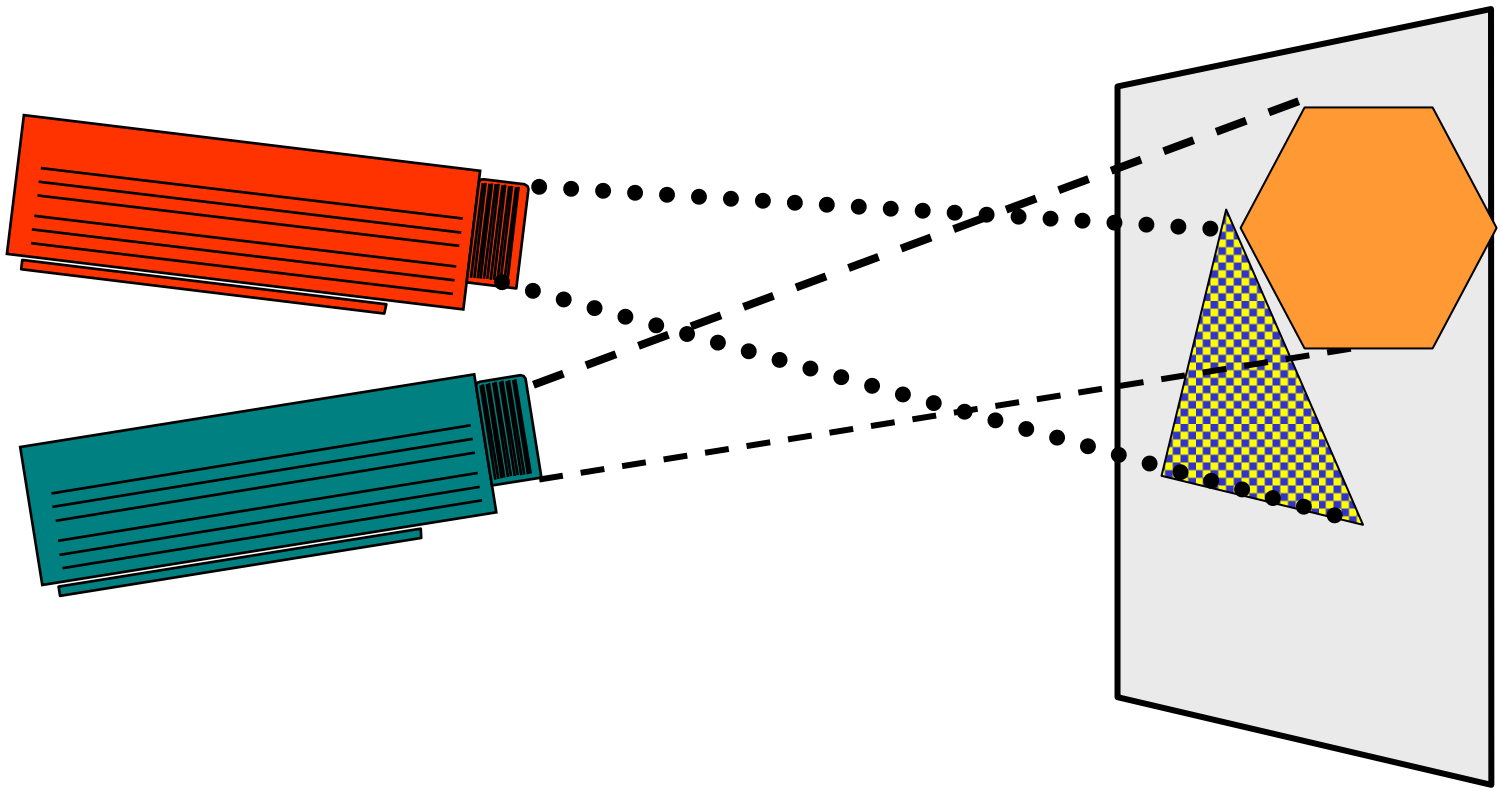
Now, imagine that the top projector is your model.

Model-Based Testing



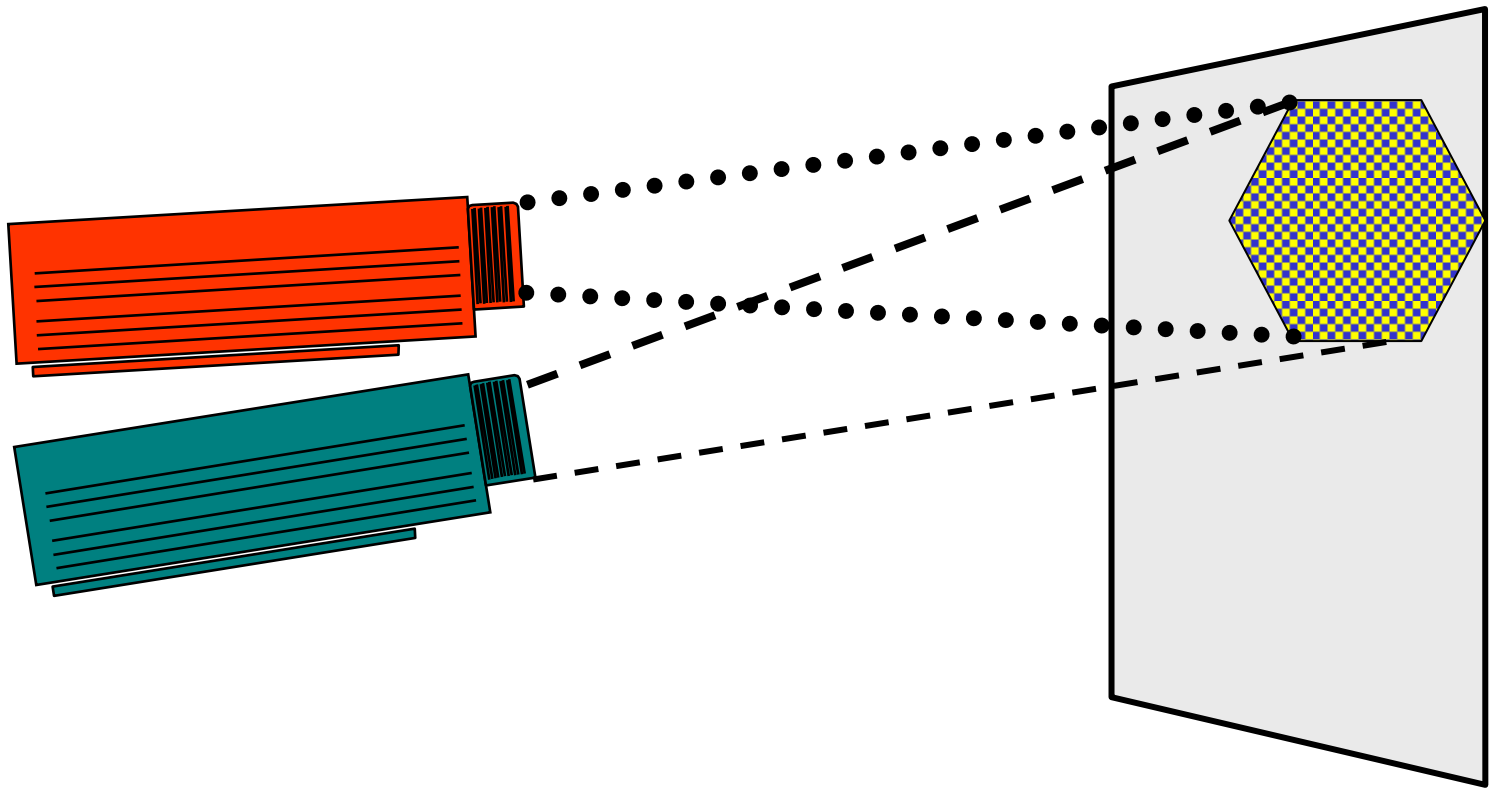
The model generates tests to cover the behavior.

Model-Based Testing



… and when the behavior changes…

Model-Based Testing



… so do the tests.

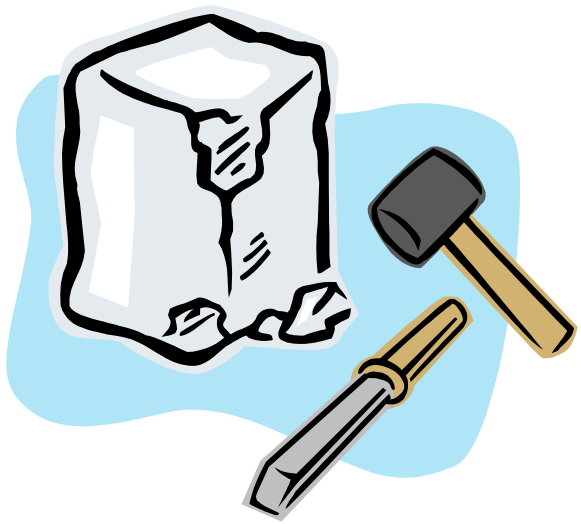
Using Models to Test

- What type of model do I use?
- How do I create the model?
- How do I choose tests?
- How do I verify results?

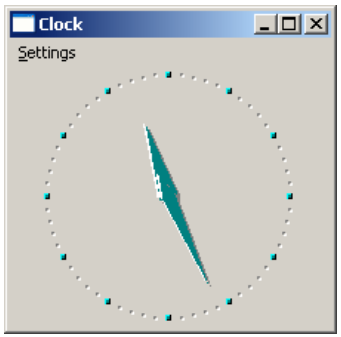
Many Types of Models

- States
- Monkeys
- Sets
- Grammars
- Combinations
- Other

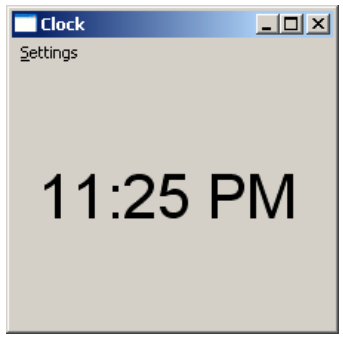
Creating a Model

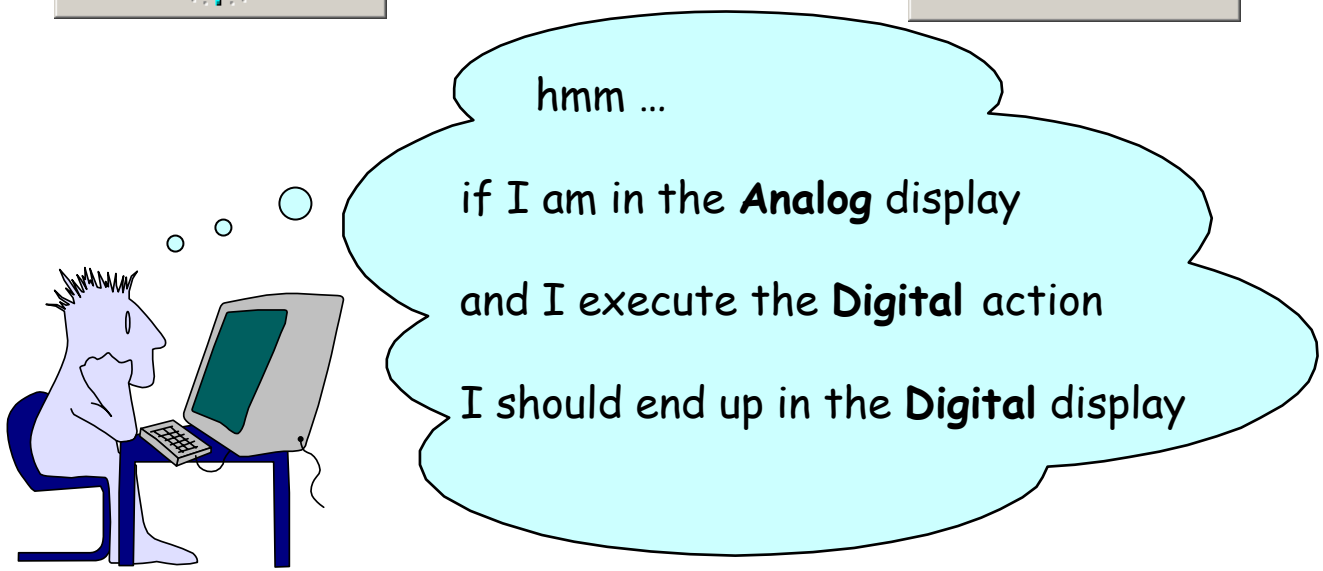


We All Use Mental Models Already



Digital



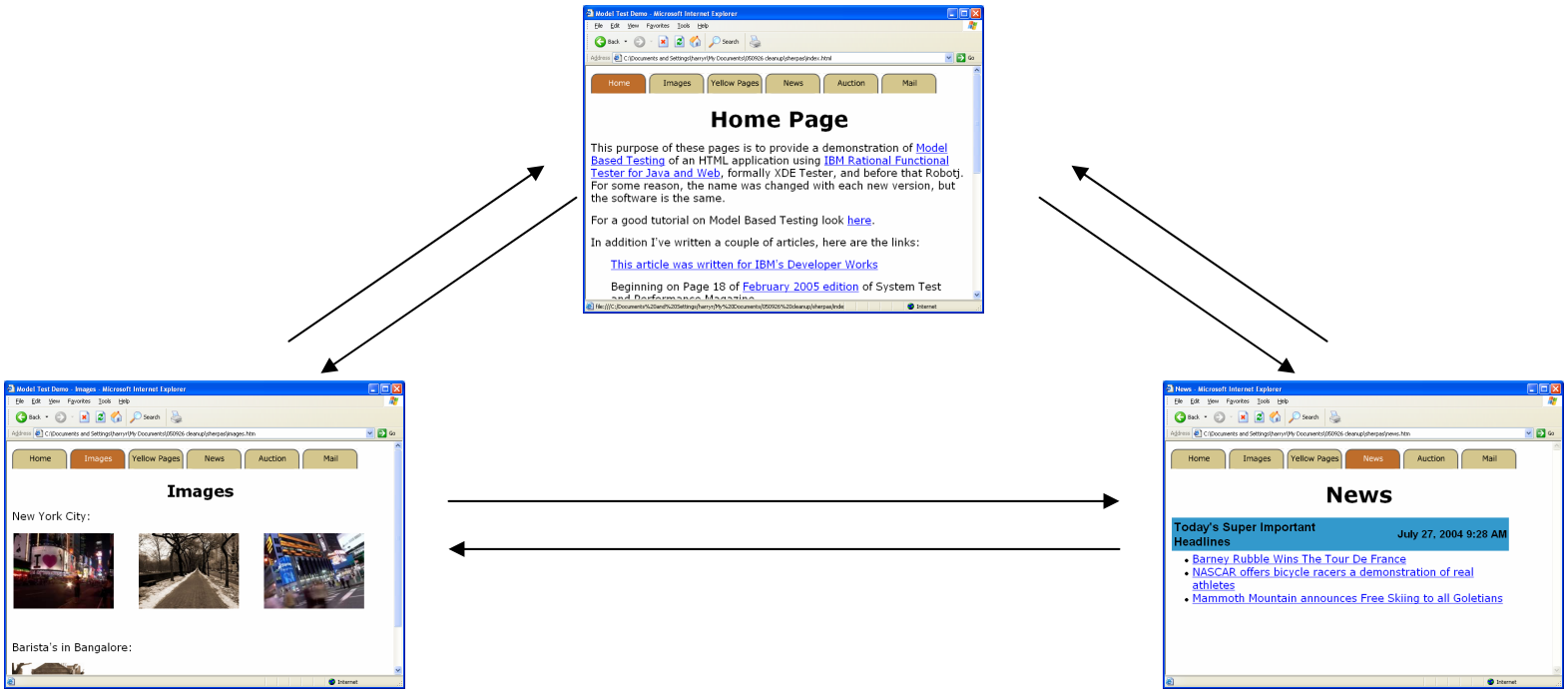


State Table Representation

StartState	Action	EndState
Analog	Digital	Digital

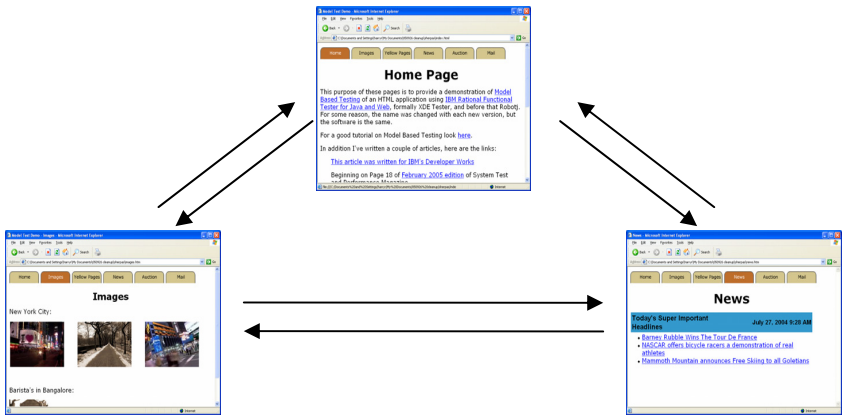
Exercise 1

Modeling a Website



Based on “Model-Based Testing: Not for Dummies” by Jeff Feldstein

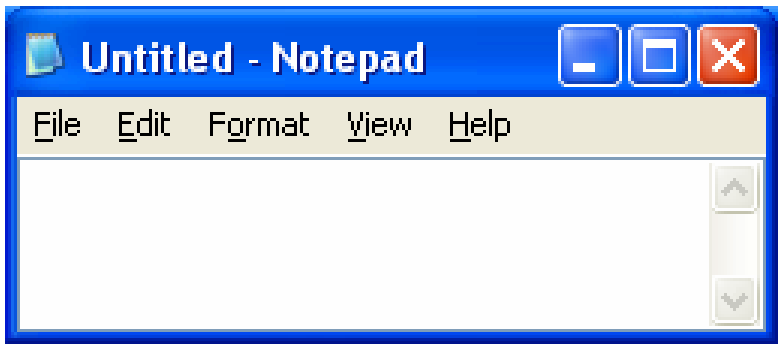
State Table Representation



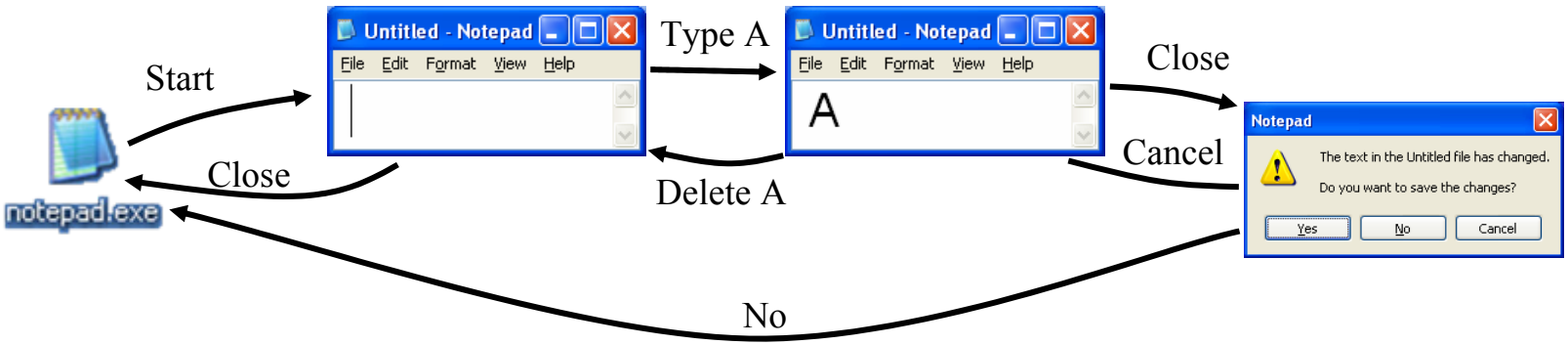
StartState	Action	EndState
HomePage	ImageTab	ImagePage
HomePage	NewsTab	NewsPage
ImagePage	HomeTab	HomePage
ImagePage	NewsTab	NewsPage
NewsPage	HomeTab	HomePage
NewsPage	ImageTab	ImagePage

Exercise 2

Modeling Notepad



Notepad Model

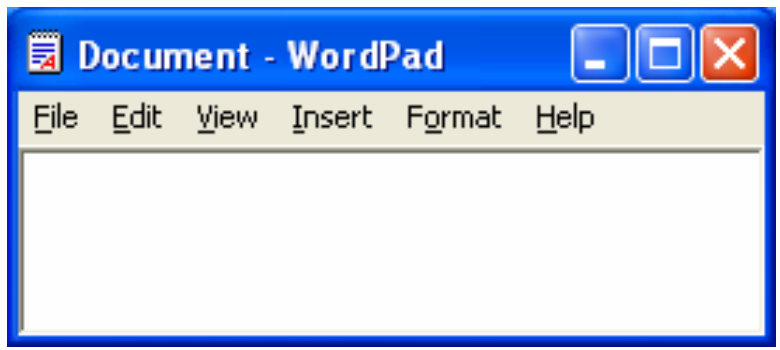


Notepad State Table

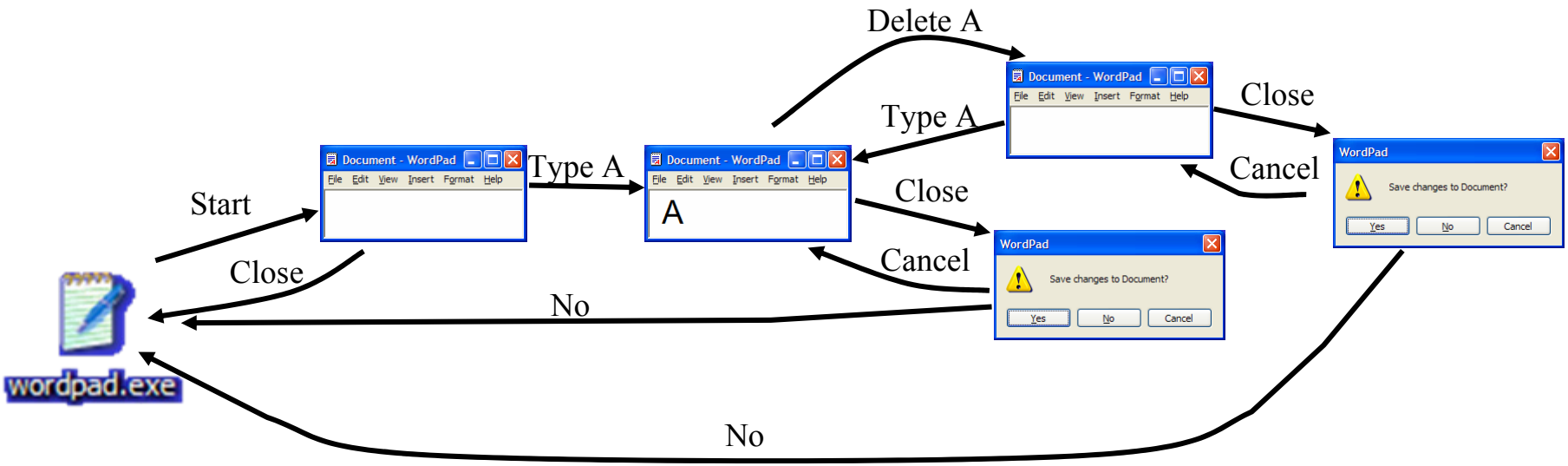
StartState	Action	EndState
NotRunning	Start	MainWindow
WindowEmpty	TypeA	WindowDirty
WindowEmpty	Close	NotRunning
WindowDirty	Delete	WindowEmpty
WindowDirty	Close	SaveDialog
SaveDialog	Cancel	WindowDirty
SaveDialog	No	NotRunning

Exercise 3

Modeling Wordpad



Wordpad Model

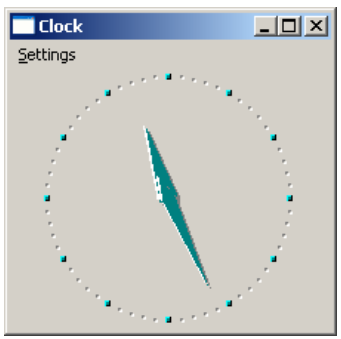


Wordpad State Table

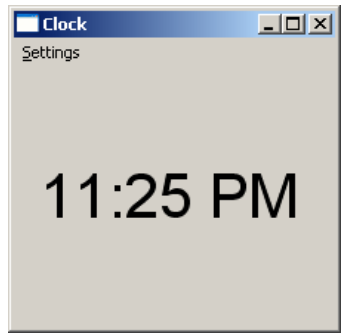
StartState	Action	EndState
NotRunning	Start	MainWindow
WindowEmpty1	TypeA	WindowDirty
WindowEmpty1	Close	NotRunning
WindowDirty	Delete	WindowEmpty2
WindowDirty	Close	SaveDialog1
WindowEmpty2	TypeA	WindowDirty
WindowEmpty2	Cancel	SaveDialog2
SaveDialog1	Cancel	WindowDirty
SaveDialog1	No	NotRunning
SaveDialog2	Cancel	WindowEmpty2
SaveDialog2	No	NotRunning

Exercise 4

Modeling Clock using Rules

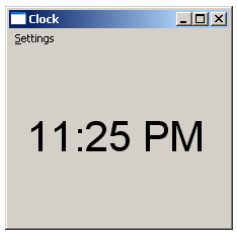


Digital



Modeling Clock States

In our discussion of the Clock behavior, we only tracked a few data values in the application:



was described as

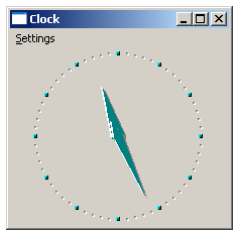
Running
Digital
Framed

was described as

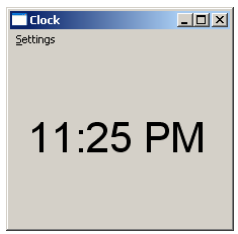
Running
Analog
Unframed

Modeling Clock Actions

We also tracked how our actions change those values:



Digital

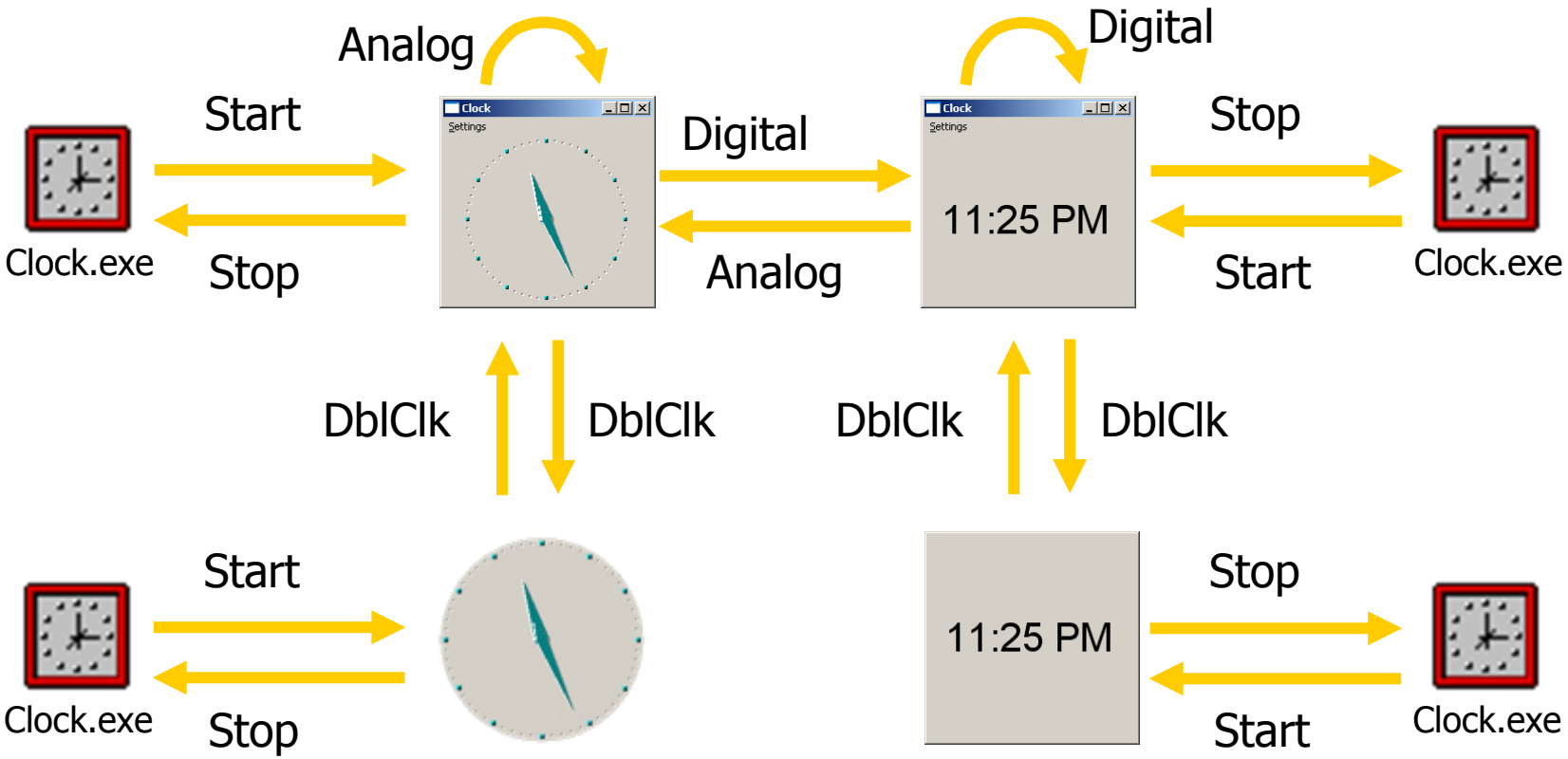


Running
Analog
Framed

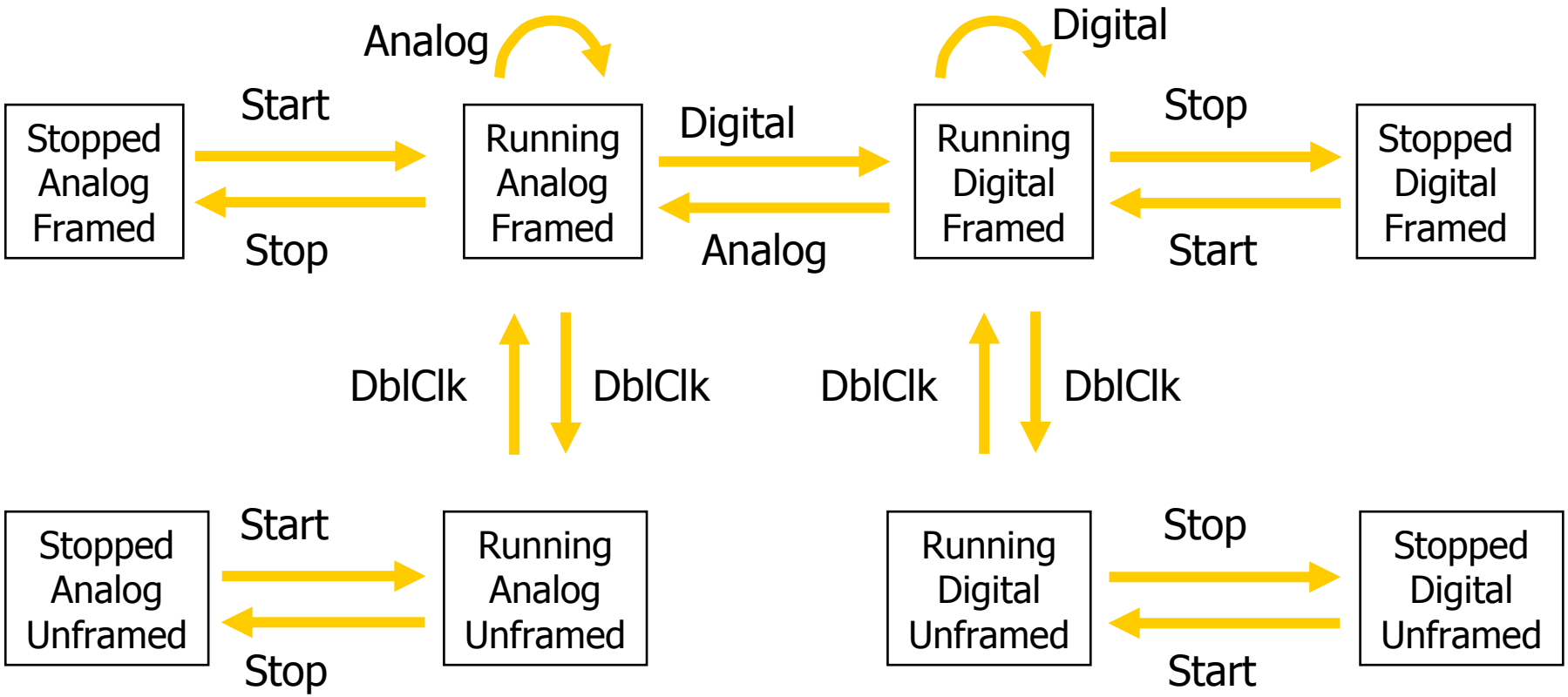
Digital

Running
Digital
Framed

So, we can replace this model …

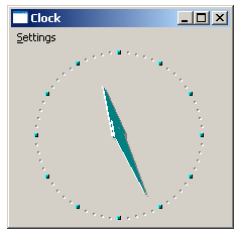


… with a state variable model



Rules for the Stop action

- If the Clock is Running, then the user can execute the 'Stop' action.
- The 'Stop' action puts you in Stopped mode



