CS 8803 ASE – Class 19

- Topics
  - Black-Box (Functional) Testing
  - Team work
- Due
  - Tech Day Assignment 3 (Wed, Oct 22 @11:59PM AOE)
  - Feasibility Study (Sun, Oct 26 @11:59PM AOE)
  - Tech Day Assignment 4 (Wed, Oct 29 @11:59PM AOE)

Functional vs. Structural Testing

BLACK-BOX TESTING EXAMPLE

Specification: inputs an integer and prints it

BLACK-BOX TESTING EXAMPLE

Specification: inputs an integer and prints it

```
1. void printNumBytes( param )
2.   if (param < 1024) printf("%d", param);
3.   else printf("%d KB", param/124);
4. }
```

WHITE-BOX TESTING EXAMPLE

```
1. int fun(int param){
2.   int result;
3.   result = param/2;
4.   return result;
5. }
```


WHITE-BOX TESTING EXAMPLE

Specification: inputs an integer param and returns half of its value if even, its value otherwise

```
1. int fun(int param){
2.   int result;
3.   result = param/2;
4.   return result;
5. }
```

SOFTWARE TESTING

BLACK-BOX TESTING

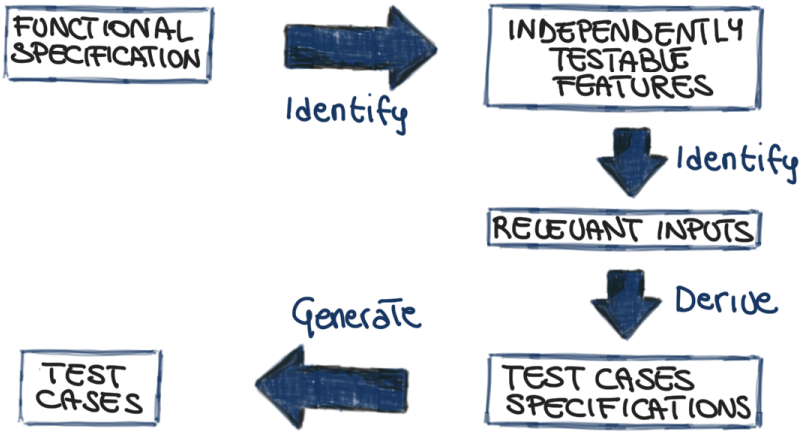
BLACK-BOX TESTING

Advantages

- focus on the domain
- No need for the code
  ⇒ early test design
- Catches logic defects
- Applicable at all granularity levels

FROM SPECIFICATIONS TO TEST CASES

A SYSTEMATIC FUNCTIONAL-TESTING APPROACH



A SYSTEMATIC FUNCTIONAL-TESTING APPROACH

IDENTIFYING TESTABLE FEATURES

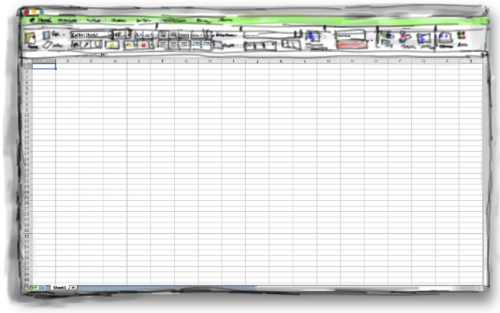
```
printSum(int a, int b)
```

How many indepentey testable features do we have here ?

[ ] 1
[ ] 2
[ ] 3
[ ] >3

IDENTIFYING TESTABLE FEATURES

Identify three possible independently testable features for a spreadsheet

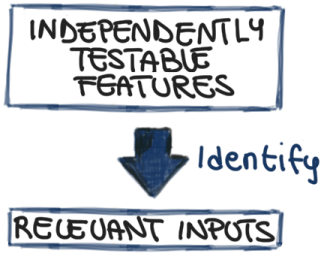


[ ]
[ ]
[ ]

A SYSTEMATIC FUNCTIONAL-TESTING APPROACH

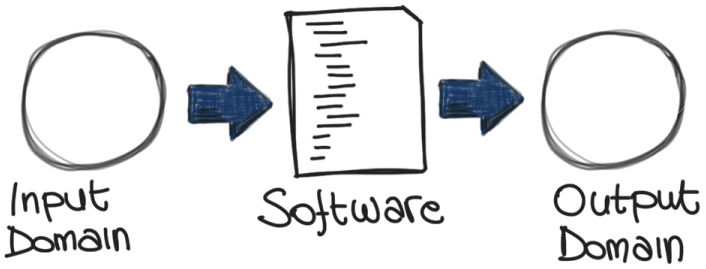
FUNCTIONAL SPECIFICATION → Identify → INDEPENDENTLY TESTABLE FEATURES

A SYSTEMATIC FUNCTIONAL-TESTING APPROACH

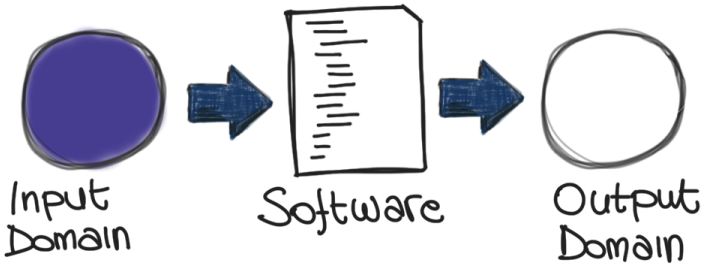


TEST DATA SELECTION

STRAW-MAN IDEA: EXHAUSTIVE TESTING!

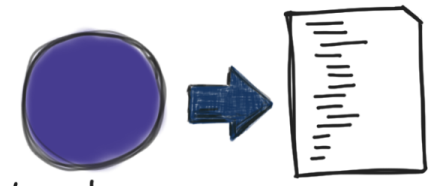


STRAW-MAN IDEA: EXHAUSTIVE TESTING!



Considers all possible inputs (executions)

STRAW-MAN IDEA: EXHAUSTIVE TESTING!



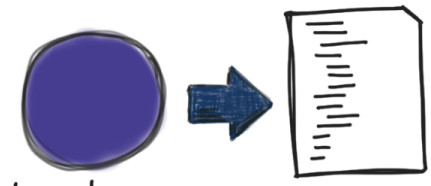
Input Domain

Software

How long would it take to exhaustively test the function `printSum(int a, int b)`?

