Testing Your Program

M. Drew LaMar
October 17, 2016

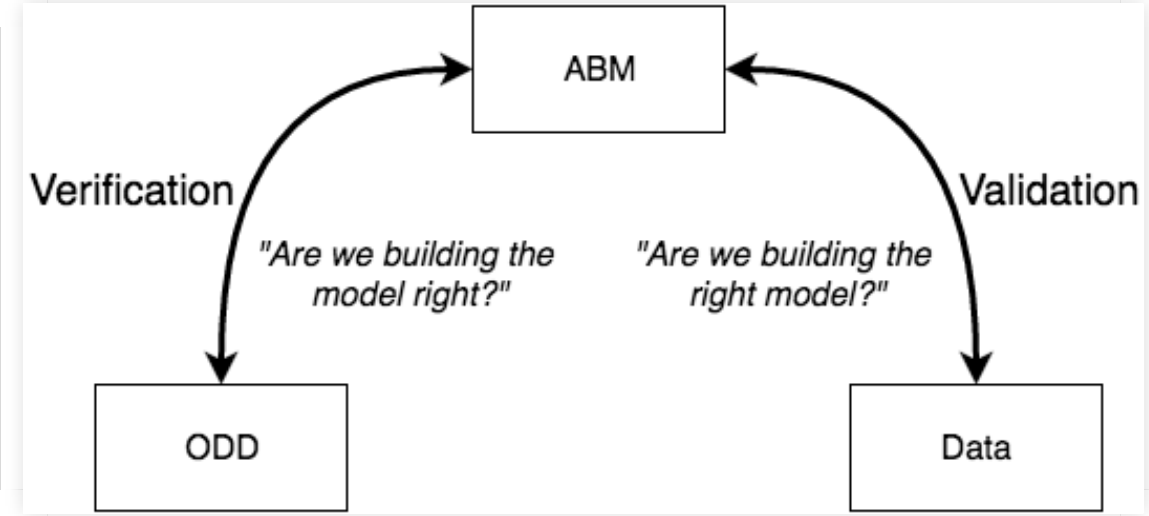
Class announcements

Chapter 6: Testing your Program - Learning Objectives

> "The productive modeler simply assumes that software mistakes are inevitable and continually searches for them."
>
> - Railsback & Grimm

- Understand the difference between validation and verification.
- **Seven** common kinds of software errors that will blow your mind!
- **Ten** important tecnhiques for finding and fixing software errors everyone should know.
- Understand why and how to document software tests.

Chapter 6: Testing your Program - Validation vs Verification



Part I. Common Kinds of Errors

- *Typographical Errors*
- *Syntax Errors*
- *Misunderstanding Primitives*: Show example
- *Wrong Display Settings*: Use `resize-world` in `setup`!
- *Logic Errors*: Runs but results incorrect
- *Run-time Errors*: No syntax or logic errors, but breaks on Go (sometimes)
- *Formulation Errors*: Incorrect assumptions & model decisions

Part II. Debugging Techniques

Syntax Checking: Chunk-it and use skeleton code!

```
ifelse (xcor >= min-marriage-age)
[ show "If" ]
[ show "Else" ]
```

```
ifelse (xcor >= min-marriage-age) and
       (random-float 1.0 < 0.1)
[ show "If" ]
[ show "Else" ]
```

```
ifelse (xcor >= min-marriage-age) and
       (random-float 1.0 < 0.1)
[ set married? true ]
[ show "Else" ]
```

Part II. Debugging Techniques

Visual Testing: Use visual cues of variables!

- Use `scale-color` to color turtles or patches based on their variables.
- Use `label` and `plabel` to check turtle or patch information.
- Use Agent and Patch Monitor.
- Use a smaller World to test things (actually, testing on smaller problem is a more general technique)
- Slow down the simulation and/or use a `step` button.