Stop

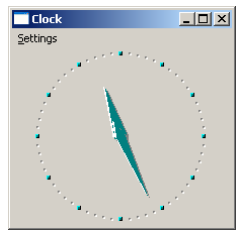
Clock.exe

```
static void Stop( )
requires (AppStatus == AppValues.Running);
{
   AppStatus = AppValues.Stopped;
}
```

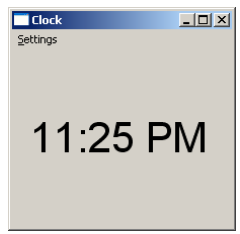
Rules for the SelectDigital action

- If the Clock is Running and Framed, then the user can execute the 'SelectDigital' action
- The 'SelectDigital' action puts you in Digital mode

```
static void SelectDigital( )
requires (AppStatus == AppValues.Running)
 && (FrameStatus == FrameValues.Framed);
{
  ModeStatus = ModeValues.Digital;
}
```



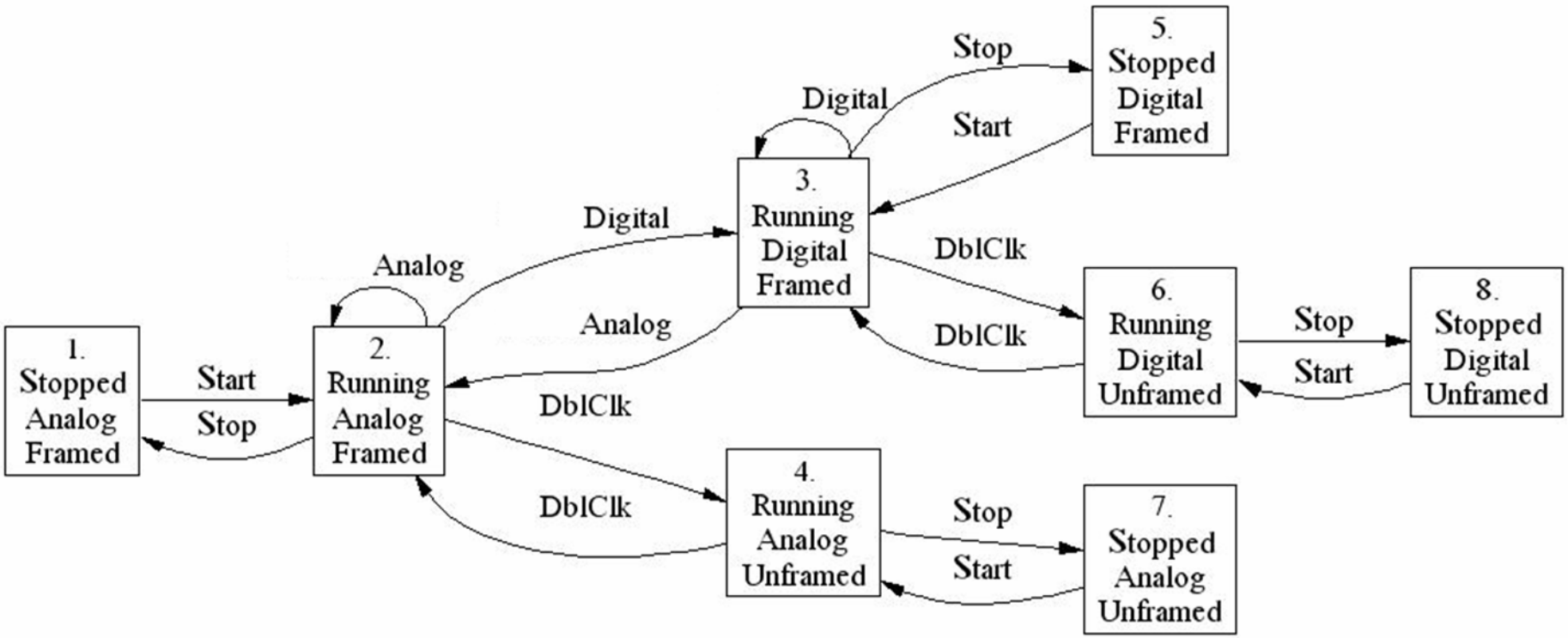
Select Digital



A generated state table!

STARTSTATE	ACTION	ENDSTATE
Stopped Analog Framed	Start	Running Analog Framed
Running Analog Framed	Stop	Stopped Analog Framed
Running Analog Framed	SelectAnalog	Running Analog Framed
Running Analog Framed	SelectDigital	Running Digital Framed
Running Analog Framed	DblClk	Running Analog Unframed
Running Digital Framed	Stop	Stopped Digital Framed
Running Digital Framed	SelectAnalog	Running Analog Framed
Running Digital Framed	SelectDigital	Running Digital Framed
Running Digital Framed	DblClk	Running Digital Unframed
Running Analog Unframed	Stop	Stopped Analog Unframed
Running Analog Unframed	DblClk	Running Analog Framed
Stopped Digital Framed	Start	Running Digital Framed
Running Digital Unframed	Stop	Stopped Digital Unframed
Running Digital Unframed	DblClk	Running Digital Framed
Stopped Analog Unframed	Start	Running Analog Unframed
Stopped Digital Unframed	Start	Running Digital Unframed

A state diagram



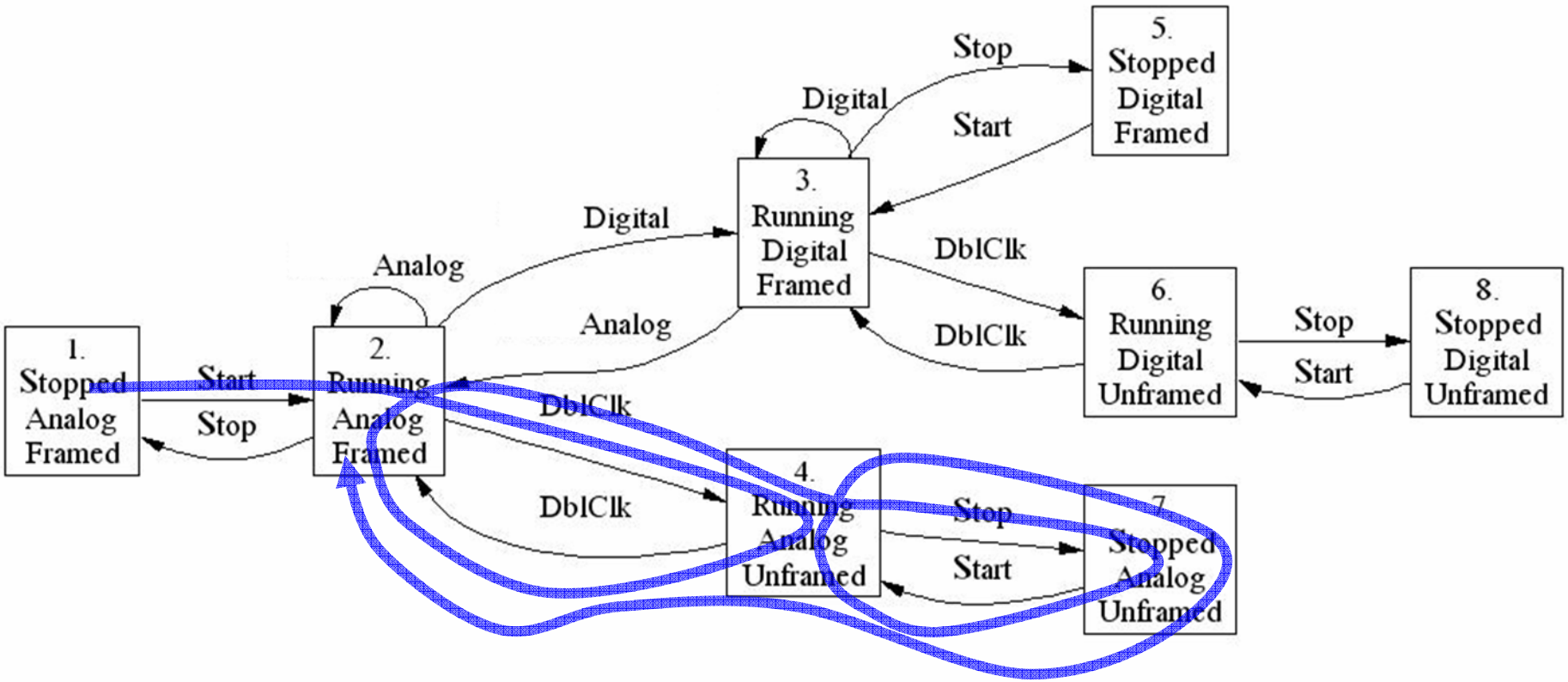
Generating Tests from a Model

Generating Test Sequences

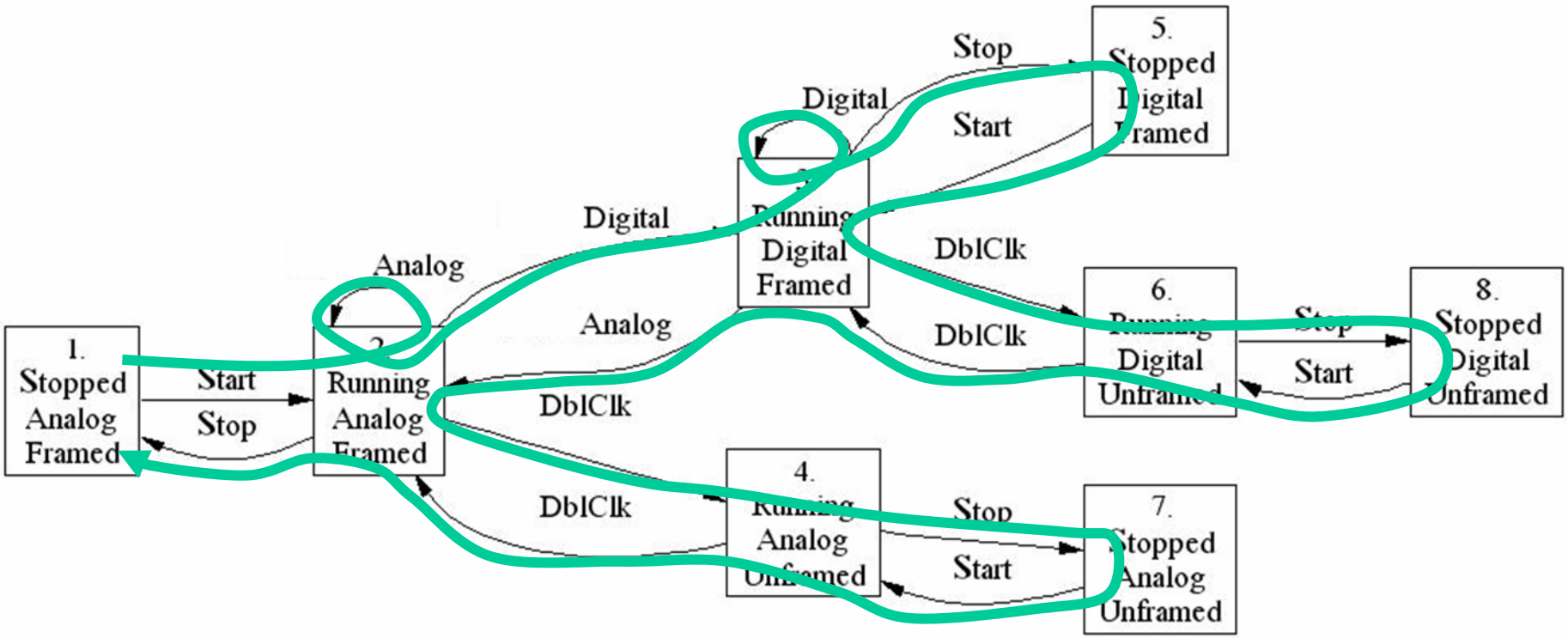
We can use the machine-readable model to create test sequences:

- Random walk
- All transitions
- Shortest paths first
- Most likely paths first

A Random Walk



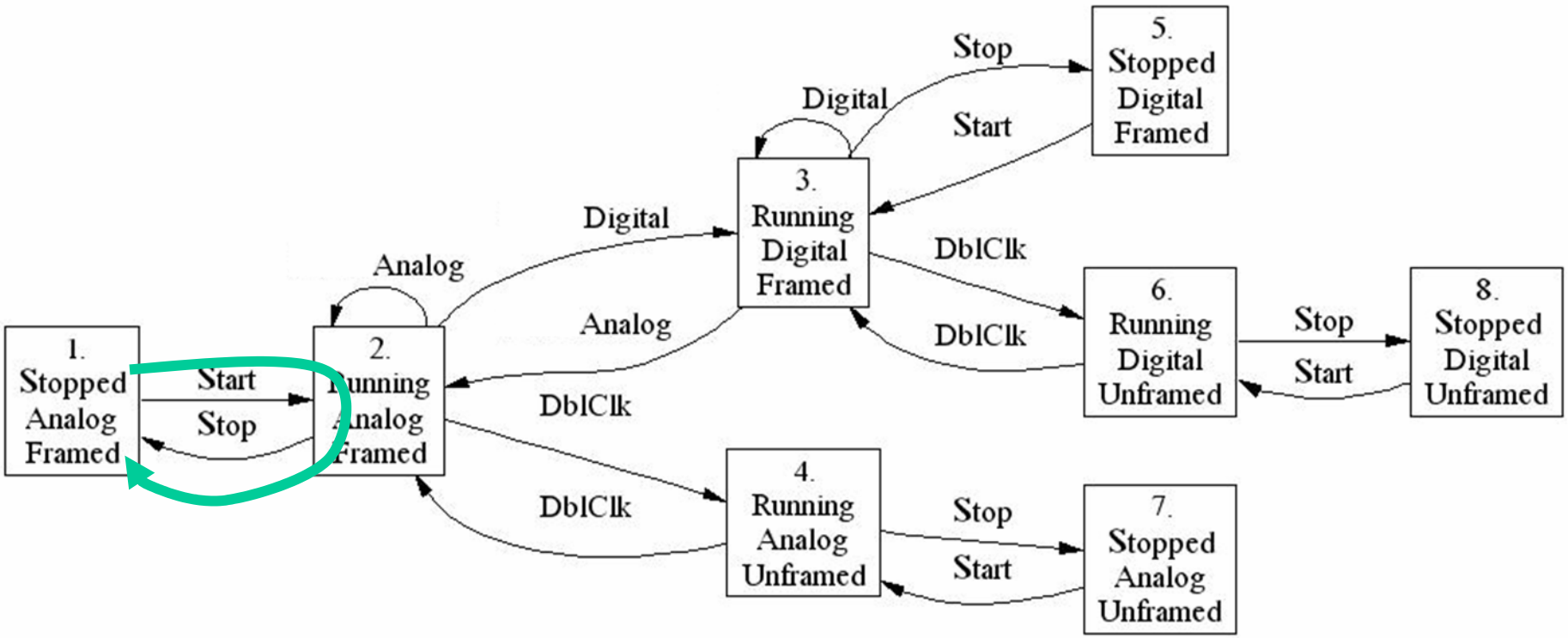
A sequence that hits all transitions



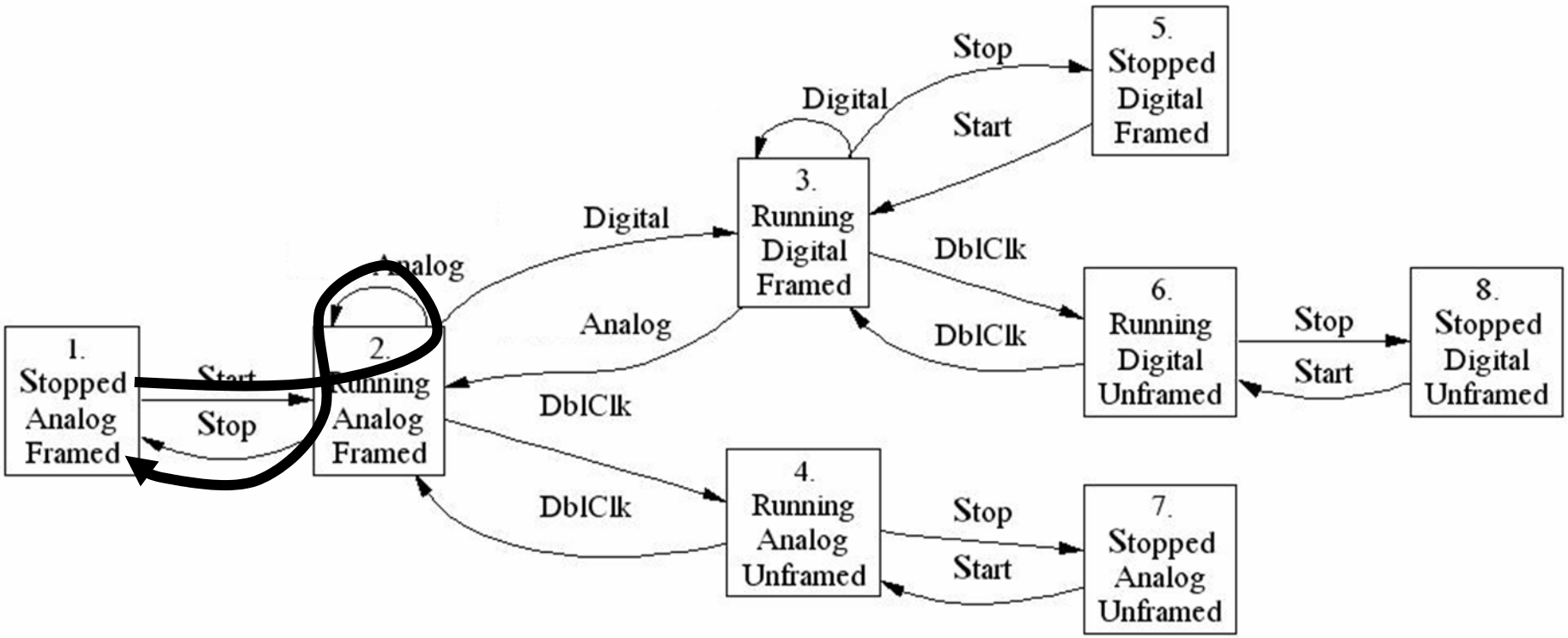
An all-transitions sequence

1. Start
2. Analog
3. Digital
4. Digital
5. Stop
6. Start
7. Double Click
8. Stop
9. Start
10. Double Click
11. Analog
12. Double Click
13. Stop
14. Start
15. Double Click
16. Stop

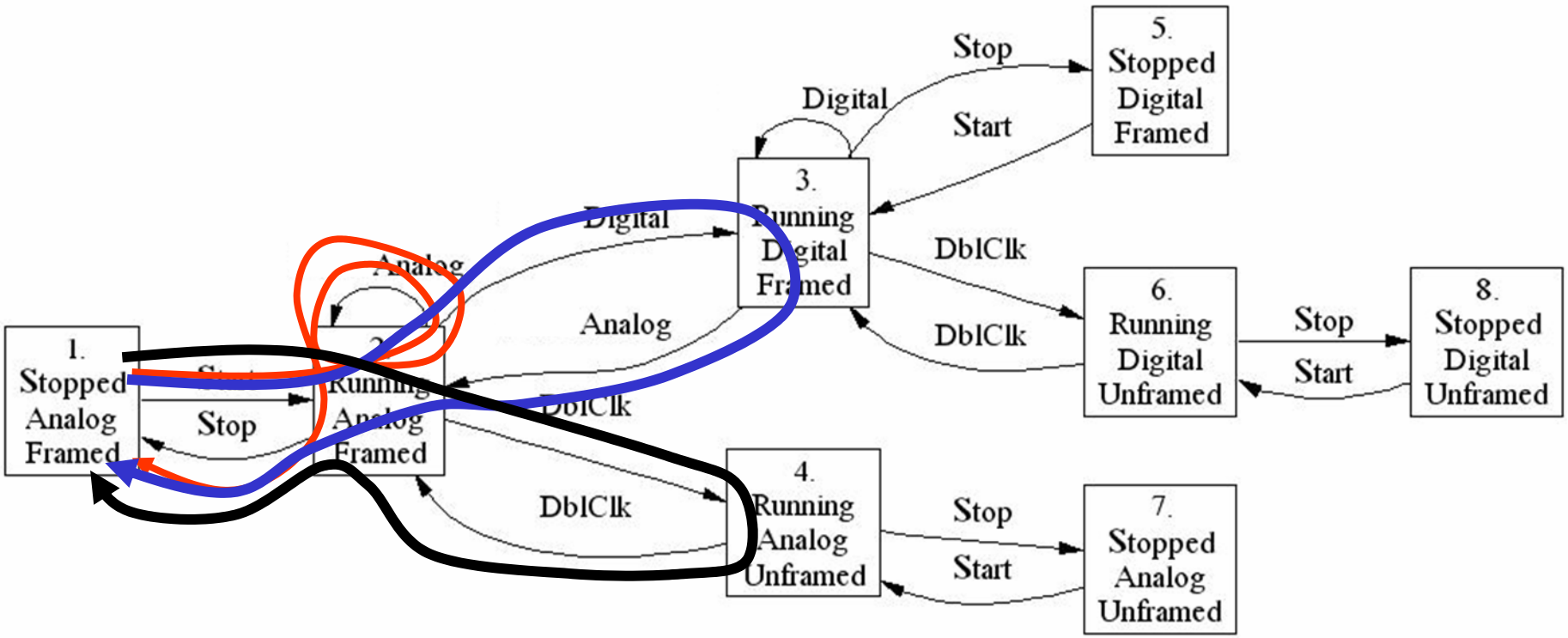
All paths of length 2



All paths of length 3



All paths of length 4

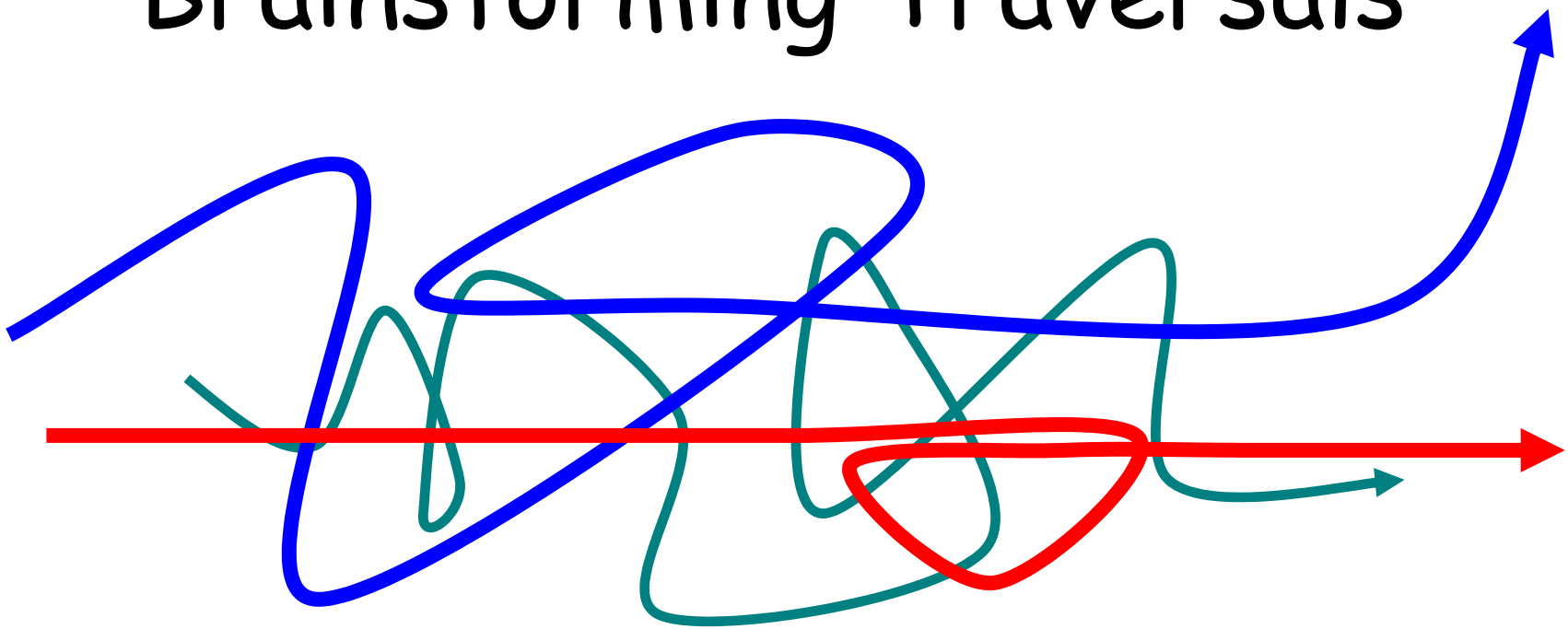


All paths of length < 5

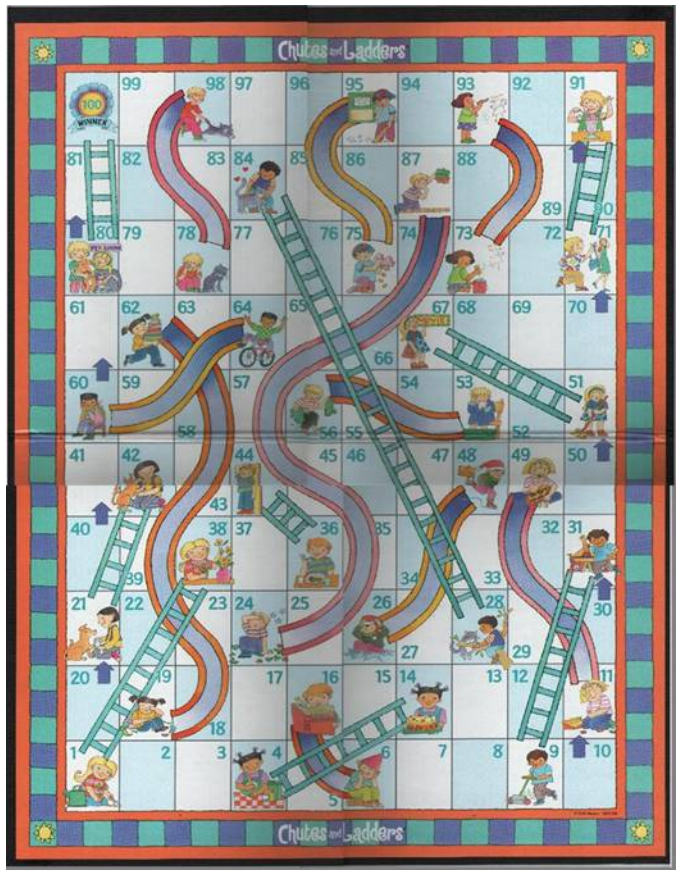
1. Start, Stop
2. Start, Analog, Stop
3. Start, Analog, Analog, Stop
4. Start, Digital, Analog, Stop
5. Start, Double Click, Double Click, Stop

