

# Chapter 19: GUI Testing

Ian Tucker

December 3, 2006

- 1 Introduction
- 2 The Currency Conversion Program
  - An Introduction to the Program
  - Testing the program
- 3 Unit Testing
  - Unit Testing Considerations
  - Unit Testing Methods
- 4 Integration Testing
  - Integration Testing Considerations
- 5 System Testing
  - System Testing Considerations
  - System Level Threads
- 6 Summary

# Introduction



- Event driven
- Unit “button level” testing
- Not much integration testing
- System level testing
- UML Behavioral models
  - StateCharts
  - Finite State Machines





# The Program:

## Testing the Program



- Identify
  - User inputs
  - Visible & observable system outputs

# Input Events

## Input Event

---

ip1	Enter U.S. dollar amount
ip2	Click on a country button
ip2.1	Click on Brazil
ip2.2	Click on Canada
ip2.3	Click on European Community
ip2.4	Click on Japan
ip3	Click on Compute Button
ip4	Click on Clear Button
ip5	Click on Quit Button
ip6	Click on OK in error message



# Output Events

	Output Event
op1	Display U.S. dollar amount
op2	Display currency name
op2.1	Display Brazilian reals
op2.2	Display Canadian dollars
op2.3	Display European Community euros
op2.4	Display Japanese Yen
op2.5	Display ellipsis
op3	Indicate selected country
op3.1	Indicate Brazil
op3.2	Indicate Canada
op3.3	Indicate European Community
op3.4	Indicate Japan

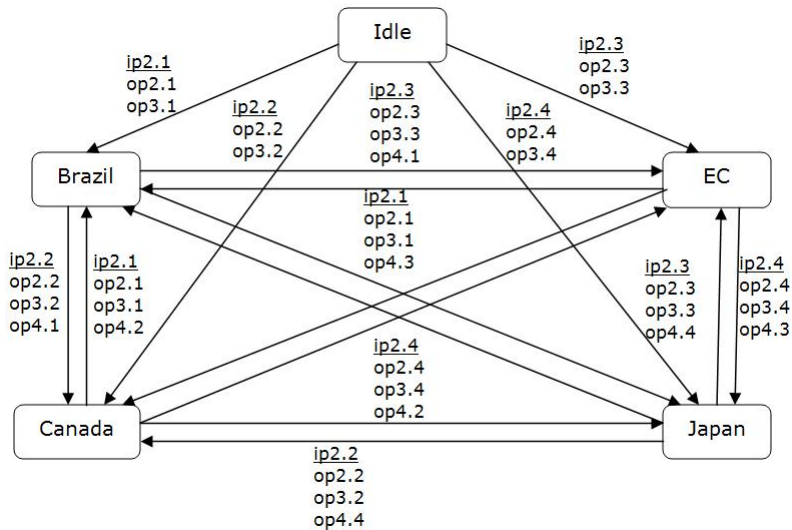
# Output Events

	Output Event
op4	Reset selected country
op4.1	Reset Brazil
op4.2	Reset Canada
op4.3	Reset European Community
op4.4	Reset Japan
op5	Display foreign currency value
op6	Error msg: must select a country
op7	Error msg: must enter USD amount
op8	Error msg: must select country and enter USD amount
op9	Reset U.S. Dollar amount
op10	Reset equivalent currency amount





## Testing the program





# Unit Testing for the Currency Conversion Program

All dem little bits



- Public domain GUI libraries
- User supplied code
- Compute, Clear, Quit



# Unit Testing Methods

- Test driver
- GUI test bed
  - System level unit testing
  - What if computation correct, but fault in output software?
  - How do you get the test execution results?
  - Seat-of-the-pants testing
  - User input & observation errors
  - Repeating tests is time-consuming



# Integration Testing for the Currency Conversion Program

How all dem little bits fit together



- Depends heavily on implementation
- Little need for most small GUIs



# System Testing for the Currency Conversion Program



- Unit and Integration testing minimally needed
- Onus on System Testing
- Testing threads
- Event-driven Petri nets (EDPNs)

# Port Input Events

## Port Input Events

---

- p1 Enter U.S. dollar amount
- p2 Click on Brazil
- p3 Click on Canada
- p4 Click on European Community
- p5 Click on Japan
- p6 Click on Compute Button
- p7 Click on Clear Button
- p8 Click on Quit Button

# Port Output Events

## Port Output Events

---

- p9 Display US Dollar Amount
- p10 Display Brazilian reals
- p11 Display Canadian dollars
- p12 Display E.U. euros
- p13 Display Japanese Yen
- p14 Display ellipsis
- p15 Indicate Brazil
- p16 Indicate Canada
- p17 Indicate European Community
- p18 Indicate Japan

