CS 3300
Intro to Software Engineering

SOFTWARE TESTING

GENERAL CONCEPTS

Mahdi Roozbahani

Slides are based on Alex Orso.

Software is Buggy!

- On average, 1-5 errors per 1KLOC
- Windows 2000
  - 35M LOC
  - 63,000 known bugs at the time of release
  - 2 per 1,000 lines

- For mass market software 100% correct is infeasible, but
- We must verify the SW as much as possible

Failure, Fault, Error

Failure

Observable incorrect behavior of a program. Conceptually related to the behavior of the program, rather than its code.

Fault (bug)

Related to the code. Necessary (not sufficient!) condition for the occurrence of a failure.

Error

Cause of a fault. Usually a human error (conceptual, typo, etc.)

Failure, Fault, Error: Example

```
1.  double doubleValue(int param) {
2.      double result;
3.      result = (double) param * param;
4.      return(result);
5.  }
```

A call to double(3) returns 9

Result 9 represents a failure

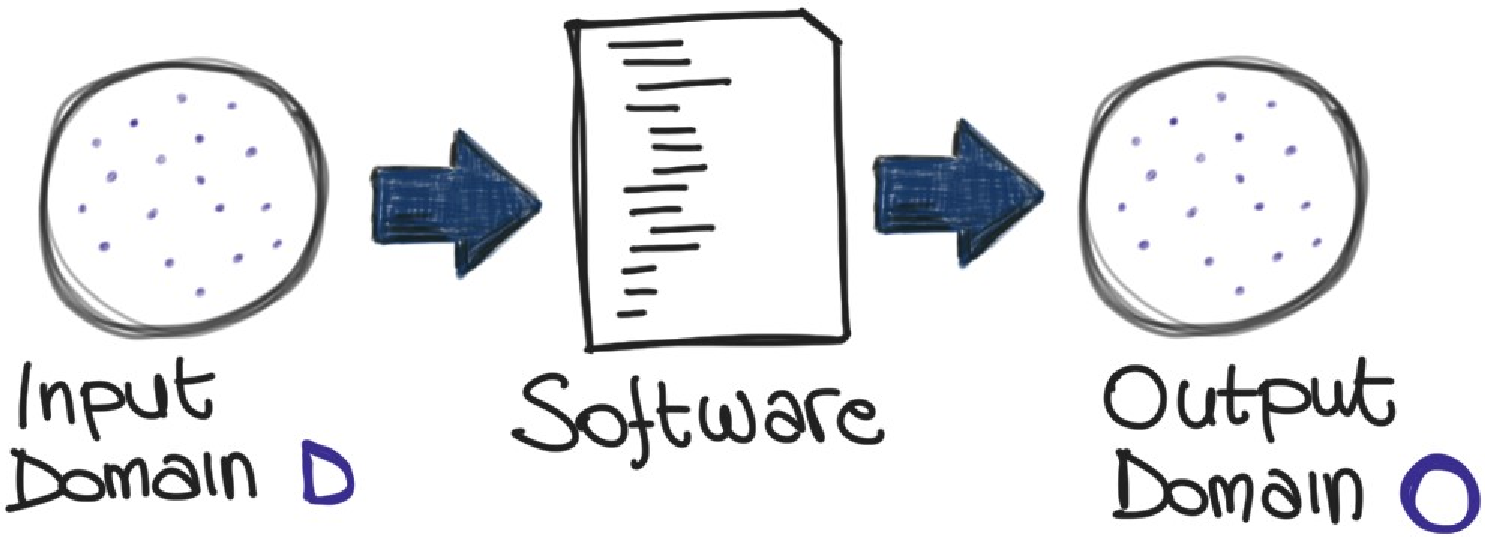
Such failure is due to the fault at line 3

The error is a typo (hopefully)

Approaches to Verification

- Testing (dynamic verification): exercising software to try and generate failures
- Static analysis: identify (specific) problems statically, that is, considering all possible executions
- Inspections/reviews/walkthroughs: systematic group review of program text to detect faults
- Formal verification (proof of correctness): proving that the program implements the program specification

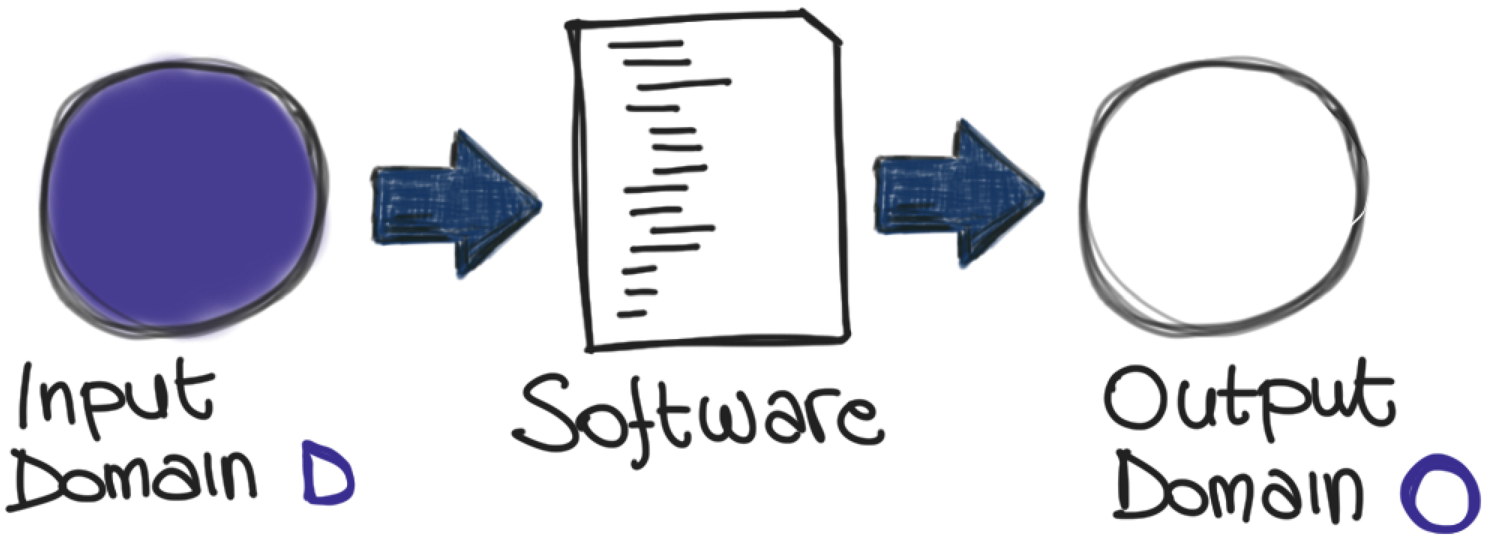
Testing (dynamic verification)



Test case : {i∈D , o∈O}

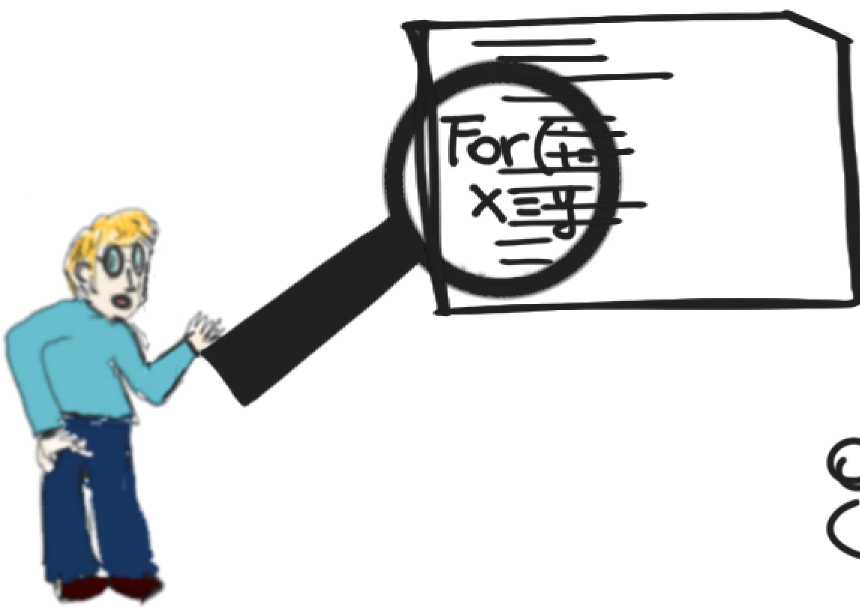
Test Suite : set of test cases

Static analysis



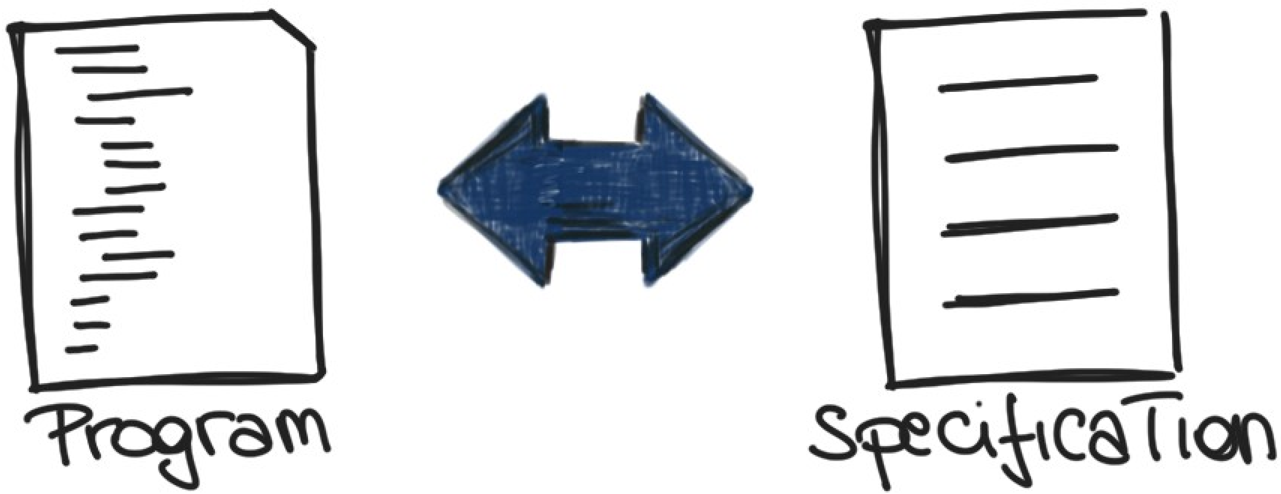
Considers all possible inputs (executions)

Inspections (AKA
- reviews
- walkthroughs)



Manual group activity

Formal verification
(proof of correctness)



Given a formal specification, checks that the code corresponds to such specification

Comparison

Testing

- Pros: no false positives
- Limits: incomplete

Static analysis

- P: complete (consider all program behaviors)
- L: false positives, expensive

Inspections

- P: systematic, thorough
- L: informal, subjective

Formal verification

- P: strong guarantees
- L: complex, expensive (requires a spec)

TODAY, QA IS MOSTLY TESTING

"50% of my company employees are Testers, and The rest spends 50% of Their Time Testing"

who said That?

TODAY, QA IS MOSTLY TESTING

"50% of my company employees are Testers, and The rest spends 50% of Their Time Testing"

who said That?

- [ ] Yogi Berra
- [ ] Steve Jobs
- [ ] Henry Ford
- [ ] Bill Gates
- [ ] Frank Gehry

What is Testing?

Testing == To execute a program with a sample of the input data

- Dynamic technique: program must be executed
- Optimistic approximation:
  - The program under test is exercised with a (very small) subset of all the possible input data
  - We **assume** that the behavior with any other input is consistent with the behavior shown for the selected subset of input data

Testing Techniques

There are a number of techniques

- Different processes
- Different artifacts
- Different approaches

There are no perfect techniques

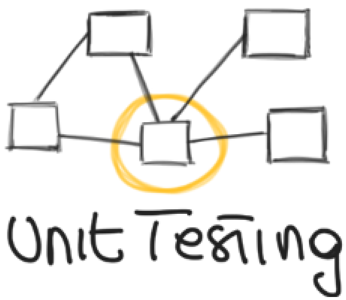
- Testing is a best-effort activity

There is no <u>best</u> technique

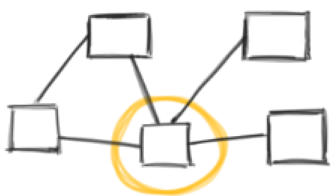
- Different contexts
- Complementary strengths and weaknesses
- Trade-offs

