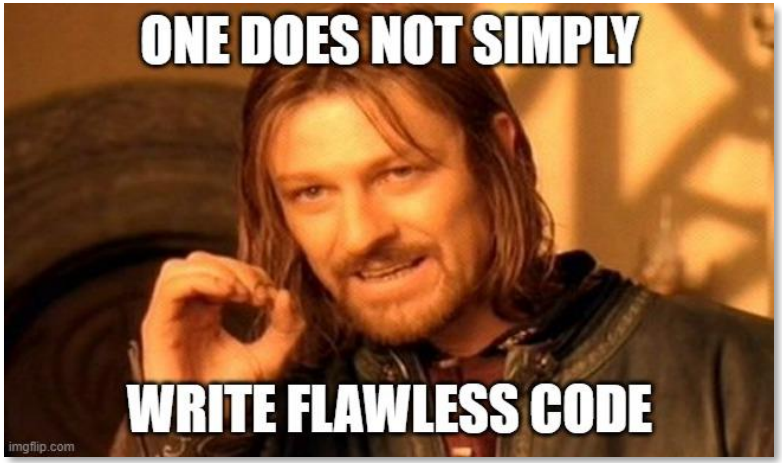
Never Work in Theory - Virtual
April 2023

Proofreading the Proofreader:
The Benefits of Unit Tests for Software Models

Allison Sullivan
The University of Texas at Arlington

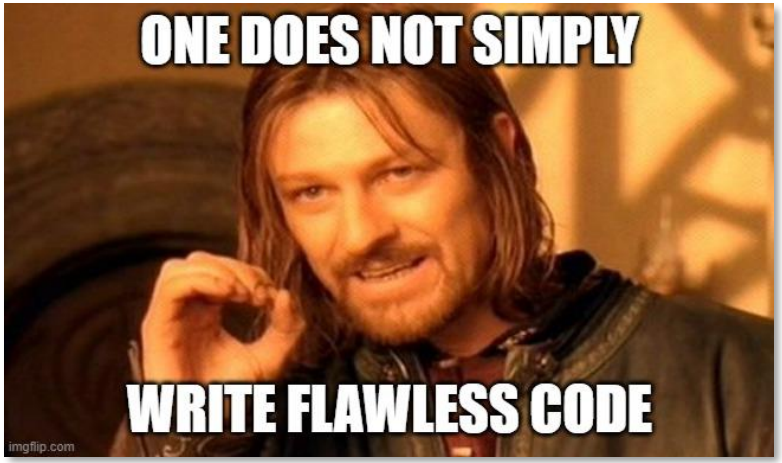
Software Modeling

Problem:



Software Modeling

Problem:



Solution:

```
one sig List { header: lone Node }
sig Node { link: lone Node }
pred acyclic(){
   no List.header or
   some n : List.header.*link | no n.link
}
run acyclic for 3
```

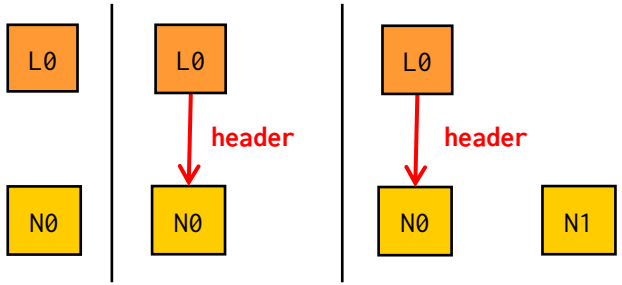
A Software Model

An Alloy Model:

```
one sig List { header: lone Node }
sig Node { link: lone Node }
pred acyclic(){
   no List.header or some n : List.header.*link | no n.link
}
run acyclic for 3
```