[ ]

STRAW-MAN IDEA: EXHAUSTIVE TESTING!



Input Domain

Software

How long would it take to exhaustively test the function `printSum(int a, int b)`?

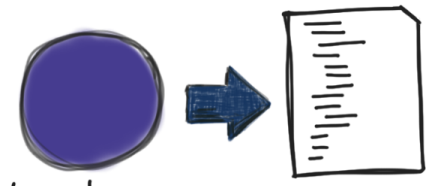
$2^{32} \times 2^{32} = 2^{64} \approx 10^{19}$ TESTS

1 TEST PER NANOSECOND (10^9 TESTS/SEC)

$\Rightarrow 10^{10}$ SECONDS OVERALL

[ ]

STRAW-MAN IDEA: EXHAUSTIVE TESTING !



[~600 YEARS]

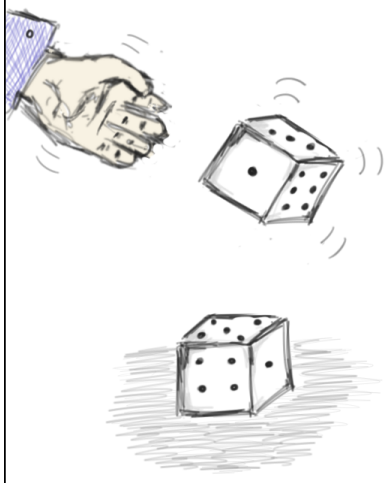
How long would it take to exhaustively test the function `printSum(int a, int b)`?

$2^{32} \times 2^{32} = 2^{64} \approx 10^{19}$ TESTS

1 TEST PER NANOSECOND (10^9 TESTS/SEC)

$\Rightarrow 10^{10}$ SECONDS OVERALL

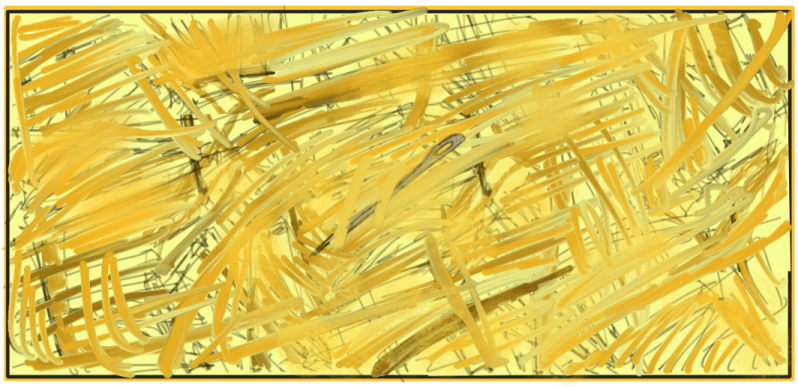
RANDOM TESTING



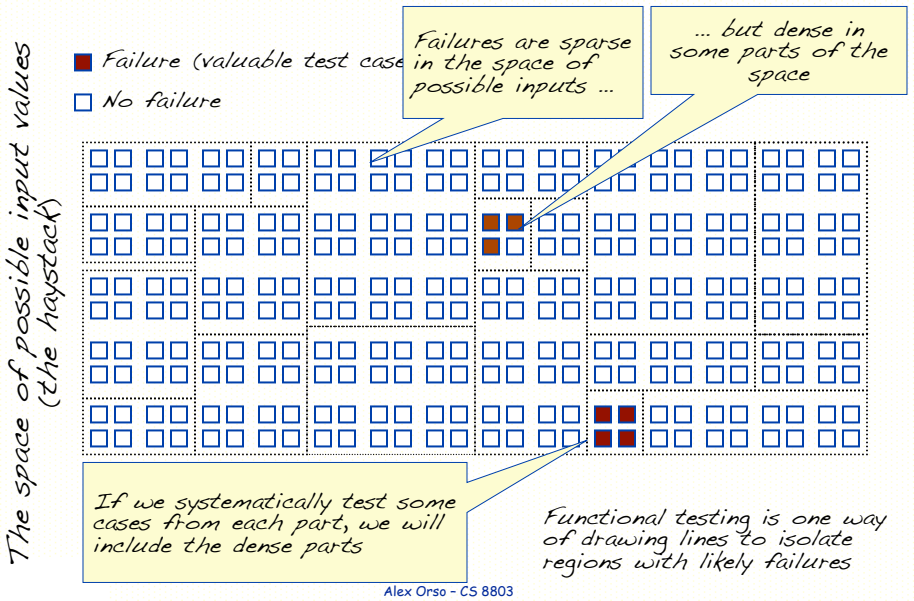
- pick inputs uniformly
- all inputs considered equal
- no designer bias

SO WHY NOT RANDOM ?

SO WHY NOT RANDOM ?



Systematic Partition Testing



EXAMPLE

```
Split (string str, int size)
```

Some possible partitions:

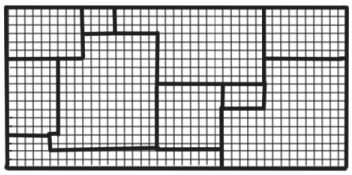
EXAMPLE

Split (string str, int size)

Some possible partitions:

- size < ∅
- size = ∅
- size > ∅
- str with length < size
- str with length in [size, size x 2]
- str with length > size x 2
- ...

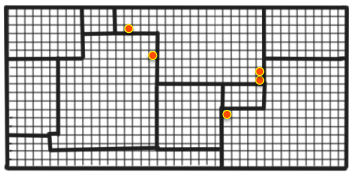
BOUNDARY VALUES



Basic idea

Errors tend to occur at the boundary of a (sub) domain

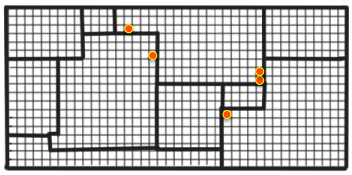
BOUNDARY VALUES



Basic idea

Errors tend to occur at the boundary of a (sub) domain

BOUNDARY VALUES



Basic idea

Errors tend to occur at the boundary of a (sub) domain

⇒ Select inputs at these boundaries

EXAMPLE

`Split (string str, int size)`

Some possible partitions:

- size < ∅
- size = ∅
- size > ∅
- str with length < size
- str with length in [size, size x 2]
- str with length > size x 2
- ...