Testing Granularity Levels

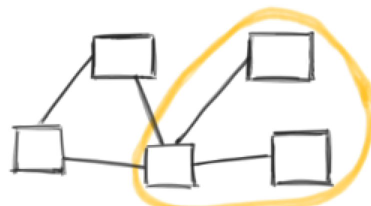
Testing Granularity Levels



TESTING GRANULARITY LEVELS

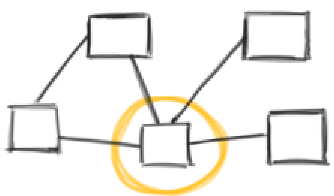


Unit Testing

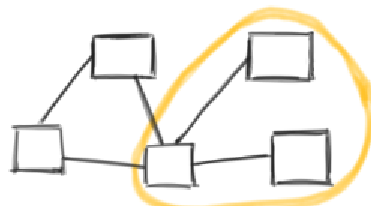


Integration Testing

TESTING GRANULARITY LEVELS



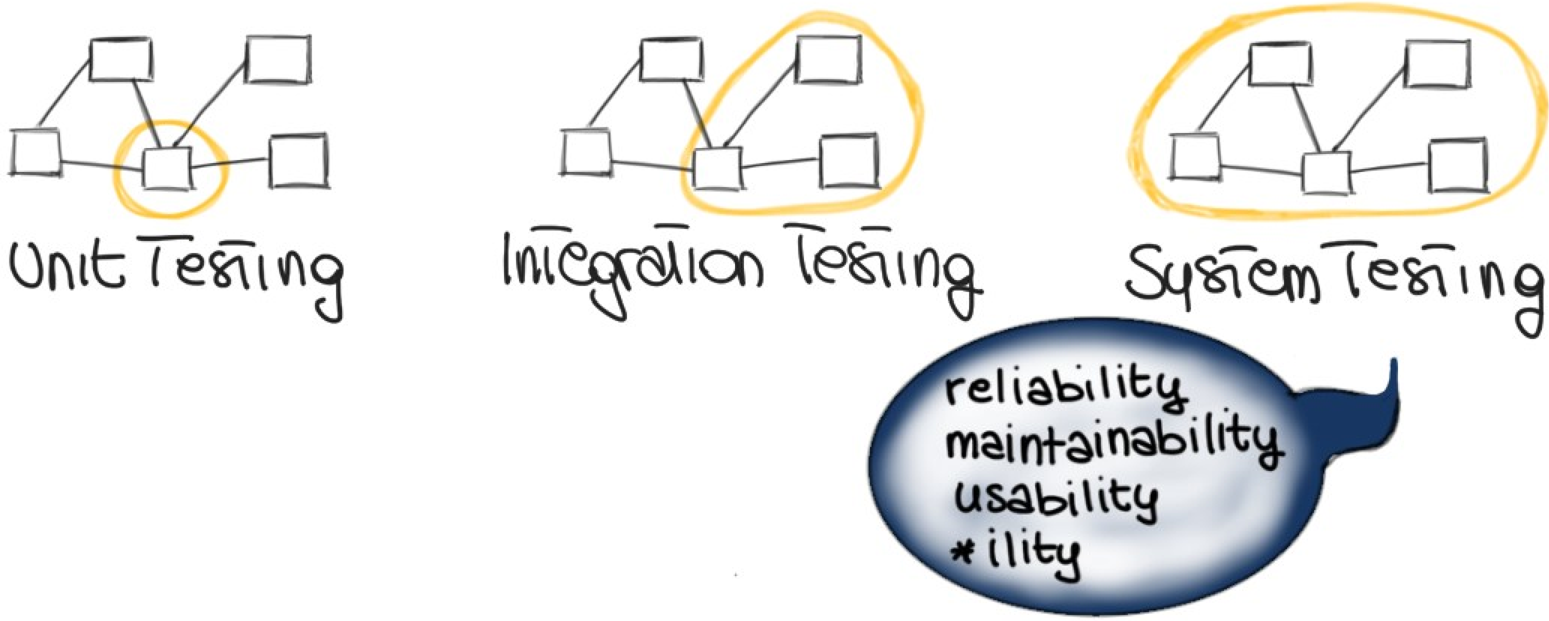
Unit Testing



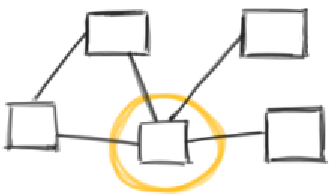
Big Bang

Integration Testing

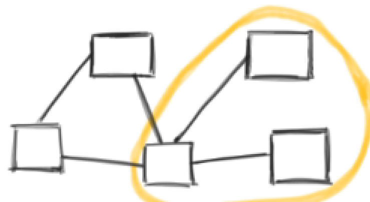
TESTING GRANULARITY LEVELS



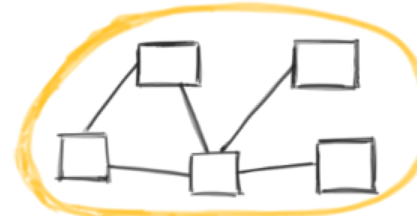
Testing Granularity Levels



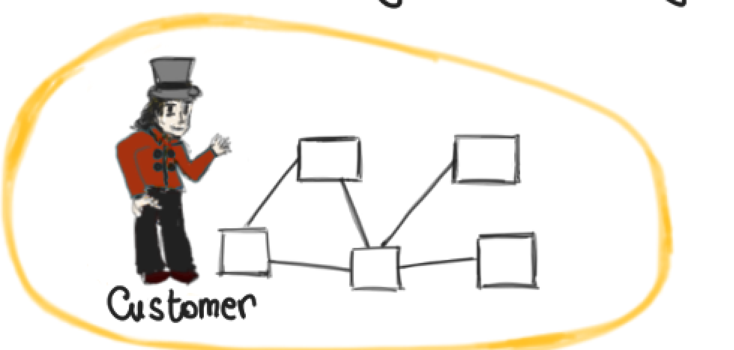
Unit Testing



Integration Testing

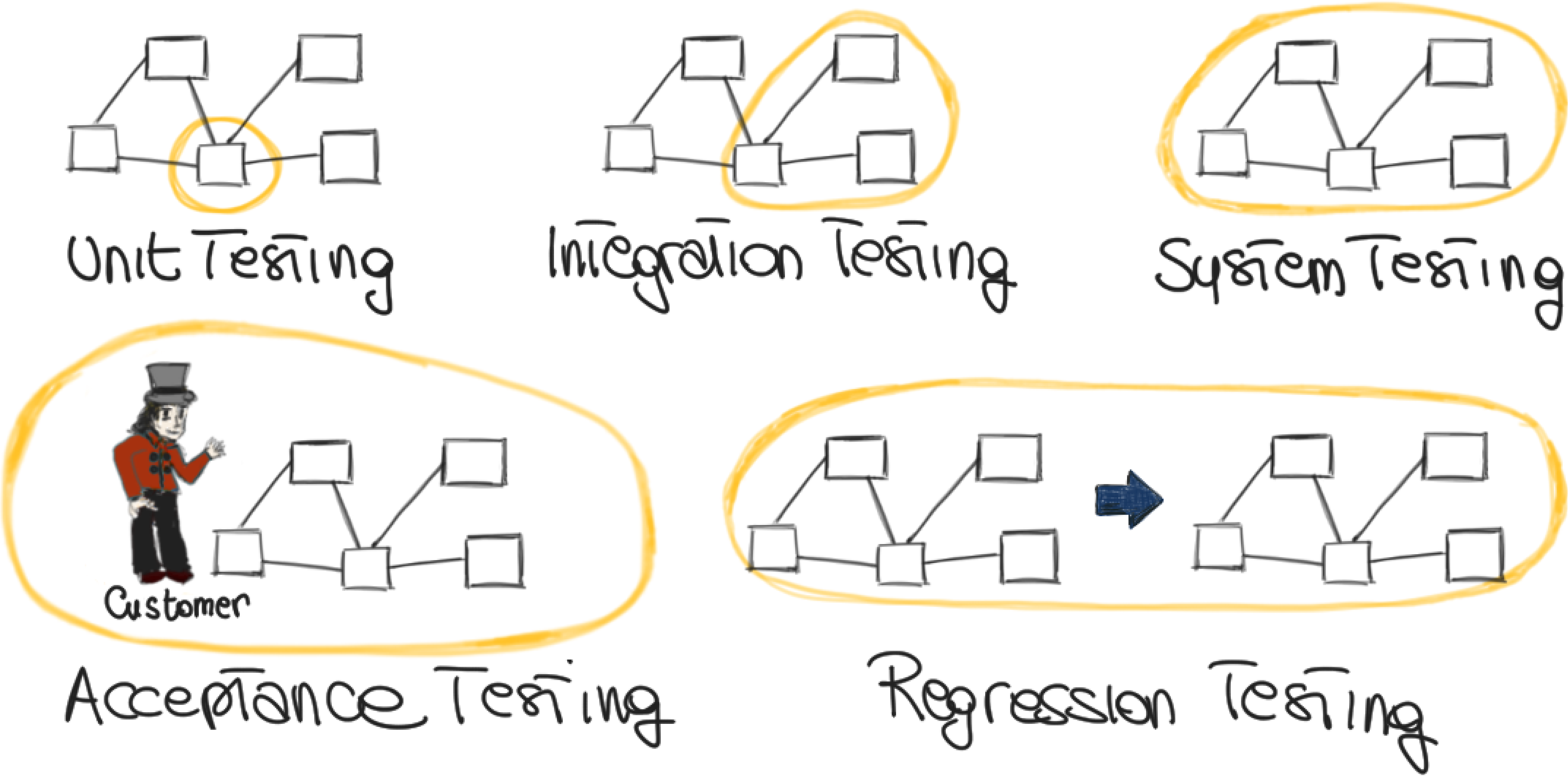


System Testing



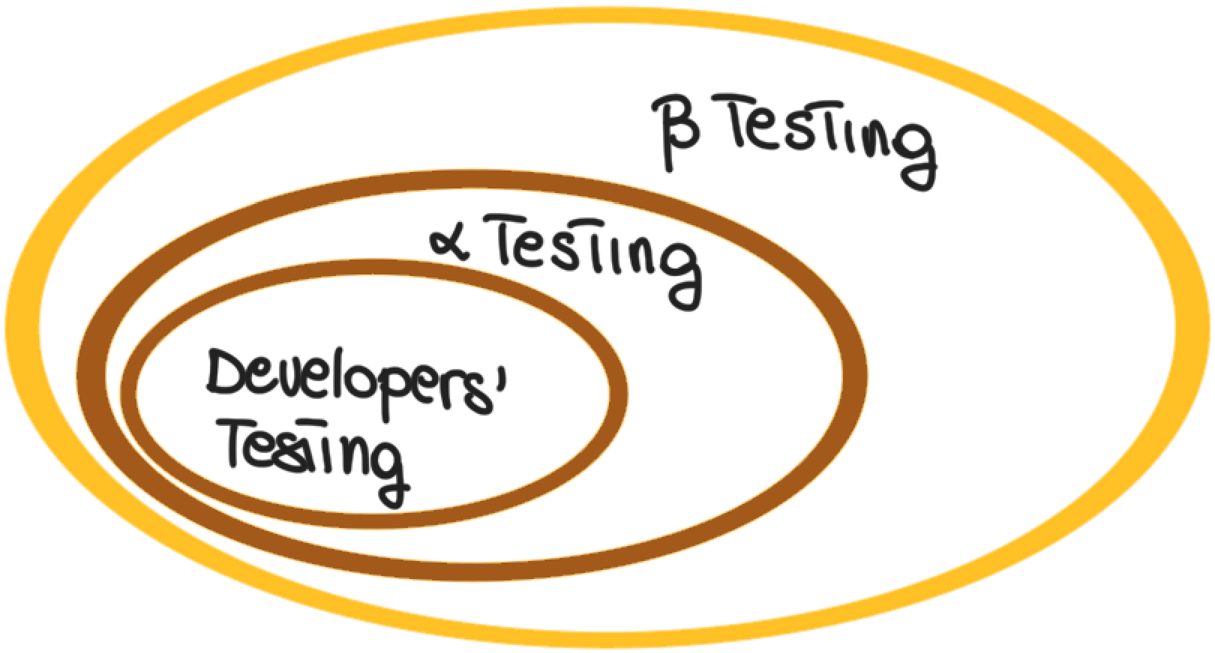
Acceptance Testing

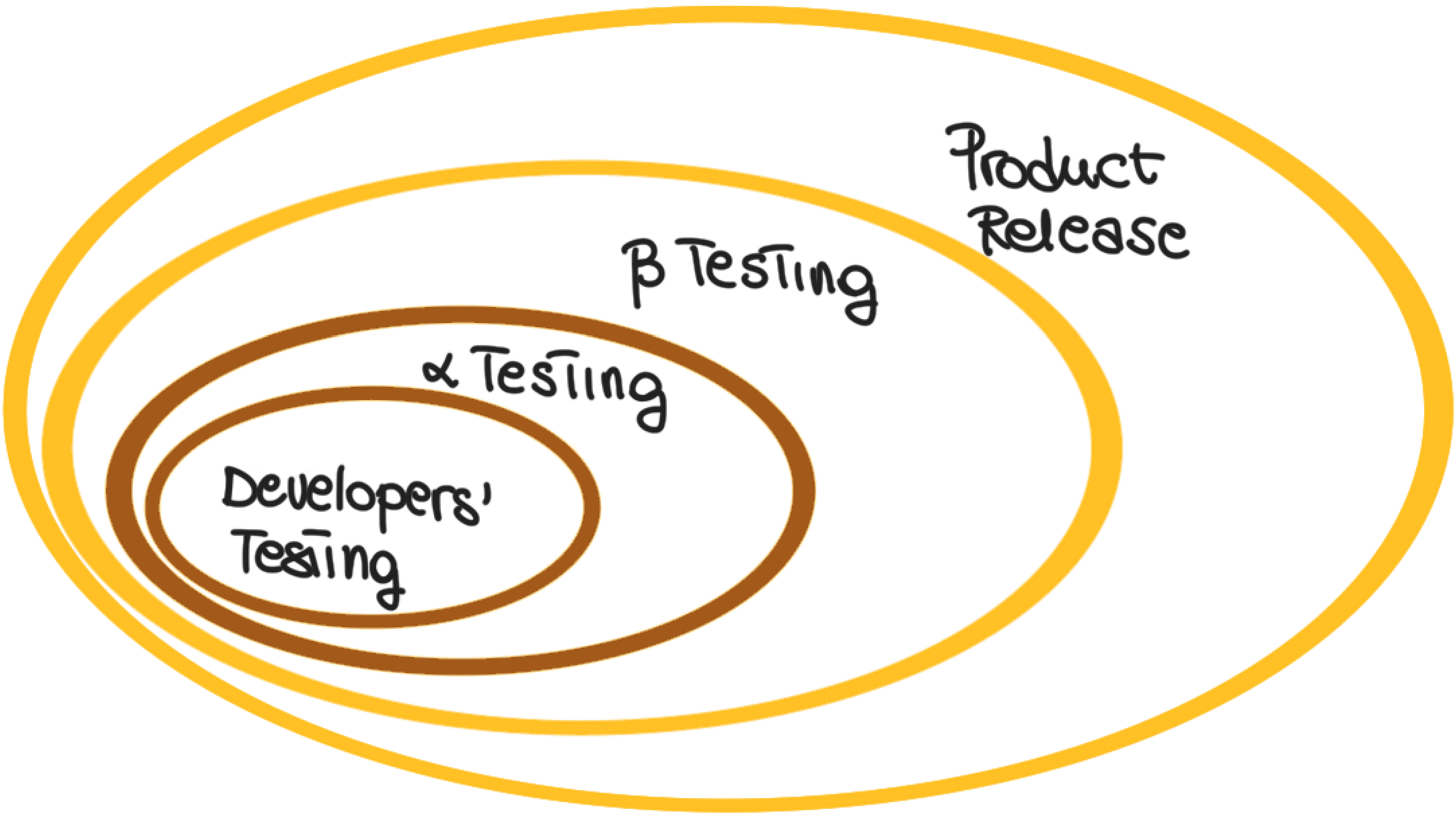
TESTING GRANULARITY LEVELS

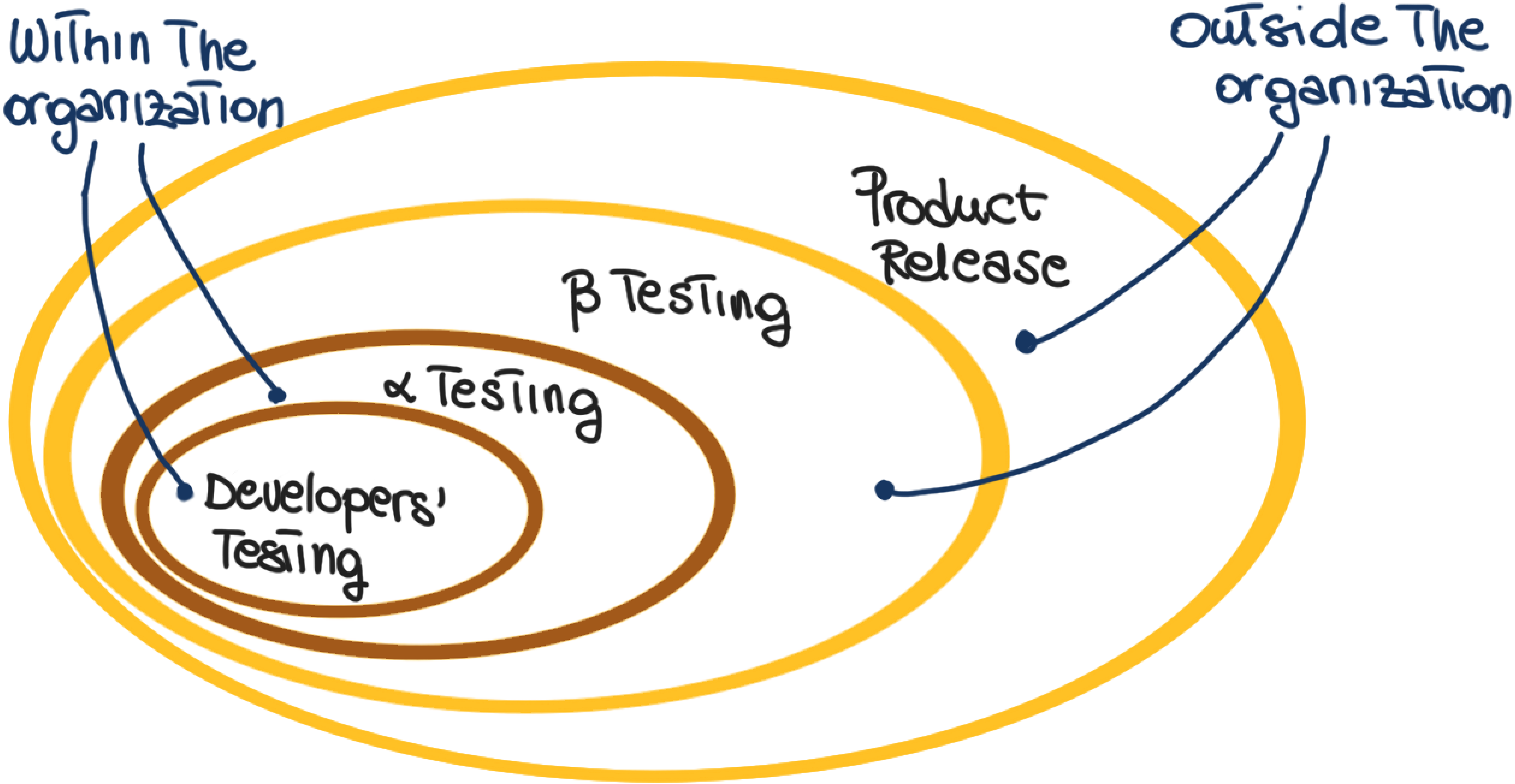



Developers' Testing


α Testing
Developers' Testing

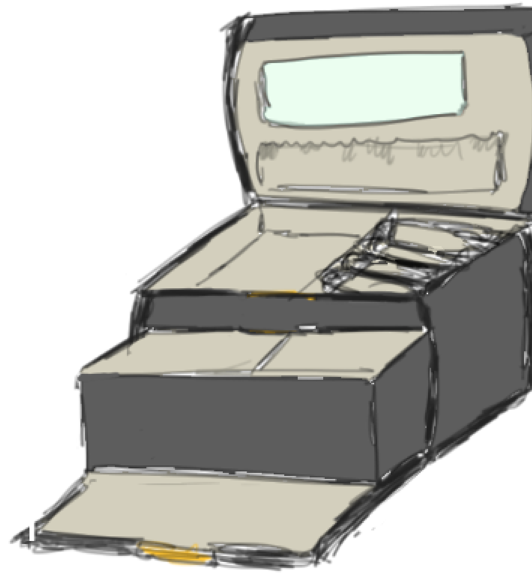

β Testing
α Testing
Developers' Testing


Product
Release
β Testing
α Testing
Developers'
Testing

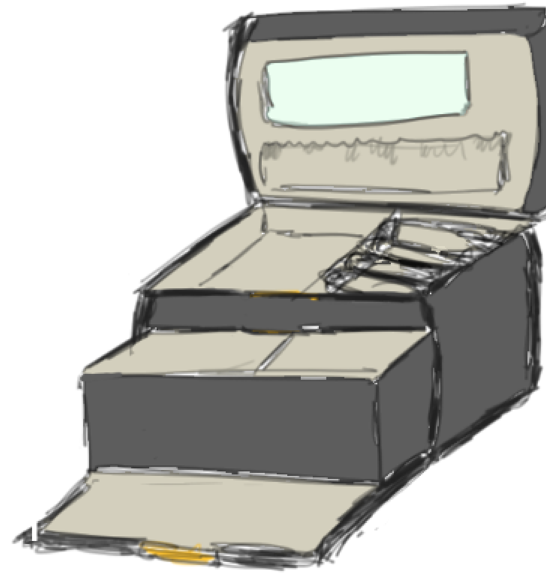

Within The organization
Outside The organization
Product Release
β Testing
α Testing
Developers' Testing

Functional vs. Structural Testing

BLACK-BOX TESTING



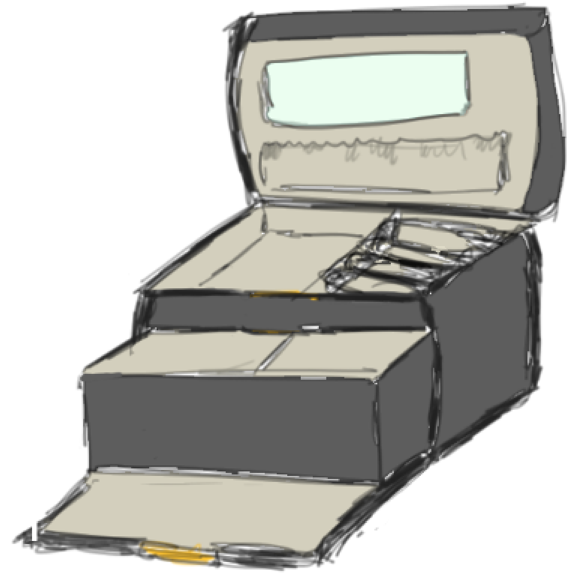
WHITE-BOX TESTING



BLACK-BOX TESTING