Exercise 5

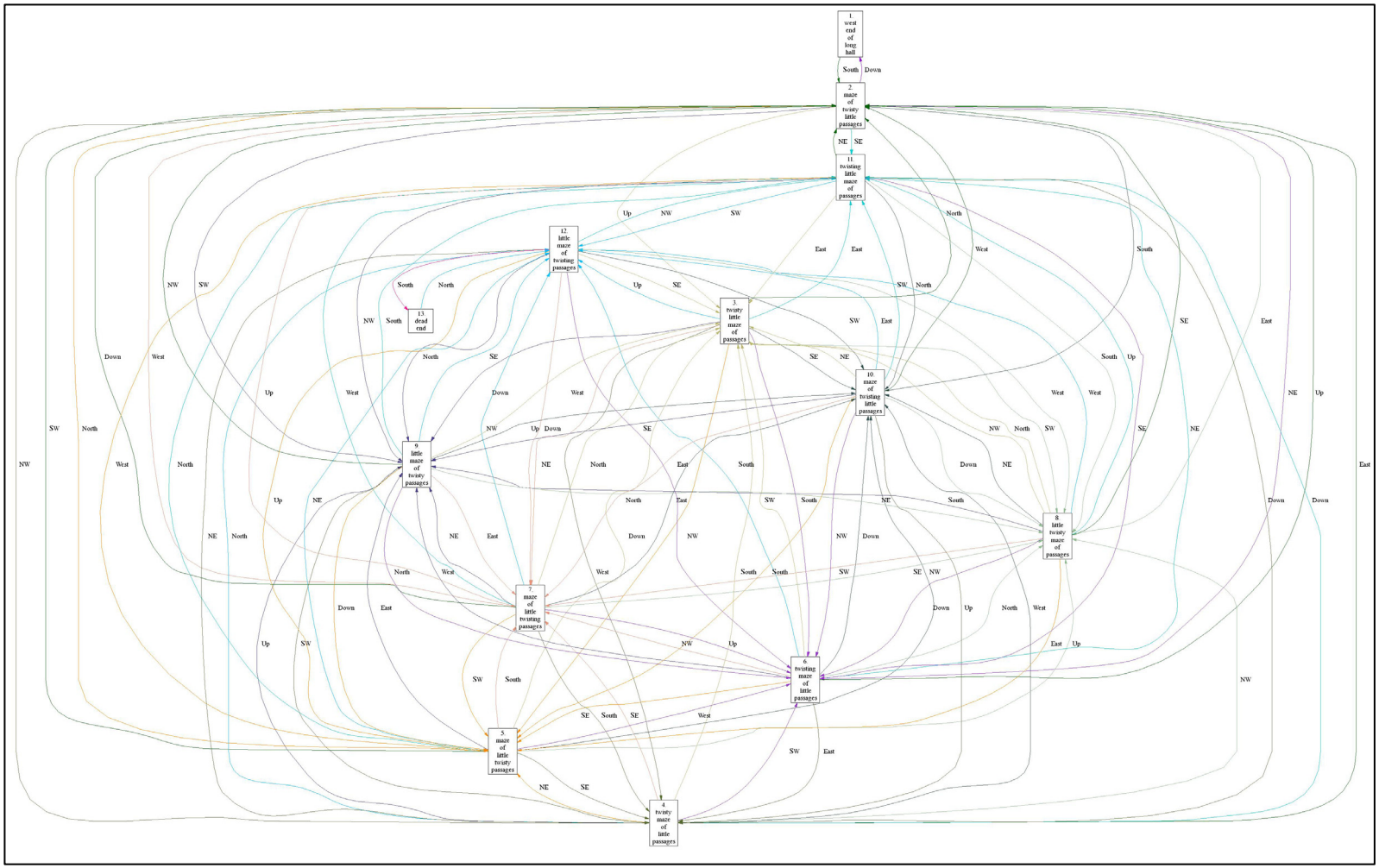
Brainstorming Traversals



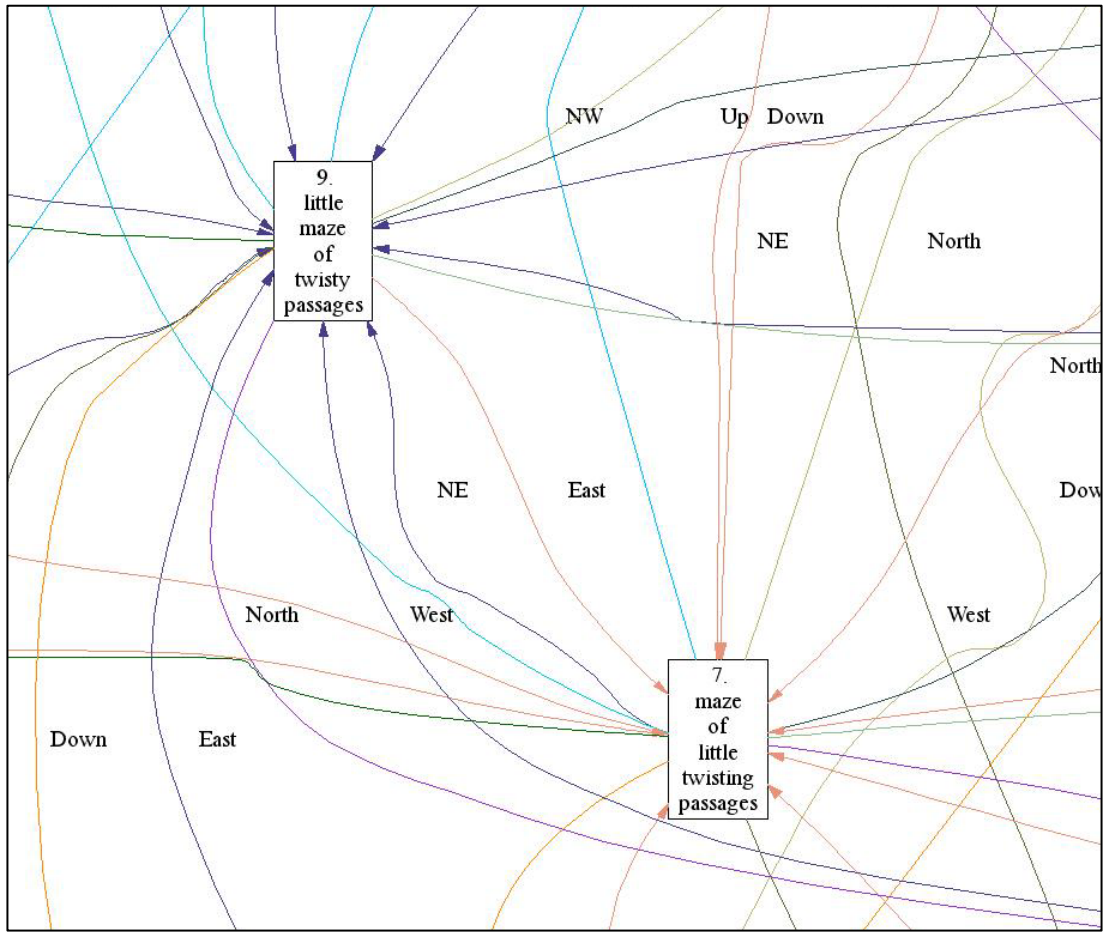
How would you test this?



Pathological



Closer in …

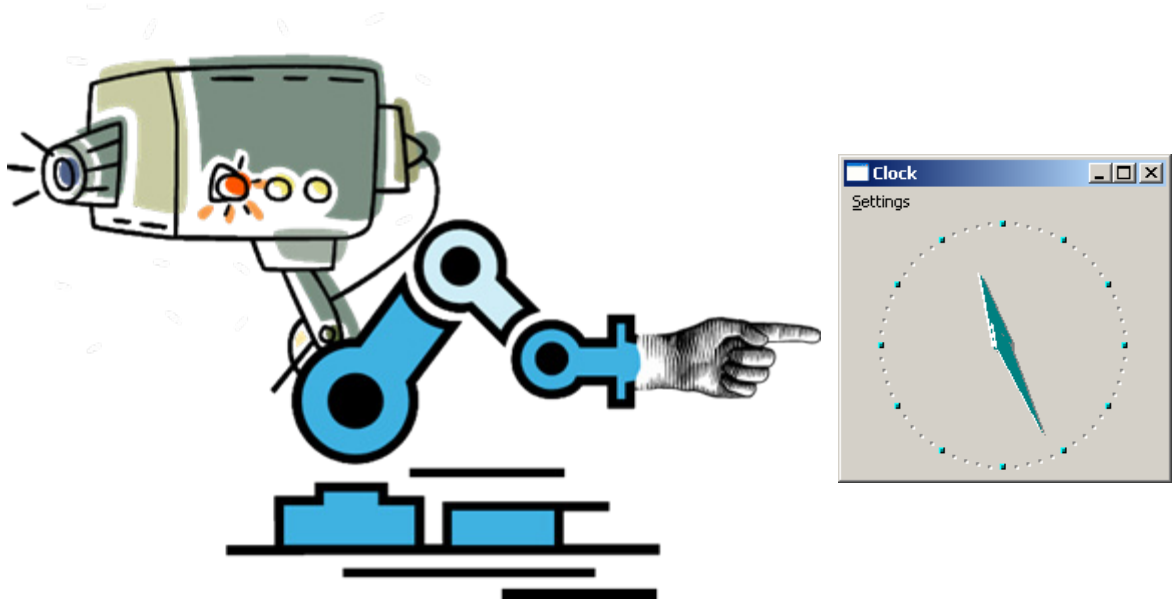


Partial state table for the maze

*west end of long hall	South	maze of twisty little passages
maze of twisty little passages	Up	twisty little maze of passages
maze of twisty little passages	NW	twisty maze of little passages
maze of twisty little passages	North	maze of little twisty passages
maze of twisty little passages	NE	twisting maze of little passages
maze of twisty little passages	West	maze of little twisting passages
maze of twisty little passages	East	little twisty maze of passages
maze of twisty little passages	SW	little maze of twisty passages

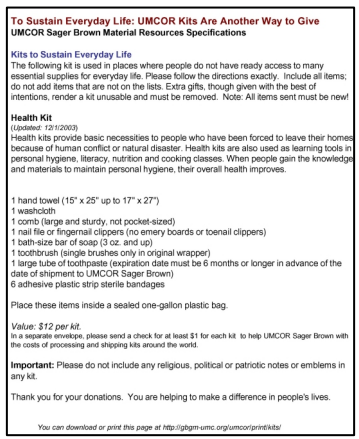
Executing the Test Actions

1. Start
2. Analog
3. Digital
4. Digital
5. Stop
6. Start
7. Double Click
8. Stop



Exercise 6

Modeling from a Spec



To Sustain Everyday Life: UMCOR Kits Are Another Way to Give
UMCOR Sager Brown Material Resources Specifications

Kits to Sustain Everyday Life
The following kit is used in places where people do not have ready access to many essential supplies for everyday life. Please follow the directions exactly. Include all items; do not add items that are not on the lists. Extra gifts, though given with the best of intentions, render a kit unusable and must be removed. Note: All items sent must be new!

Health Kit
(*Updated: 12/1/2003*)
Health kits provide basic necessities to people who have been forced to leave their homes because of human conflict or natural disaster. Health kits are also used as learning tools in personal hygiene, literacy, nutrition and cooking classes. When people gain the knowledge and materials to maintain personal hygiene, their overall health improves.

1 hand towel (15" x 25" up to 17" x 27")
1 washcloth
1 comb (large and sturdy, not pocket-sized)
1 nail file or fingernail clippers (no emery boards or toenail clippers)
1 bath-size bar of soap (3 oz. and up)
1 toothbrush (single brushes only in original wrapper)
1 large tube of toothpaste (expiration date must be 6 months or longer in advance of the date of shipment to UMCOR Sager Brown)
6 adhesive plastic strip sterile bandages

Place these items inside a sealed one-gallon plastic bag.

Value: $12 per kit.
In a separate envelope, please send a check for at least $1 for each kit to help UMCOR Sager Brown with the costs of processing and shipping kits around the world.

Important: Please do not include any religious, political or patriotic notes or emblems in any kit.

Thank you for your donations. You are helping to make a difference in people's lives.

You can download or print this page at http://gbgm-umc.org/umcor/print/kits/

“Accessing an Account”

1. customer needs to log in to use the system
2. customer stays logged in to the system until she logs out
3. customer starts with no account
4. customer can create an account
5. customer can delete her account
6. customer can open an existing account
7. customer can close an existing account

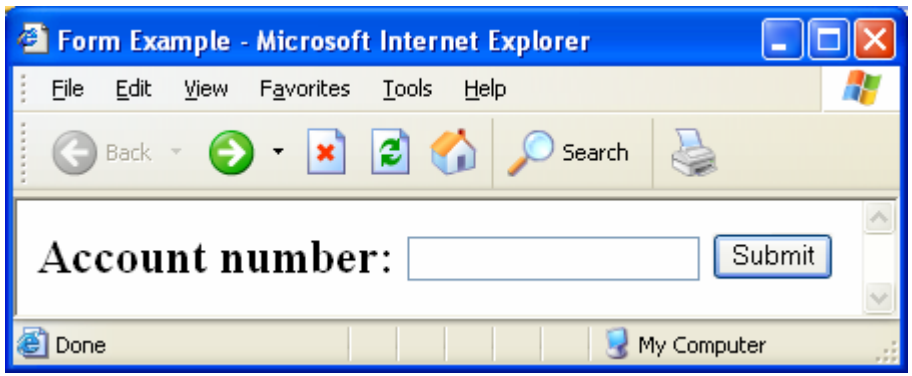
Verifying the Outcomes

Oracles

- Crashes
- Prediction
- Checking
- Pre-oracling
- Heuristics
- Filtering + Humans
- Assertions

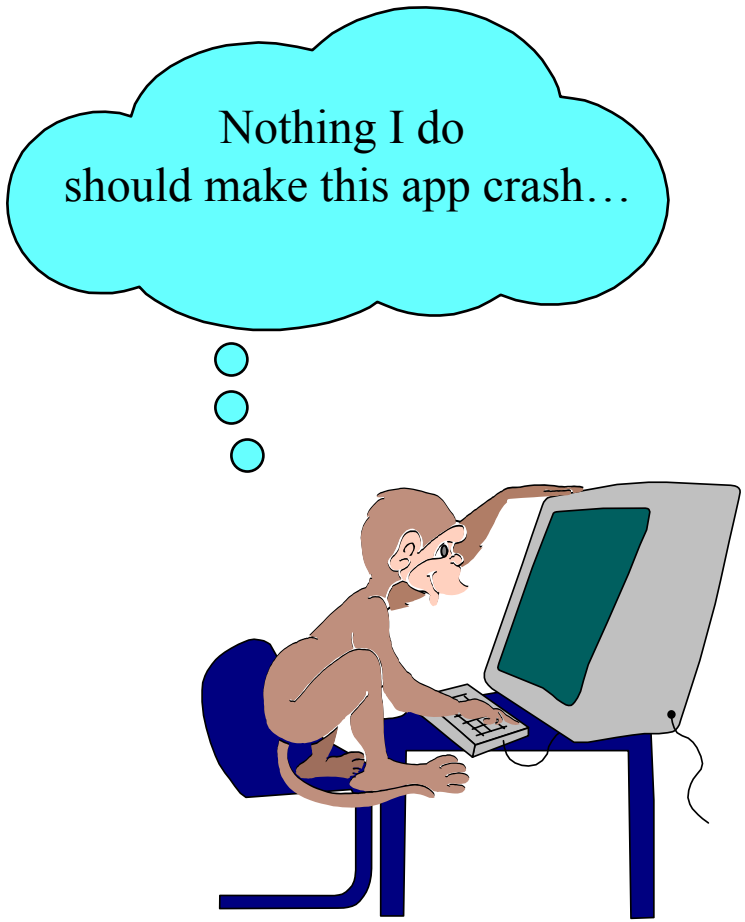
Exercise 7

Monkeying an Input Field



Inspired by Noel Nyman's article "Using Monkey Test Tools"

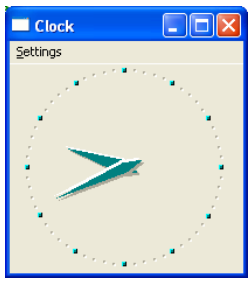
Monkey Models

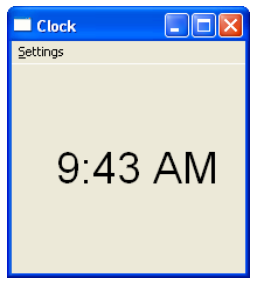


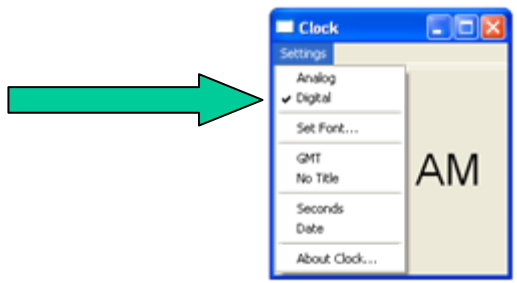
1.2
a P, a P, ab P, ab P 2, abc P, P, P, P,
\\scbuild1
Type 5
containing message laurie Bill's October
? A
? v?-0Qrg+ 'cQ?_<$ ` Z`i7c} oV? E1X …
nov 31, 2000 Oct, 2000 dates Wednesday …
alike. In Word documents, for example …
3 M note
RNL in Office 10

Predicting the Outcome

StartState	Action	EndState
Analog	Digital	Digital







Exercise 8

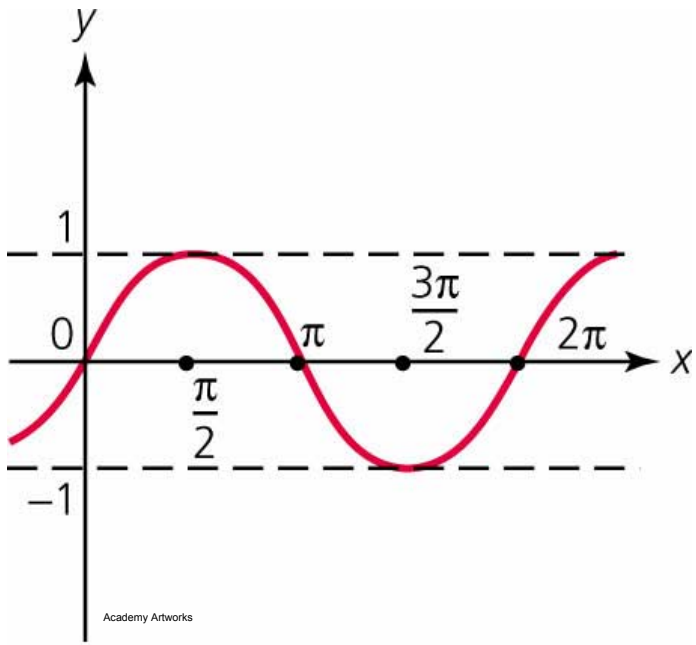
Oracling the Square Root Function

$$\sqrt{a}$$

$$\sqrt{a} * \sqrt{a} = a$$

Exercise 9

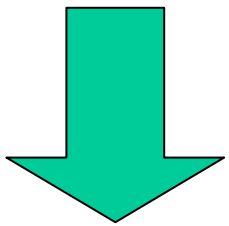
Oracling the Sine Function



Inspired by Doug Hoffman's article "Heuristic Test Oracles"