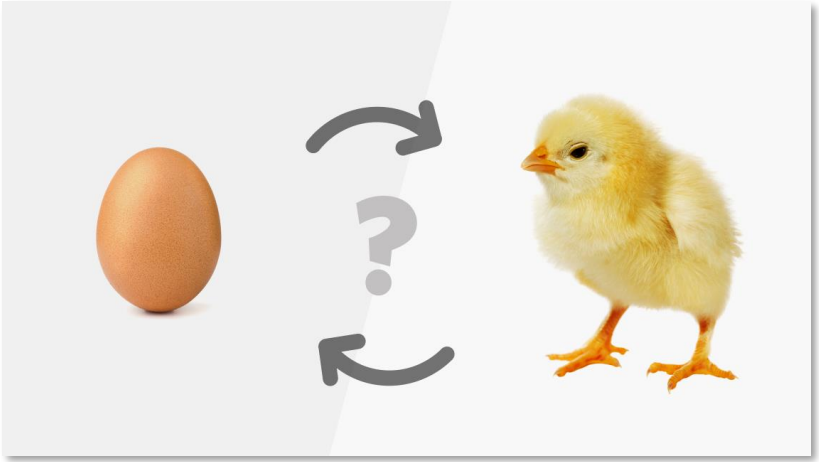
Show me all acyclic lists with up to 3 nodes

Discovered Scenario:



Software Modeling

***New* Problem:**

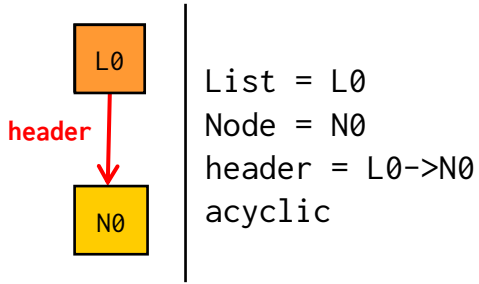


New Solution: Unit Tests for Models

An Alloy Model:

```
one sig List { header: lone Node }
sig Node { link: lone Node }
pred acyclic(){
   no List.header or some n : List.header.*link | no n.link
}
run acyclic for 3
```

~~Discovered Scenario:~~ Unit Test

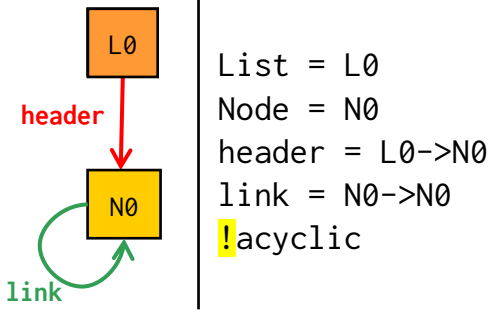


New Solution: Unit Tests for Models

An Alloy Model:

```
one sig List { header: lone Node }
sig Node { link: lone Node }
pred acyclic(){
   no List.header or some n : List.header.*link | no n.link
}
run acyclic for 3
```

~~Discovered Scenario:~~ Unit Test




```
List = L0
Node = N0
header = L0->N0
link = N0->N0
!acyclic
```

Ex: Mutation Testing

```
public int min(int x, int y) {
   int v;
   if(x < y)
      v = x;
   else
      v = y;
   return v;
}
```

Original

```
public int min(int x, int y) {
   int v;
   if(x >= y)
      v = x;
   else
      v = y;
   return v;
}
```

Mutant 1.

```
public int min(int x, int y) {
   int v;
   if(x <= y)
      v = x;
   else
      v = y;
   return v;
}
```

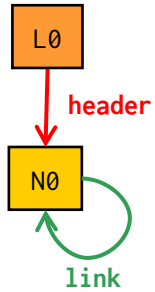
Mutant 2.

Ex: Mutation Testing

```
pred acyclic(){
   no List.header or some n : List.header.*link | no n.link
}
pred acyclicMUTATED(){
   no List.header or some n : List.header.*link | some n.link
}
check {acyclic[] <=> acyclicMUTATED} for 3
```

Ask: *Do these two properties differ? (detect equivalent mutant)*
If yes, show me a scenario where they differ. (test to kill mutant)

Answer:



What you can walk away with:

- Software models can be intimidating, but enable a really robust automated testing environment
- If working with a non-traditional language, consider investing in unit testing

Thank you! Any questions?

Doing a bunch of testing to make sure software is correct

Doing a bunch of testing to make sure a model of that software is correct

Allison Sullivan

The University of Texas at Arlington

allison.sullivan@uta.edu