WHITE-BOX TESTING

- based on a description of the software (specification)
- cover as much specified behavior as possible
- cannot reveal errors due to implementation details



BLACK-BOX TESTING

- based on a description of the software (specification)
- cover as much specified behavior as possible
- cannot reveal errors due to implementation details

WHITE-BOX TESTING

- based on the code
- cover as much coded behavior as possible
- cannot reveal errors due to missing paths

BLACK-BOX TESTING EXAMPLE

Specification: inputs an integer and prints it

BLACK-BOX TESTING EXAMPLE

Specification: inputs an integer and prints it

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

WHITE-BOX TESTING EXAMPLE

```
int fun(int param){
    int result;
    result = param/2;
    return result;
}
```

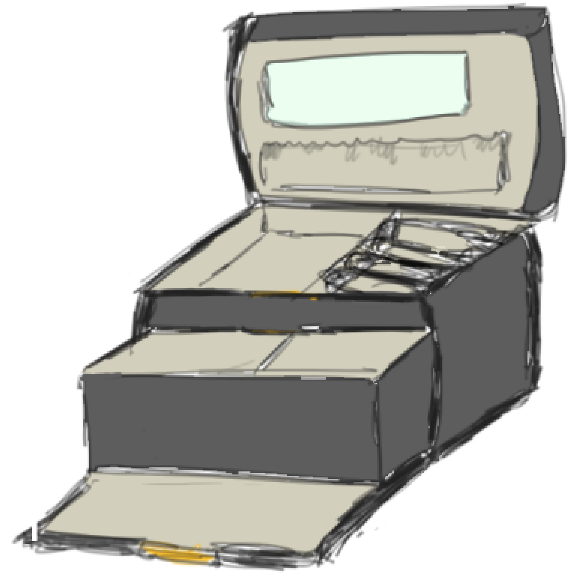
WHITE-BOX TESTING EXAMPLE

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

```
int fun(int param){
    int result;
    result = param/2;
    return result;
}
```

Software Testing

General Concepts



BLACK-BOX TESTING

- based on a description of the software (specification)
- cover as much specified behavior as possible
- cannot reveal errors due to implementation details

WHITE-BOX TESTING

- based on the code
- cover as much coded behavior as possible
- cannot reveal errors due to missing paths