Part II. Debugging Techniques

Print Statements

- For procedures:

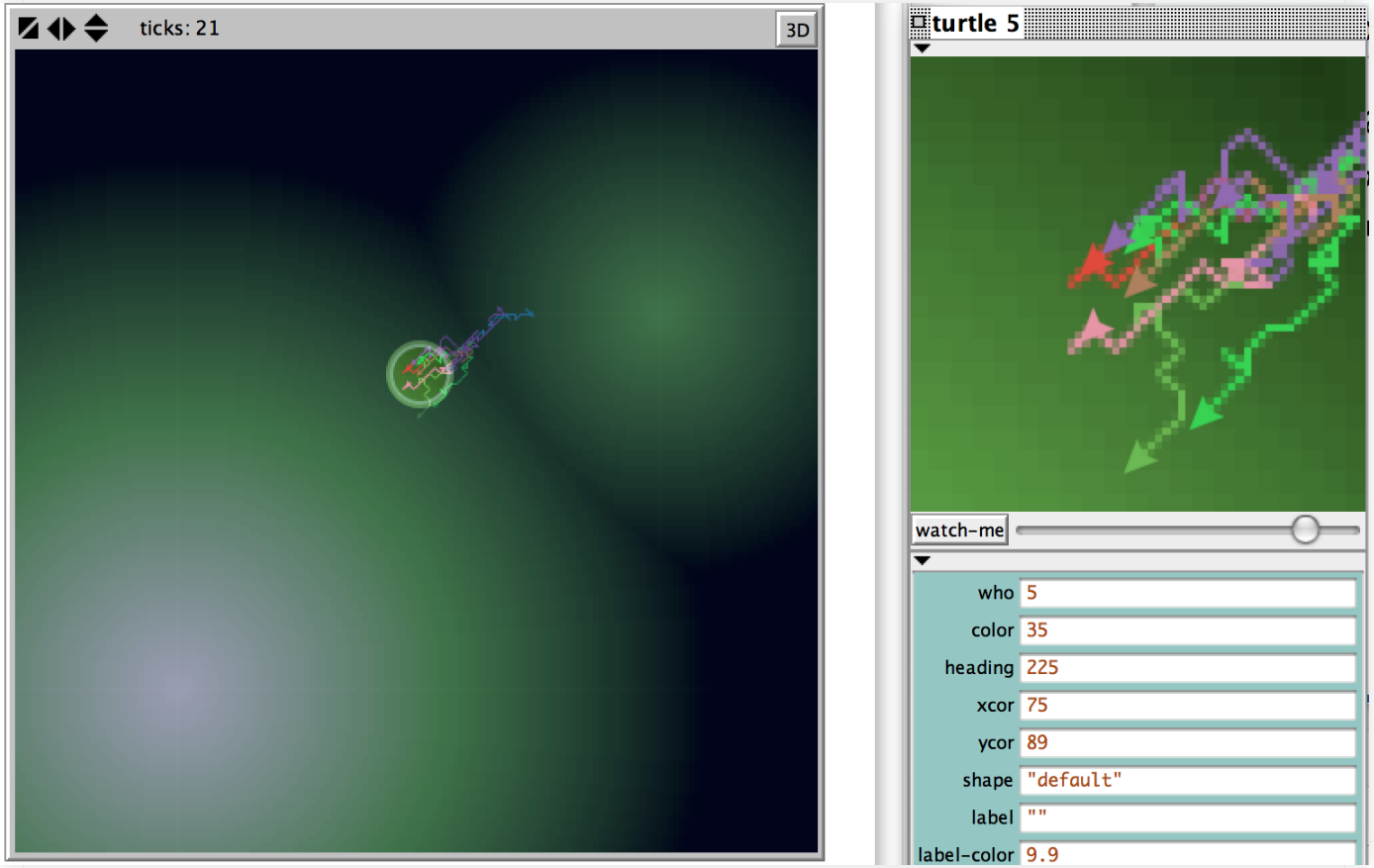
```
to dostuff
  show "Starting procedure X"
  ; Do stuff
  show "Ending procedure X"
end
```

- For variables:

```
observer> show word "num turtles = " count
turtles
observer: "num turtles = 0"
```

