Testing Tactics

Functional
"black box"

Structural
"white box"

- Tests based on *spec*
- Test covers as much *specified* behavior as possible

- Tests based on *code*
- Test covers as much *implemented* behavior as possible

Systematic Functional Testing

Functional specification → *identify* → Independently testable feature

identify → Representative values

derive → Model

derive → Test case specifications

generate → Test case
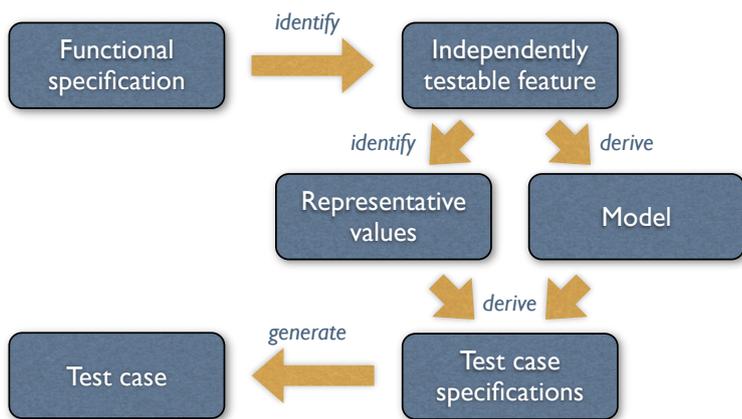
The main steps of a systematic approach to functional program testing (from Pezze + Young, "Software Testing and Analysis", Chapter 10)

Systematic Partition Testing

■ Failure (valuable test case)
□ No failure

The space of possible input values (the haystack)

Failures are sparse in the space of possible inputs ...

... but dense in some parts of the space

If we systematically test some cases from each part, we will include the dense parts

Functional testing is one way of drawing orange lines to isolate regions with likely failures
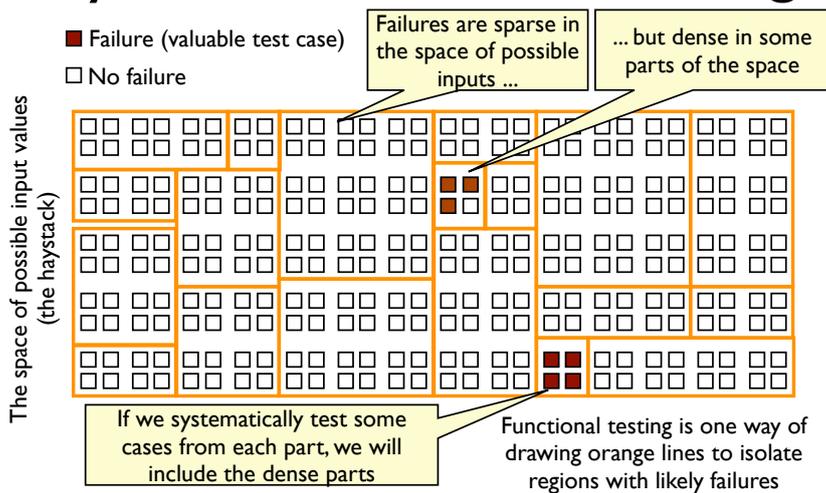
We can think of all the possible input values to a program as little boxes ... white boxes that the program processes correctly, and colored boxes on which the program fails. Our problem is that there are a lot of boxes ... a huge number, and the colored boxes are just an infinitesimal fraction of the whole set. If we reach in and pull out boxes at random, we are unlikely to find the colored ones. Systematic testing says: Let's not pull them out at random. Let's first subdivide the big bag of boxes into smaller groups (the pink lines), and do it in a way that tends to concentrate the colored boxes in a few of the groups. The number of groups needs to be much smaller than the number of boxes, so that we can systematically reach into each group to pick one or a few boxes.
Functional testing is one variety of partition testing, a way of drawing the orange lines so that, when one of the boxes within a orange group is a failure, many of the other boxes in that group may also be failures. Functional testing means using the program specification to draw pink lines. (from Pezze + Young, "Software Testing and Analysis", Chapter 10)

Maintenance: The *Maintenance* function records the history of items undergoing maintenance.

If the product is covered by warranty or maintenance contract, maintenance can be requested either by calling the maintenance toll free number, or through the Web site, or by bringing the item to a designated maintenance station.

If the maintenance is requested by phone or Web site and the customer is a US or EU resident, the item is picked up at the customer site, otherwise, the customer shall ship the item with an express courier.

If the maintenance contract number provided by the customer is not valid, the item follows the procedure for items not covered by warranty.

If the product is not covered by warranty or maintenance contract, maintenance can be requested only by bringing the item to a maintenance station. The maintenance station informs the customer of the estimated costs for repair. Maintenance starts only when the customer accepts the estimate. If the customer does not accept the estimate, the product is returned to the customer.

Small problems can be repaired directly at the maintenance station. If the maintenance station cannot solve the problem, the product is sent to the maintenance regional headquarters (if in US or EU) or to the maintenance main headquarters (otherwise).

If the maintenance regional headquarters cannot solve the problem, the product is sent to the maintenance main headquarters.

Maintenance is suspended if some components are not available.

Once repaired, the product is returned to the customer.

70

…based on these (informal) requirements
(from Pezze + Young, "Software Testing and Analysis", Chapter 14)

Finite State Machine

71
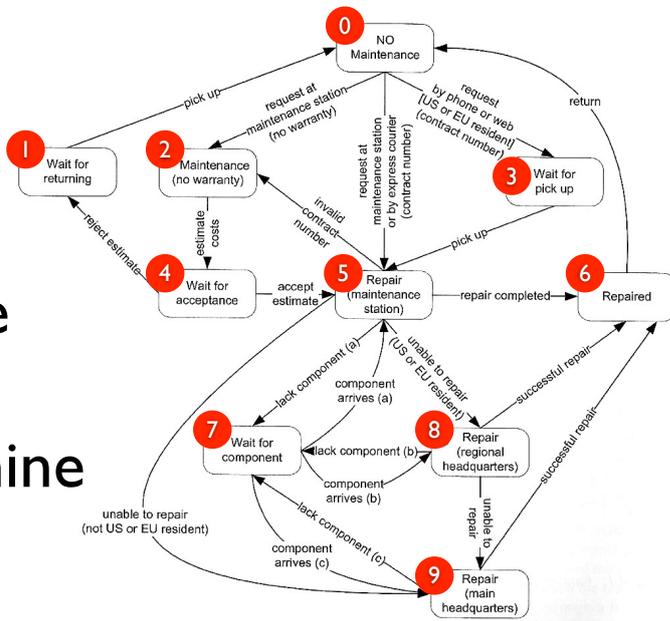
As an example, consider these steps modeling a product maintenance process…
(from Pezze + Young, "Software Testing and Analysis", Chapter 14)

Decision Tables

	Education		Individual					
Education account	T	T	F	F	F	F	F	F
Current purchase > Threshold 1	–	–	F	F	T	T	–	–
Current purchase > Threshold 2	–	–	–	–	F	F	T	T
Special price < scheduled price	F	T	F	T	–	–	–	–
Special price < Tier 1	–	–	–	–	F	T	–	–
Special price < Tier 2	–	–	–	–	–	–	F	T
Out	Edu discount	Special price	No discount	Special price	Tier 1 discount	Special price	Tier 2 discount	Special Price

72

A decision table describes under which conditions a specific outcome comes to be. This decision table, for instance, determines the discount for a purchase, depending on specific thresholds for the amount purchased.
(from Pezze + Young, "Software Testing and Analysis", Chapter 14)