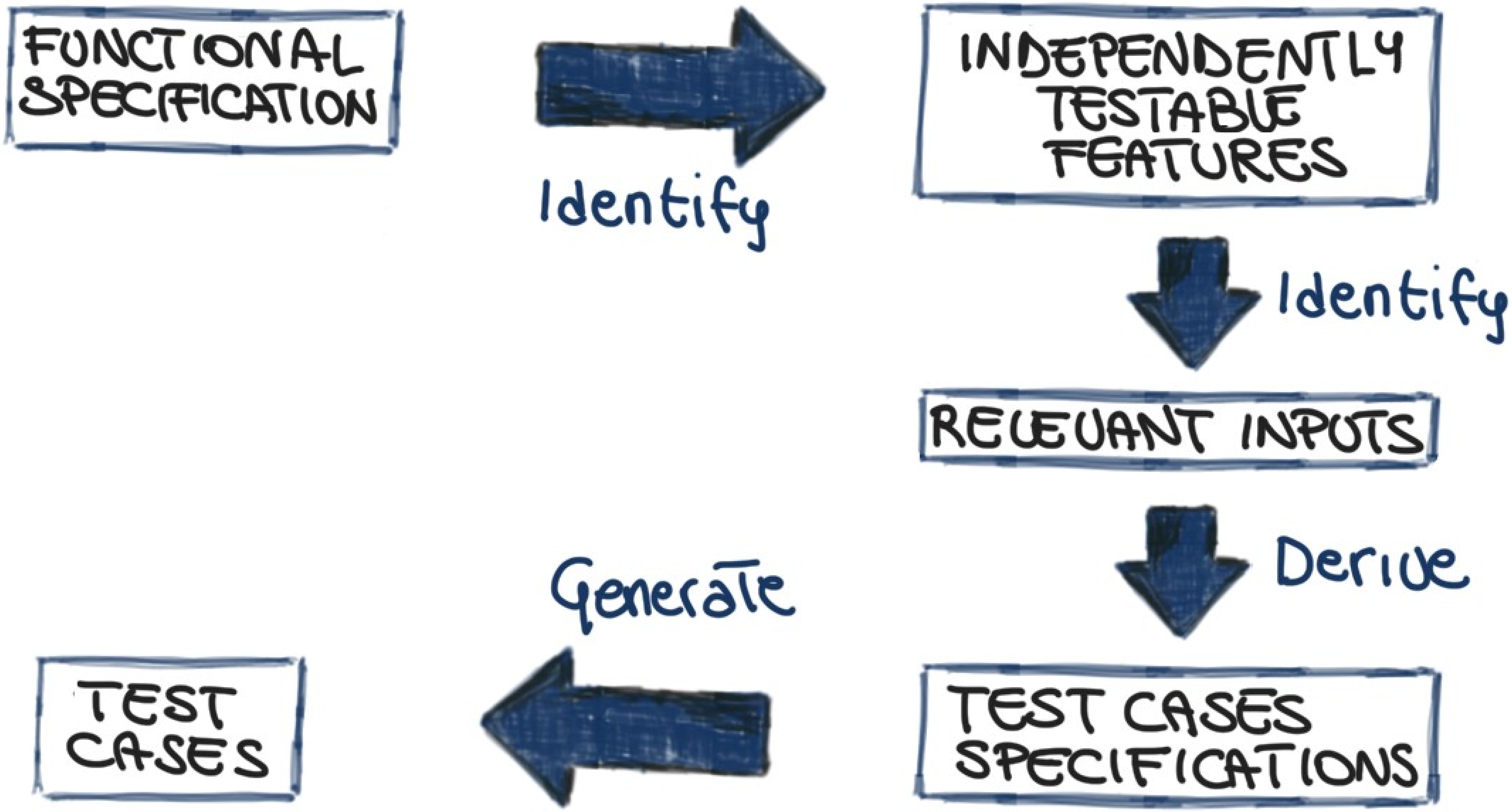
BLACK-BOX TESTING

Advantages

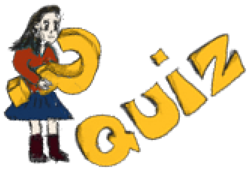
- focus on the domain
- No need for the code
  ⇒ early test design
- Catches logic defects
- Applicable at all granularity levels

FROM SPECIFICATIONS TO TEST CASES

A SYSTEMATIC FUNCTIONAL-TESTING APPROACH



A SYSTEMATIC FUNCTIONAL-TESTING APPROACH



IDENTIFYING TESTABLE FEATURES

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

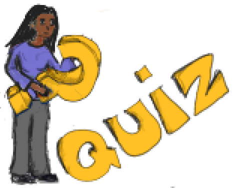
How many indepentey testable features do we have here ?

[ ] 1

[ ] 2

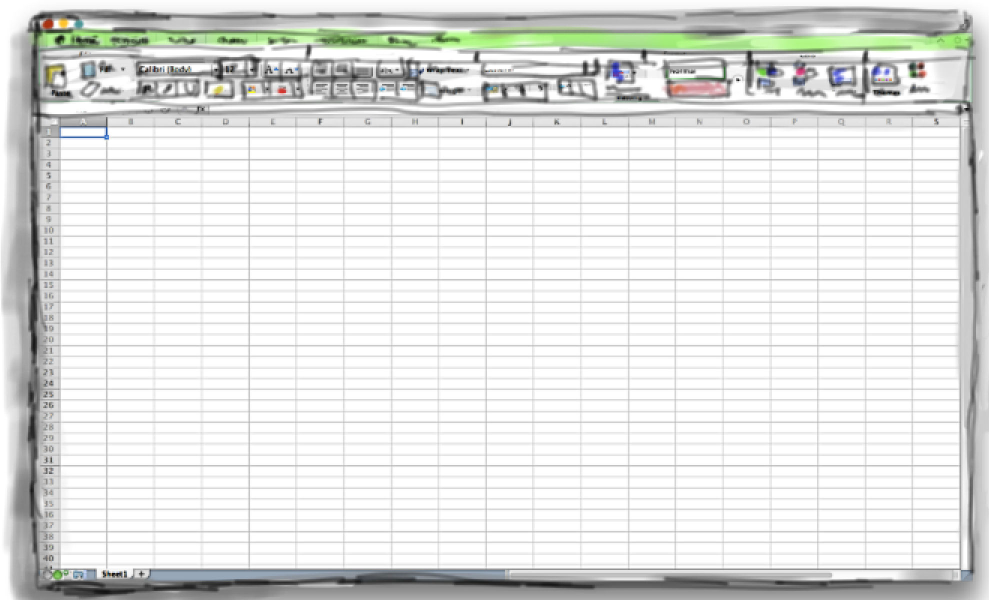
[ ] 3

[ ] > 3



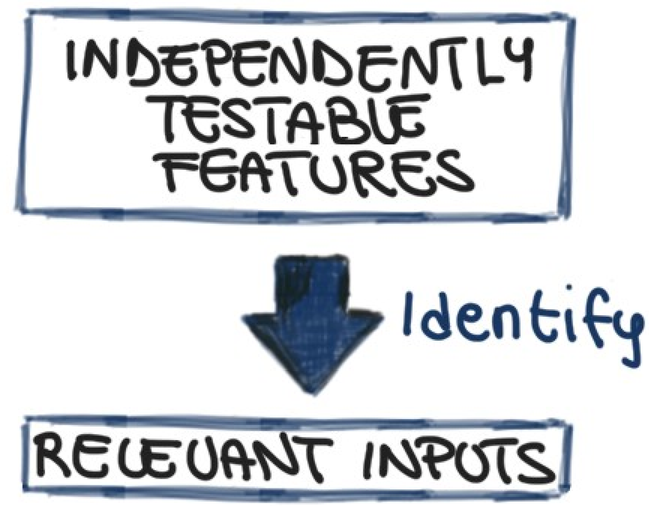
IDENTIFYING TESTABLE FEATURES

Identify three possible independently testable features for a spreadsheet

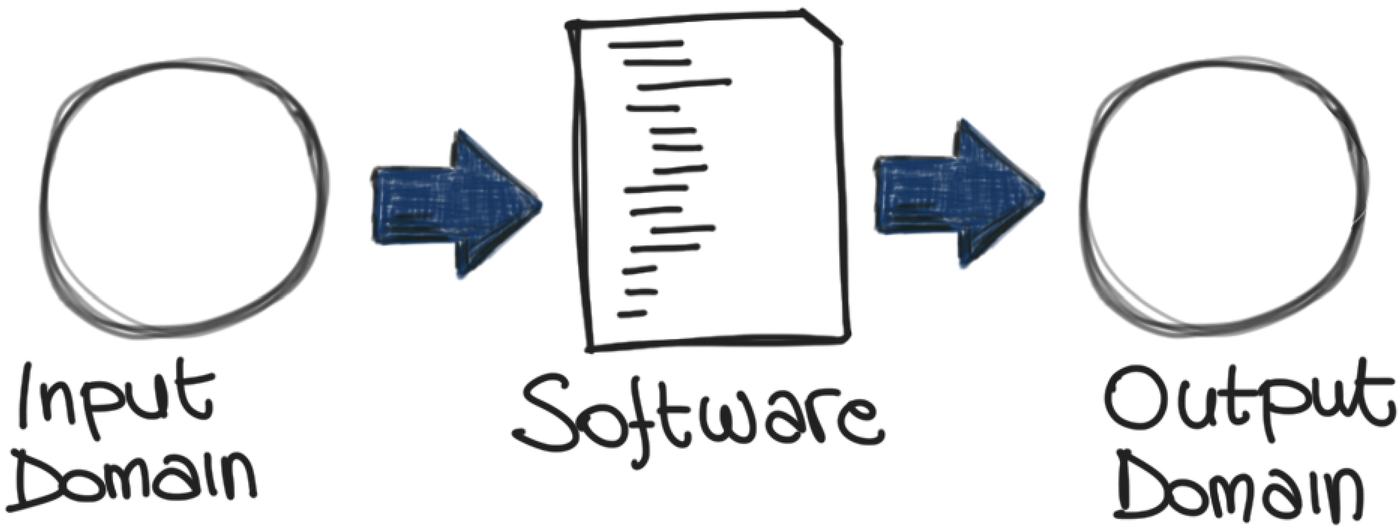


A SYSTEMATIC FUNCTIONAL-TESTING APPROACH

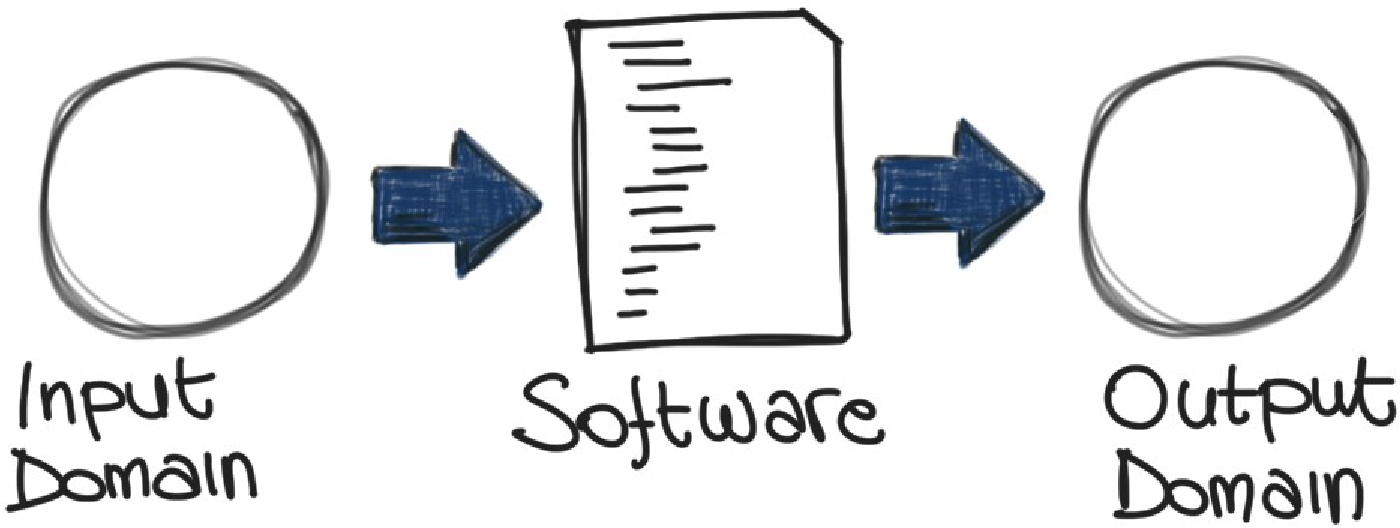
A SYSTEMATIC FUNCTIONAL-TESTING APPROACH



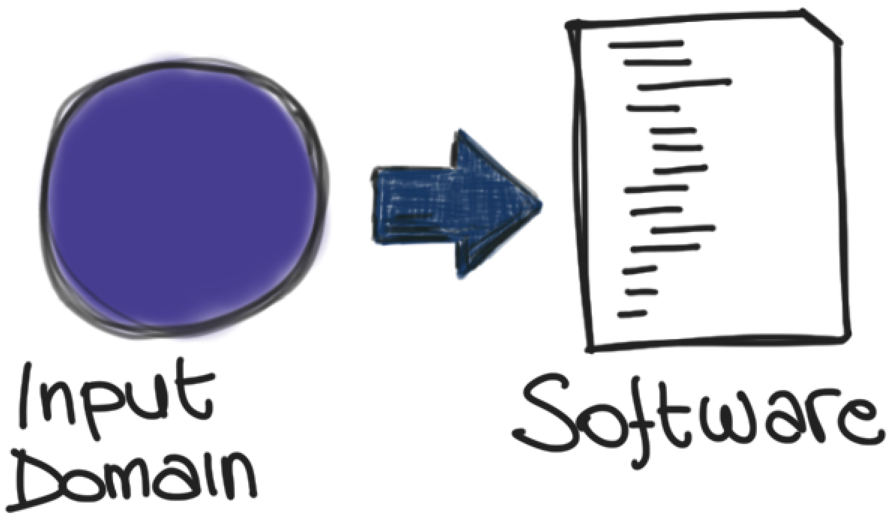
TEST DATA SELECTION



STRAW-MAN IDEA: EXHAUSTIVE TESTING!



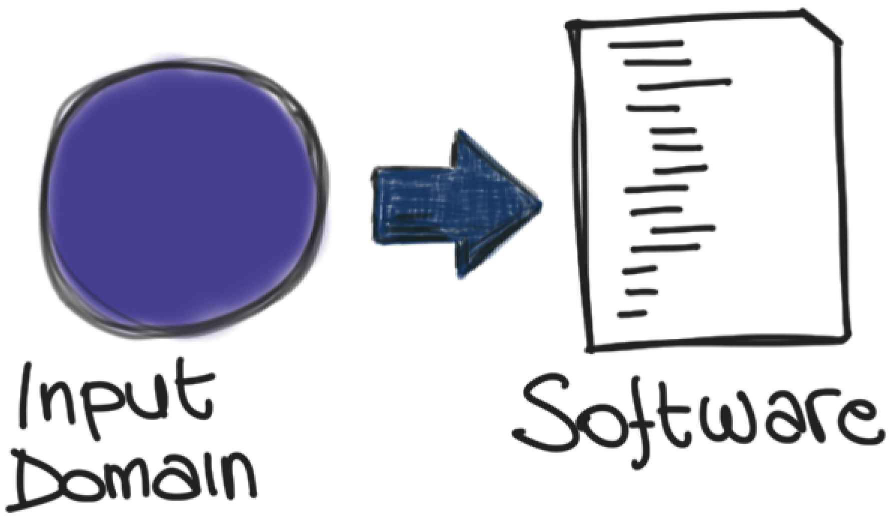
STRAW-MAN IDEA: EXHAUSTIVE TESTING!