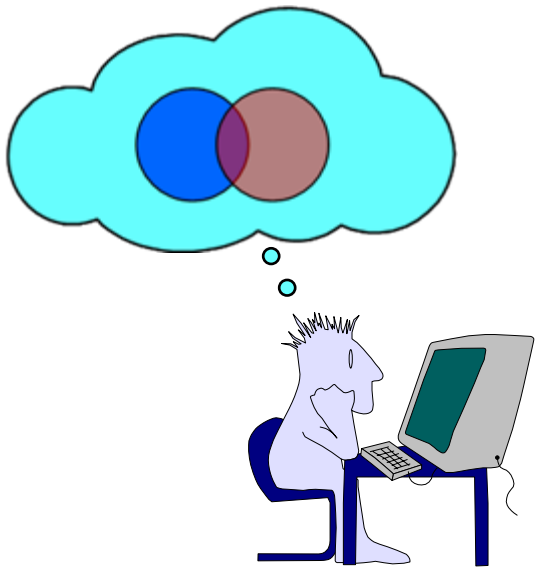
$$\sin^2(x) + \cos^2(x) = 1$$

$$\cos(x) = \sin(x - \pi)$$



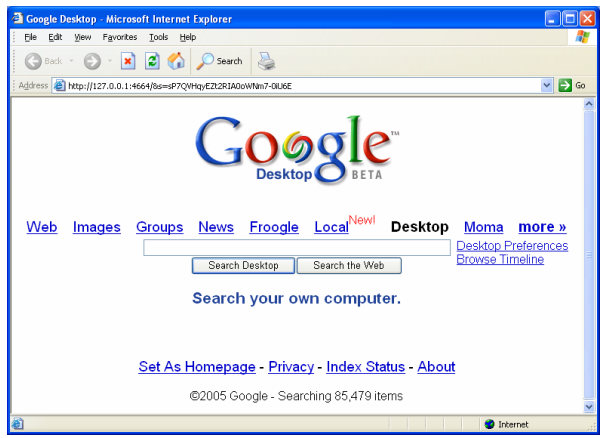
$$\sin^2(x) + \sin^2(x - \pi) = 1$$

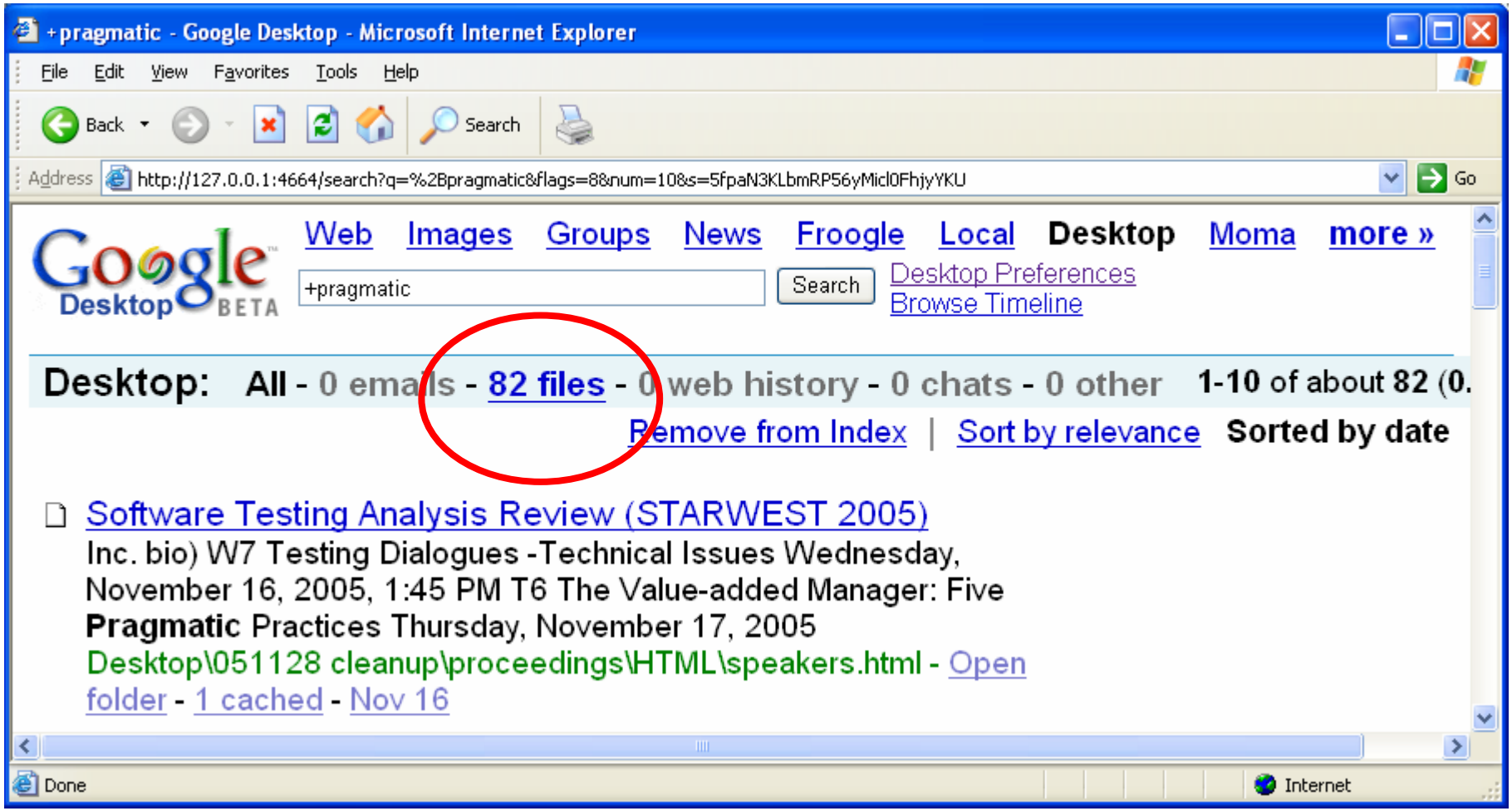
Modeling with Sets

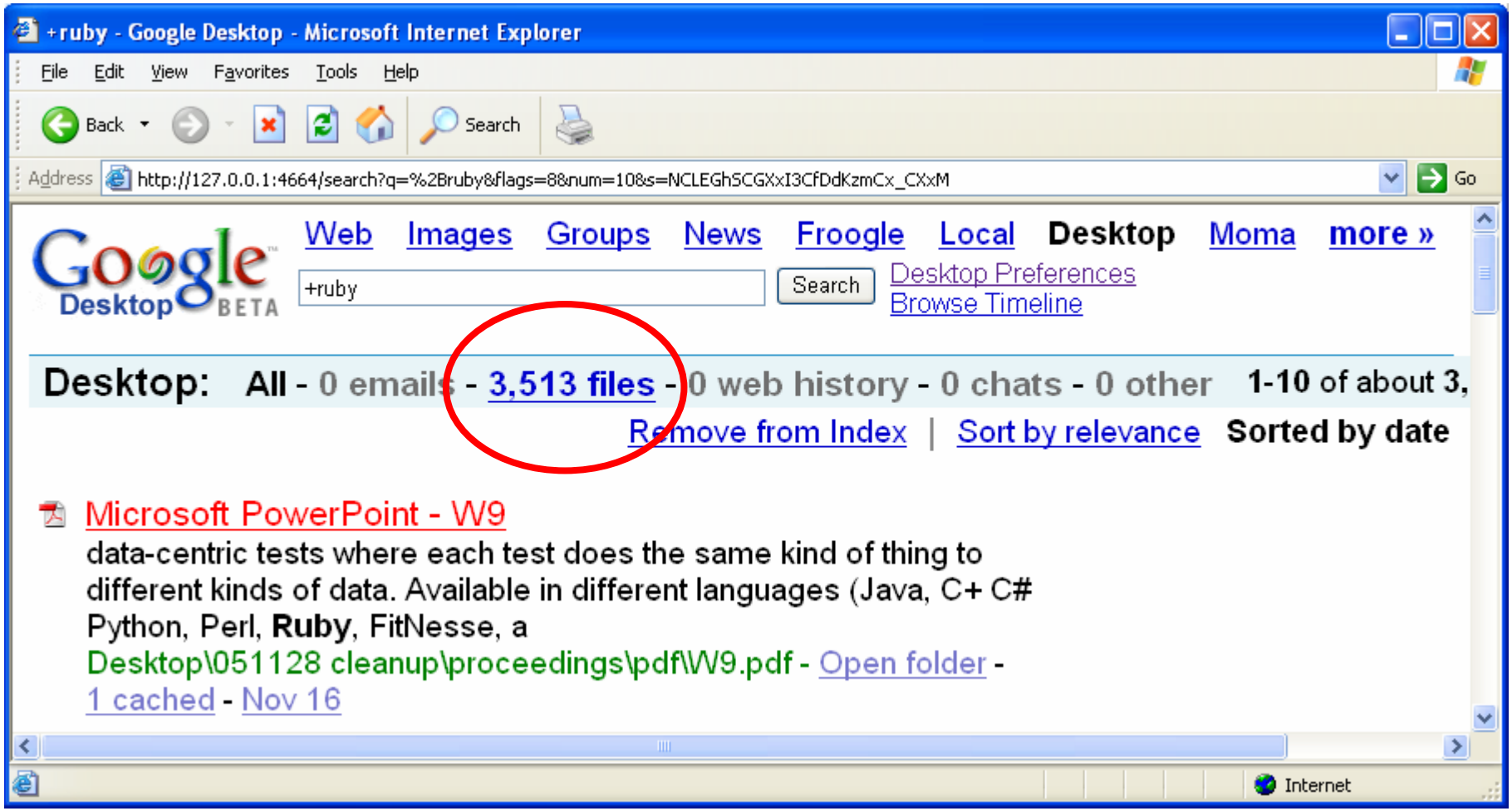


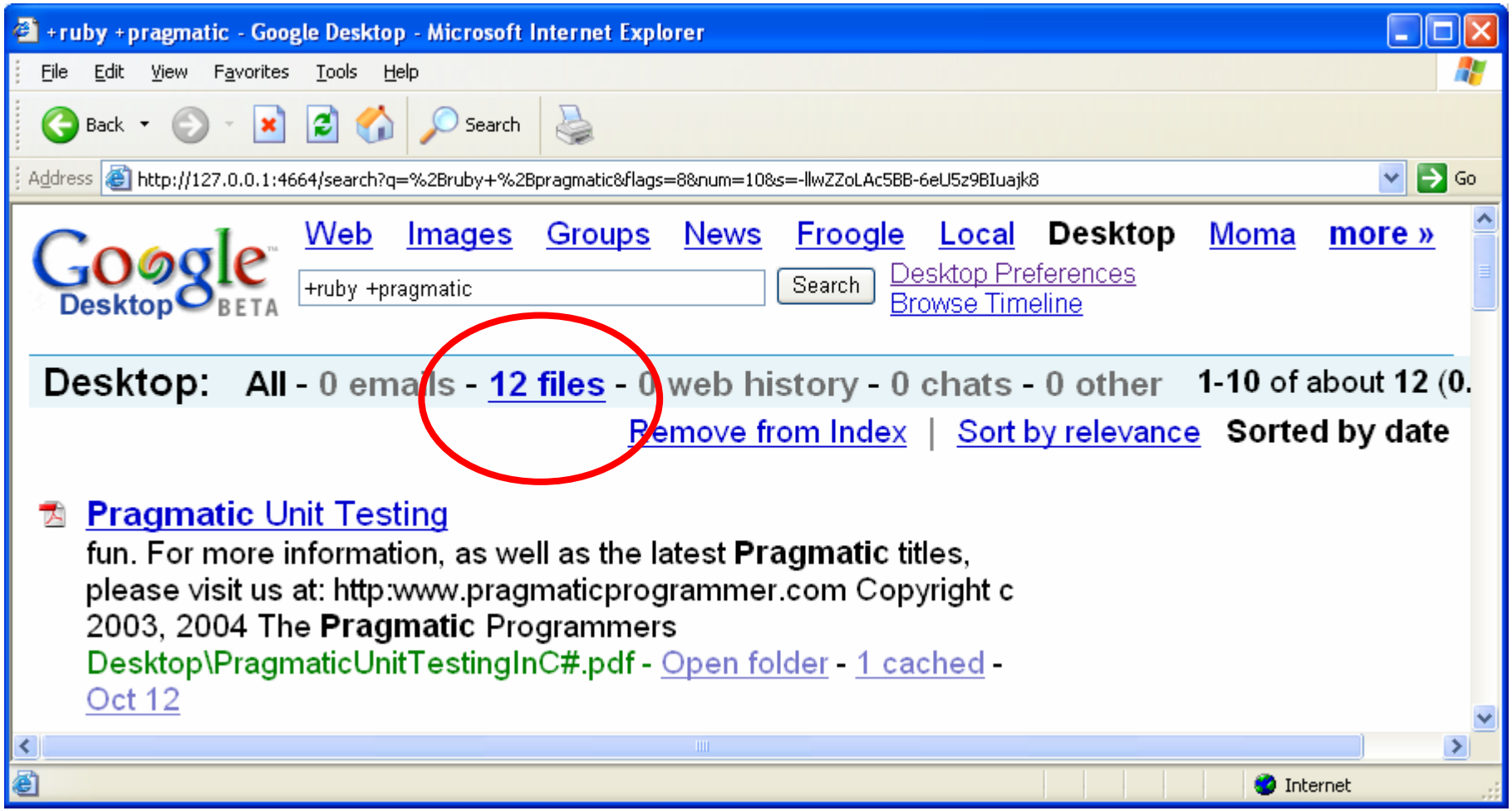
Exercise 10

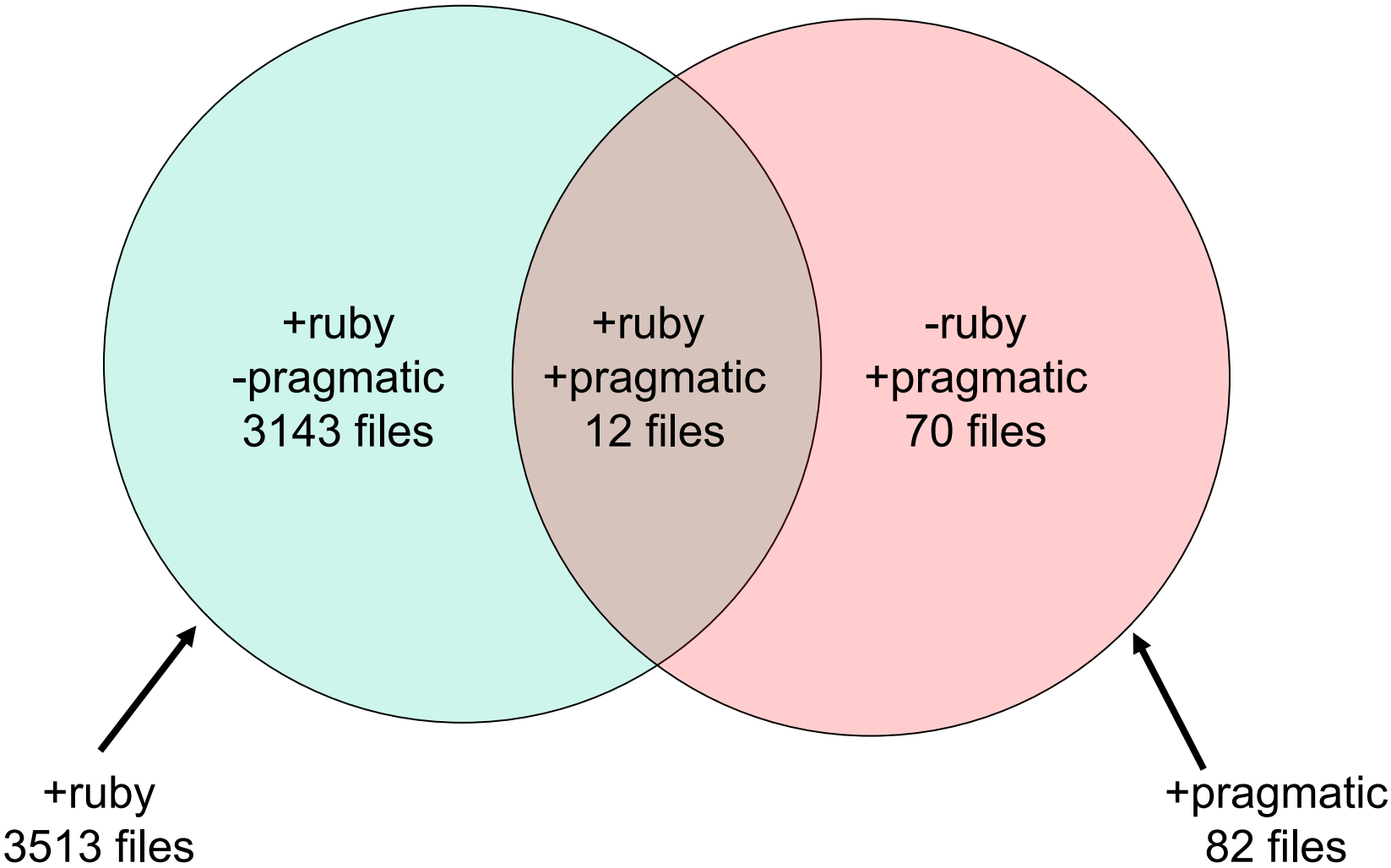
Modeling Search

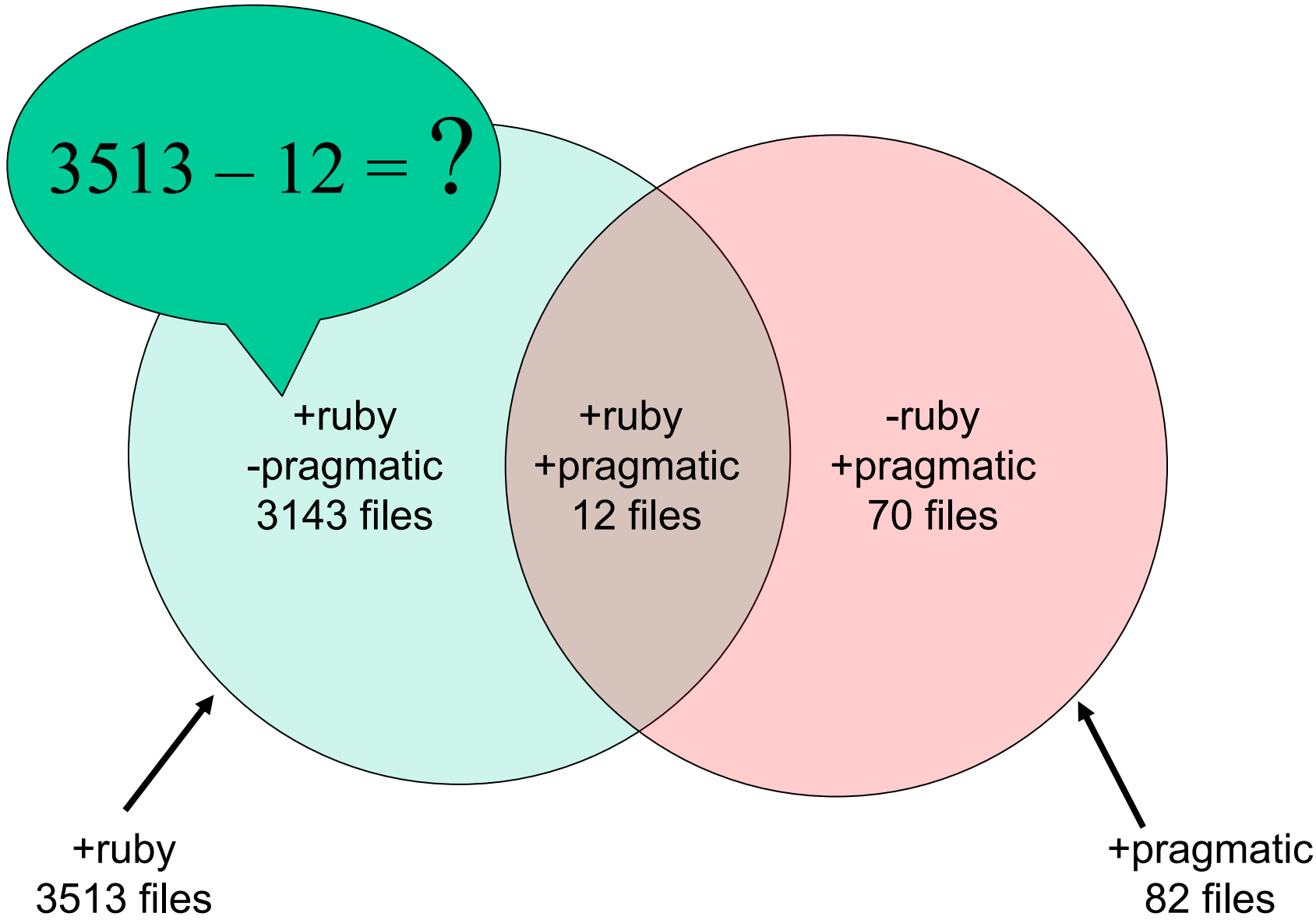




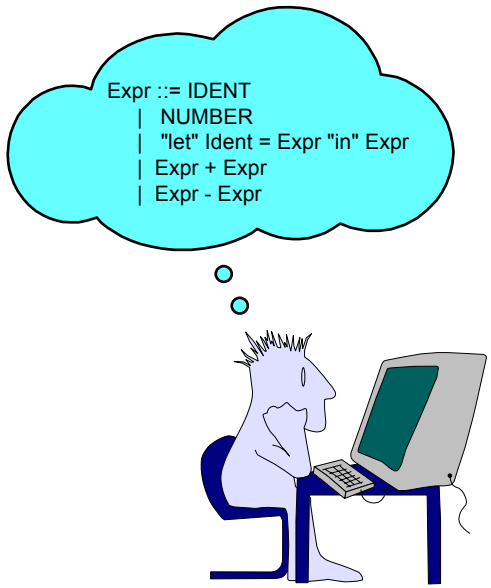








Modeling with Grammars

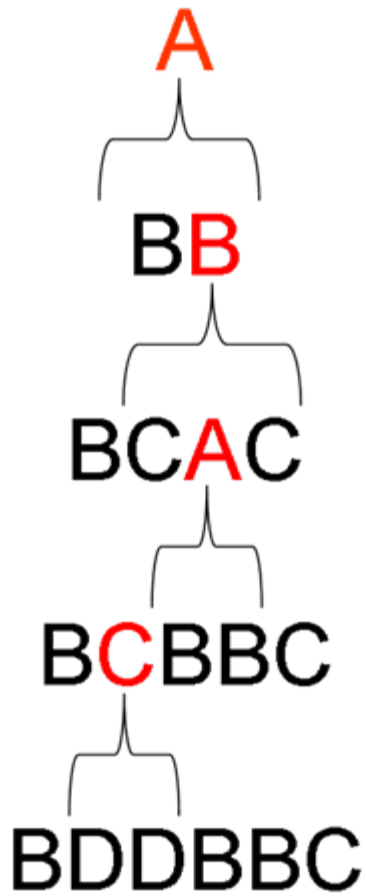


How a Production Grammar Works

$A \rightarrow BB$

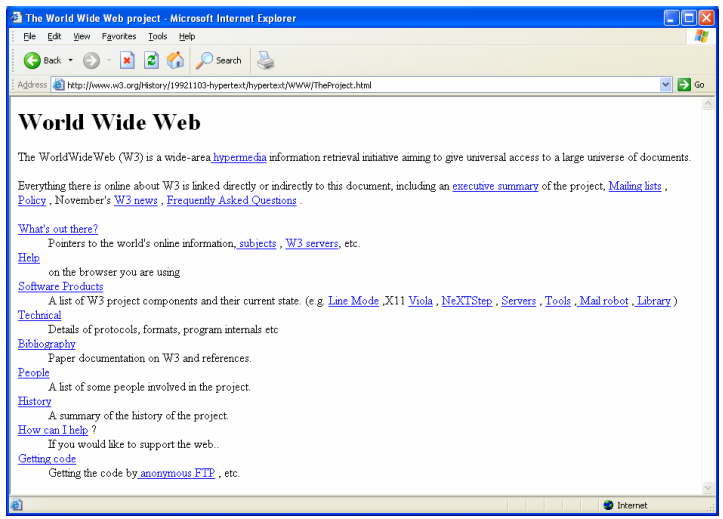
$B \rightarrow CAC$

$C \rightarrow DD$



Exercise 11

Modeling HTML



A mini-html grammar

{seed}:=<html>{body}</html>;

{body}:={body}<P>{body}|<H1>{body2}</H1>|<H4>{body2}</H4>;

{body2}:= {body3} | {body3} ;

{body3}:=<U>{body4}</U>|{body4};

{body4}:=<I> {body5} </I> | {body5};

{body5}:=the quick brown fox;

Grammar Example 1

Modeling Arithmetic Expressions

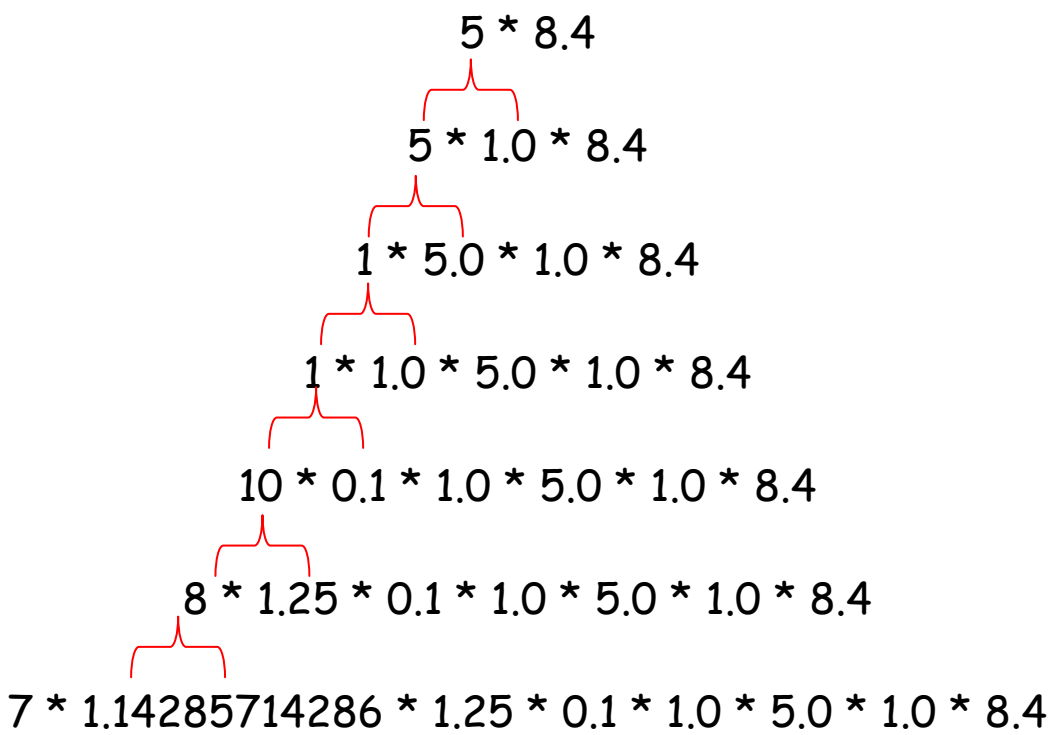
```
100 * 5 / (6 - 5.01 * 3.14159)
```

Inspired by Pete TerMaat's article "Adventures in Automated Testing"

Evaluate the following expressions

1. 10 * 1.0 * 0.1 * 6.0 * 1.16666666667 * 1.0 * 1.14285714286 * 5.25
2. 2 * 2.5 * 0.4 * 4.0 * 1.25 * 1.0 * 0.8 * 5.25
3. 4 * 0.5 * 3.5 * 1.0 * 0.285714285714 * 3.0 * 0.166666666667 * 42.0
4. 2 * 5.0 * 0.2 * 1.5 * 1.0 * 2.66666666667 * 1.125 * 4.66666666667
5. 5 * 1.6 * 1.0 * 0.5 * 1.75 * 0.428571428571 * 0.333333333333 * 42.0
6. 4 * 2.0 * 0.75 * 1.5 * 0.444444444444 * 0.75 * 2.66666666667 * 5.25
7. 6 * 1.5 * 0.777777777778 * 0.428571428571 * 1.33333333333 * 0.25 * 7.0 * 6.0
8. 2 * 1.5 * 0.333333333333 * 10.0 * 0.5 * 0.8 * 1.5 * 7.0
9. 5 * 0.6 * 1.66666666667 * 0.2 * 1.0 * 1.0 * 10.0 * 4.2

42



Evaluate the following expression

10 * 0.6 * 1.5 * 0.666666666667 * 1.0 * 1.0 * 0.833333333333 * 1.2 * 1.5 * 0.666666666667 * 1.33333333333 * 0.375 * 1.66666666667 * 1.0 * 2.0 * 1.0 * 0.9 * 0.777777777778 * 1.0 * 0.428571428571 * 3.0 * 0.666666666667 * 1.5 * 0.222222222222 * 0.5 * 5.0 * 2.0 * 0.4 * 2.0 * 0.5 * 0.75 * 1.33333333333 * 0.75 * 2.33333333333 * 1.42857142857 * 0.7 * 1.0 * 1.28571428571 * 1.0 * 0.333333333333 * 2.0 * 0.833333333333 * 0.2 * 6.0 * 1.33333333333 * 0.375 * 0.666666666667 * 1.0 * 2.0 * 2.25 * 0.222222222222 * 4.5 * 0.888888888889 * 0.125 * 4.0 * 2.0 * 0.75 * 1.0 * 1.5 * 0.333333333333 * 2.33333333333 * 0.428571428571 * 3.0 * 0.888888888889 * 0.625 * 0.2 * 8.0 * 0.375 * 0.333333333333 * 7.0 * 1.42857142857 * 0.1 * 3.0 * 3.33333333333 * 1.0 * 0.6 * 0.333333333333 * 4.0 * 0.5 * 2.5 * 0.5 * 1.4 * 1.14285714286 * 0.25 * 0.5 * 7.0 * 0.428571428571 * 2.0 * 1.0 * 0.5 * 0.666666666667 * 4.5 * 0.333333333333 * 2.33333333333 * 0.571428571429 * 1.75 * 0.428571428571 * 2.0 * 0.5 * 14.0

Grammar Example 2

Modeling C Code

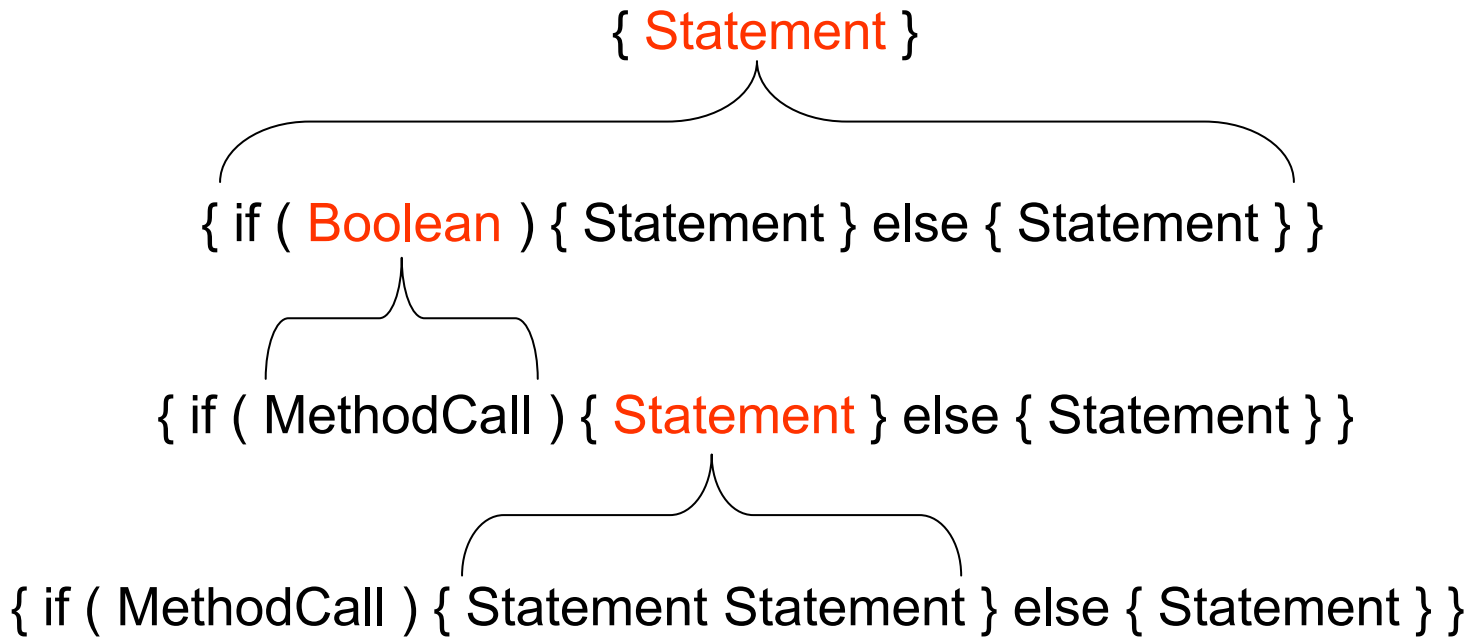
```
static void TestCase(){
while( Fun("L21") ){
L22: break; }}
else {
L3:{
L31:      goto L22;
L32:      goto L31; }}
```

Grammar Model for a Subset of C

Statement :: if (BooleanCondition) { Statement } else { Statement }
Statement :: while (BooleanCondition) { Statement }
Statement :: { Statement Statement }
Statement :: break;
Statement :: goto Label;
Statement :: Label : Statement
Statement :: ExpressionStatement;
ExpressionStatement :: MethodCall
BooleanExpression :: MethodCall
MethodCall :: Fun(Label)

Based on “A High-Level Modular Definition of the Semantics of C#”
- Boerger, Fruja, Gervasi, Stark

Deriving Programs from the Grammar



Tests Generated from the Grammar

suite123.cs

```
static void TestCase(){
L:
if ( Fun("L1") ){
L2:
if ( Fun("L21") ){
L22: goto L;
}
else{
L23: goto L;
}}
else{
L3:
{
L31: goto L32;
L32: goto L;
}}}
```

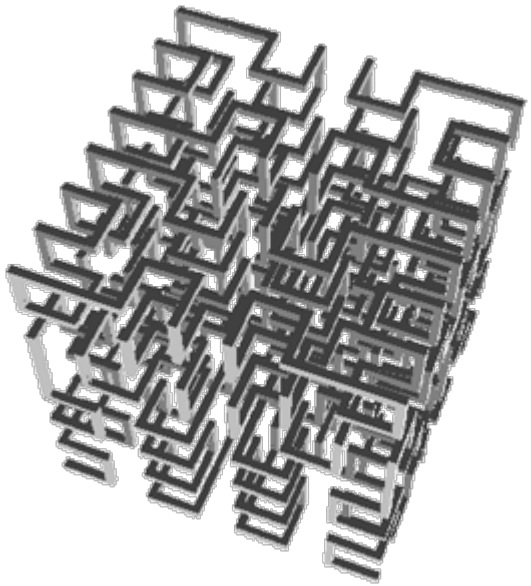
suite321.cs

```
static void TestCase(){
L:
if ( Fun("L1") ){
L2:
while( Fun("L21") ){
L22:
break;
}}
else{
L3:
{
L31: goto L;
L32: goto L31;
}}}
```

"Experiments on Semantics Based Testing of a Compiler" - Esin, Novikov, Yavorskiy

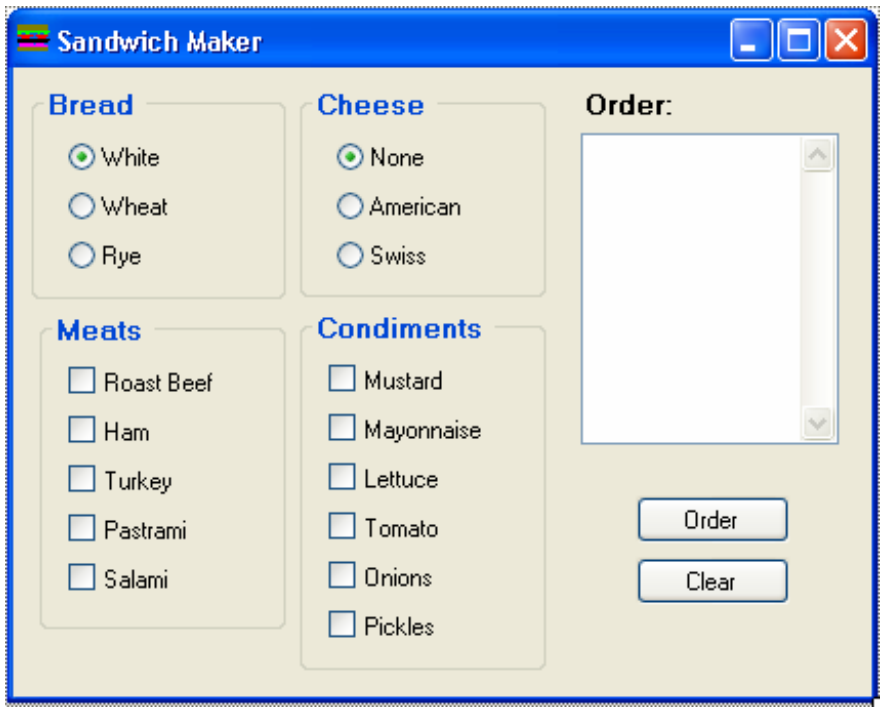
Exercise 12

Modeling Combinations



Exercise 13

Selecting Sandwiches



Inspired by an example in Lou Tylee's book "Visual C# Express for Kids"

Setting Up the Combinations

Bread	**Cheese**	**H**	**MU**	**MA**	**L**	**T**
white	none	ham	mustard	mayo	lettuce	tomato
wheat	american	no_ham	no_must	no_mayo	no_lett	no_tom
rye	swiss					

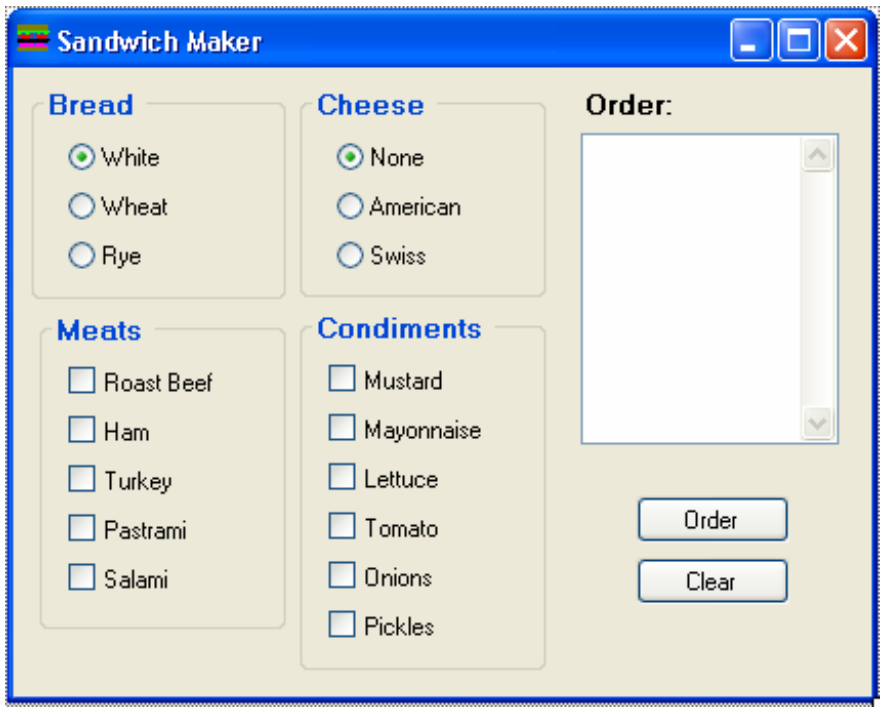
Using James Bach's AllPairs program from www.satisfice.com

Generating the Pairs

TEST CASES

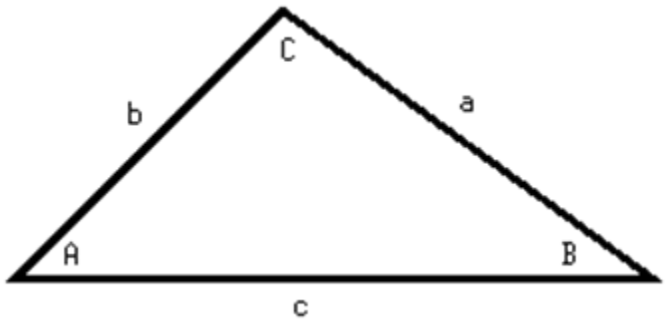
case	Bread	Cheese	H	M	MA	L	T
1	white	none	ham	mustard	mayo	lettuce	tomato
2	white	american	no_ham	no_must	no_mayo	no_lett	no_tom
3	wheat	none	no_ham	mustard	no_mayo	lettuce	no_tom
4	wheat	american	ham	no_must	mayo	no_lett	tomato
5	rye	swiss	ham	mustard	no_mayo	no_lett	tomato
6	rye	swiss	no_ham	no_must	mayo	lettuce	no_tom
7	white	none	ham	no_must	no_mayo	no_lett	no_tom
8	white	american	no_ham	mustard	mayo	lettuce	tomato
9	wheat	swiss	ham	mustard	mayo	no_lett	no_tom
10	rye	none	no_ham	no_must	no_mayo	lettuce	tomato
11	rye	american	ham	mustard	no_mayo	lettuce	no_tom
12	white	swiss	no_ham	no_must	no_mayo	no_lett	tomato

But, did we miss any behavior of interest?