EXAMPLE

`Split (string str, int size)`

Some possible partitions:

- size < 0
- size = 0
- size > 0
- str with length < size
- str with lenght in [size, size x 2]
- str with lenght > size x 2

Some possible inputs

EXAMPLE

Split (string str, int size)

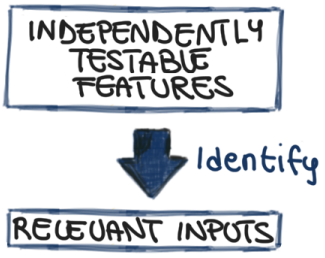
Some possible partitions:

- size < 0
- size = 0
- size > 0
- str with length < size
- str with lenght in [size, size x 2]
- str with lenght > size x 2

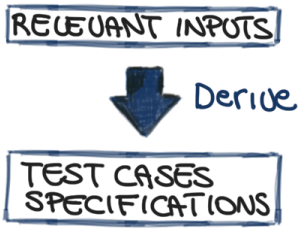
Some possible inputs

- size = -1
- size = 1
- size = MAXINT
- string with length size - 1
- string with length size
- ...

A SYSTEMATIC FUNCTIONAL-TESTING APPROACH



A SYSTEMATIC FUNCTIONAL-TESTING APPROACH



EXAMPLE

Split (string str, int size)

Some possible inputs

- size = -1
- size = 1
- size = MAXINT
- string with length size -1
- string with length size
- ...

EXAMPLE

```
Split (string str, int size)
```

Some possible inputs

- size = -1
- size = 1
- size = MAXINT

X

- string with length size -1
- string with length size
- ...

EXAMPLE

```
Split (string str, int size)
```

Some possible inputs

- size = -1
- size = 1
- size = MAXINT

X

- string with length size -1
- string with length size
- ...

Test case specifications

EXAMPLE

Split (string str, int size)

Some possible inputs

- size = -1
- size = 1
- size = MAXINT

X

- string with length size -1
- string with length size
- ...

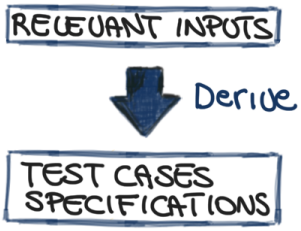
Test case specifications

~~- size = 1 str with length 2~~

~~- size = 1 str with length 1~~

- size = 1 str with length 0
- ...

A SYSTEMATIC FUNCTIONAL-TESTING APPROACH



A SYSTEMATIC FUNCTIONAL-TESTING APPROACH

Implement test cases in code

Requires building scaffolding

- Drivers
- Stubs

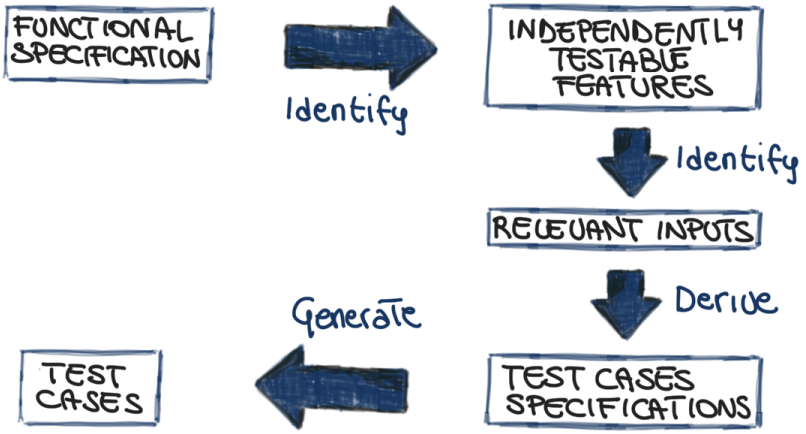
Structure of tests

- Set fixture
- Invoke
- Check
- Cleanup

Infrastructure for unit testing

- XUnit: automated unit testing frameworks
  - Available for multiple languages & environments
    - JUnit (Java)
    - cppUnit (C++)
    - nUnit (.NET languages)
    - dbUnit (database testing)
    - HTTPUnit (testing web sites)
    - PHPUnit (PHP)
    - ...
  - Acts as a driver
  - Automates test runs
  - Automates result checks

A SYSTEMATIC FUNCTIONAL-TESTING APPROACH



A SPECIFIC BLACK-BOX TESTING APPROACH

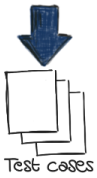
THE CATEGORY-PARTITION METHOD

[Ostrand & Balcer, CACM, June 1988]

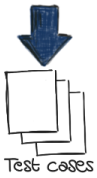
A SPECIFIC BLACK-BOX TESTING APPROACH

THE CATEGORY-PARTITION METHOD

[Ostrand & Balcer, CACM, June 1988]



1. Identify independently testable features
2. Identify categories
3. Partition categories into choices
4. Identify constraints among choices
5. Produce/Evaluate test case specifications
6. Generate test cases from test case specifications



IDENTIFY CATEGORIES

Characteristics of each input element

Example: split(string str, int size)

IDENTIFY CATEGORIES

Characteristics of each input element

Example: split(string str, int size)

Input str — Input size

IDENTIFY CATEGORIES

Characteristics of each input element

Example: split(string str, int size)

Input str
- lenght

- content

Input size
- value

PARTITION CATEGORIES INTO CHOICES

Interesting cases (subdomains)

Example: split(string str, int size)

Input str
- lenght

- content

Input size
- value

PARTITION CATEGORIES INTO CHOICES

Interesting cases (subdomains)

Example: split(string str, int size)

Input str
- lenght
  - 0
  - size - 1
  - ...
- content

Input size
- value

PARTITION CATEGORIES INTO CHOICES

Interesting cases (subdomains)

Example: split(string str, int size)

Input str
- lenght
  - 0
  - size - 1
  - ...
- content
  - spaces
  - special characters
  - ...

Input size
- value

Alex Orso - CS 8803

PARTITION CATEGORIES INTO CHOICES

Interesting cases (subdomains)

Example: split(string str, int size)

Input str
- lenght
  - 0
  - size - 1
  - ...
- content
  - spaces
  - special characters
  - ...