# Port Output Events

## Port Output Events

---

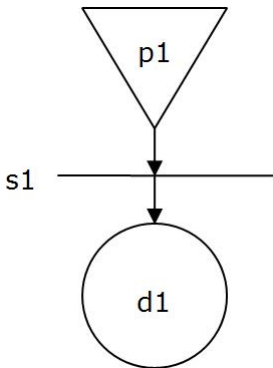
p19	Reset Canada, E.U., Japan
p20	Reset Brazil, E.U., Japan
p21	Reset Brazil, Canada, Japan
p22	Reset Brazil, Canada, E.U.
p23	Reset U.S. Dollar amount
p24	Reset equivalent currency amount
p25	End Application
p26	Display equivalent currency amount

# Atomic System Functions

## Atomic System Functions & Data Places

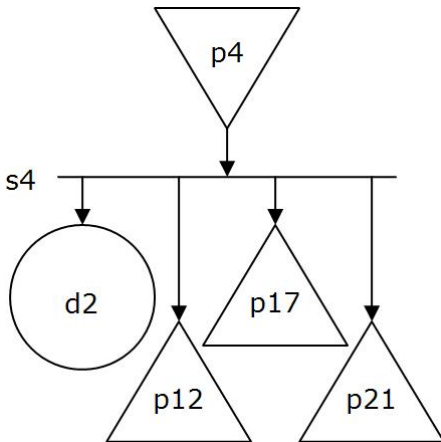
s1	US Dollar Amount Stored
s2	Sense Click on Brazil
s3	Sense Click on Canada
s4	Sense Click on EU
s5	Sense Click on Japan
s6	Sense Click on Compute Button
s7	Sense Click on Clear Button
s8	Sense Click on Quit Button
d1	US Dollar amount entered
d2	County selected

# EPDNs: A Simple Example



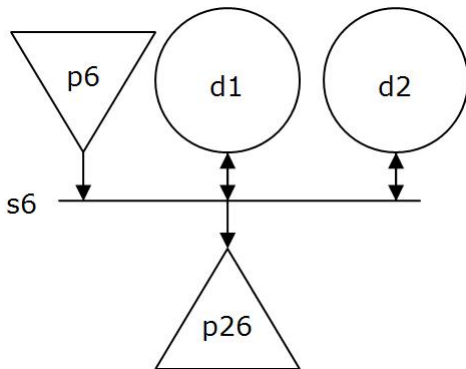
s1: Store Dollar Amount

# EPDNs: Little bit harder now



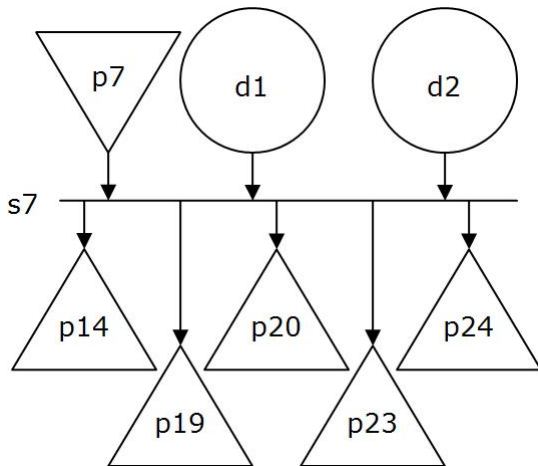
s4: Sense Click on E.U.