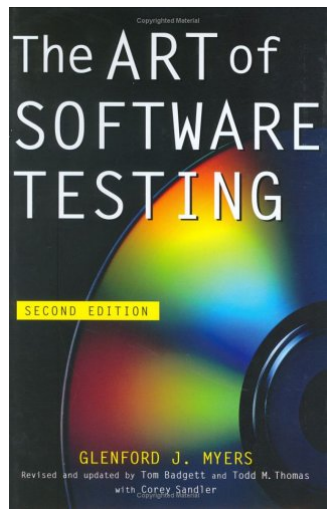
Exercise 14

Modeling the Triangle



Inspired by B J Rollison's conference paper "Dissecting the Triangle Problem"

triangle test cases for the triangle program, that in no way guarantees the correct detection of all equilateral triangles. The program could contain a special check for values 3842,3842,3842 and denote such a triangle as a scalene triangle. Since the program is a black box, the only way to be sure of detecting the presence of such a statement is by trying every input condition.

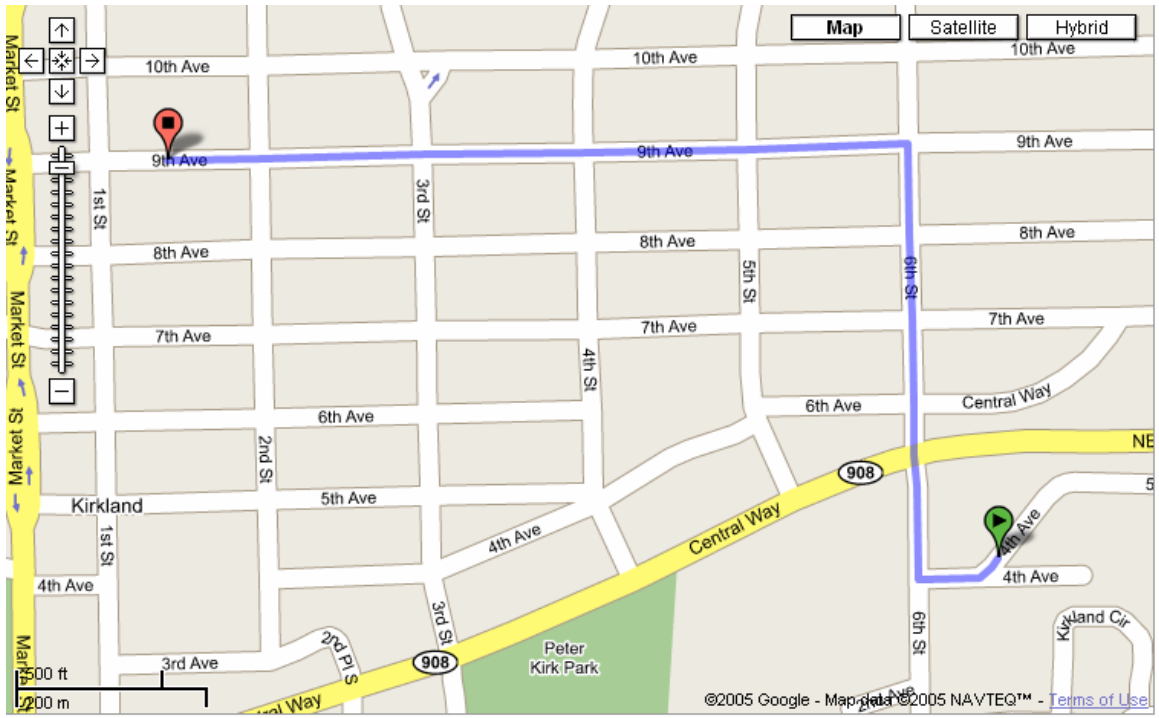


Exhaustion?

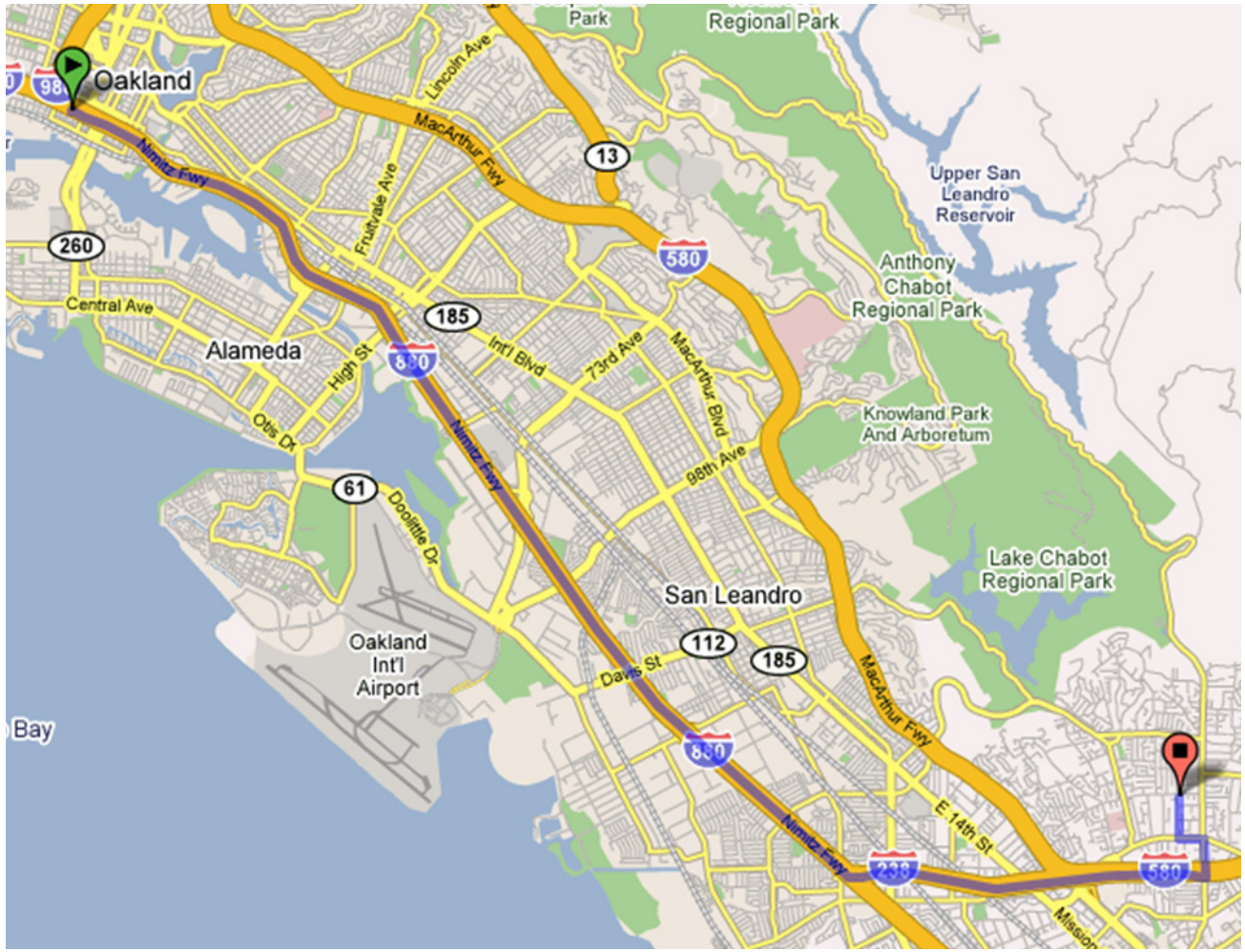
```
for ( a=0; a<=32767; a++)
   for ( b=0; b<=32767; b++)
      for ( c=0; c<=32767; c++)
         evaluate( a, b, c);
```

Exercise 15

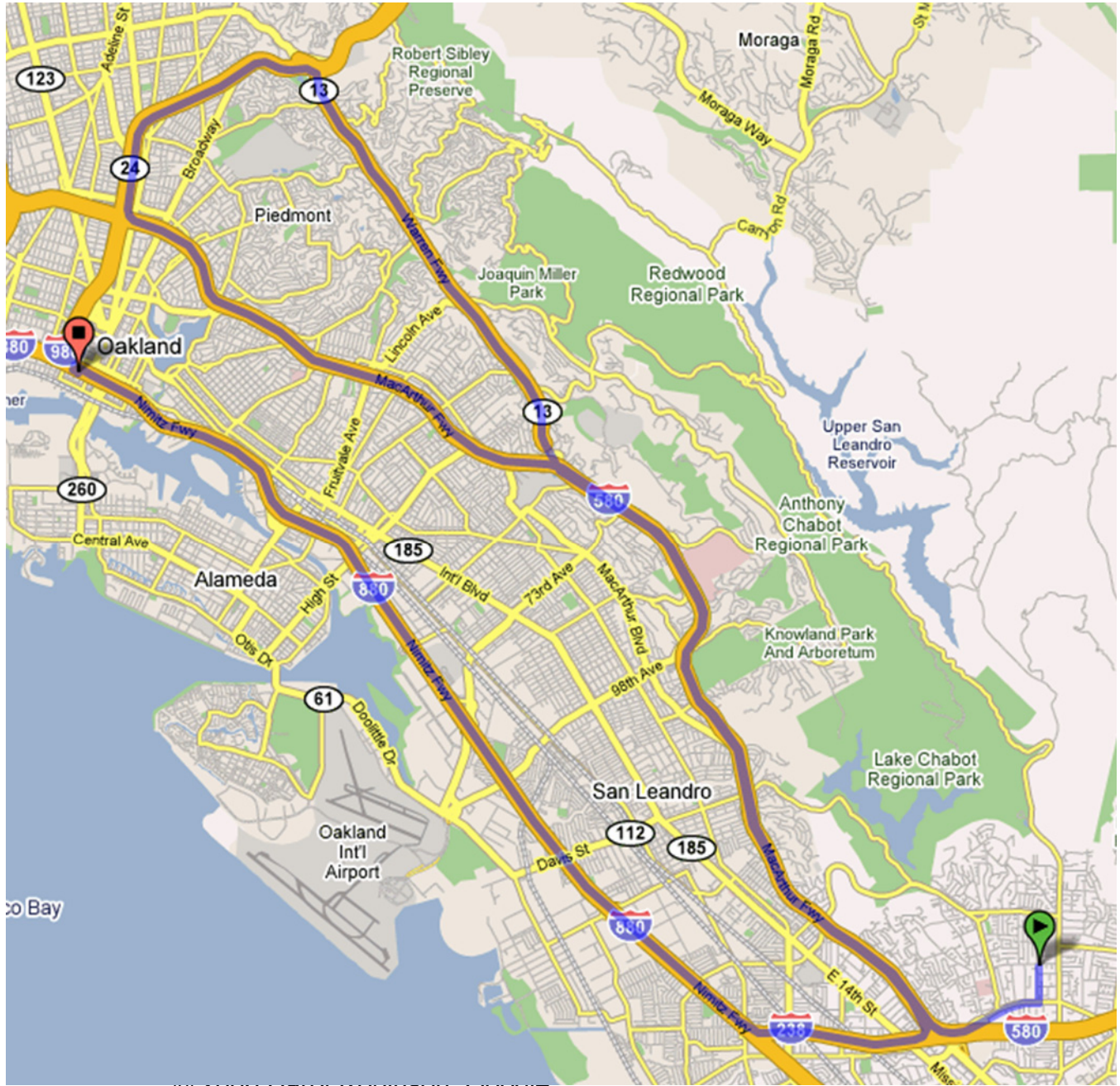
Modeling Google Maps

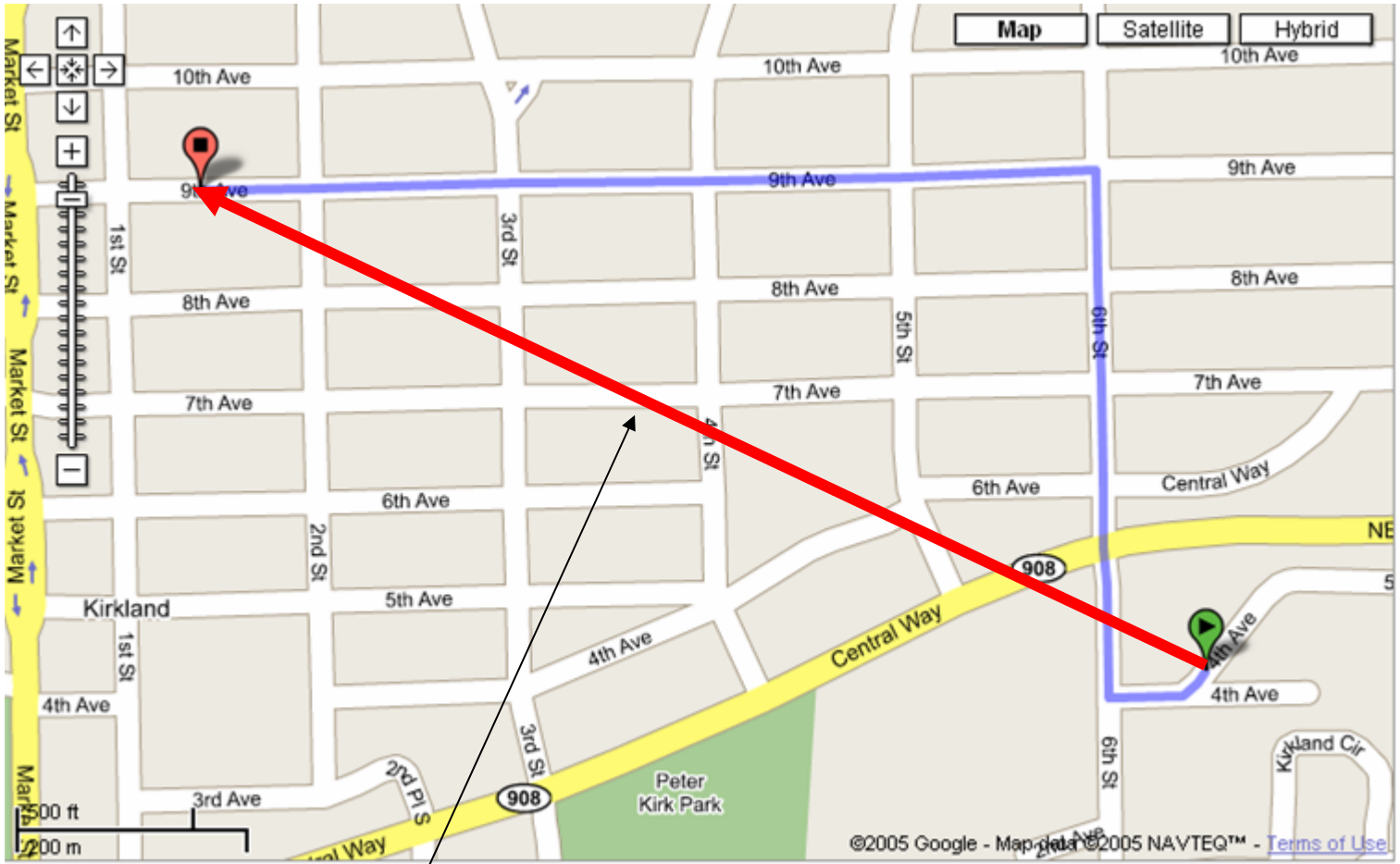


Start address:	Clay St Oakland, CA 94607
End address:	CA 94546
Distance:	15.6 mi (about 21 mins)



Start address:	CA 94546
End address:	Clay St Oakland, CA 94607
Distance:	45.8 mi (about 54 mins)





0.60 miles

Start address:	720 4th Ave Kirkland, WA 98033
End address:	Kirkland, WA
Distance:	0.9 mi (about 1 min)

Trip #115

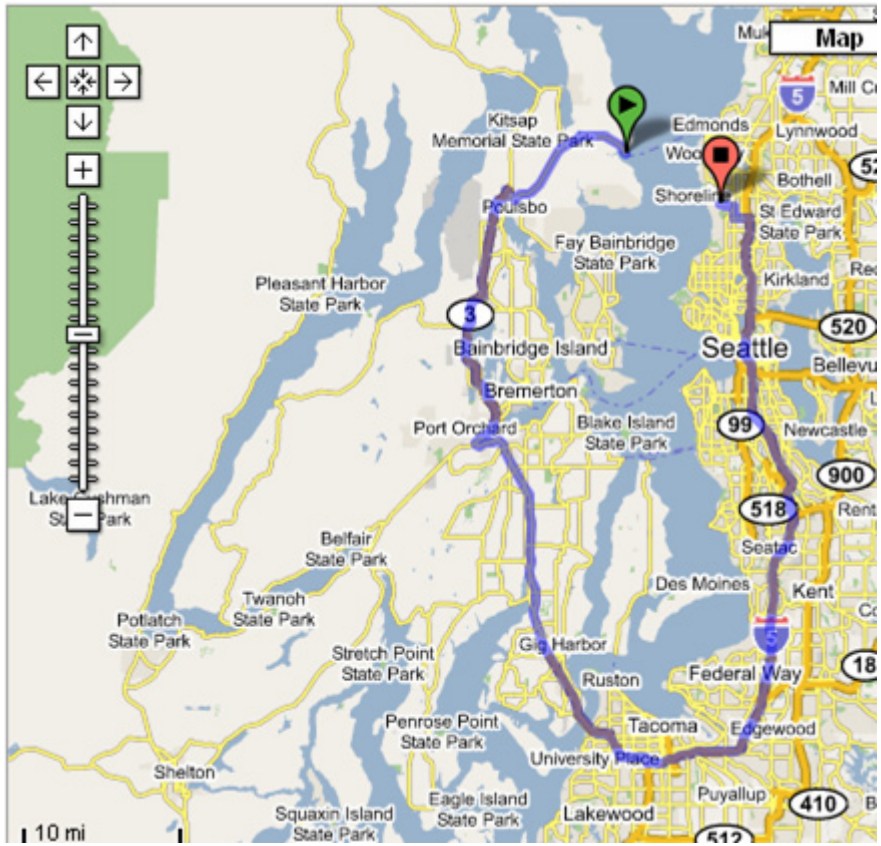
Trip start: 47.7945, -122.493

Trip end: 47.74753 -122.36585

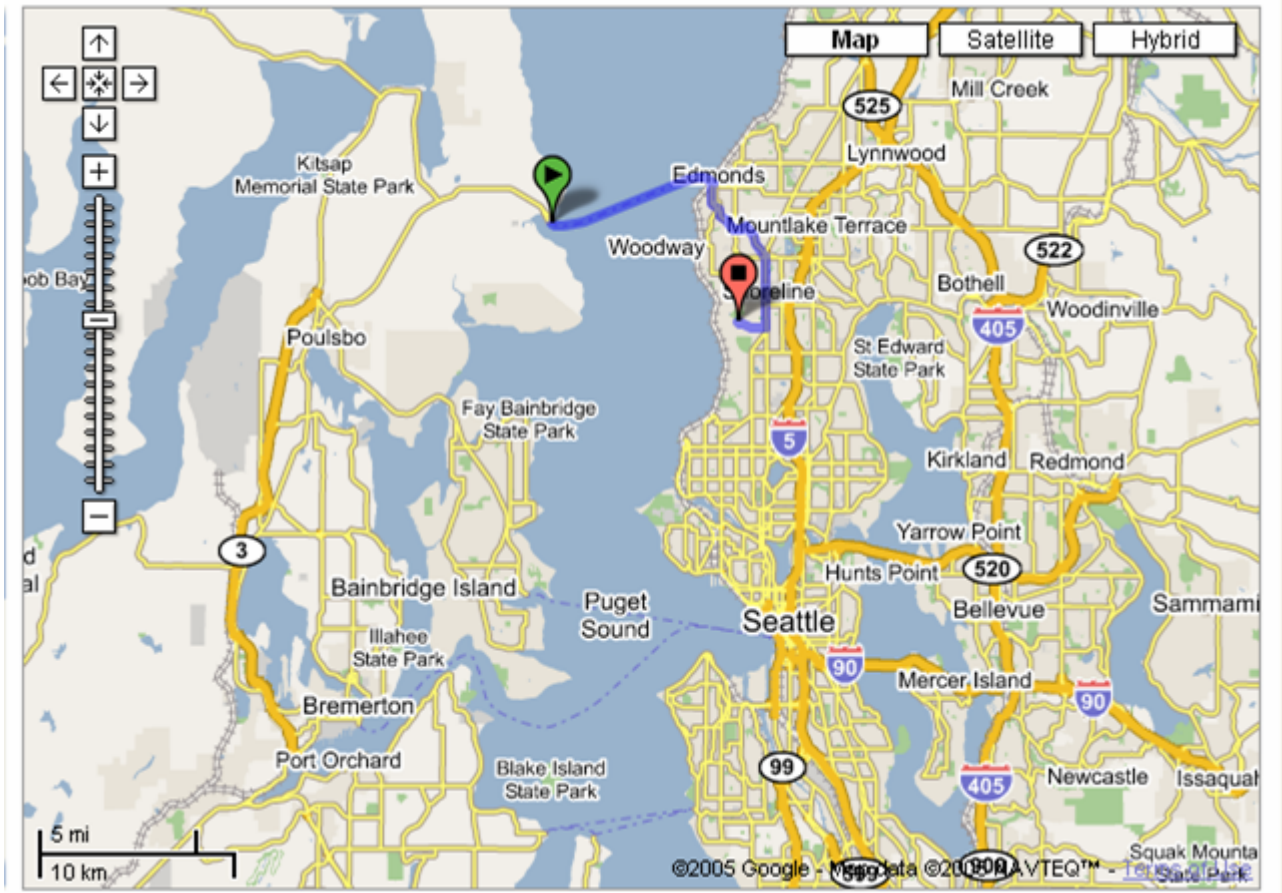
Google route distance is 102.0 miles

Straightline distance is 6.7 miles

Ratio is 15.2 ---------- SUSPICIOUS...



Start address:	**47.794500, -122.493000** +47° 47' 40.20", -122° 29' 34.80"
End address:	**47.747530, -122.365850** +47° 44' 51.11", -122° 21' 57.06"
Distance:	102 mi (about 2 hours 25 mins)



Start address:	**47.794500, -122.492000** +47° 47' 40.20", -122° 29' 31.20"
End address:	**47.747530, -122.365850** +47° 44' 51.11", -122° 21' 57.06"
Distance:	12.3 mi (about 2 hours 20 mins)

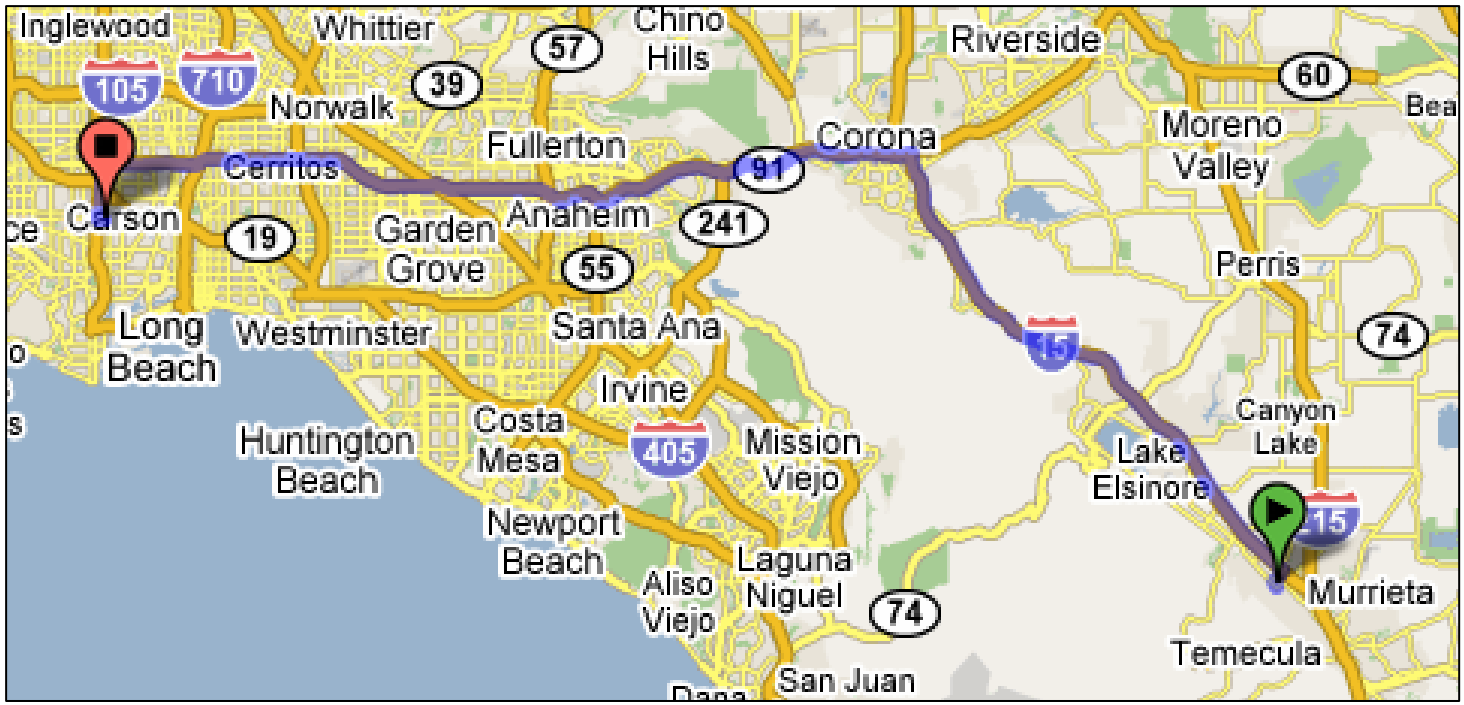
Start address:	**47.794500, -122.493000** +47° 47' 40.20", -122° 29' 34.80"
End address:	**47.747530, -122.365850** +47° 44' 51.11", -122° 21' 57.06"
Distance:	102 mi (about 2 hours 25 mins)

Start address:	**47.794500, -122.492000** +47° 47' 40.20", -122° 29' 31.20"
End address:	**47.747530, -122.365850** +47° 44' 51.11", -122° 21' 57.06"
Distance:	12.3 mi (about 2 hours 20 mins)

Checking Driving Directions

Maps

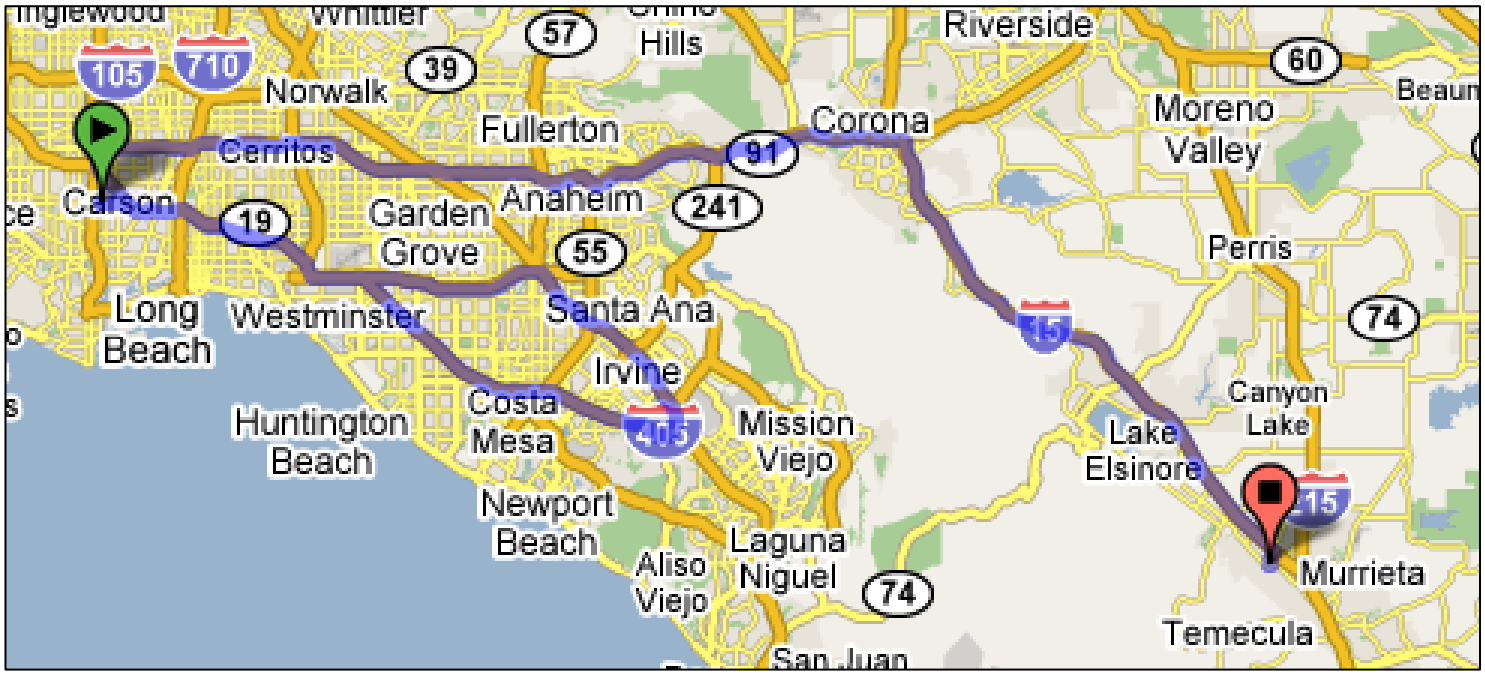
Start address:	CA 92564
End address:	CA 90747
Distance:	79.8 mi (about 1 hour 21 mins)