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

[ ]

STRAW-MAN IDEA: EXHAUSTIVE TESTING!



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

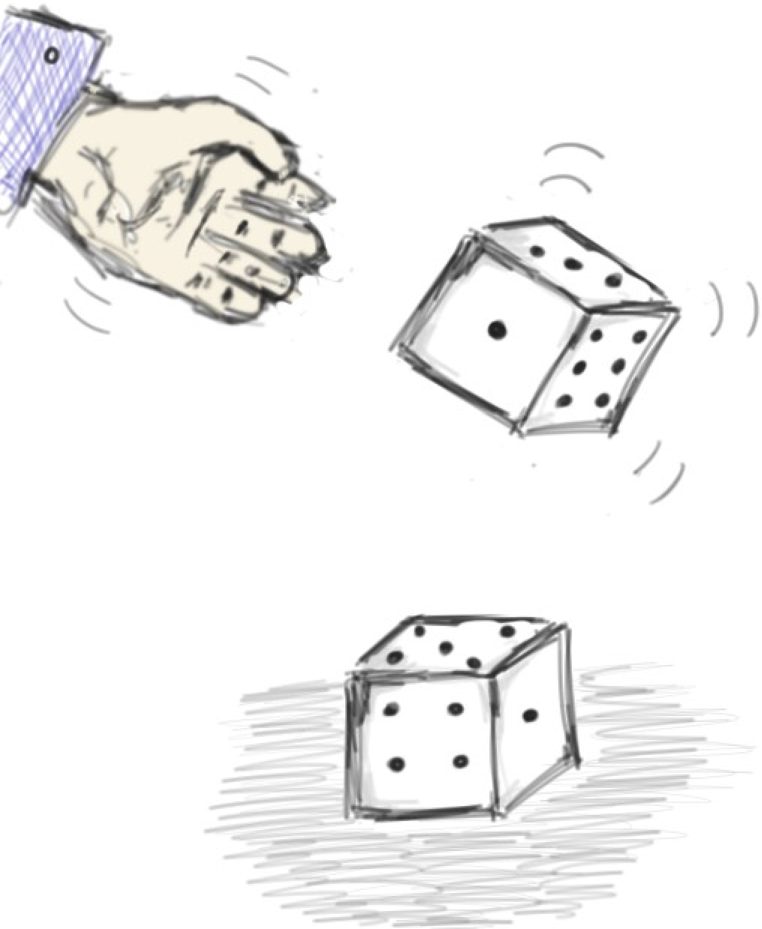
$2^{32} \times 2^{32} = 2^{64} \cong 10^{19}$ tests

1 test per nanosecond (10^9 tests/sec)

=> 10^{10} seconds

[~600 years]

RANDOM TESTING

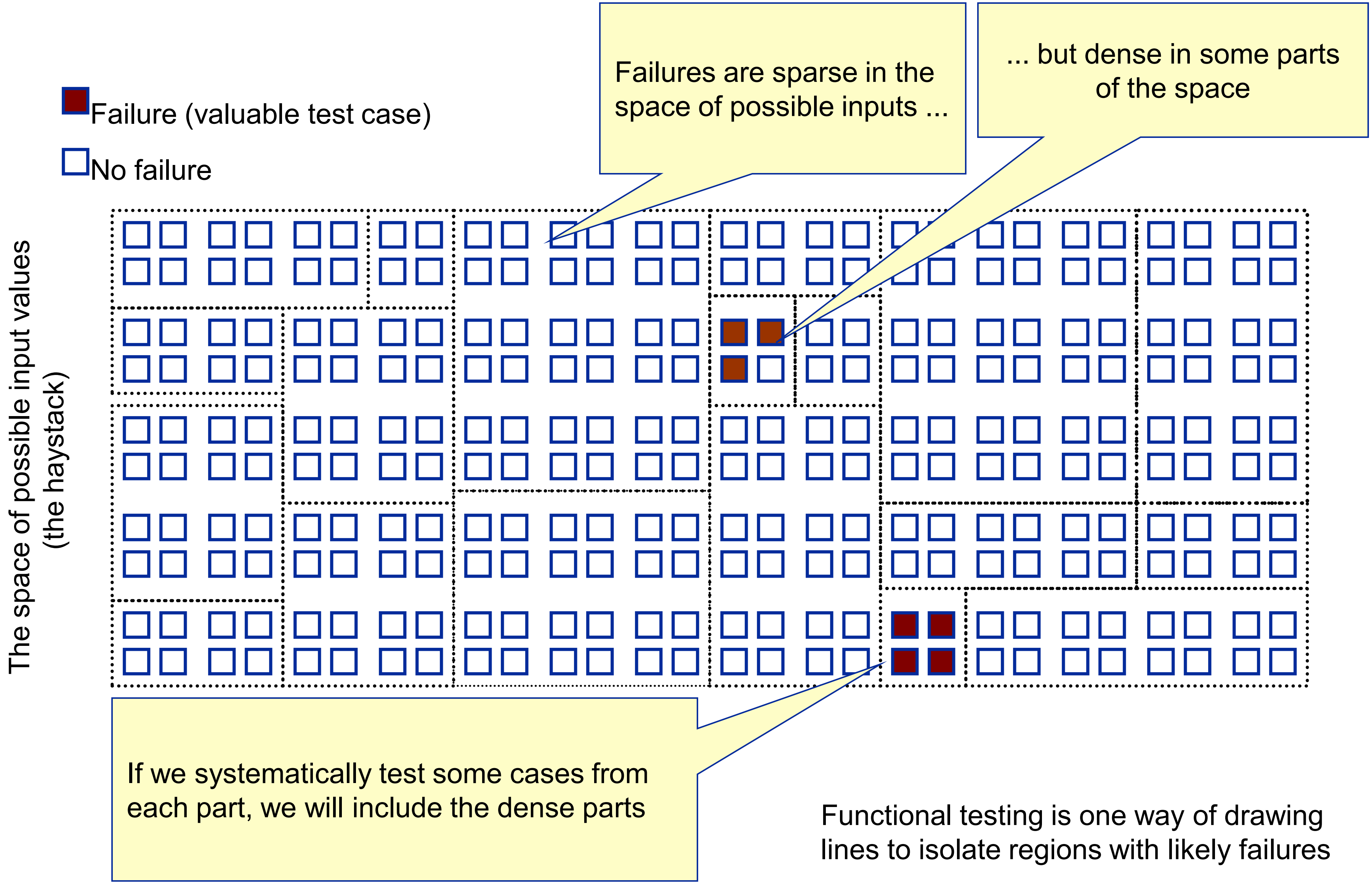


- pick inputs uniformly
- all inputs considered equal
- no designer bias

SO WHY NOT RANDOM ?

SO WHY NOT RANDOM ?

Systematic Partition Testing



Functional testing is one way of drawing lines to isolate regions with likely failures

EXAMPLE

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

Some possible partitions:

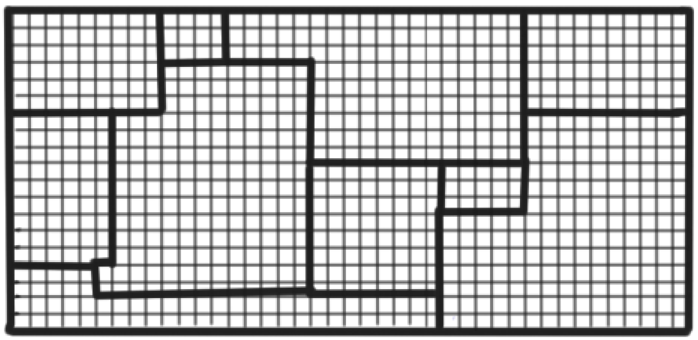
EXAMPLE

Split(string str, int size)

Some possible partitions:

- $size < 0$
- $size = 0$
- $size > 0$
- str with length $< size$
- str with length in $[size, size \times 2]$
- str with length $> size \times 2$
- ...

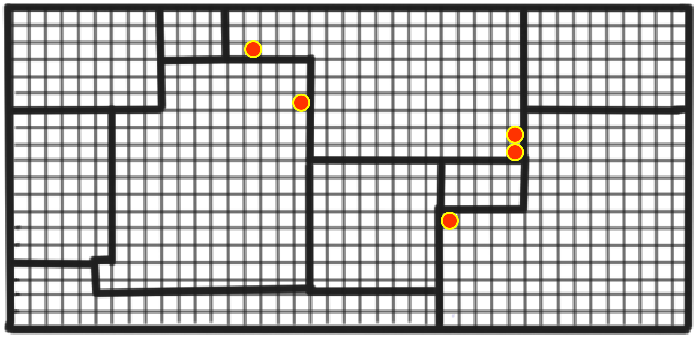
BOUNDARY VALUES



Basic idea

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

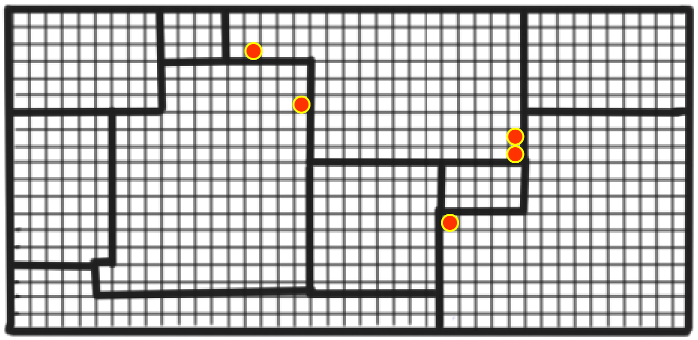
BOUNDARY VALUES



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 x2]
- str with length > size x2
- ...

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

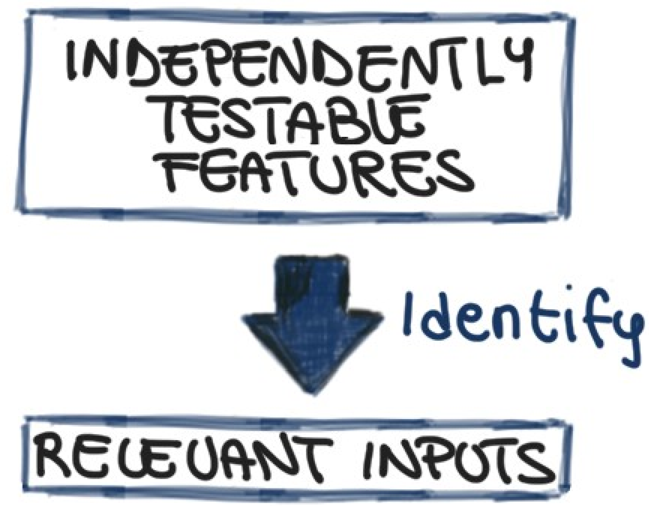
- str with length < size
- str with lenght in [size , size x 2]
- str with lenght > size x 2

Some possible inputs

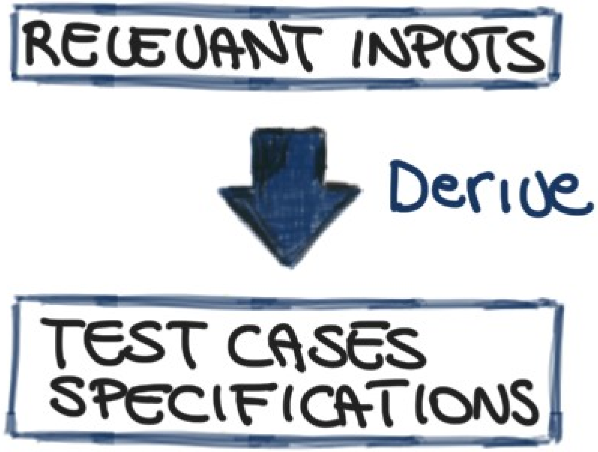
- size = -1
- size = 1
- size = MAXINT

- 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 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
- ...

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

×

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

Test case specifications

EXAMPLE

Split(string str, int size)

Some possible inputs

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

Test case specifications

- size = -1 str with length -2
- size -1 str with length -1
- size = 1 str with length 0
- ...