Input size
- value
  - 0
  - >0
  - <0
  - MAXINT
  - ...

IDENTIFY CONSTRAINTS AMONG CHOICES

To eliminate meaningless combinations

To reduce the number of test cases

IDENTIFY CONSTRAINTS AMONG CHOICES

To eliminate meaningless combinations

To reduce the number of test cases

Three types: PROPERTY ... IF, ERROR, SINGLE

IDENTIFY CONSTRAINTS AMONG CHOICES

To eliminate meaningless combinations

To reduce the number of test cases

Three types: PROPERTY ... IF, ERROR, SINGLE

Examples

Input str
- lenght
  - 0
- content
  - special characters

Input size
- value
  - <0
  - MAXINT

IDENTIFY CONSTRAINTS AMONG CHOICES

To eliminate meaningless combinations

To reduce the number of test cases

Three types: PROPERTY ... IF, ERROR, SINGLE

Examples

Input str
- lenght
  - 0 PROPERTY zerovalue
- content
  - special characters

Input size
- value
  - <0
  - MAXINT

IDENTIFY CONSTRAINTS AMONG CHOICES

To eliminate meaningless combinations

To reduce the number of test cases

Three types: PROPERTY ... IF, ERROR, SINGLE

Examples

Input str
- lenght
  - 0 PROPERTY zerovalue
- content
  - special characters if !zerovalue

Input size
- value
  - <0
  - MAXINT

IDENTIFY CONSTRAINTS AMONG CHOICES

To eliminate meaningless combinations

To reduce the number of test cases

Three types: PROPERTY ... IF, ERROR, SINGLE

Examples

Input str
- lenght
  - 0 PROPERTY zerovalue
- content
  - special characters if !zerovalue

Input size
- value
  - <0
  - MAXINT

IDENTIFY CONSTRAINTS AMONG CHOICES

To eliminate meaningless combinations

To reduce the number of test cases

Three types: PROPERTY ... IF, ERROR, SINGLE

Examples

Input str
- lenght
  - 0 PROPERTY zerovalue
- content
  - special characters if !zerovalue

Input size
- value
  - <0 ERROR
  - MAXINT SINGLE

Alex Orso - CS 8803

PRODUCE AND EVALUATE TEST CASE SPECIFICATIONS

Can be automated

Produces test frames

PRODUCE AND EVALUATE TEST CASE SPECIFICATIONS

Can be automated

Produces test frames

Example

Test frame #36
- input str
  - lenght : size - 1
  - content : special characters
- input size
  - value : > 0

GENERATE TEST CASES FROM TEST CASE SPECIFICATIONS

Simple instantiation of frames

Final result : set of concrete tests

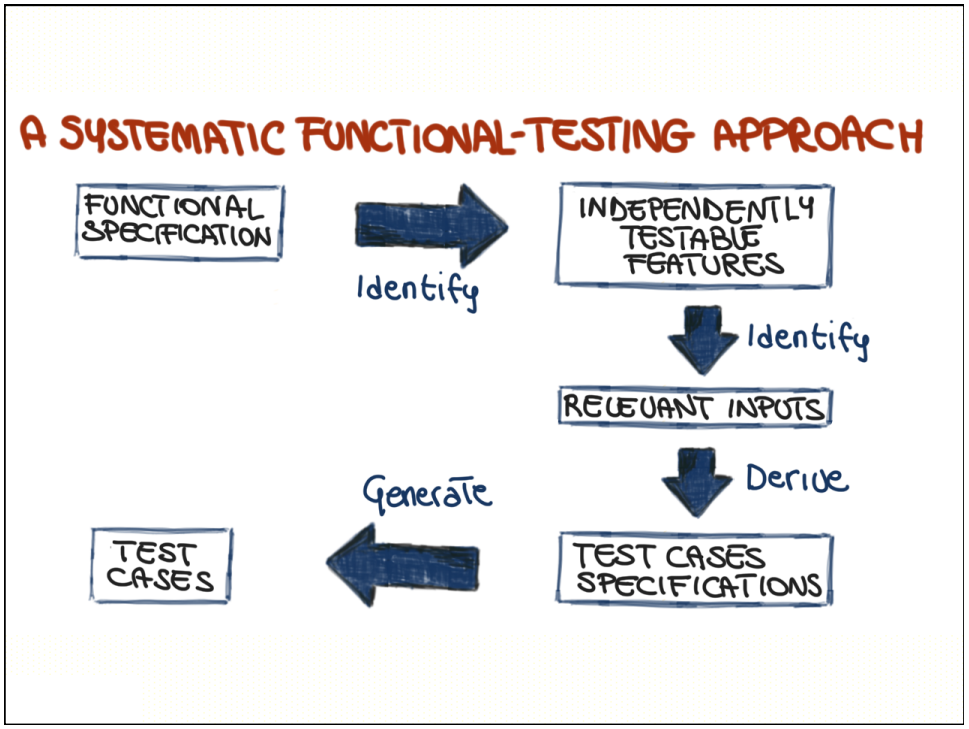
GENERATE TEST CASES FROM TEST CASE SPECIFICATIONS