Checking Driving Directions

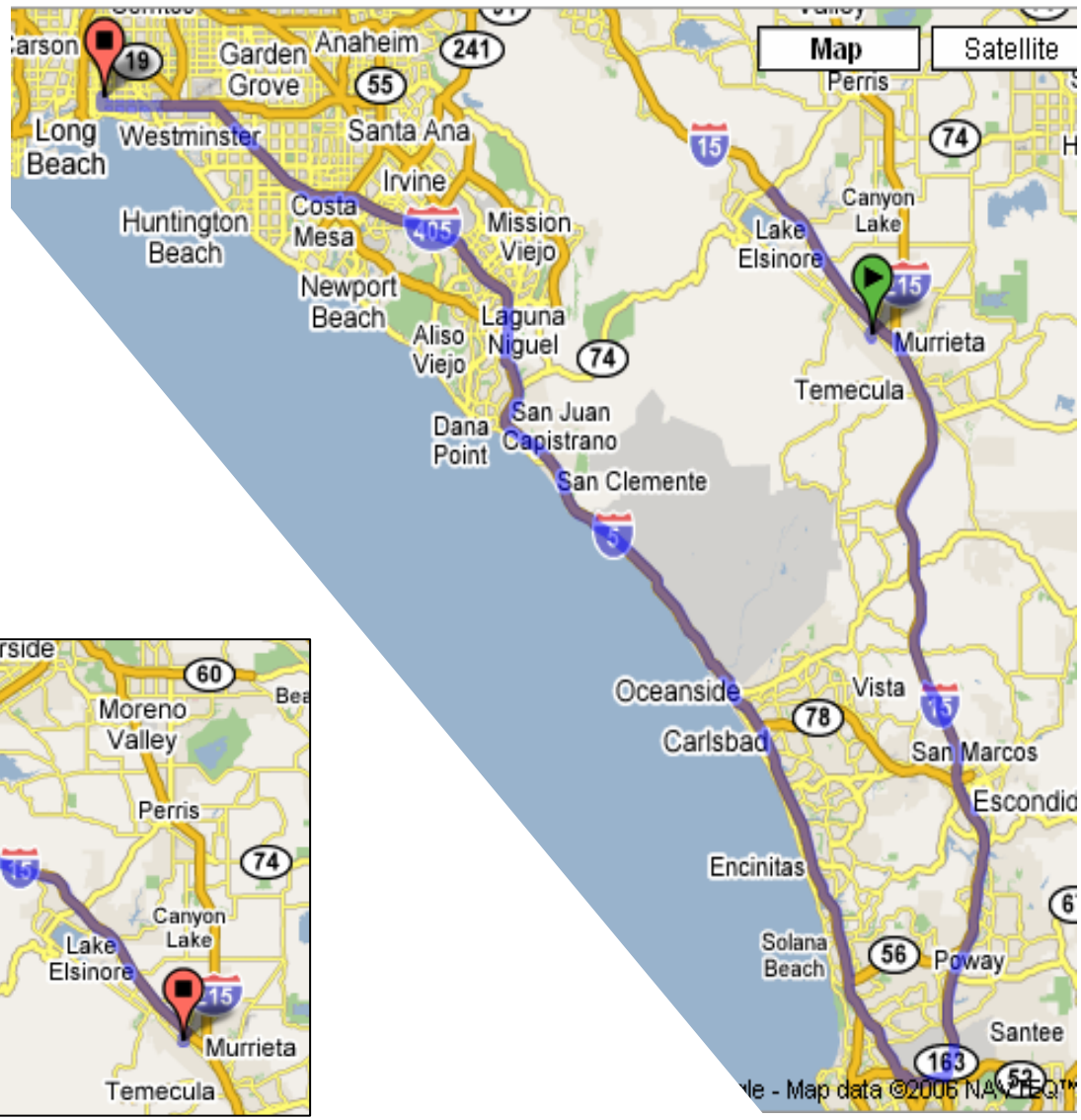
Maps

Start address:	CA 90747
End address:	CA 92564
Distance:	149 mi (about 2 hours 29 mins)

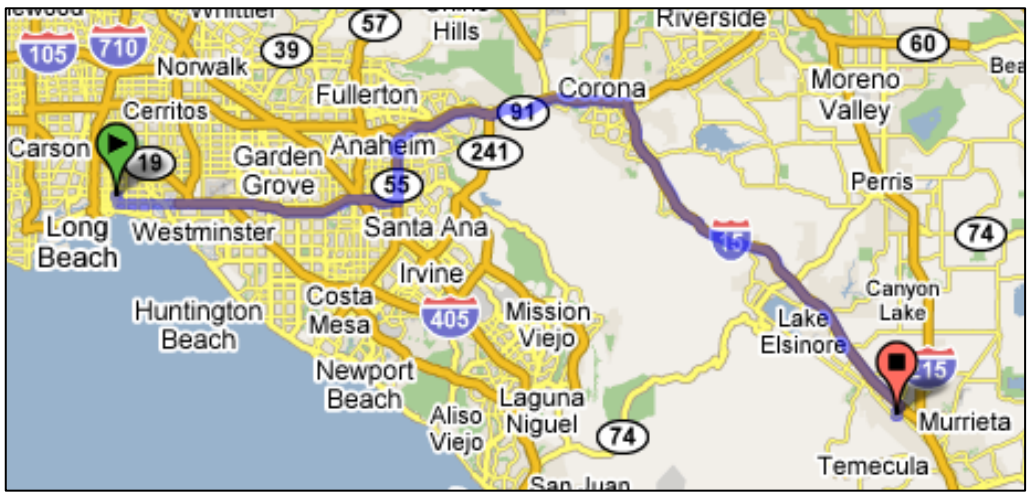


The Long Road Home

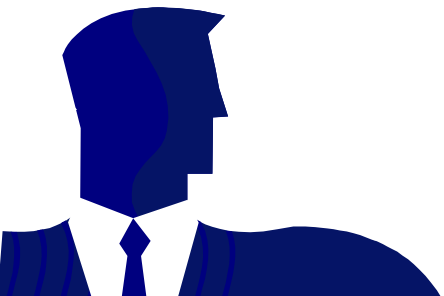
Start address:	CA 92564
End address:	Long Beach, CA 90813
Distance:	180 mi (about 3 hours 3 mins)



Start address:	Long Beach, CA 90813
End address:	CA 92564
Distance:	75.1 mi (about 1 hour 21 mins)

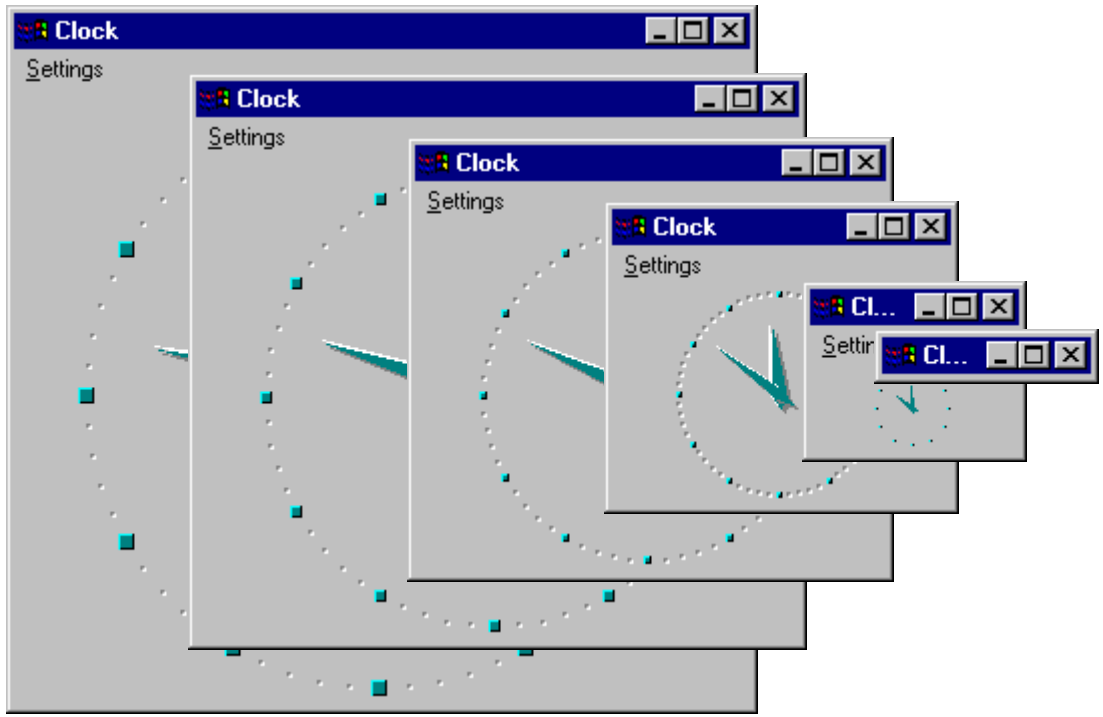


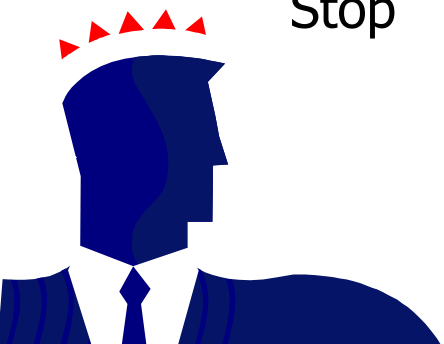
What Kinds of Bugs do Models Find?



The Incredible Shrinking Clock

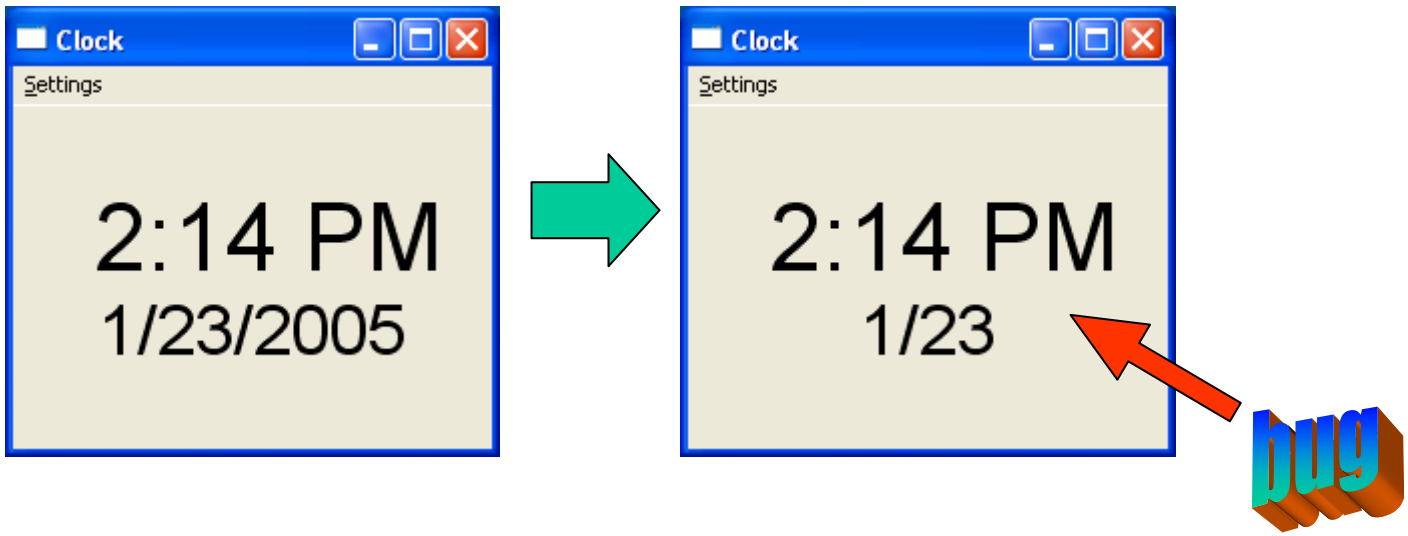
Start
Maximize
Stop
Start
Minimize
Stop
Start
Restore
Stop

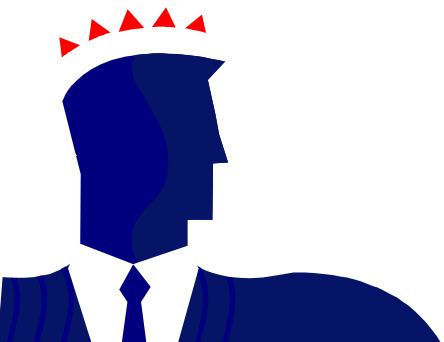




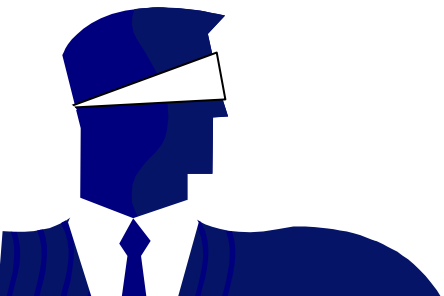
That Was the Year that Wasn't

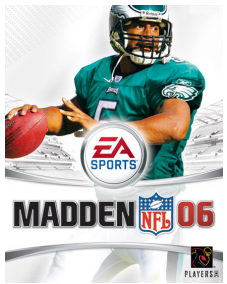
Start
Minimize
Stop
Start
Restore
Date

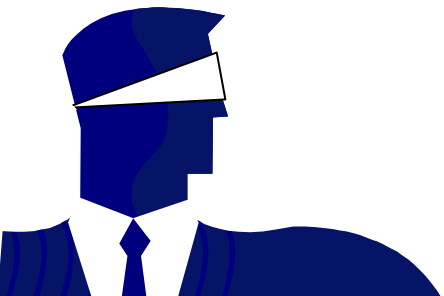


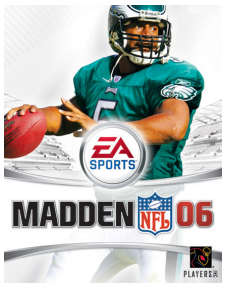


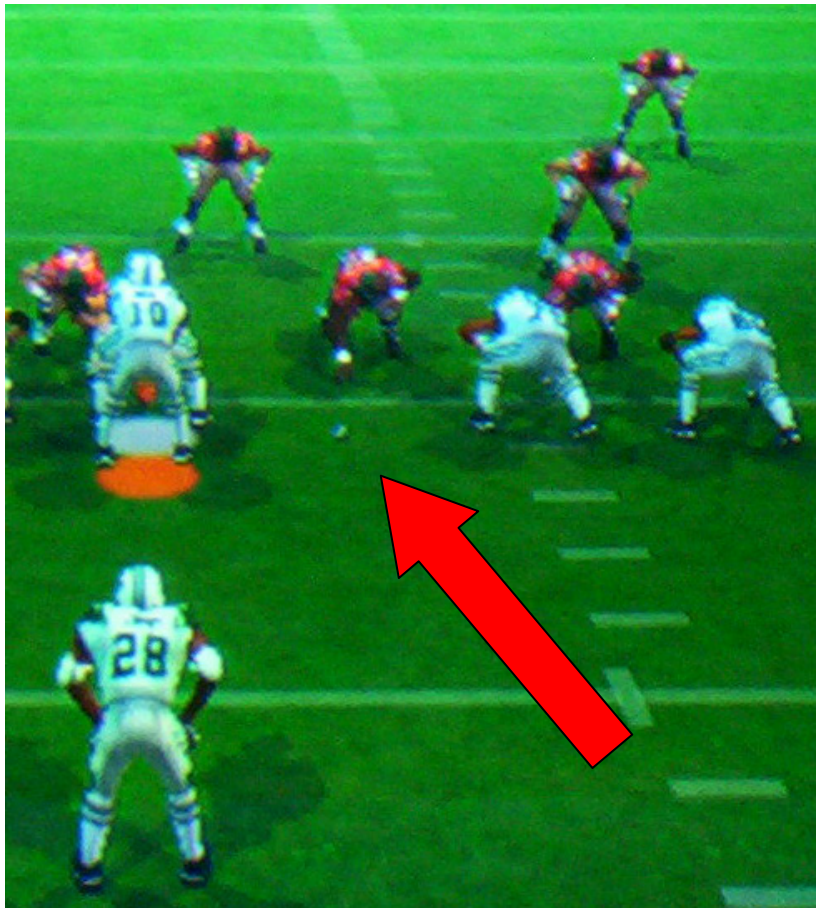
What Kinds of Bugs do Models NOT Find?

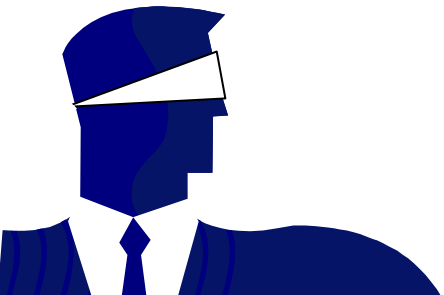


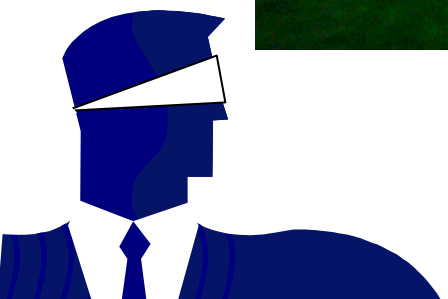


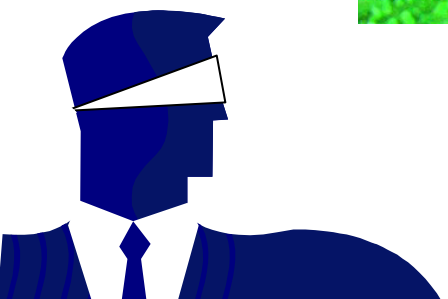












Where is Model-Based Testing Heading?

1. Security Testing
2. Shortening Bug Repro Scenarios
3. Meaningful Regression Testing

4. Machine Learning?

Security Testing

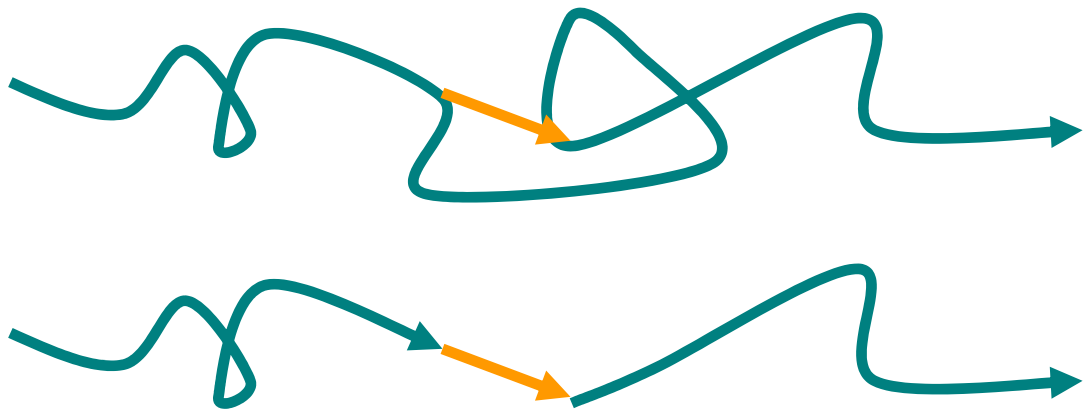
"The testing method developed in the PROTOS project is uniquely practical .... Tests are conducted by bombing software with illegally formatted or unexpected input."

Tekes - National Technology Agency of Finland

CERT Advisories

- CA-2001-18 Multiple Vulnerabilities in LDAP Implementations
- CA-2002-03 Multiple Vulnerabilities in SNMP Implementations
- CA-2003-06 Multiple Vulnerabilities in SIP Implementations
- CA-2003-11 Multiple Vulnerabilities in Lotus Notes and Domino

Shortening Repro Scenarios



The Motivation

"The fewer steps that it takes to reproduce a bug,
the fewer places the programmer has to look (usually).

If you make it easier to find the cause and test the change,
you reduce the effort required to fix the problem.
Easy bugs gets fixed even if they are minor."

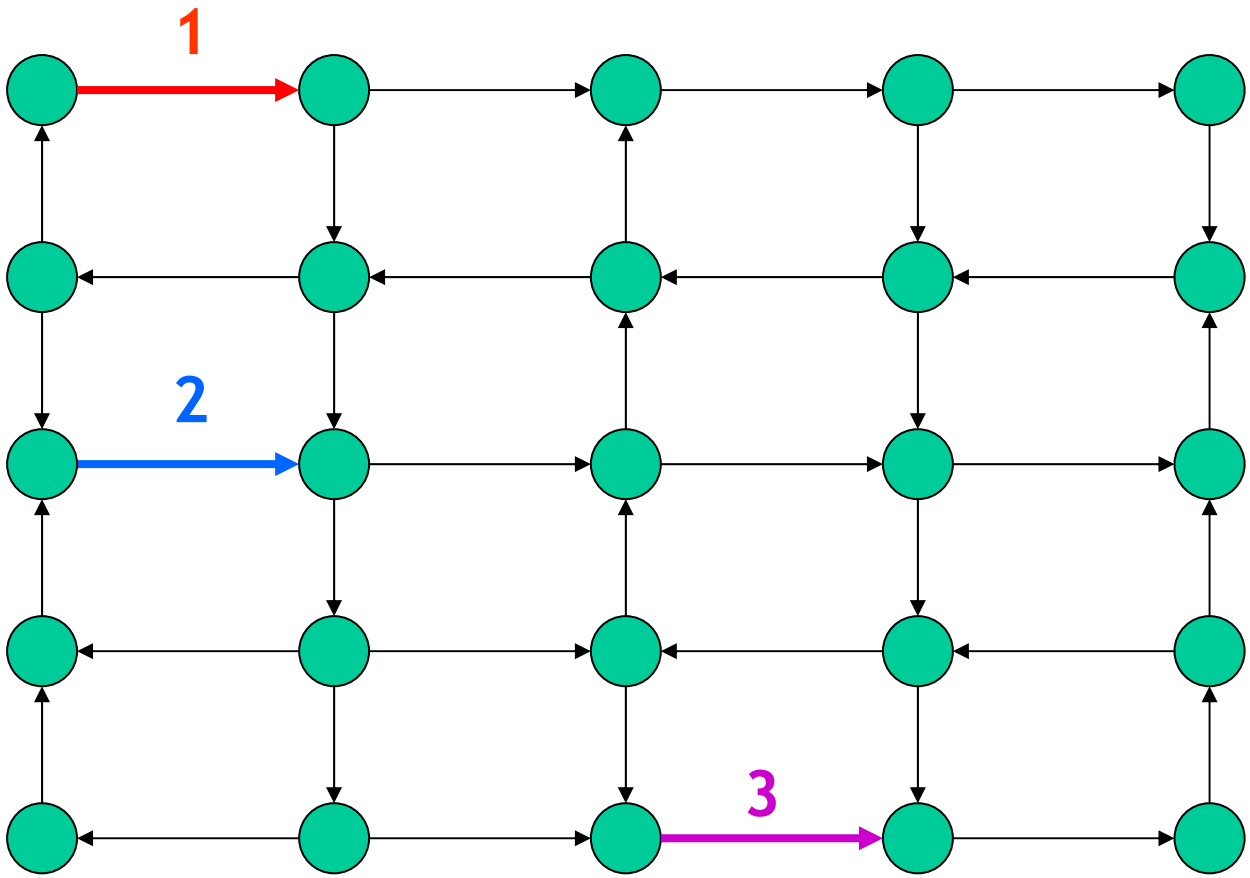
- from Testing Computer Software

The Beeline Approach

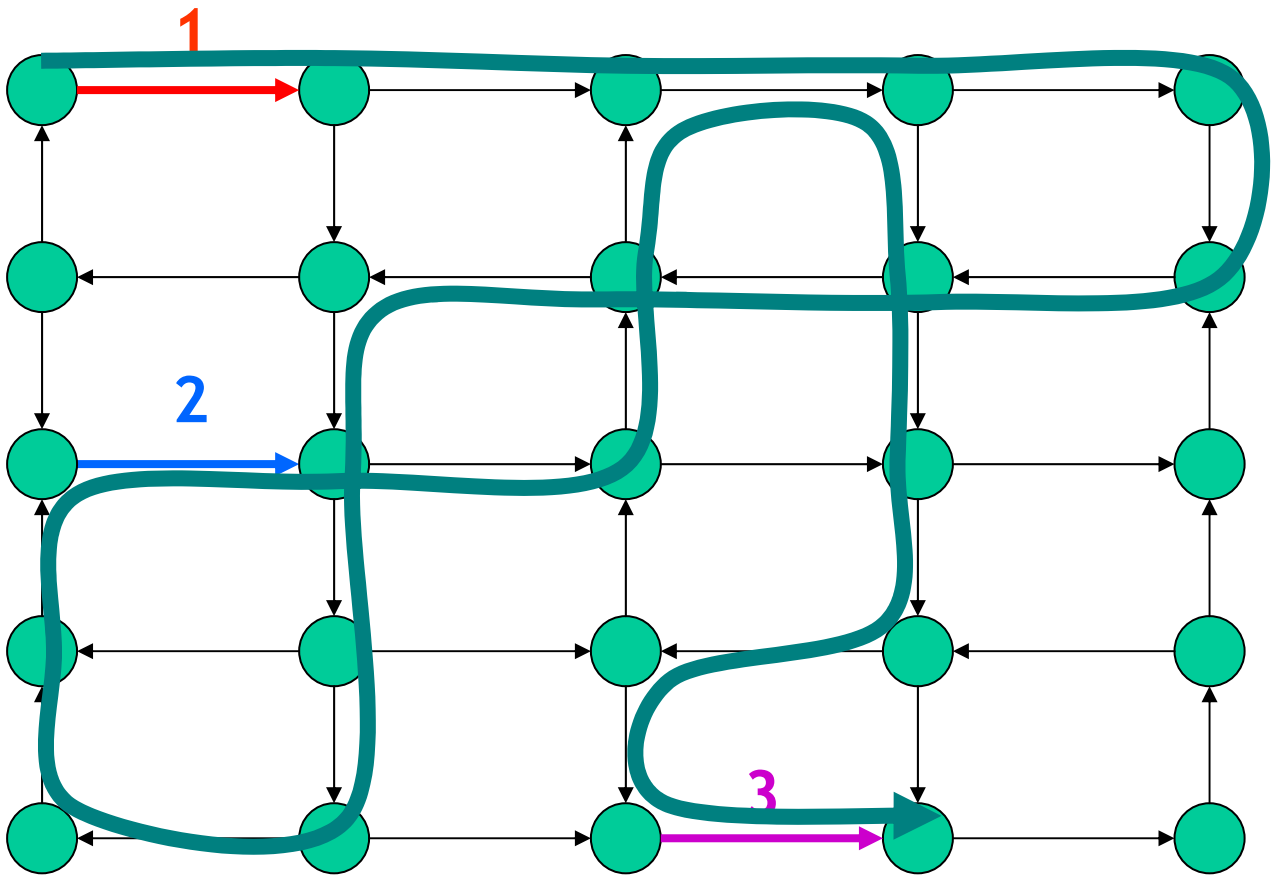
A repro path is simply another traversal through the state model, so …

1. Choose any 2 nodes in the repro path
2. Find the shortest path between them
3. Execute the spliced 'shortcut' path
4. Evaluate the results and repeat