EXAMPLE

Split(string str, int size)

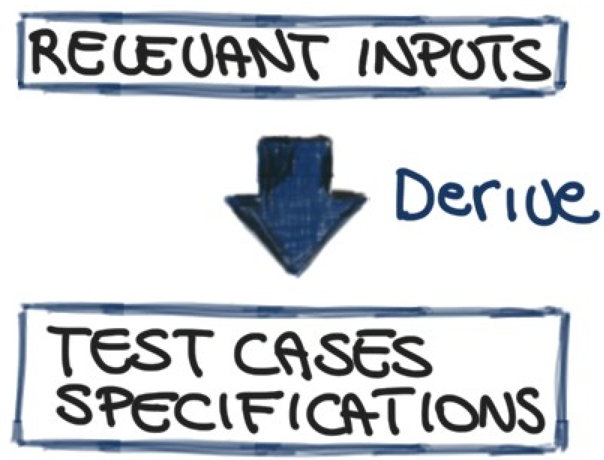
Some possible inputs

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

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

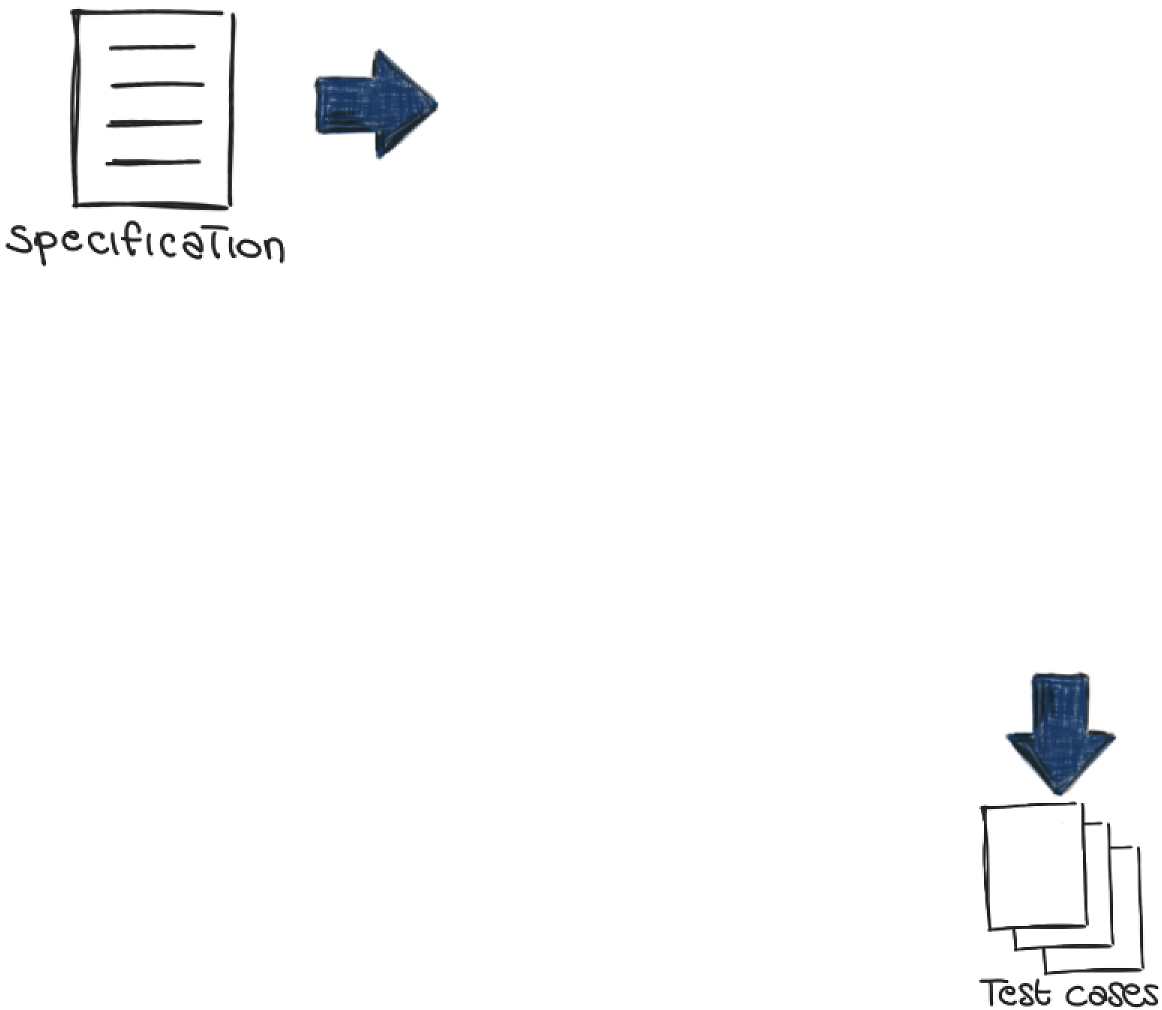
Requires building scaffolding

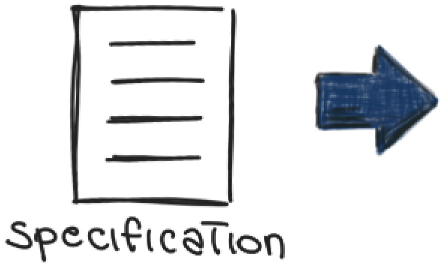
- Drivers
- Stubs

A SPECIFIC BLACK-BOX TESTING APPROACH

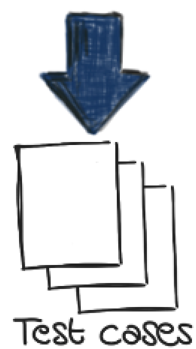
THE CATEGORY-PARTITION METHOD

[Ostrand & Balcer, CACM, June 1988]


Specification
Test cases



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	Input size
- lenght	- value
- content	

PARTITION CATEGORIES INTO CHOICES

Interesting cases (subdomains)

Example: split(string str, int size)