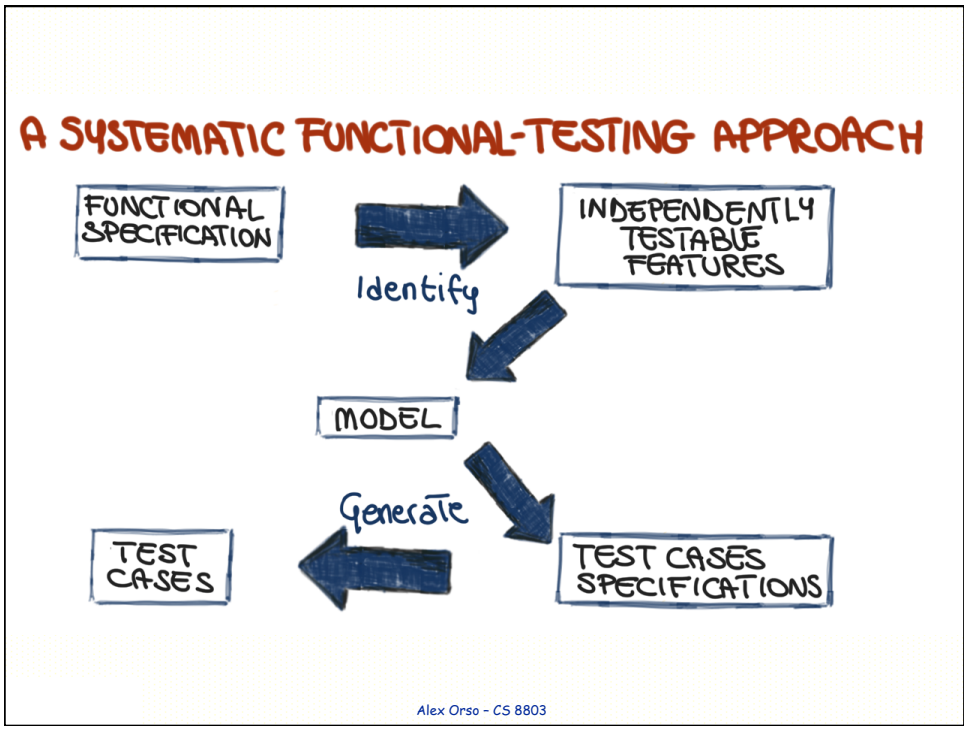
Simple instantiation of frames

Final result : set of concrete tests

Example

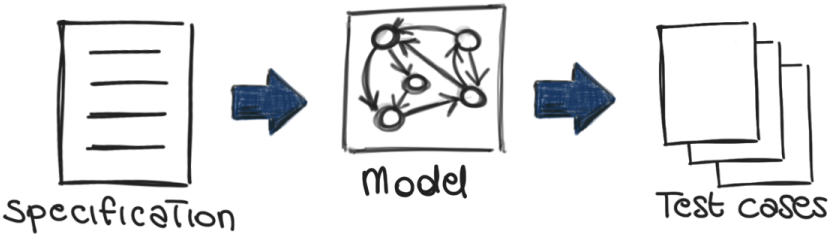
Test case #36
str = "ABCC!\n\t∅"
size = 10


A SYSTEMATIC FUNCTIONAL-TESTING APPROACH
FUNCTIONAL SPECIFICATION
Identify
INDEPENDENTLY TESTABLE FEATURES
Identify
RELEVANT INPUTS
Derive
TEST CASES SPECIFICATIONS
Generate
TEST CASES


A SYSTEMATIC FUNCTIONAL-TESTING APPROACH
FUNCTIONAL SPECIFICATION
Identify
INDEPENDENTLY TESTABLE FEATURES
MODEL
Generate
TEST CASES SPECIFICATIONS
TEST CASES
Alex Orso - CS 8803

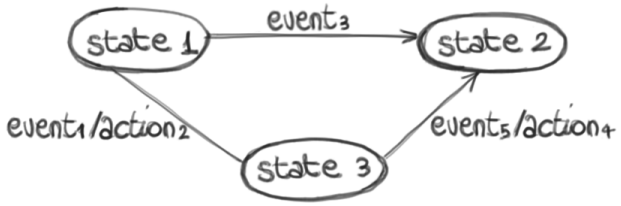
MODEL-BASED TESTING

MODEL-BASED TESTING



FINITE STATE MACHINES (FSM)