The repro path reduction problem

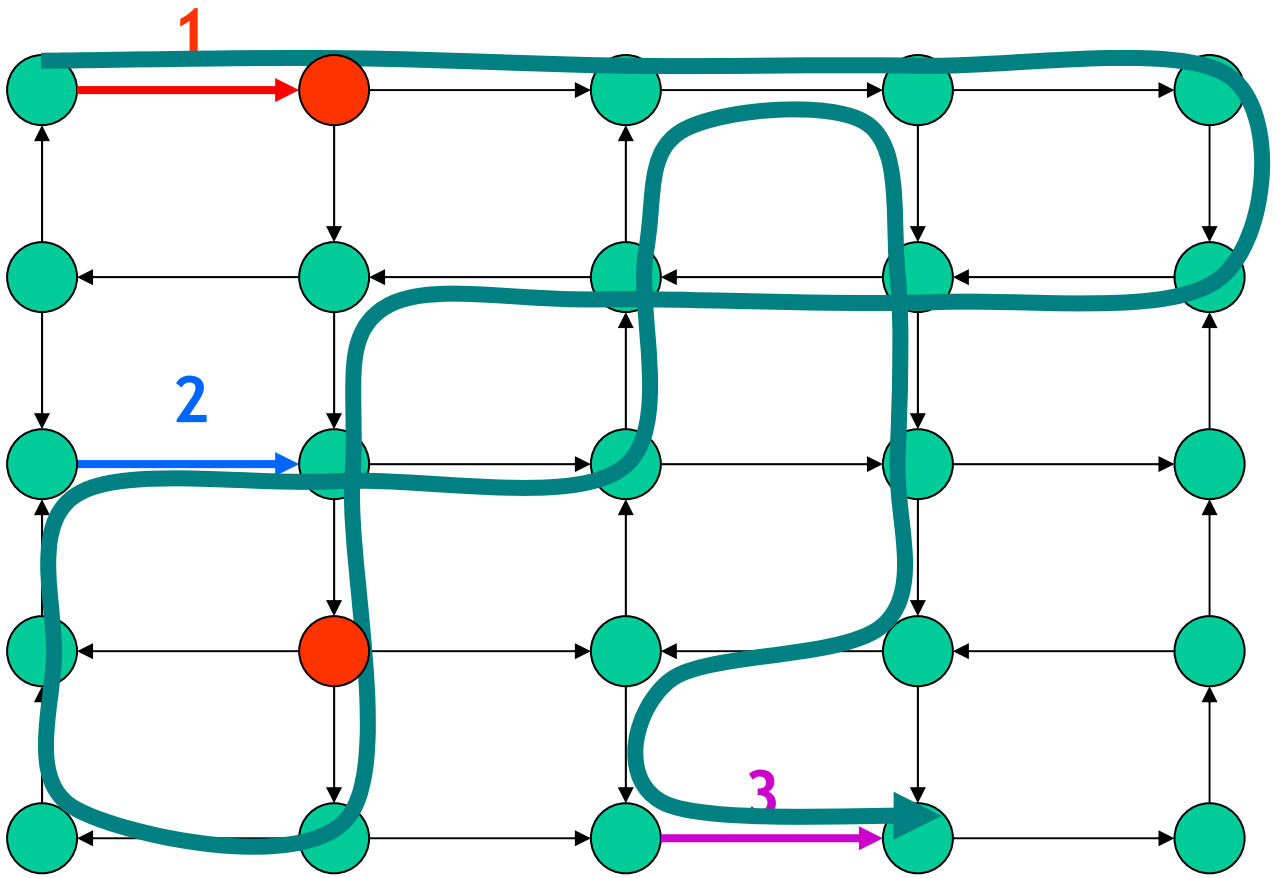


Random walk finds a bug

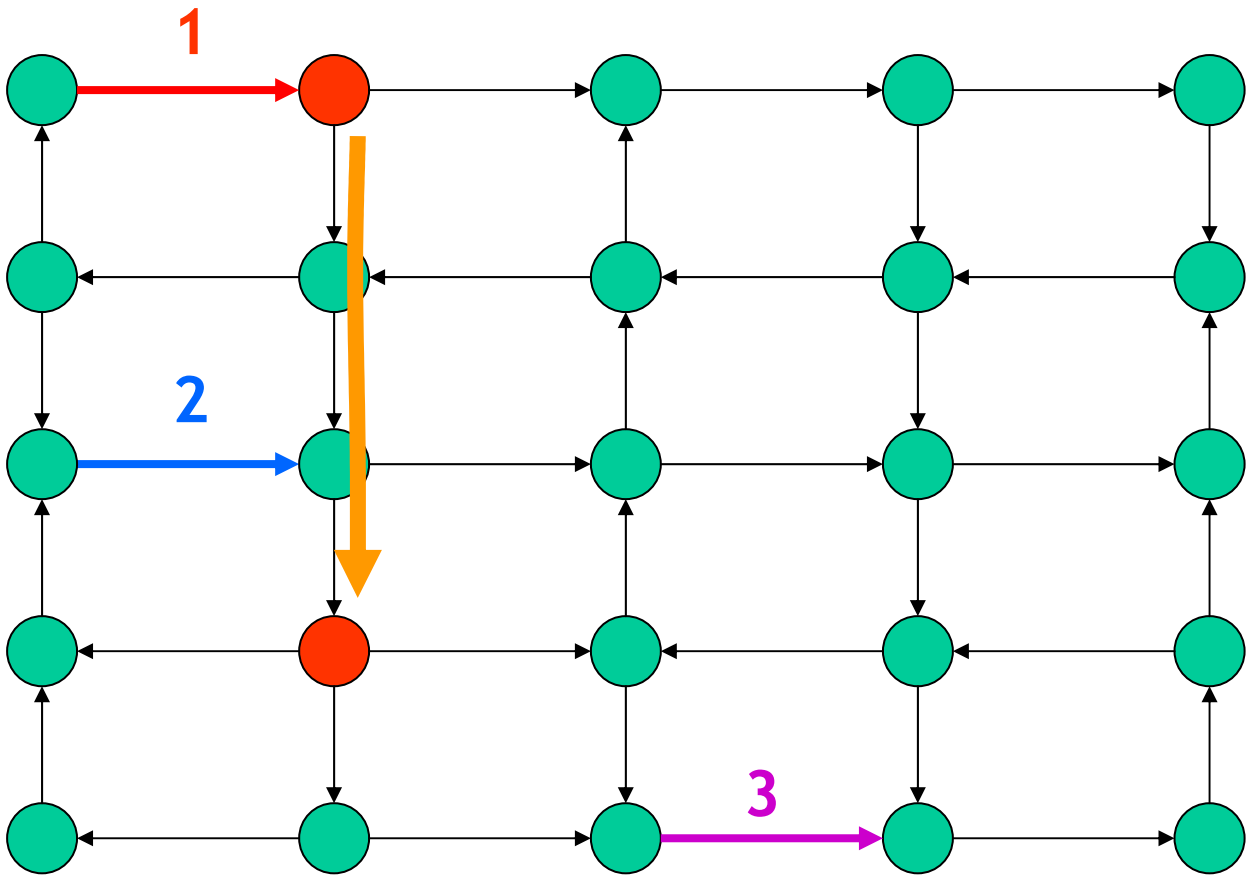


… but the repro path is inconveniently long

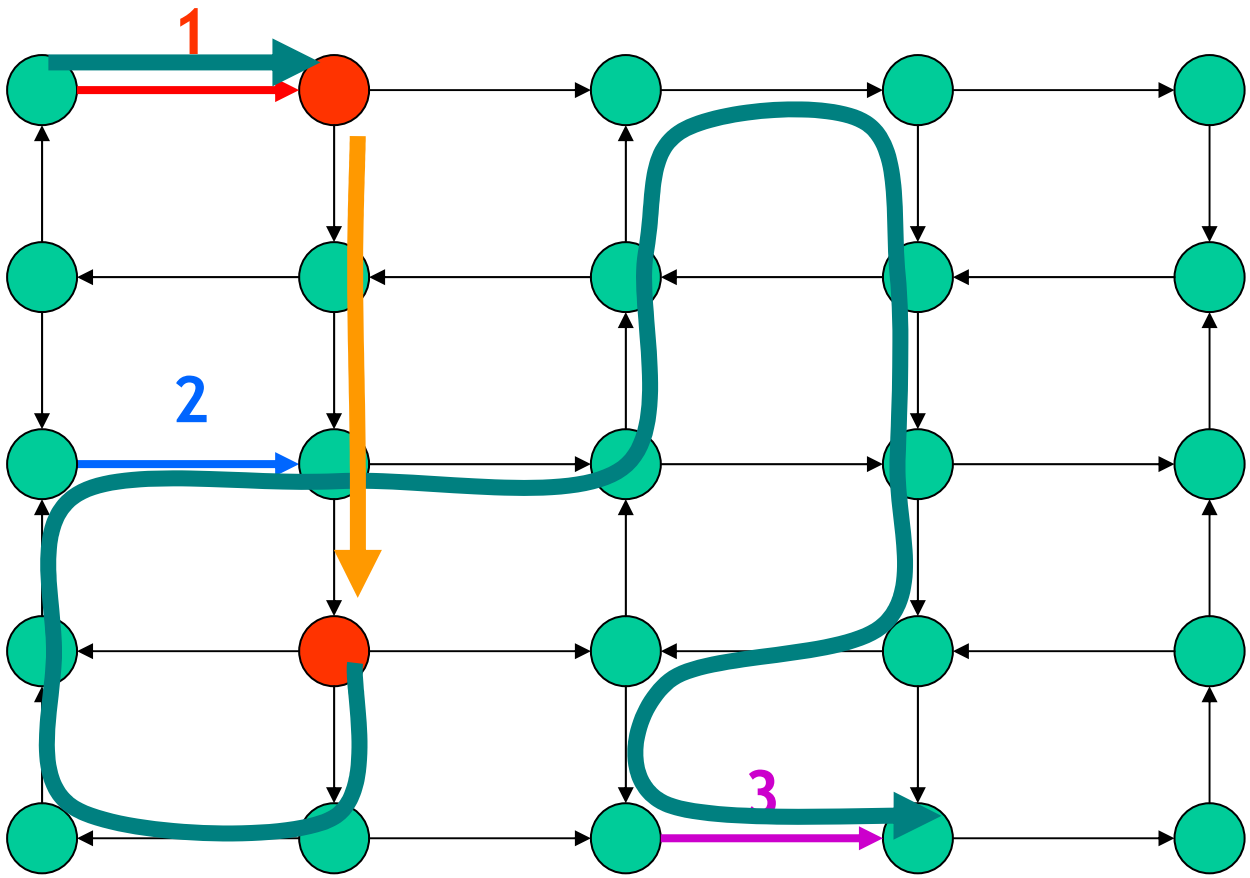
1. Choose any 2 nodes in the path



2. Find shortest path between them

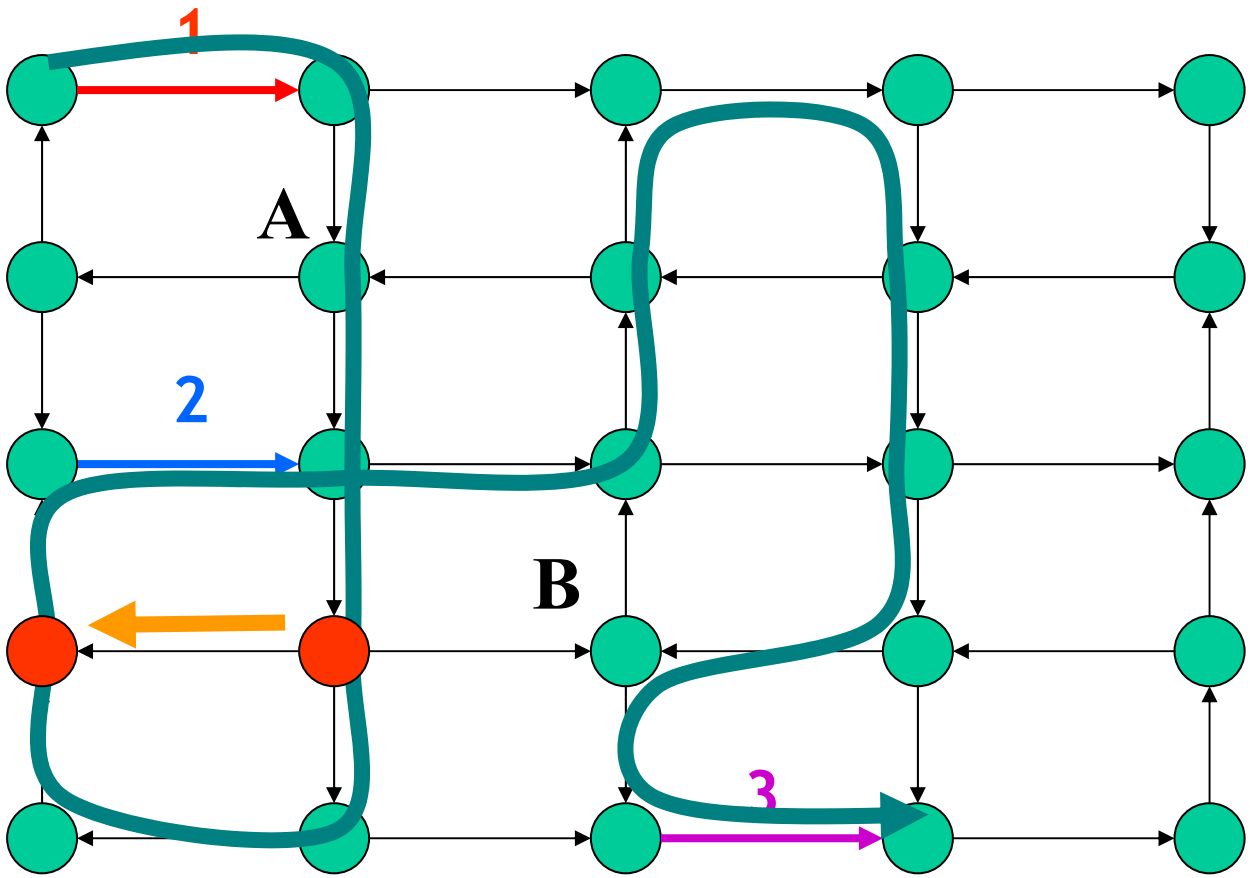


3. Execute the spliced shortcut path

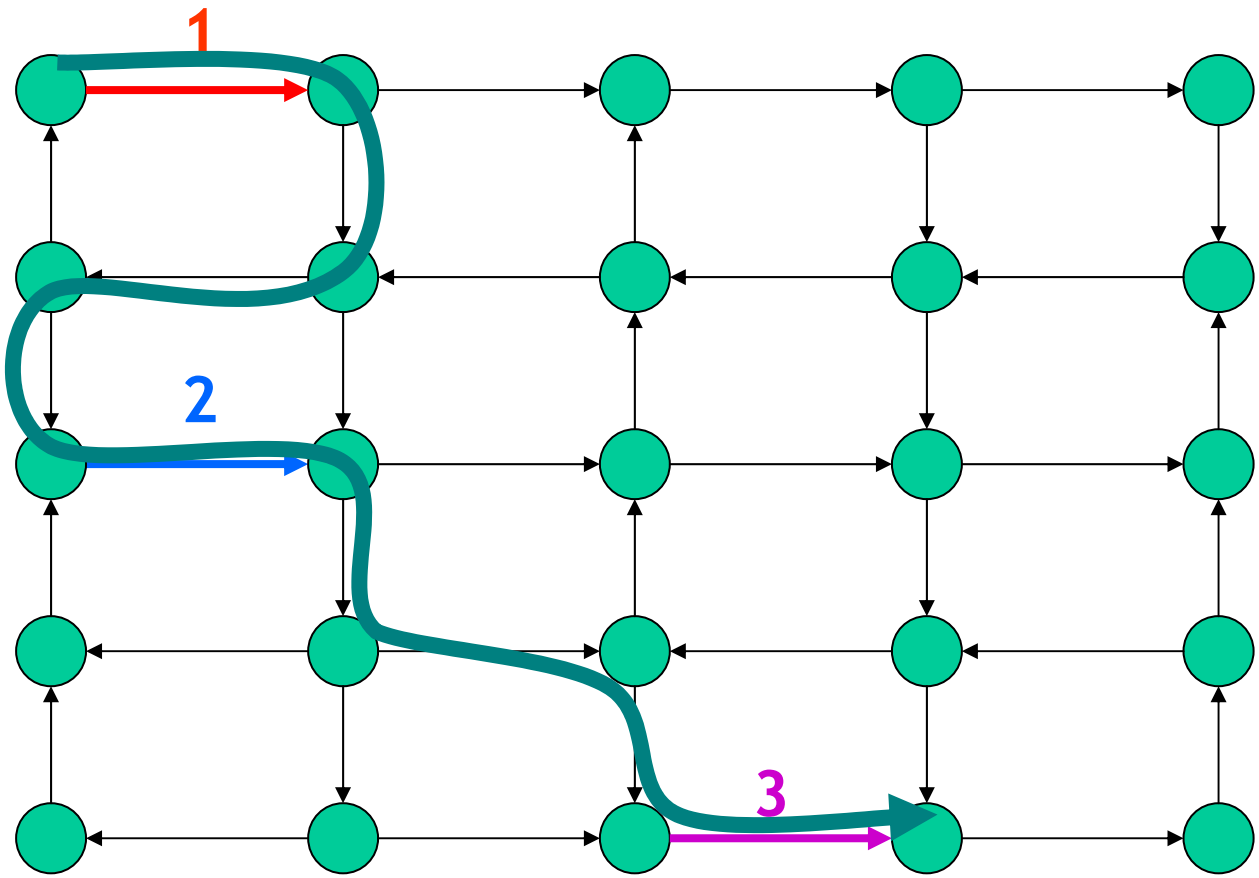


The bug repro'ed - this is the new shortest path

Continue trimming …



… until you stop.

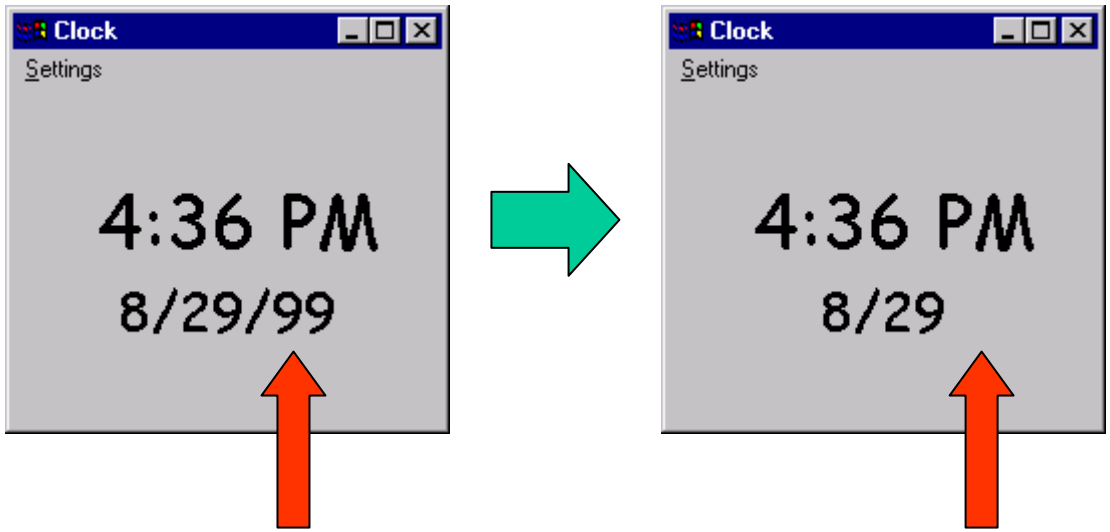


Why Use a Model for Reducing?

- The model can detect (and therefore reduce) both crashing AND non-crashing bugs.
- Finding a shortcut is simple in a model, so the reduction is more efficient.
- Finding bugs is good. Getting them fixed is better.

That Was The Year That Wasn't

Start
Minimize
Stop
Start
Restore
Date



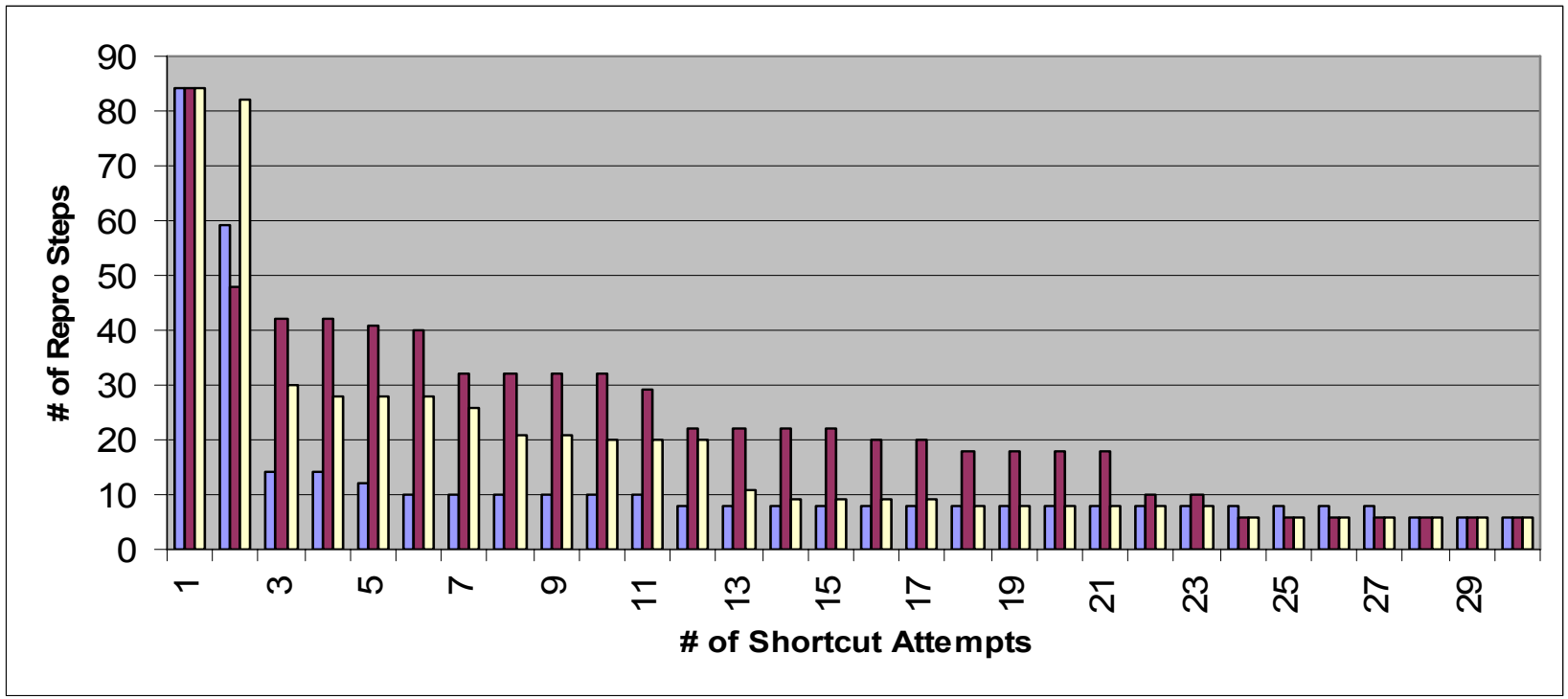
An 84-step repro sequence

invoke about ok_about no_title doubleclick seconds restore seconds doubleclick doubleclick date about ok_about restore gmt maximize doubleclick doubleclick date seconds date close_clock invoke close_clock invoke close_clock invoke seconds date restore about ok_about no_title doubleclick digital doubleclick doubleclick no_title doubleclick no_title doubleclick seconds restore restore doubleclick doubleclick gmt analog maximize date digital minimize restore **minimize close_clock invoke restore** digital date minimize close_clock invoke maximize gmt digital restore doubleclick doubleclick about ok_about maximize digital digital digital seconds analog about ok_about about ok_about minimize close_clock invoke restore **date**

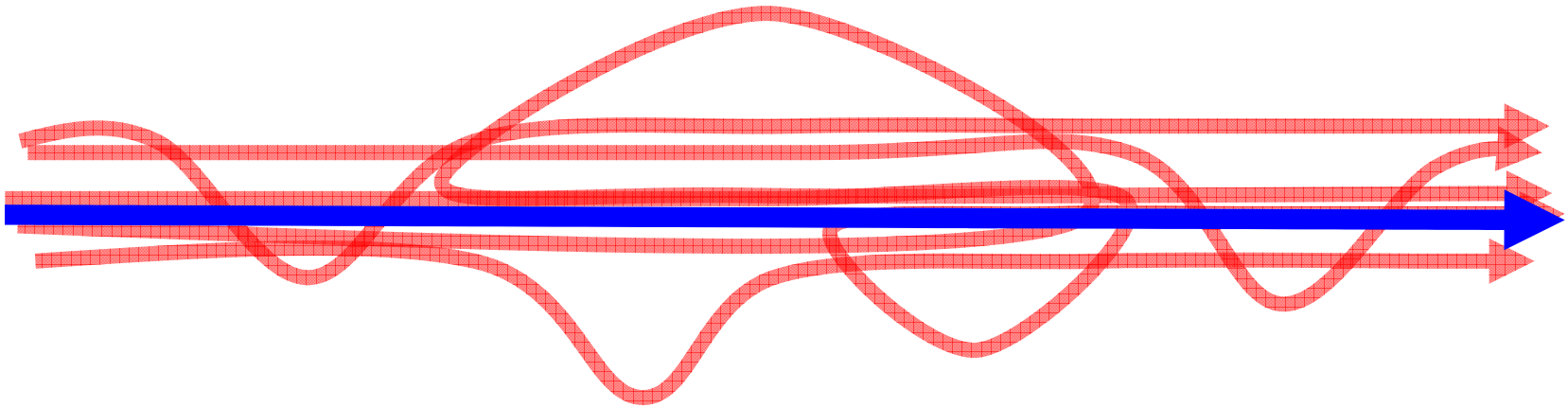
Reducing the Sequence:

- Initial path length: 84 steps
- Shortcut attempt 2 : repro sequence: 83 steps
- Shortcut attempt 3 : repro sequence: 64 steps
- Shortcut attempt 4 : repro sequence: 37 steps
- Shortcut attempt 5 : repro sequence: 11 steps
- Shortcut attempt 7 : repro sequence: 9 steps
- Shortcut attempt 20 : repro sequence: 8 steps
- Shortcut attempt 29 : repro sequence: 6 steps

Repro Steps Over Time



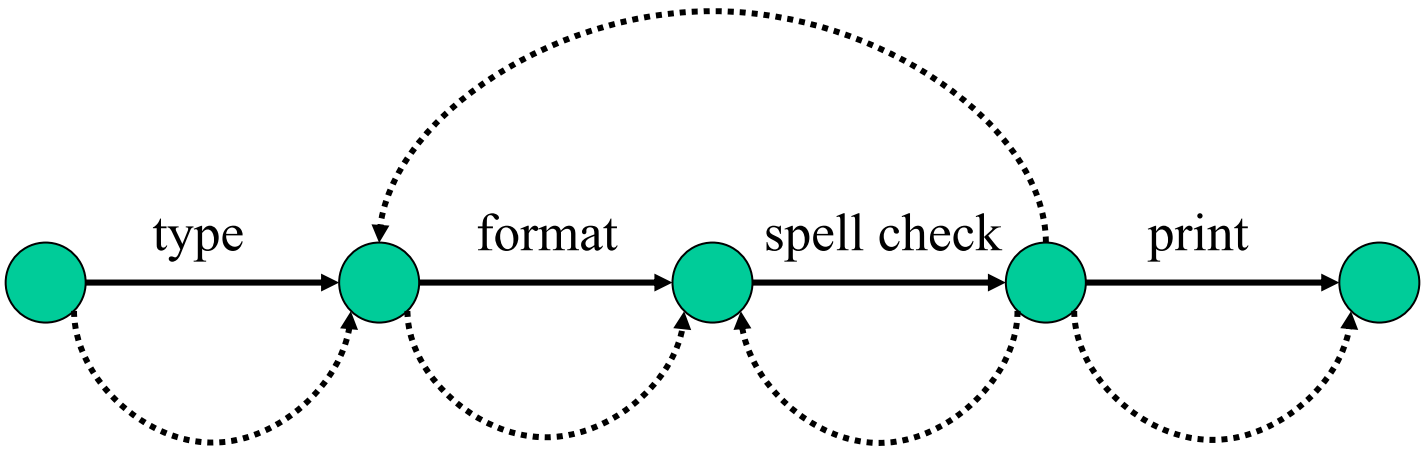
Useful Regression Testing



The Motivation

Q: What scenario does a developer use to test a fix?

A: The repro scenario you provided!

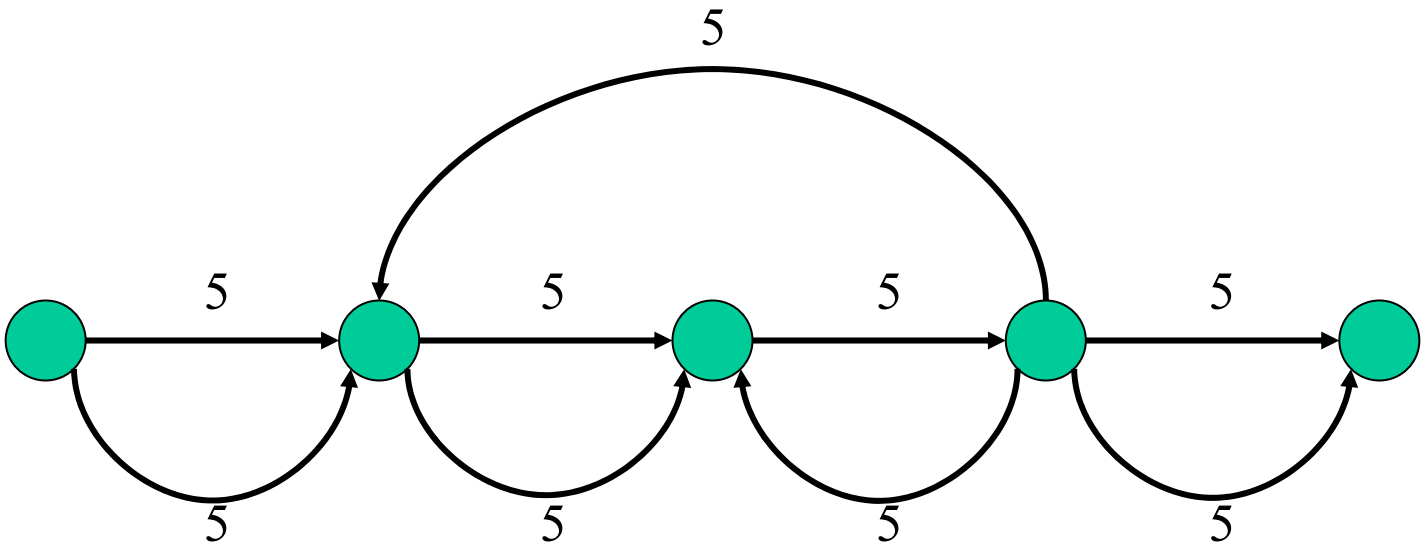


The Gawain* Approach

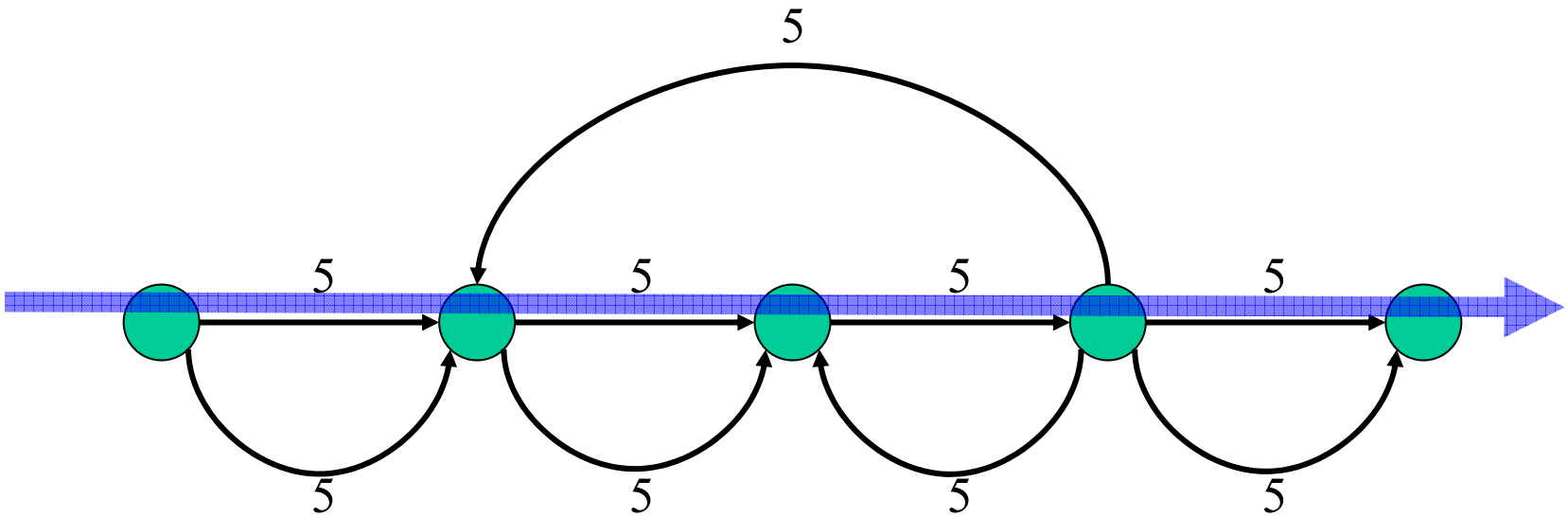
1. Assign the same weight to each arc in a graph
2. Choose a path through the graph
3. Assign a low weight to each arc in that path
4. Exercise paths in graph in weight-increasing order