Input str	Input size
- lenght	- value
- content	

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

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

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

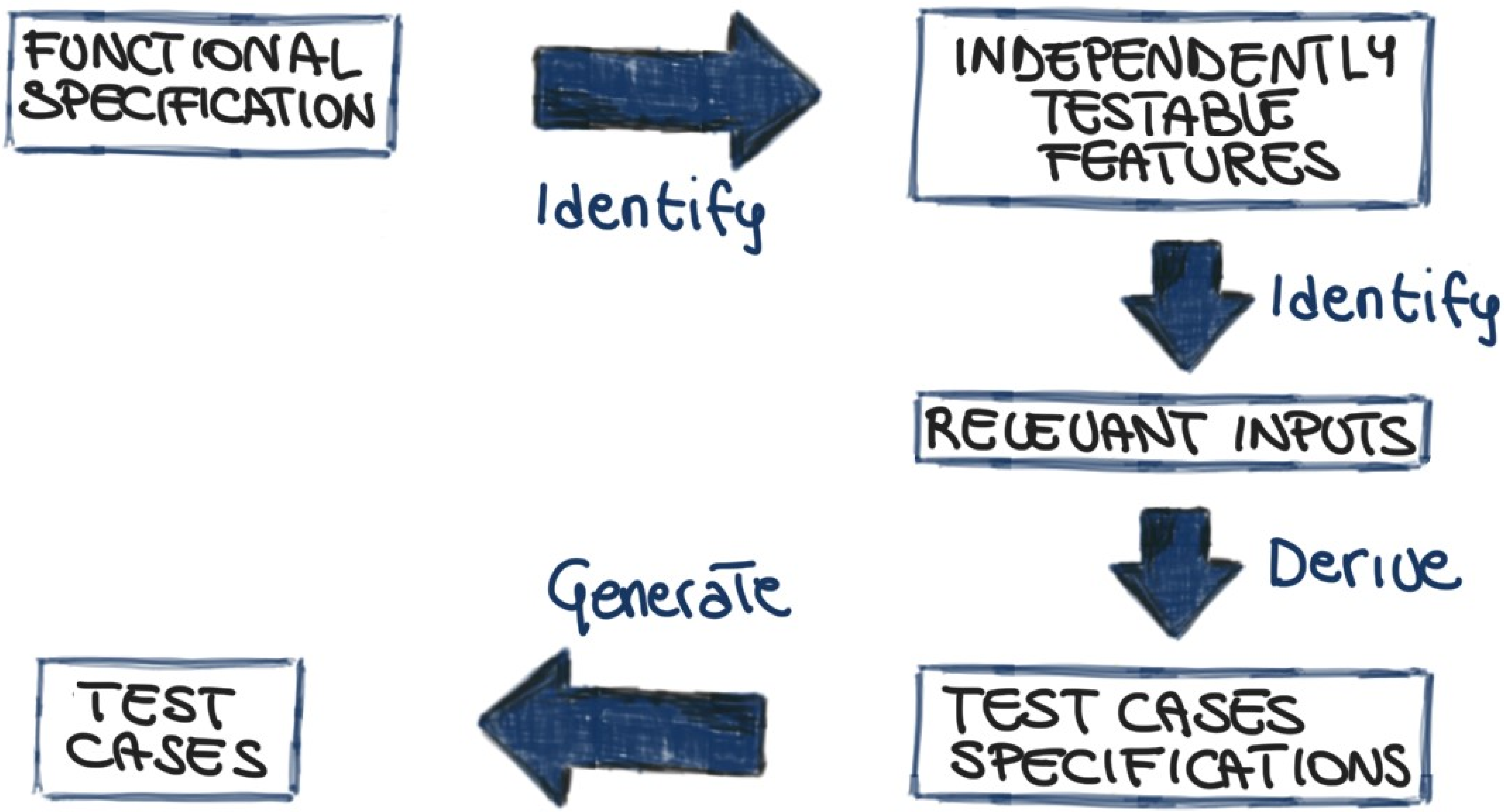
Simple instantiation of frames

Final result : set of concrete tests

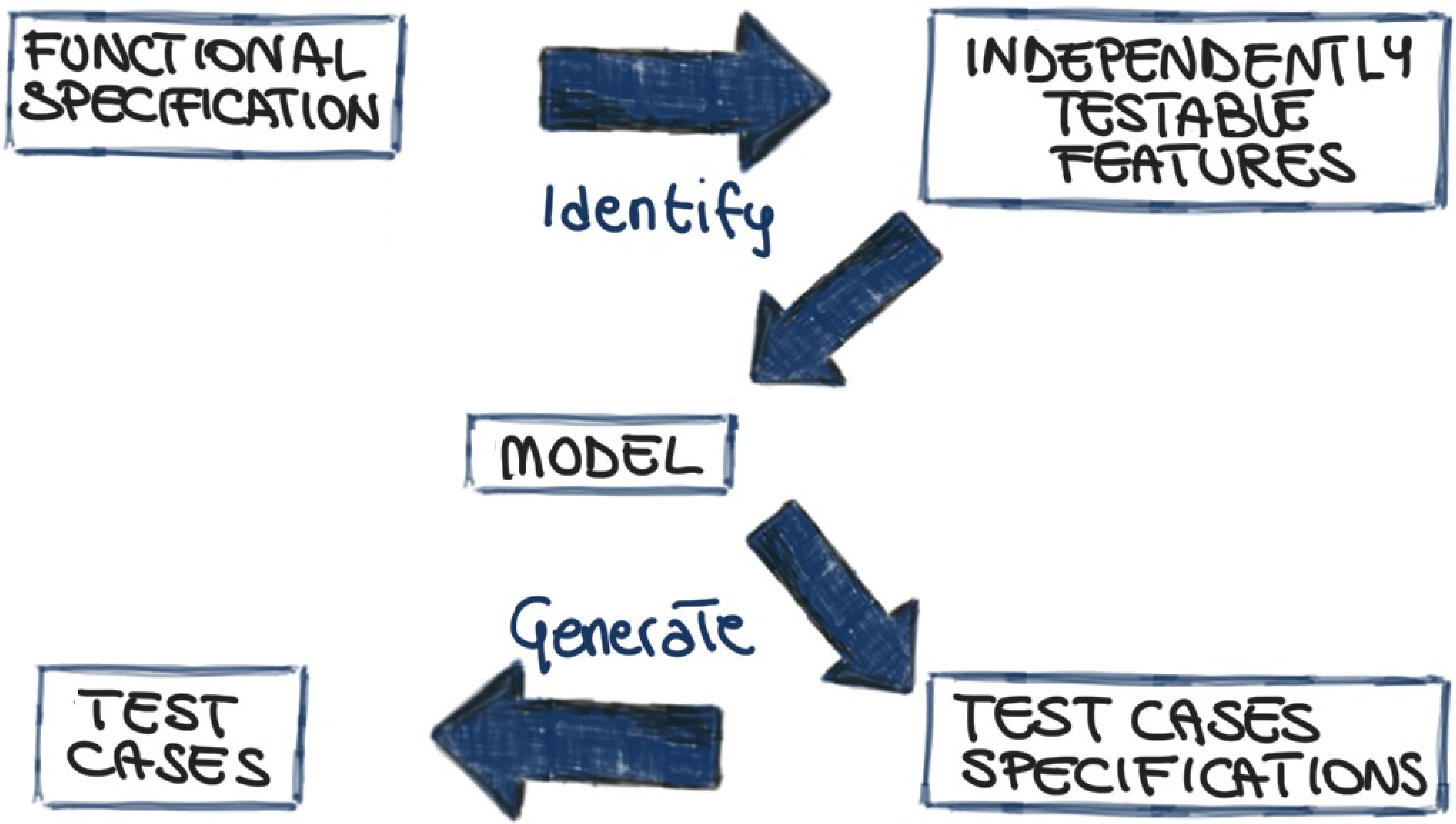
Example

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

A SYSTEMATIC FUNCTIONAL-TESTING APPROACH

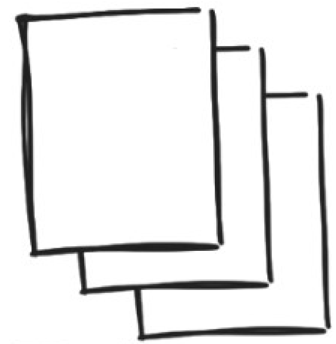


A SYSTEMATIC FUNCTIONAL-TESTING APPROACH



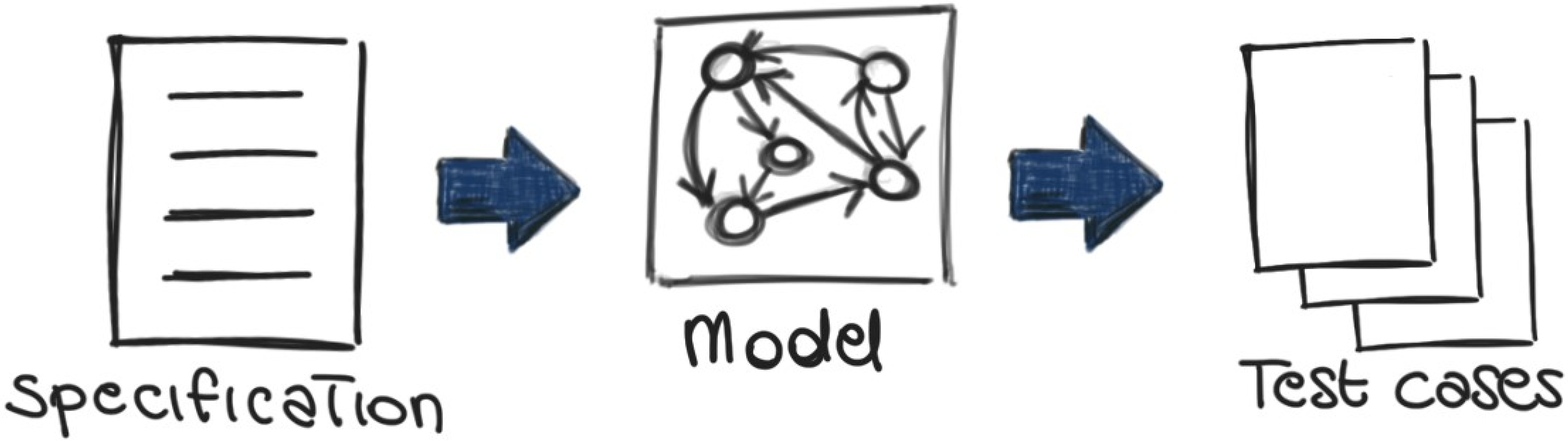
MODEL-BASED TESTING

Specification



Test cases

MODEL-BASED TESTING

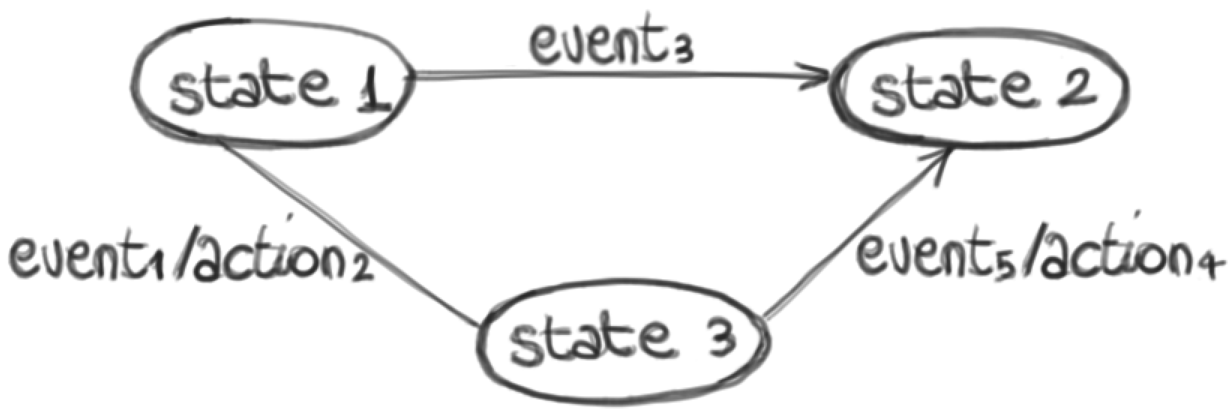


FINITE STATE MACHINES (FSM)

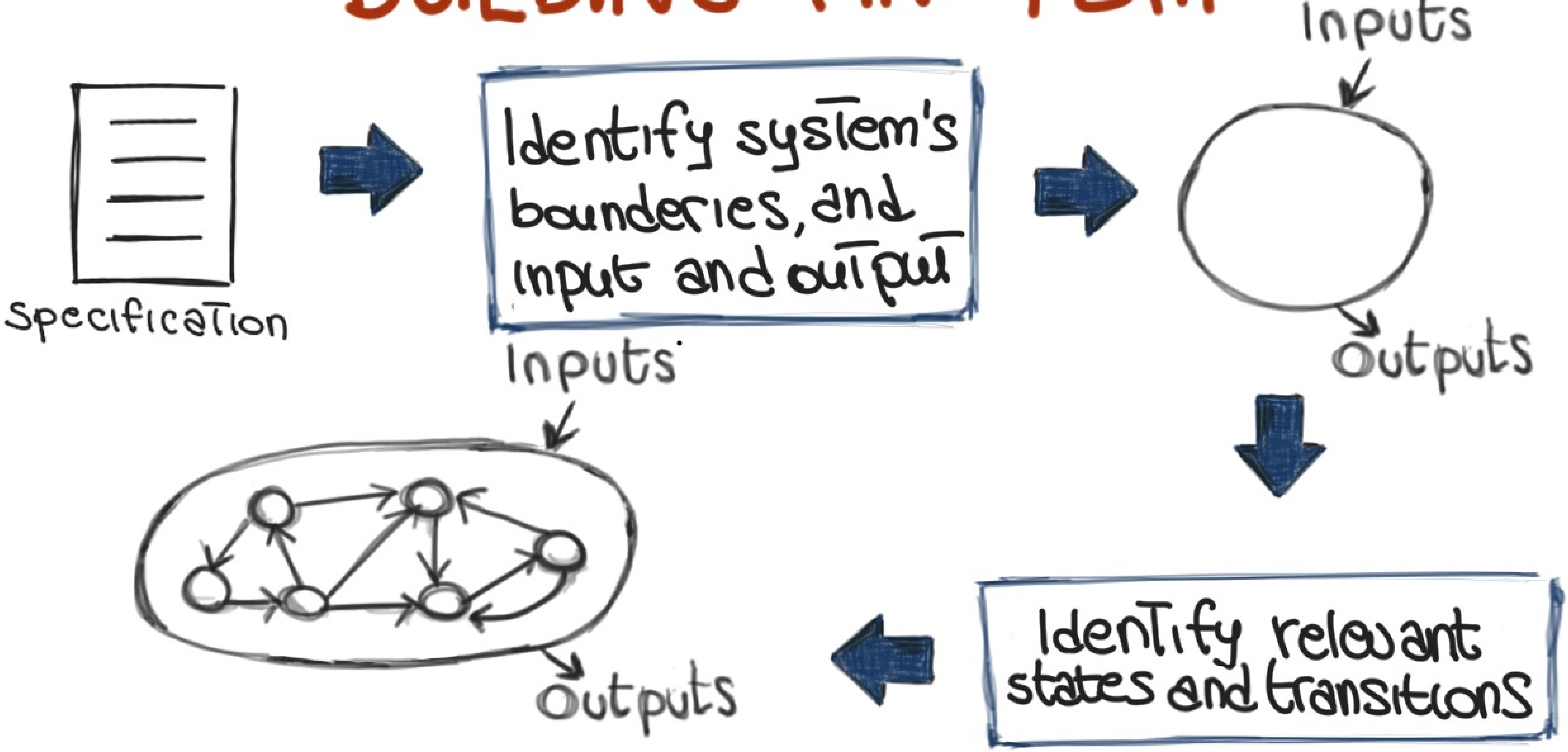
Nodes = states

Edges = transitions

Edge labels = events/actions



BUILDING AN FSM



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.
```

FROM AN INFORMAL SPECIFICATION...

Multiple choices here

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 →

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 →

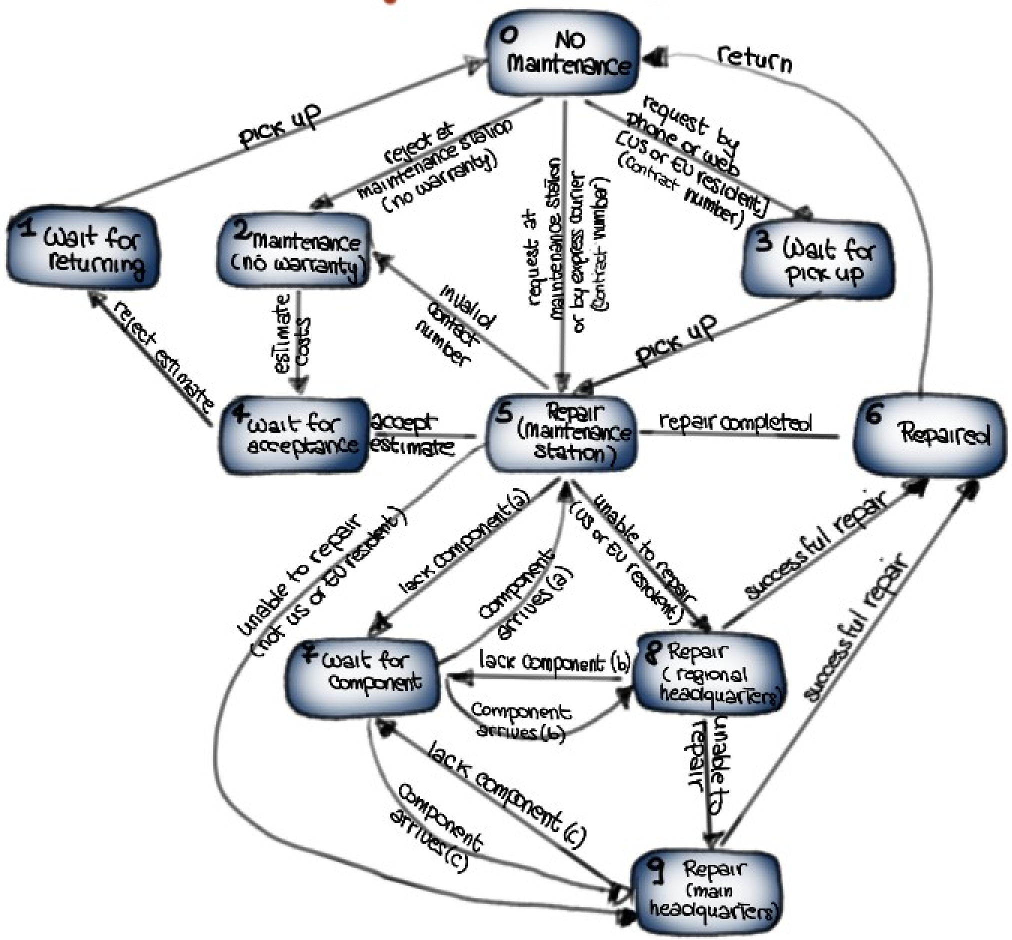
Determine the next step →

and so on →

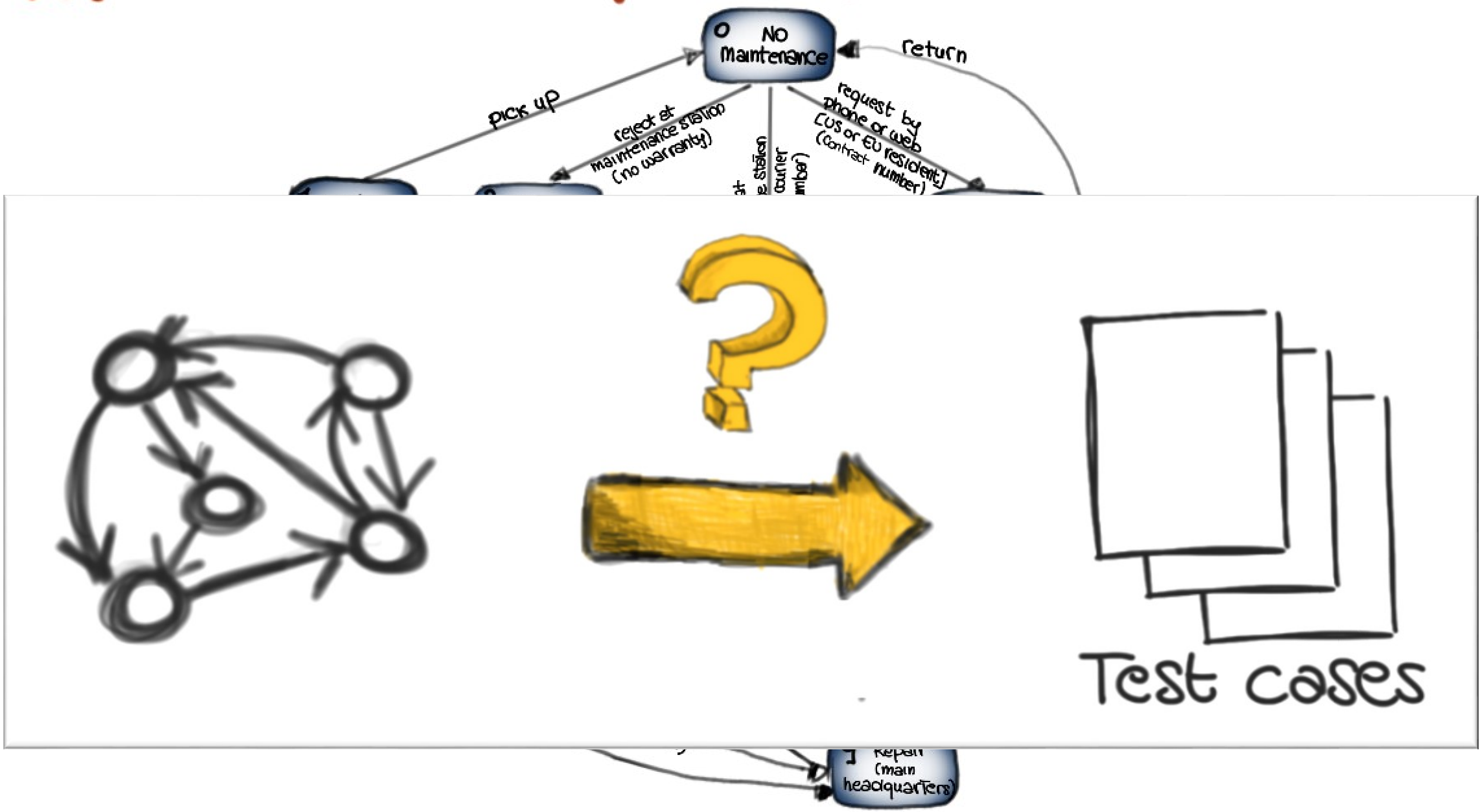
Spec.txt