# EPDNs: Getting tougher



s6 Sense Click on Compute button

# EPDNs: Good luck

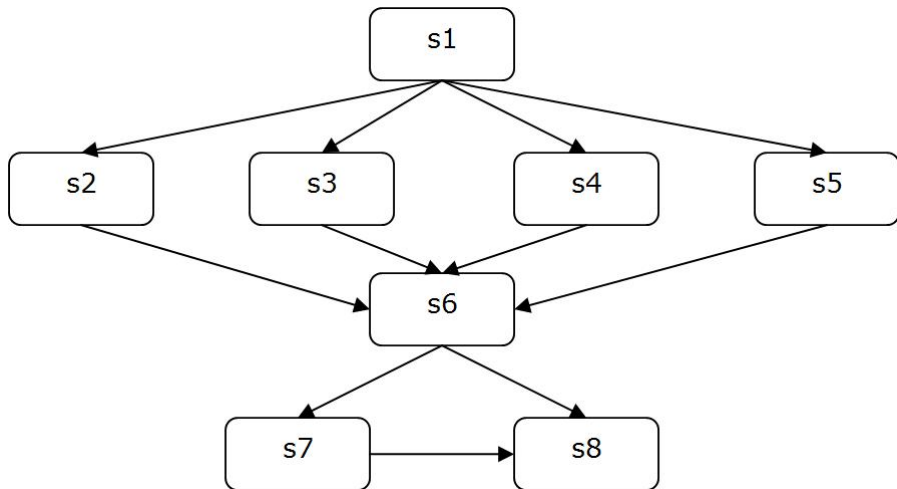


s7: Sense Click on Clear button



- Can combine ASF EDPNs into sequences:
- **System Level Threads**





Partial Directed graph of mainline ASF Sequences



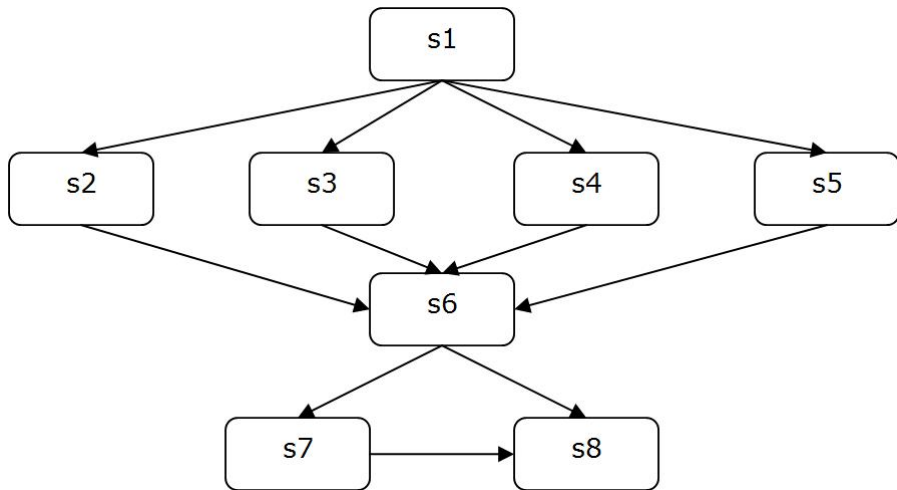
# How we test a GUI System

- Lowest Level: exercise all ASFs
  - Not a good idea – no visible system-level outputs
  - Even worse – ASFs with no port outputs (s1)
  - Cannot tell if an amount is correctly stored
- Next idea: exercise a suitable set of threads...
- What the hell is a “suitable” set?



# Some possibilities

- Every ASF
- Every Port Input
- Every Port Output



Partial Directed graph of mainline ASF Sequences



- Consider a set of threads T [T1, T2, T3, T4]
  - T1 = [s1, s4, s6, s7]
  - T2 = [s1, s2, s6, s7]
  - T3 = [s3, s1, s6, s7]
  - T4 = [s5, s1, s7, s8]
- Coverage:
  - Every ASF
  - Every port input
  - Every port output



## More Detail?

- Next-level user behavior
- A good example:
  - $T5 = [s1, s2, s6, s3, s6, s4, s6, s5, s6, s7, s8]$
  - Converts amount to all 4 currencies, clears and quits
- Abnormal behavior sequences
  - $T6 = [s1, s2, s3, s4, s5, s6, s7, s8]$
  - Changes mind about currency
- Silly threads
  - $T7 = [s1, s2, s3, s2, s3, s2, s3, s8]$
  - Toggles between 2 countries, then quits
- An infinite amount of threads for this GUI



# An Incidence Matrix

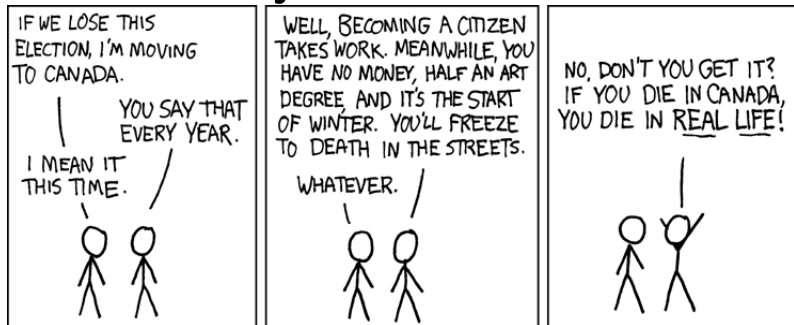
	s1	s2	s3	s4	s5	s6	s7	s8
s1	1	1	1	1	1	1	1	1
s2	1	0	1	1	1	1	1	1
s3	1	1	0	1	1	1	1	1
s4	1	1	1	0	1	1	1	1
s5	1	1	1	1	0	1	1	1
s6	1	1	1	1	1	1	1	1
s7	1	1	1	1	1	1	1	1
s8	0	0	0	0	0	0	0	0



# Summary

- Unit testing & integration testing not important
- System testing most complicated test.
- ASFs and EDPNs can help identify and validate test cases

# Any Questions?



[www.xkcd.com](http://www.xkcd.com)