Part II. Debugging Techniques

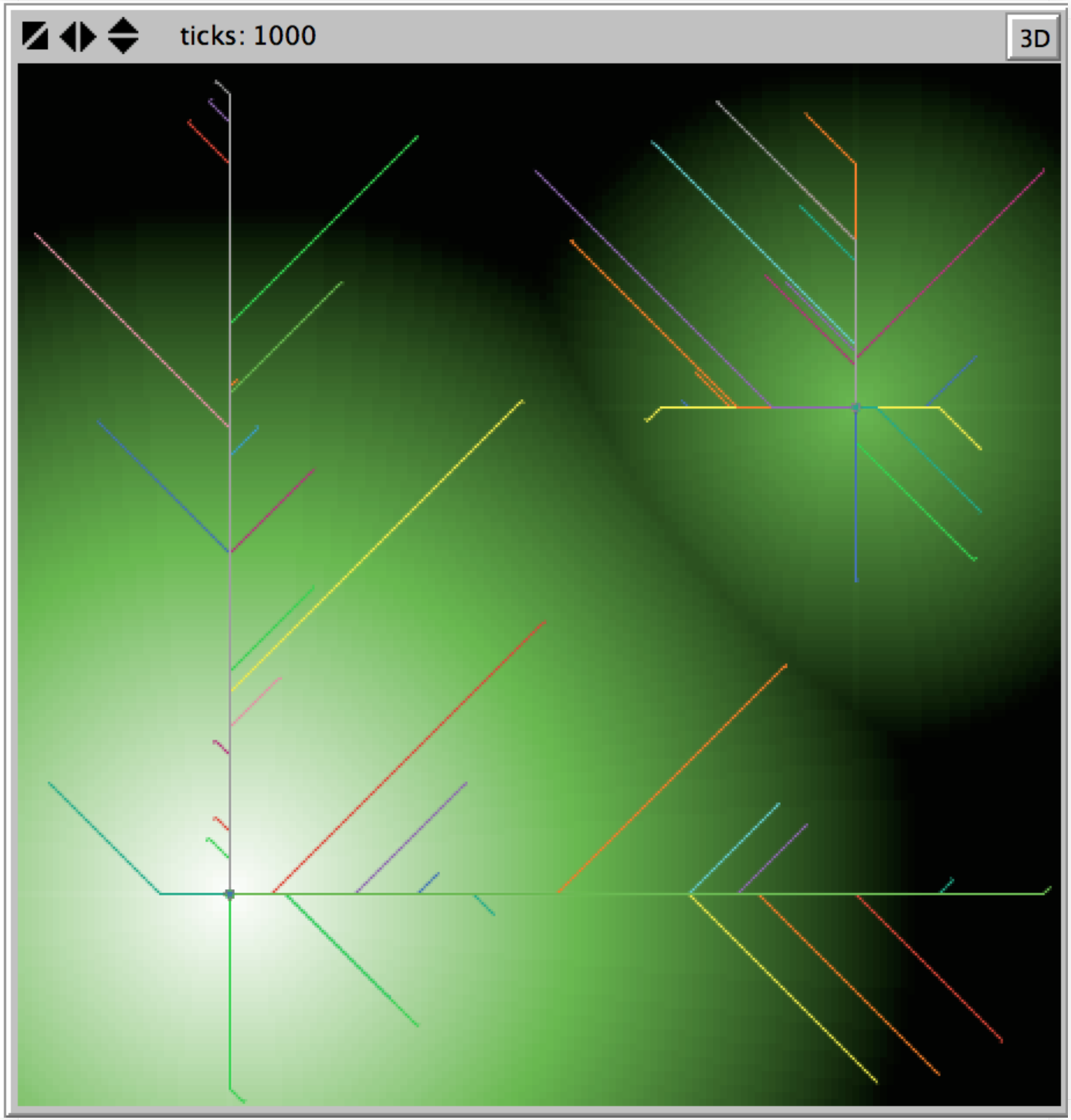
Spot Tests with Agent Monitors



Part II. Debugging Techniques

Stress Tests

Use parameters and initial data at the extreme values, and possibly outside normal ranges (e.g. $q = 1.0$)