Nodes = states
Edges = transitions
Edge labels = events/actions

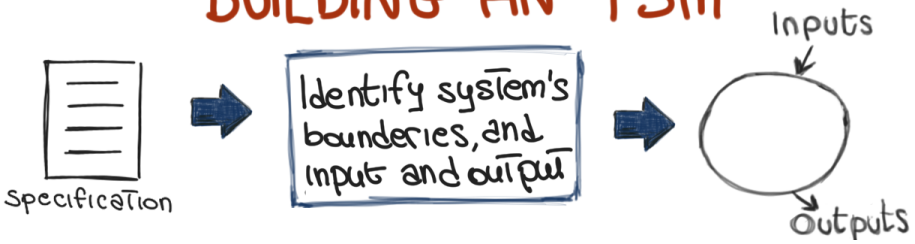


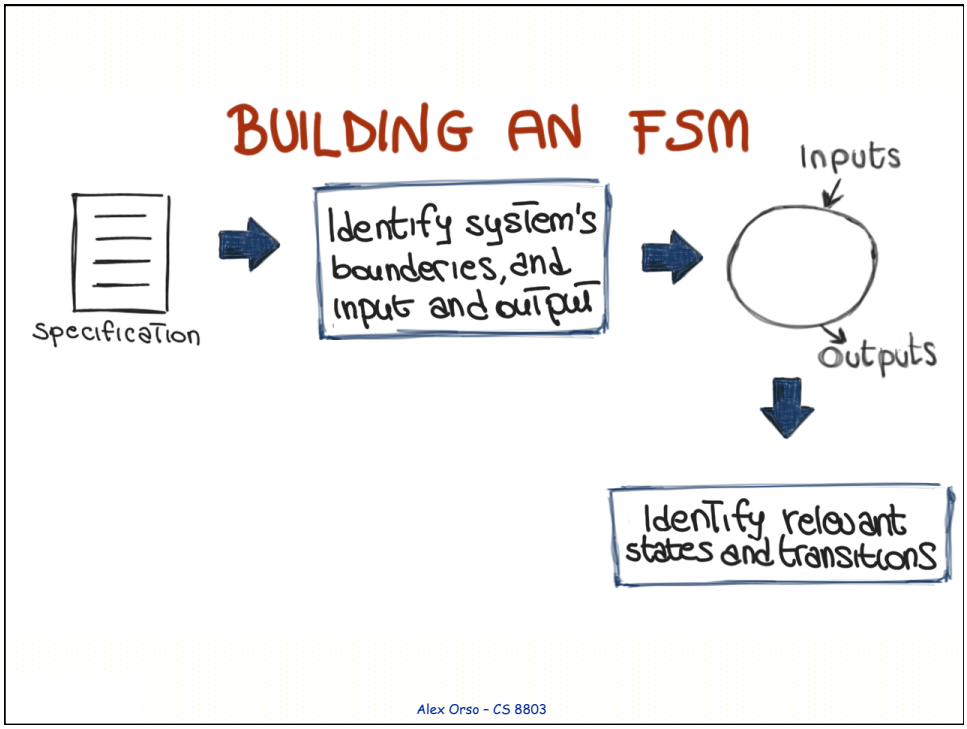
BUILDING AN FSM

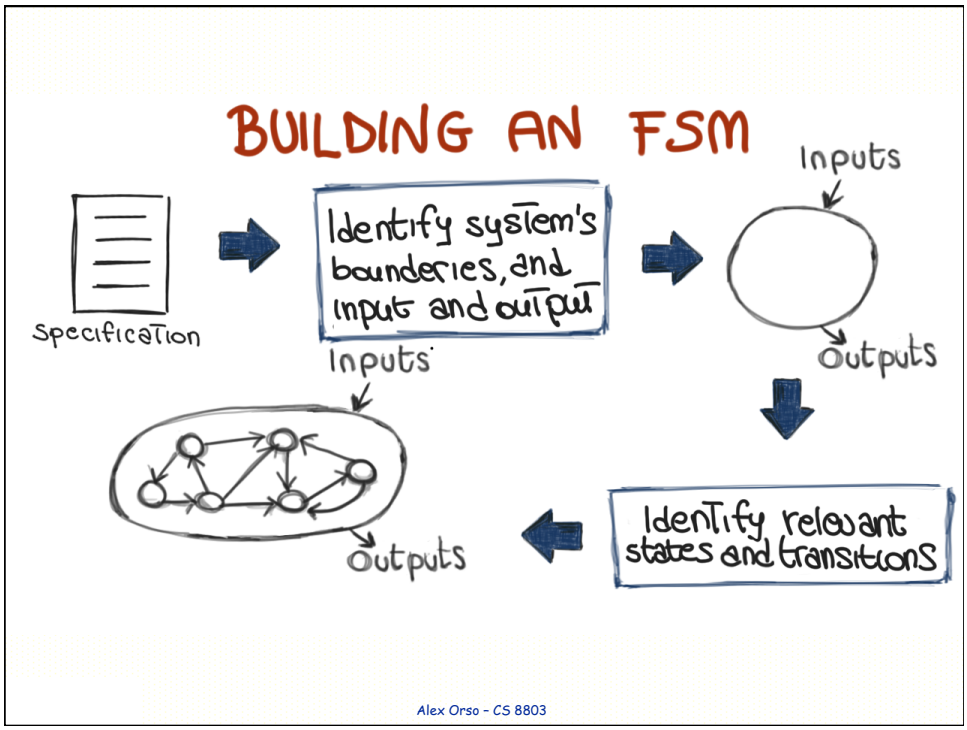
Specification

BUILDING AN FSM

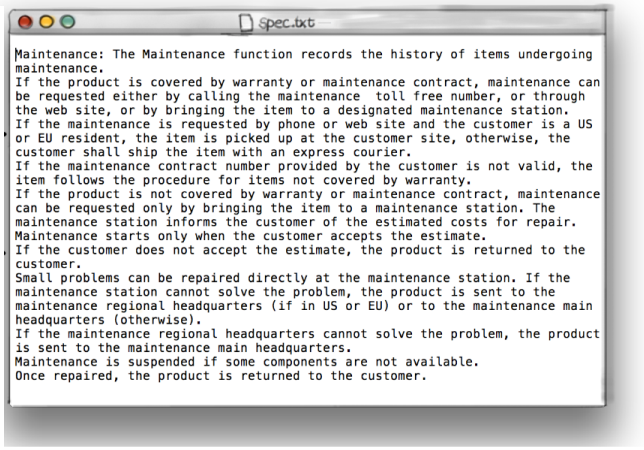
BUILDING AN FSM




BUILDING AN FSM
Specification
Identify system's boundaries, and input and output
Inputs
Outputs
Identify relevant states and transitions
Alex Orso - CS 8803

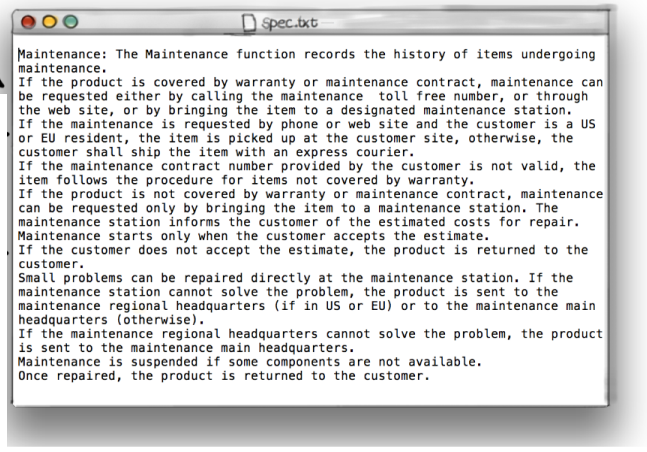

BUILDING AN FSM
Specification
Identify system's boundaries, and input and output
Inputs
Outputs
Inputs
Outputs
Identify relevant states and transitions
Alex Orso - CS 8803

FROM AN INFORMAL SPECIFICATION...



FROM AN INFORMAL SPECIFICATION...

Multiple choices here



FROM AN INFORMAL SPECIFICATION...

Multiple choices here

Determine the next step

spec.txt

```
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.
```


FROM AN INFORMAL SPECIFICATION...