* Graph Algorithm Without An Interesting Name

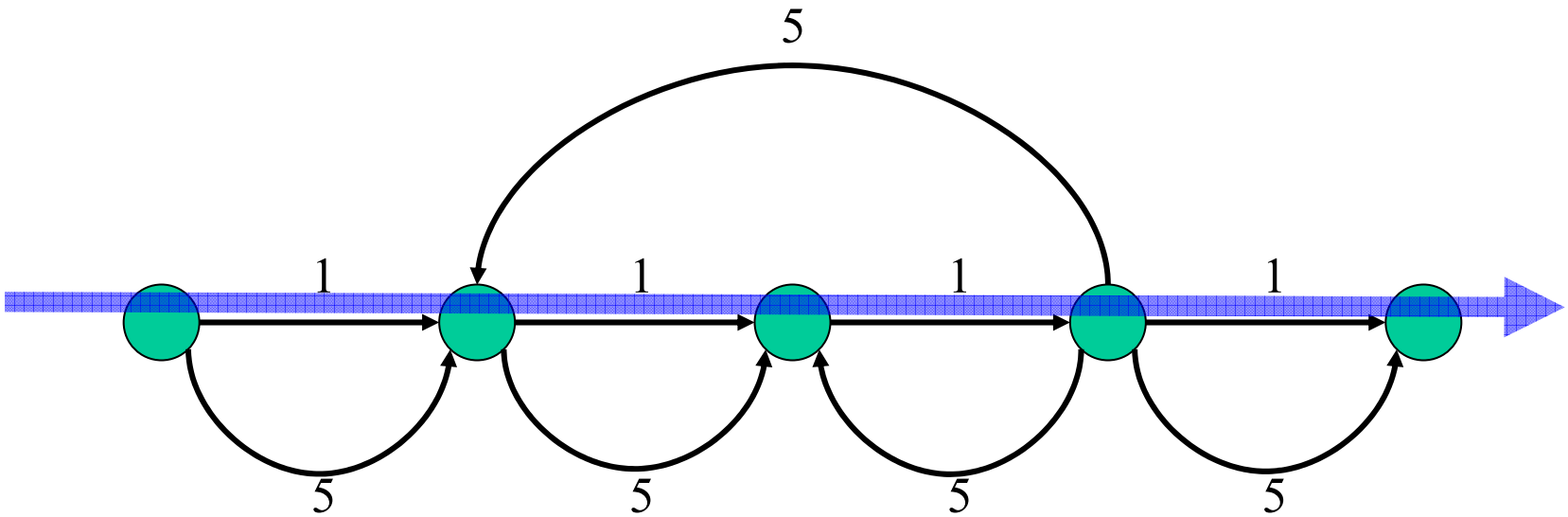
Assign the same weight to each arc



Choose a path through the graph

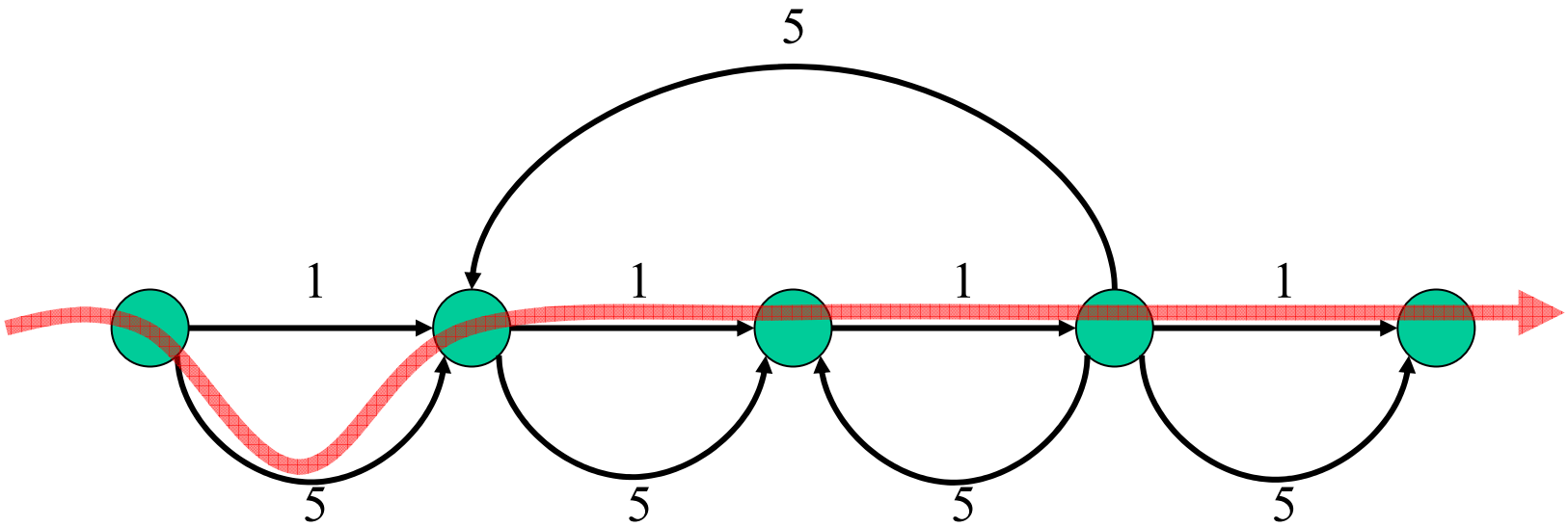


Assign a lower weight to each arc in that path

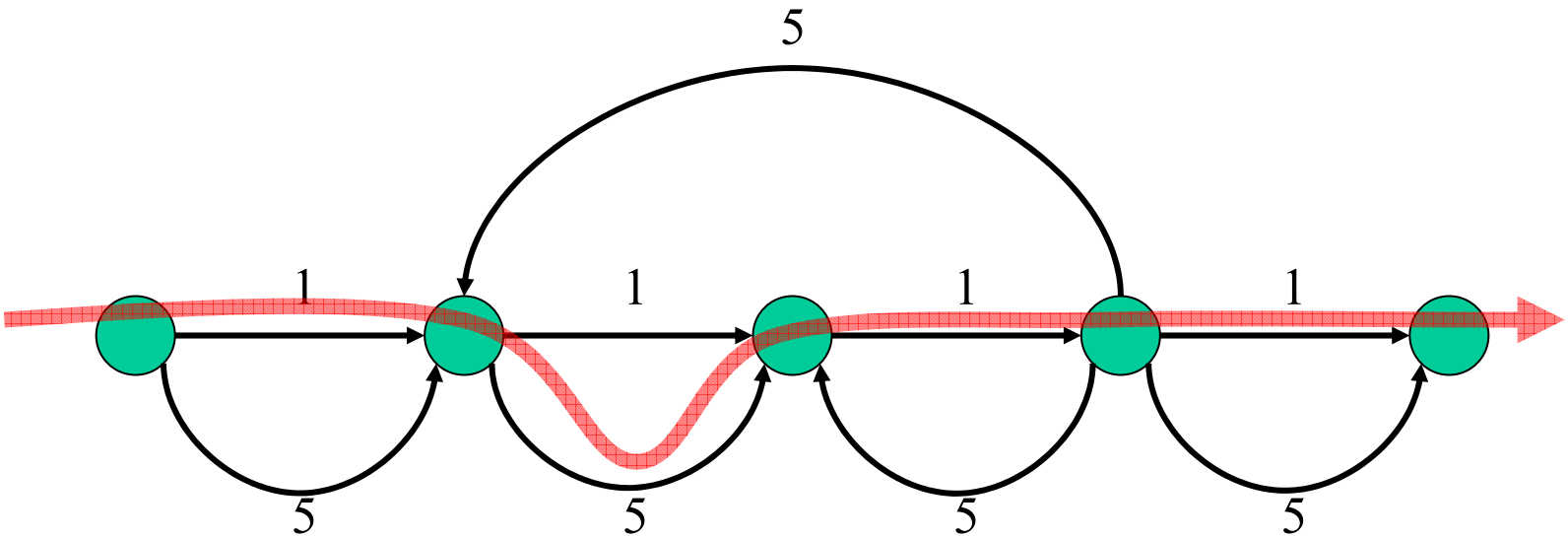


Weight of this path = 4

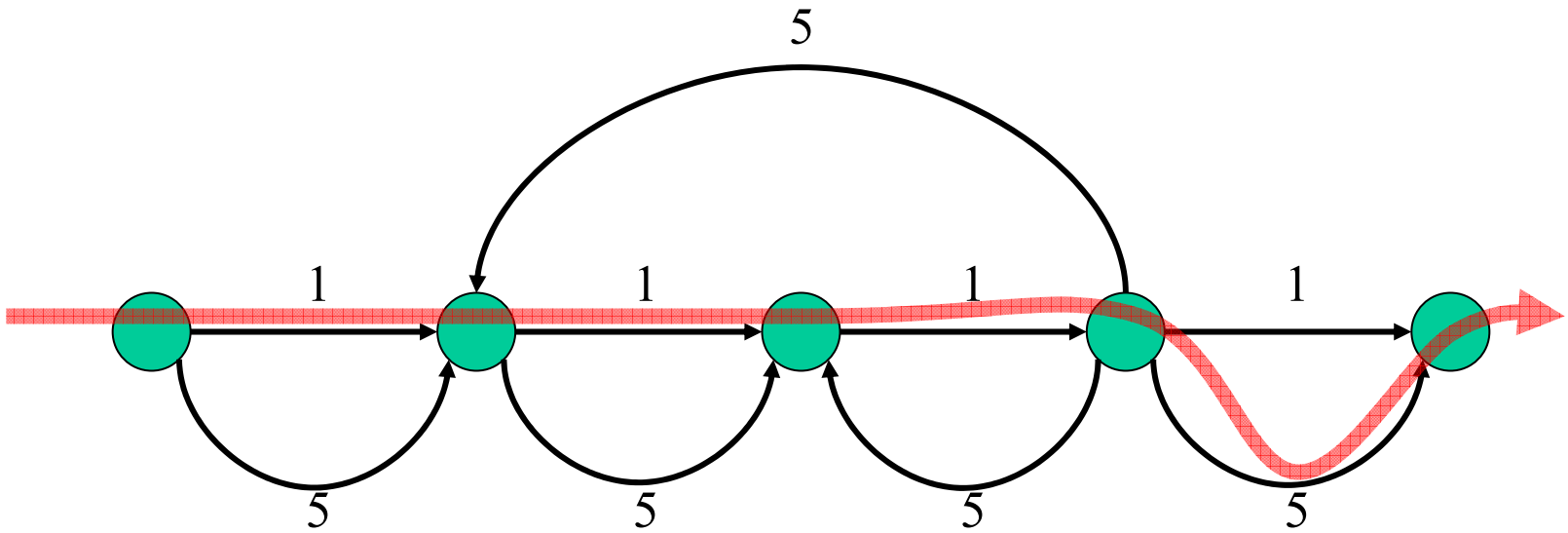
Execute all paths with total weight less than some amount "X"



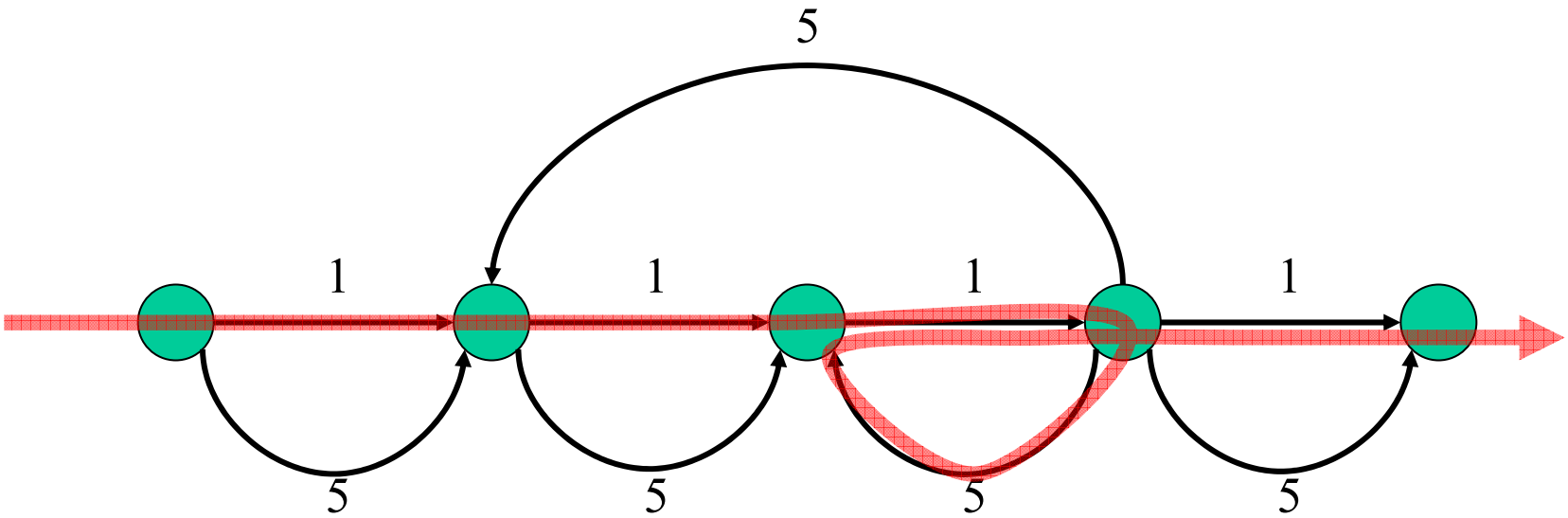
E.g., weight of this path = 8



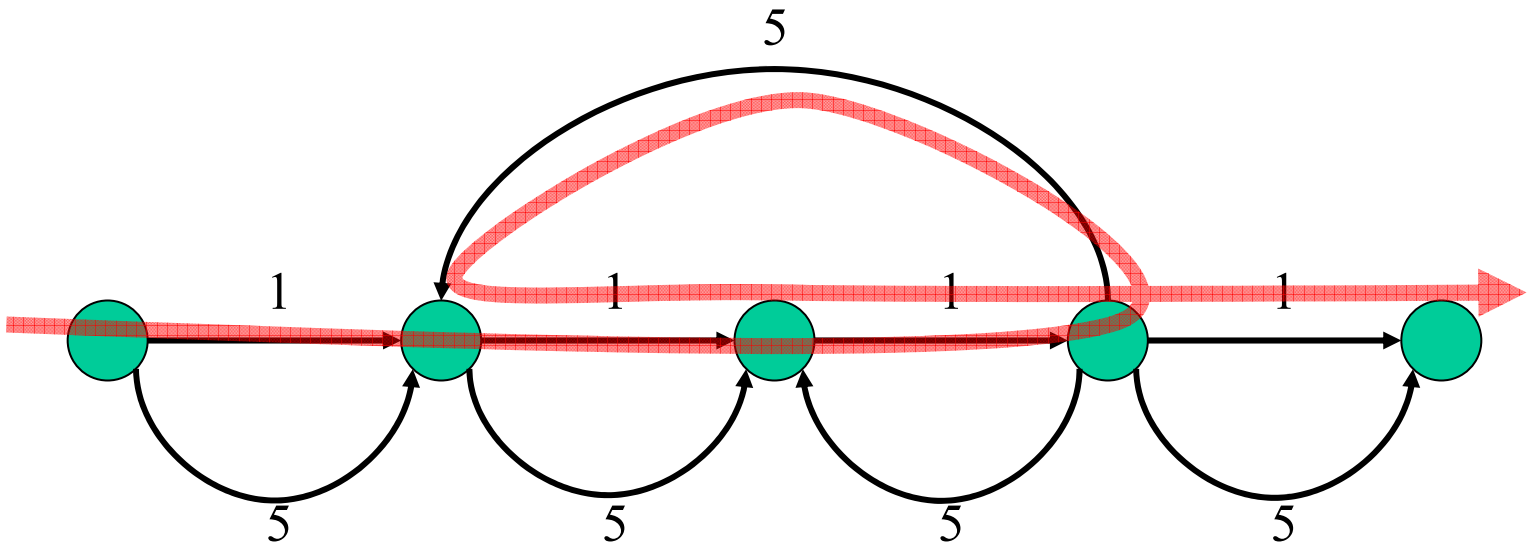
Weight of this path = 8



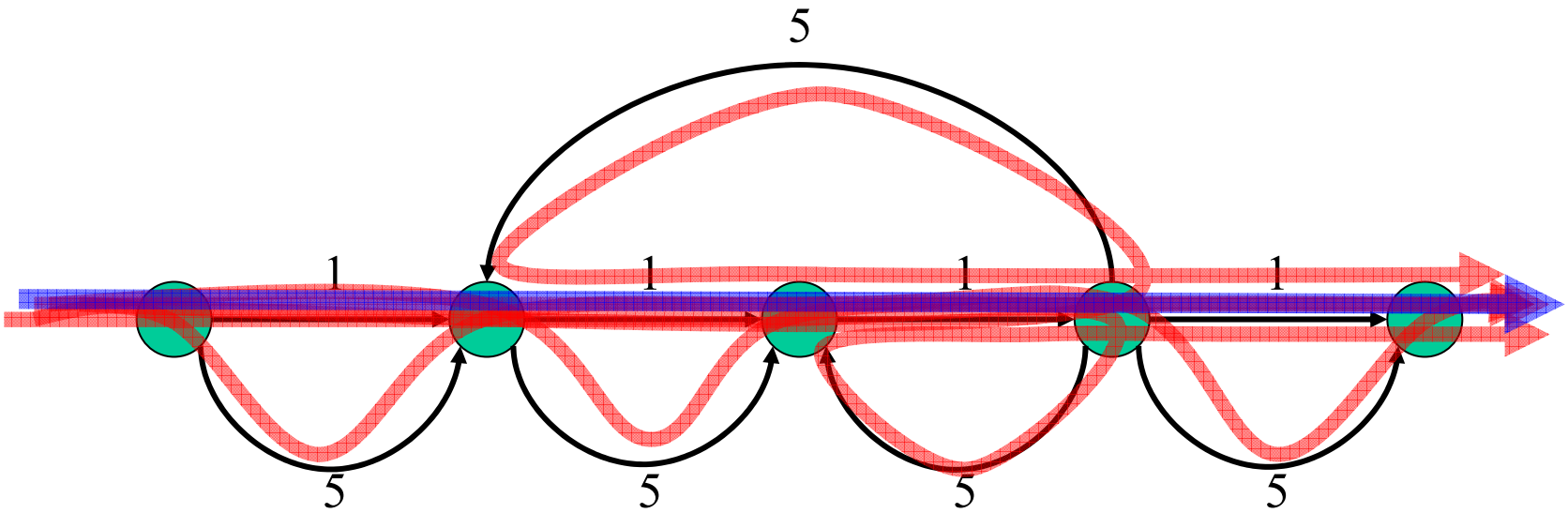
Weight of this path = 8



Weight of this path = 9

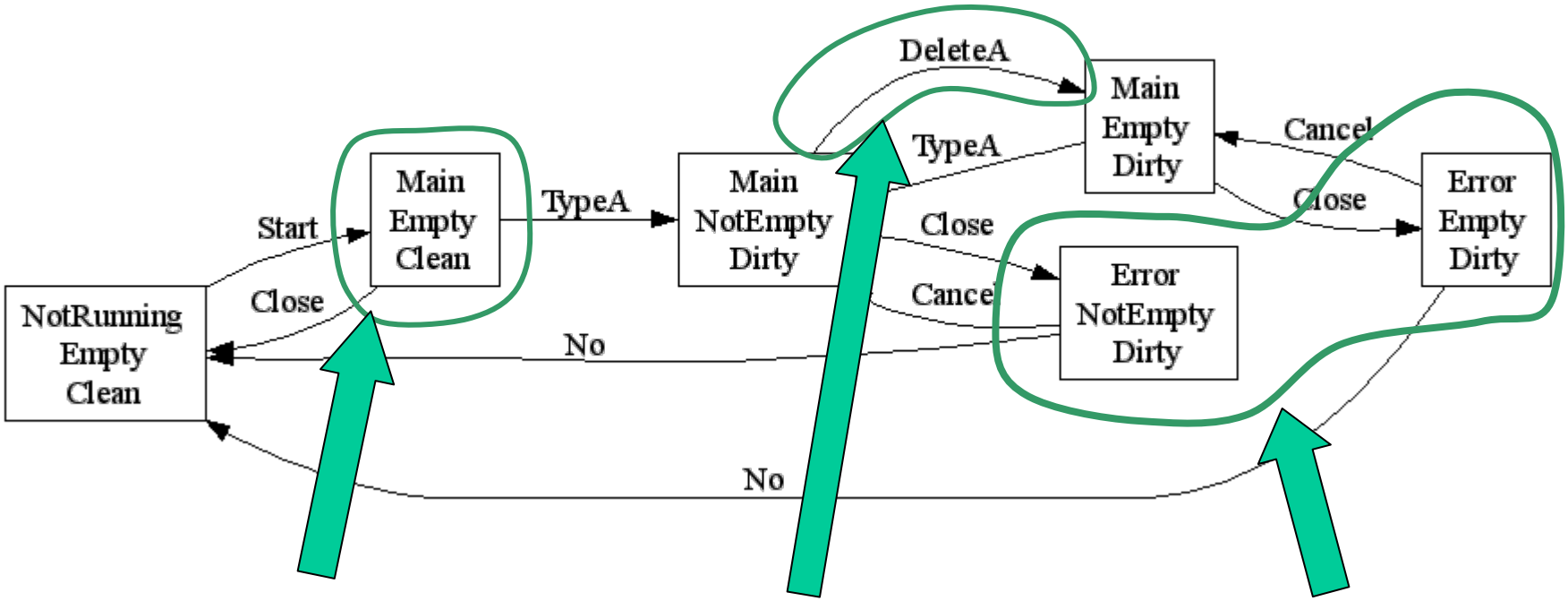


Weight of this path = 11



You end up “Cocooning” the regression path

Machine Learning?



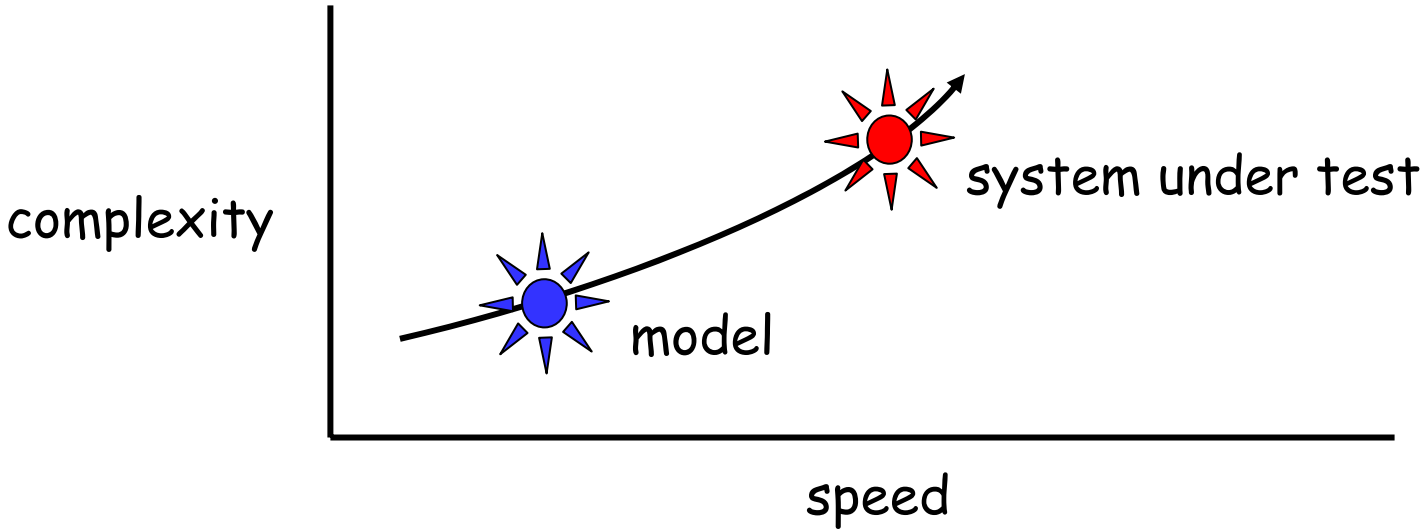
This developer is new.

New bug fix here.

This feature is new.

Observations on Model-Based Testing

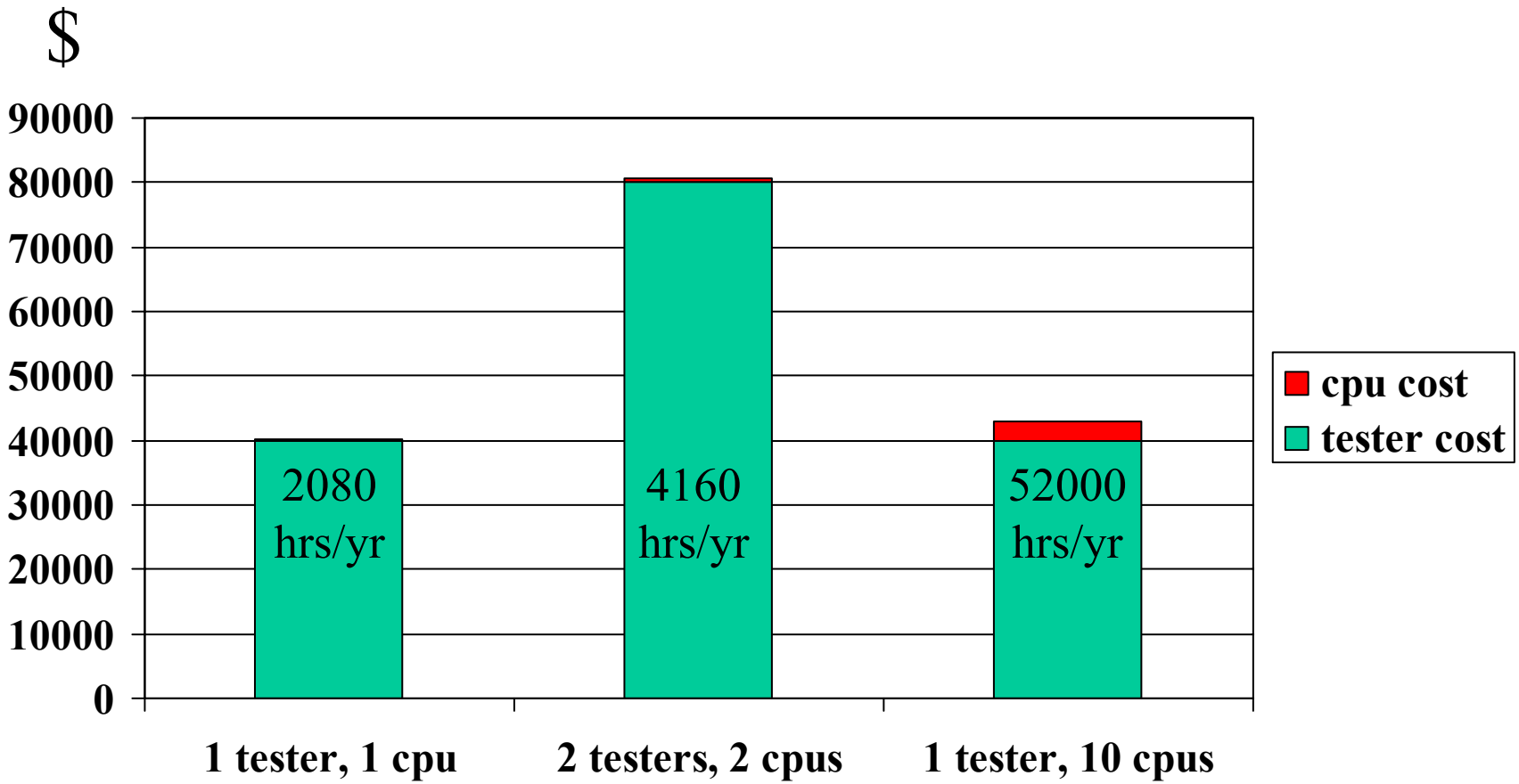
Why Does Model-Based Testing Work?



"… I think that less than 10 percent of most programs' code is specific to the application. Furthermore, that 10 percent is often the easiest 10 percent. Therefore, it is not unreasonable to build a model program to use as an oracle."

-Boris Beizer, Black Box Testing, p.63

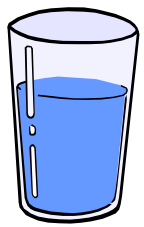
Economics of Model-Based Testing

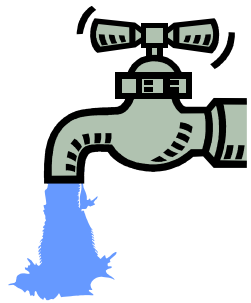


Metrics Issues

Should you count bugs you prevented?

Should you count how many test cases you've generated?





Benefits of Model-Based Testing

- Easy test case maintenance
- Reduced costs
- More test cases
- Early bug detection
- Increased bug count
- Time savings
- Time to address bigger test issues
- Improved tester job satisfaction

Obstacles to Model-Based Testing

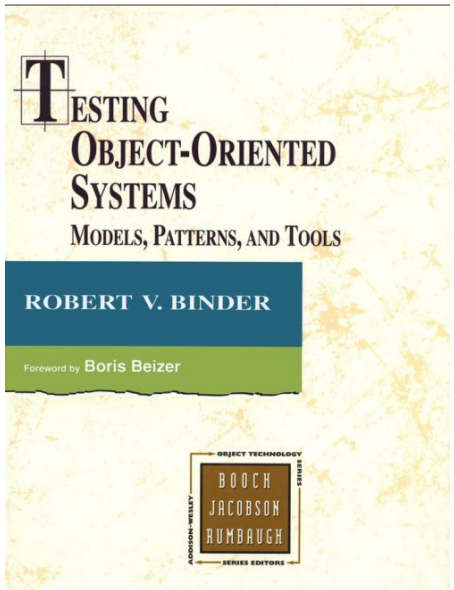
- Comfort factor
  - This is not your parents' test automation
- Skill sets
  - Need testers who can design
- Expectations
  - Models can be a significant upfront investment
  - Will never catch all the bugs
- Metrics
  - Bad metrics: bug counts, number of test cases
  - Better metrics: spec coverage, code coverage

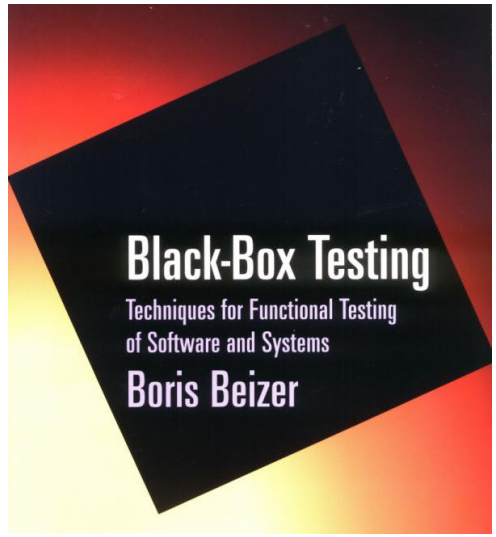
A Useful Resource

The Model-Based Testing Home Page

www.model-based-testing.org

Recommended reading





Thank you!