ticks: 1000
3D

Part II. Debugging Techniques

Test Procedures

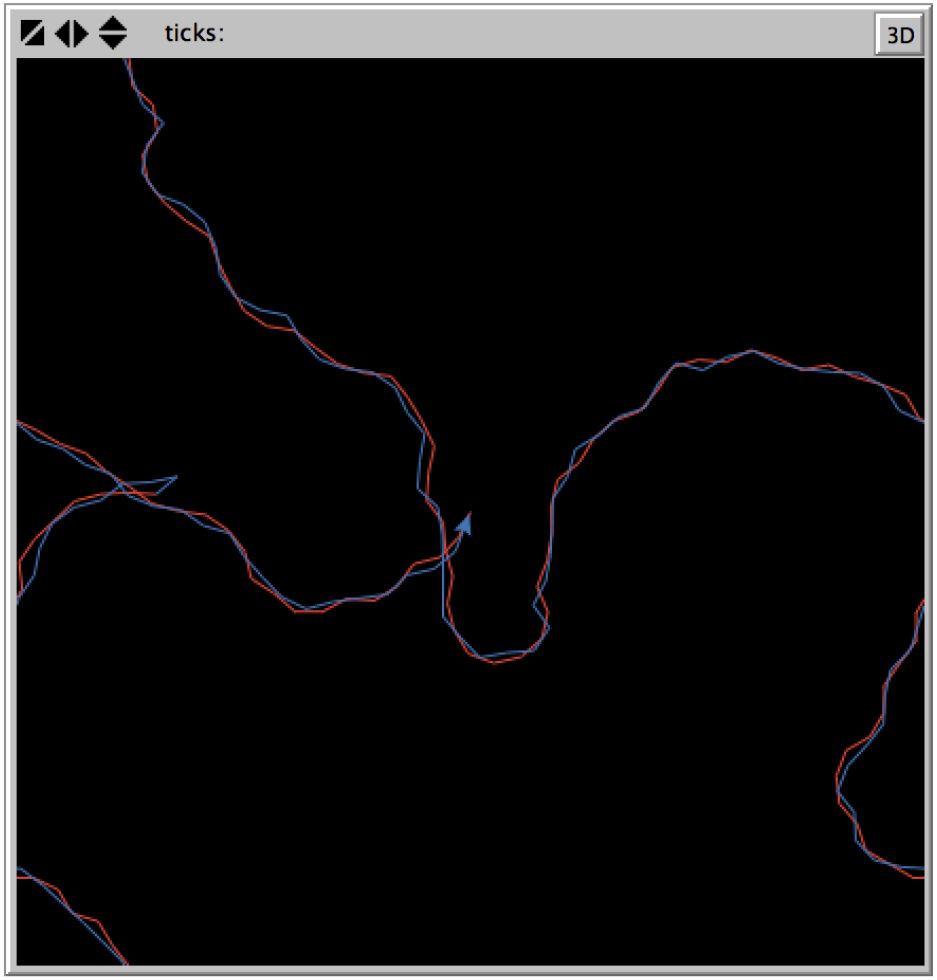
and

Test Programs

setup

go

go-back



Part II. Debugging Techniques

Code Reviews

- Reviewer's job:
  - Verification: Does the code match the ODD model formulation?
  - Fresh set of eyes can more easily find bugs (sometimes).
  - Make sure code is well-organized and easy to understand.
  - Write code as if someone will eventually read AND USE IT, even if you do not plan on it being used.

Part II. Debugging Techniques

Statistical Analysis of File Output

Edit: This slide has been modified from lecture to correct an error.

Question: Is the probability butterflies move to the highest neighbor patch really q?

Answer: No. It is the approximate proportion $q + \frac{1-q}{8}$.

For $q = 0.4$, we would expect the butterfly to move to the highest neighbor patch with probability 0.475.

Part II. Debugging Techniques

Statistical Analysis of File Output

```
file-type
file-print
file-open
file-close
```

Part II. Debugging Techniques

Statistical Analysis of File Output

```
mydata <- read.csv("TestOutput.csv",
header=FALSE)
str(mydata)
```

```
'data.frame':   1000 obs. of  9 variables:
 $ V1: num  15.8 14.9 17.1 17.8 16.4 ...
 $ V2: num  15.6 15.6 15.6 17 16.3 ...
 $ V3: num  16.3 15.5 16.9 18.5 17.1 ...
 $ V4: num  14.7 17 18.4 19.2 18.4 ...
 $ V5: num  15.6 15.7 16.4 19.1 16.9 ...
 $ V6: num  14.8 16.3 17.7 19.8 17.7 ...
 $ V7: num  15.5 16.4 16.3 17.7 15.6 ...
 $ V8: num  15.5 15.5 17.8 18.3 17.8 ...
 $ V9: num  15.6 17 18.4 17 15.6 ...
```

Part II. Debugging Techniques

Statistical Analysis of File Output

Note: There were errors here during lecture. The code below has been changed to correct the error. I will discuss in class on Wednesday.

```
moved.to.highest <- sapply(1:1000, function (x)
{max(mydata[x,1:8]) == mydata[x,9]})

moved.to.highest <- as.integer(moved.to.highest)
```

Part II. Debugging Techniques

Statistical Analysis of File Output

```
prop.test(sum(moved.to.highest), 1000)
```

```
    1-sample proportions test with continuity
correction

data:  sum(moved.to.highest) out of 1000, null
probability 0.5
X-squared = 23.409, df = 1, p-value = 1.31e-06
alternative hypothesis: true p is not equal to
0.5
95 percent confidence interval:
 0.3922385 0.4543604
sample estimates:
    p
0.423
```