Multiple choices here

Determine the next step

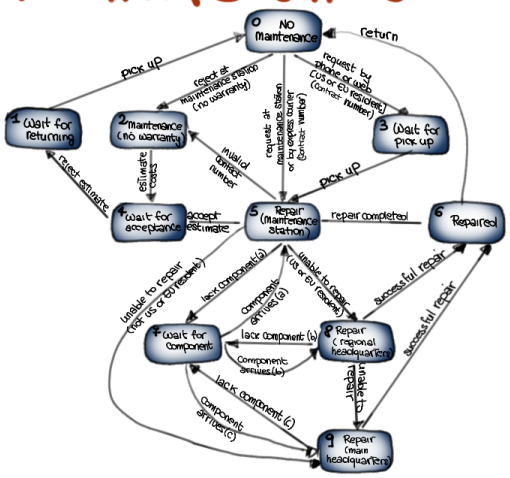
and so on

spec.txt

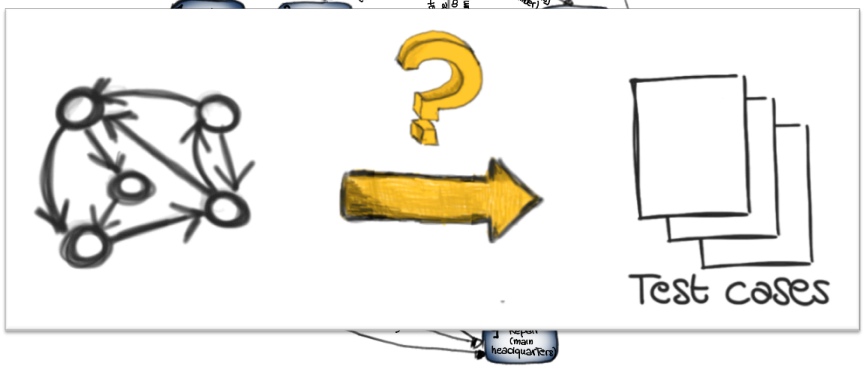
```
Maintenance: The Maintenance function records the history of items undergoing
maintenance.
If the product is covered by warranty or maintenance contract, maintenance can
be requested either by calling the maintenance  toll free number, or through
the web site, or by bringing the item to a designated 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.
```

Alex Orso - CS 8803

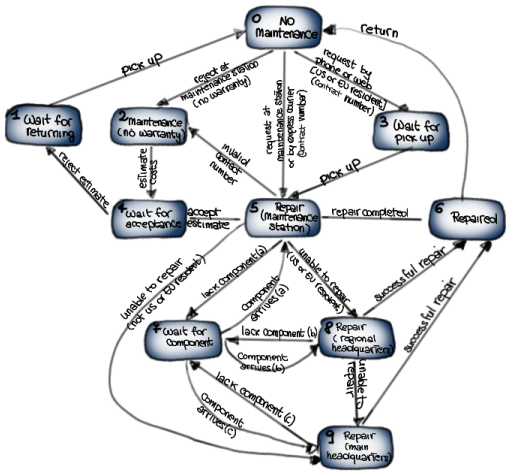
... TO A FINITE STATE MACHINE



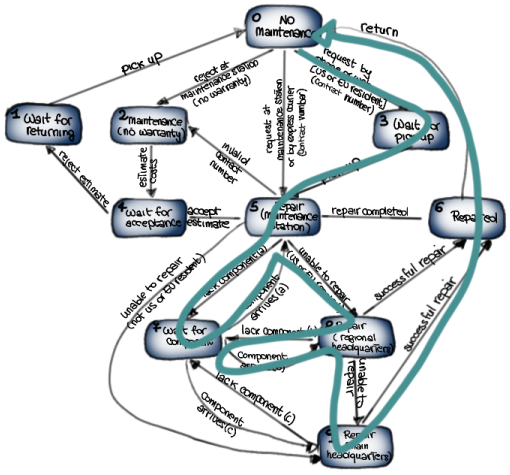
... TO A FINITE STATE MACHINE



TO A SET OF TEST CASES

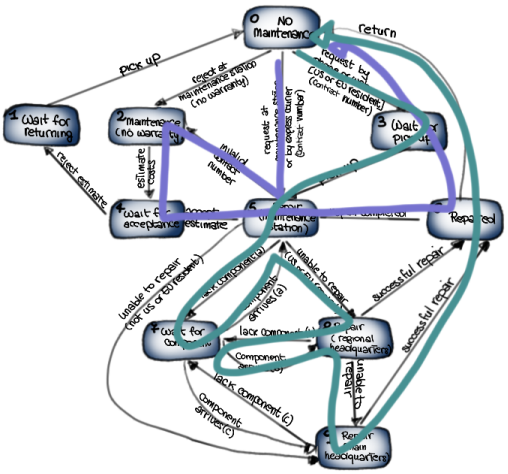


TO A SET OF TEST CASES

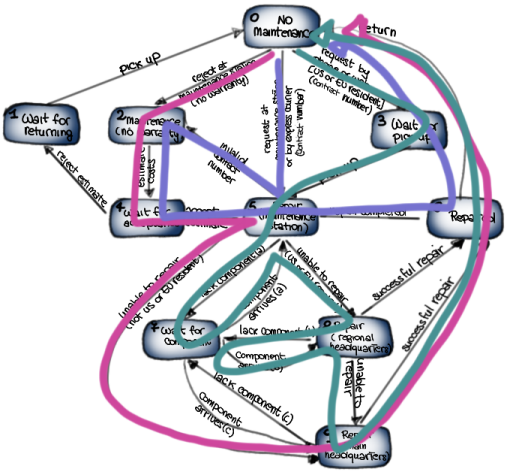


Alex Orso – CS 8803

TO A SET OF TEST CASES

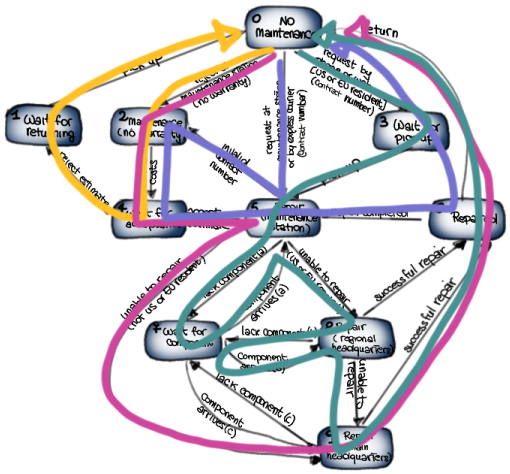


TO A SET OF TEST CASES

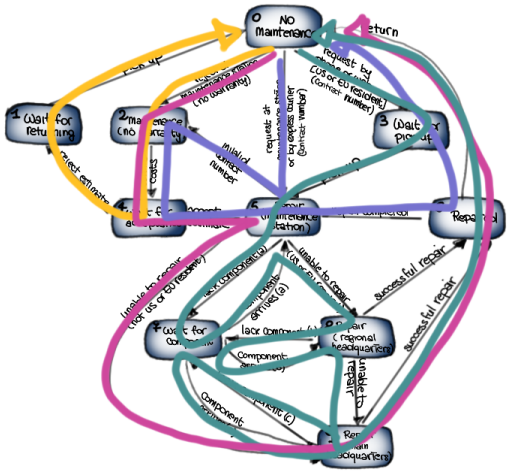


Alex Orso – CS 8803

TO A SET OF TEST CASES

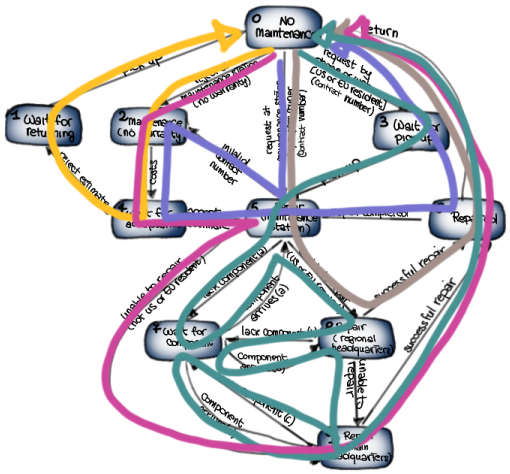


TO A SET OF TEST CASES

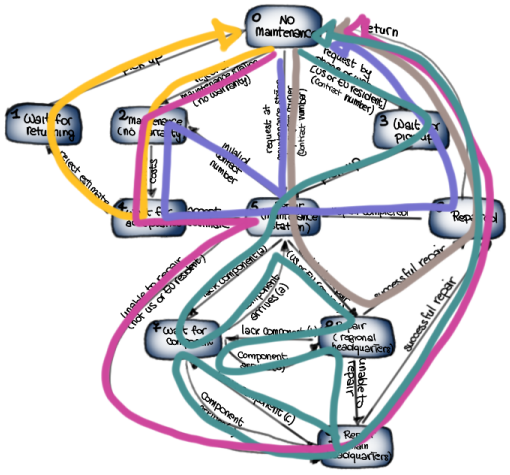


Alex Orso – CS 8803

TO A SET OF TEST CASES



TO A SET OF TEST CASES



TC1: ∅, 3, 5, 7, 5, 8
7, 8, 9, 7, 9, 6, ∅

TC2: ∅, 5, 2, 4, 5, 6, ∅

TC3: ∅, 2, 4, 1, ∅

TC4: ∅, 4, 5, 9, 6, ∅

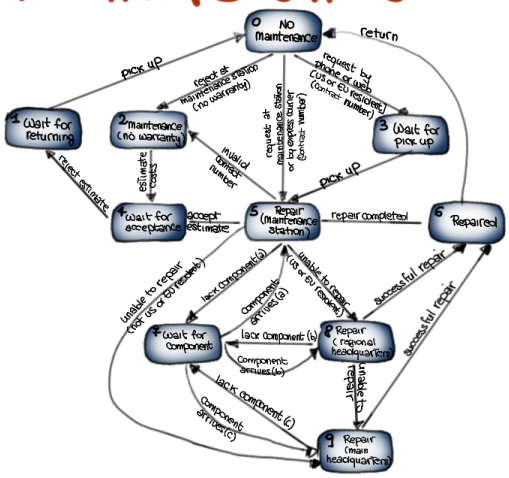
TC5: ∅, 5, 6, ∅

FROM AN INFORMAL SPECIFICATION...

spec.txt

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.

... TO A FINITE STATE MACHINE



SOME CONSIDERATIONS

SOME CONSIDERATIONS

Applicability

- very general approach
- in UML, state machine are readily available

SOME CONSIDERATIONS

Applicability

- very general approach
- in UML, state machine are readily available

Abstraction is key

SOME CONSIDERATIONS

Applicability

- very general approach
- in UML, state machine are readily available

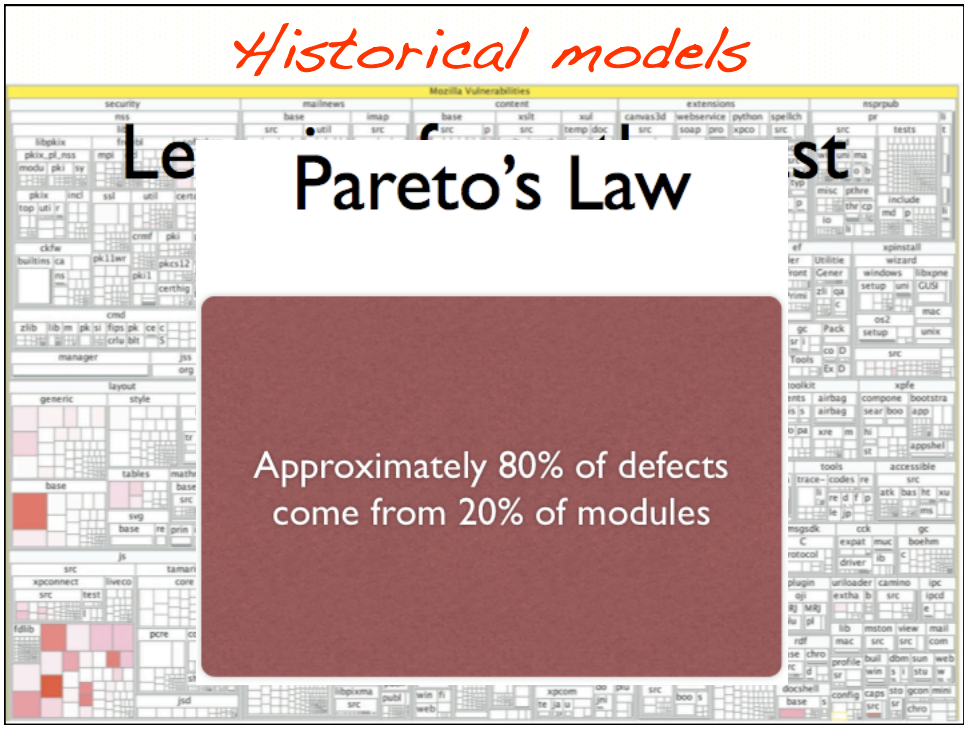
Abstraction is key

Many other approaches

- decision tables
- flow graphs
- historical models

...

Historical models



A SYSTEMATIC FUNCTIONAL-TESTING APPROACH