```
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.
```

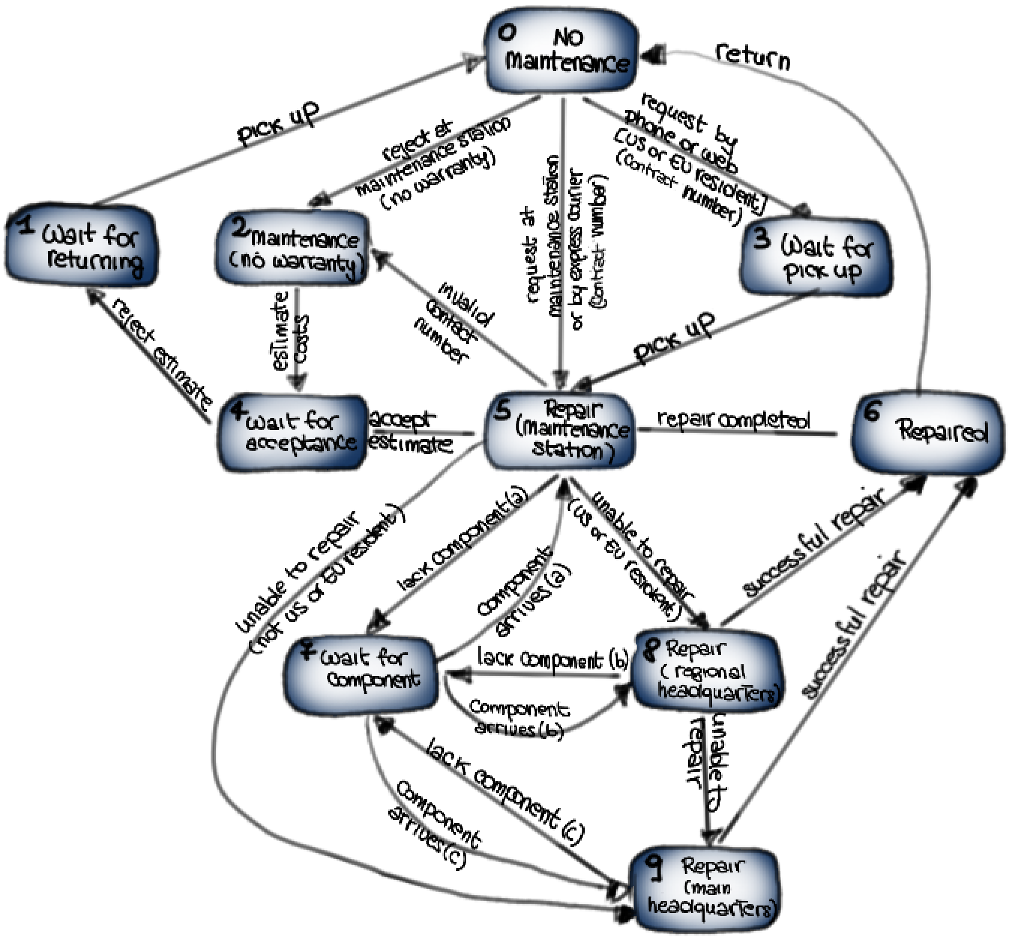
... TO A FINITE STATE MACHINE



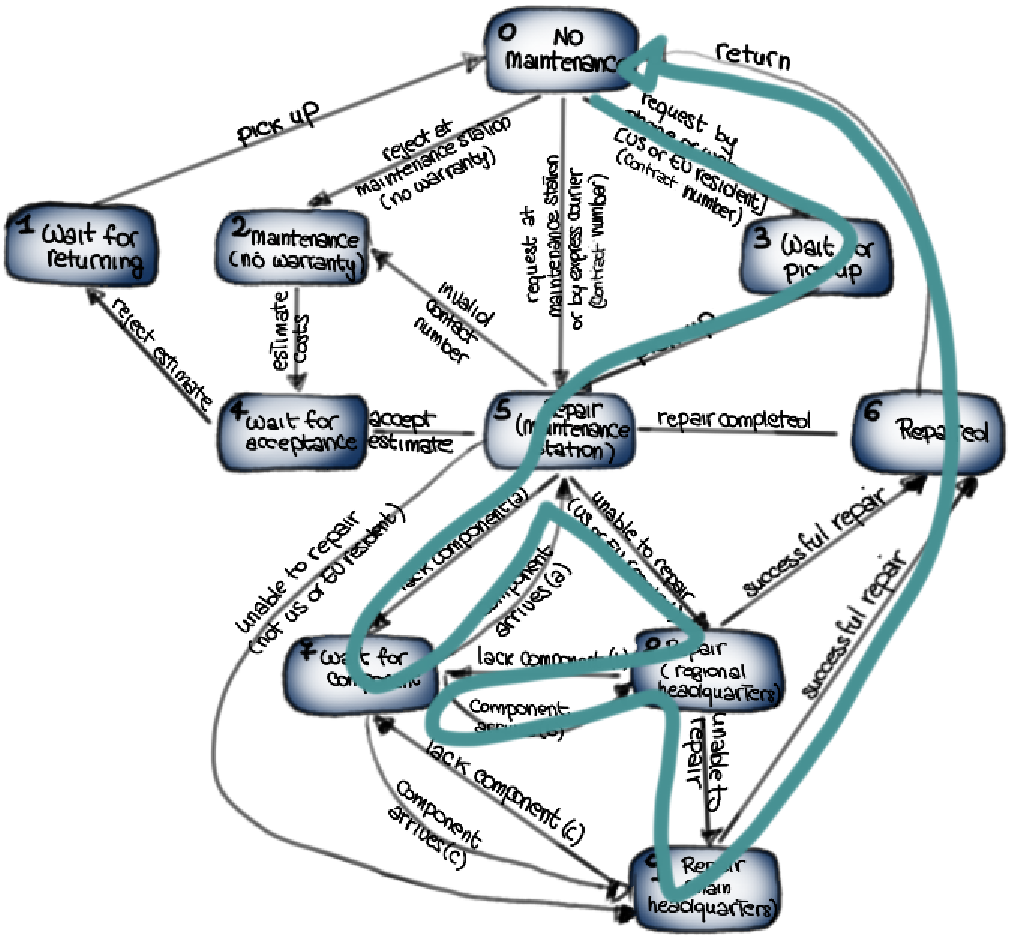
... TO A FINITE STATE MACHINE



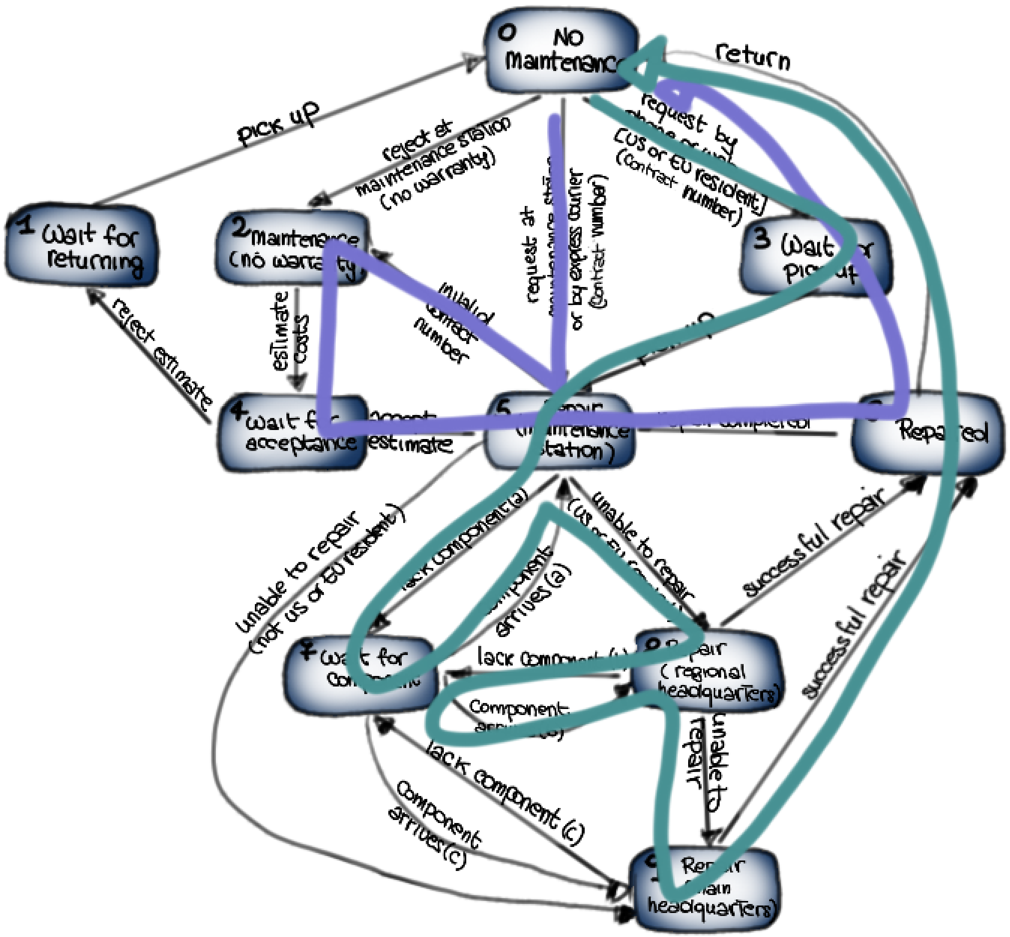
TO A SET OF TEST CASES



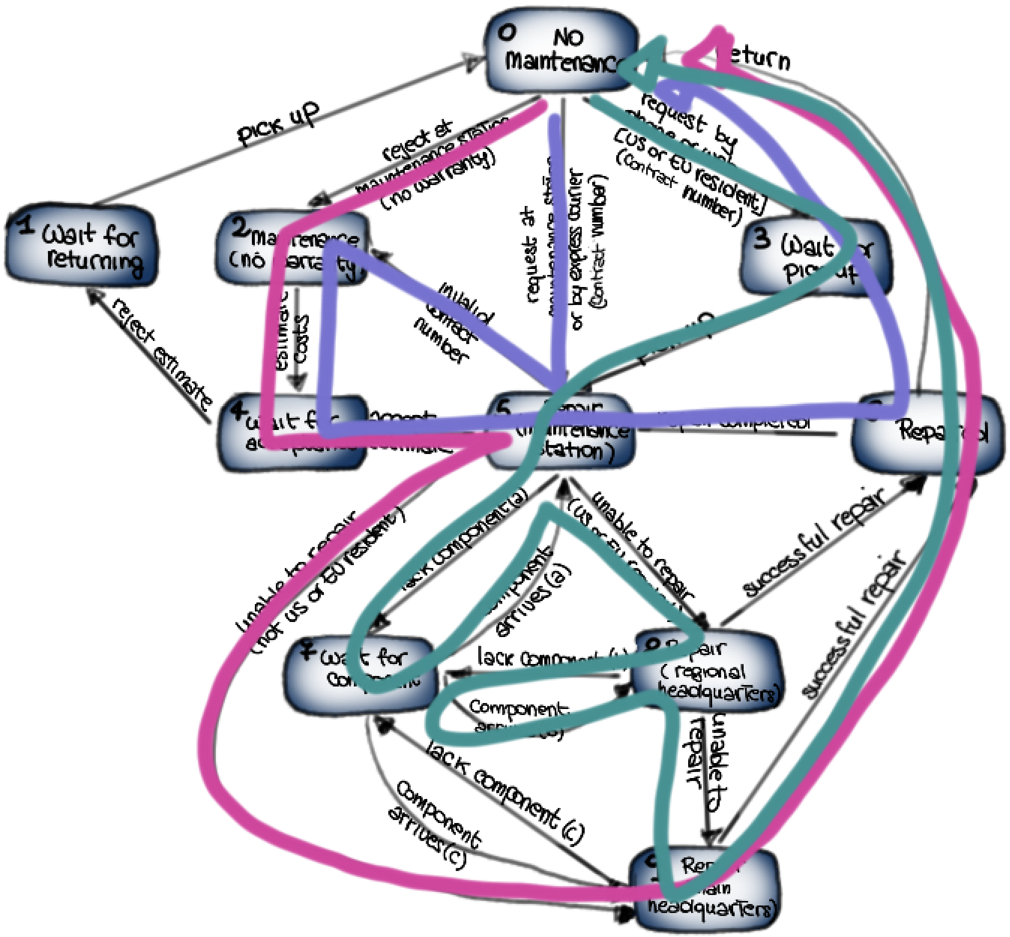
TO A SET OF TEST CASES



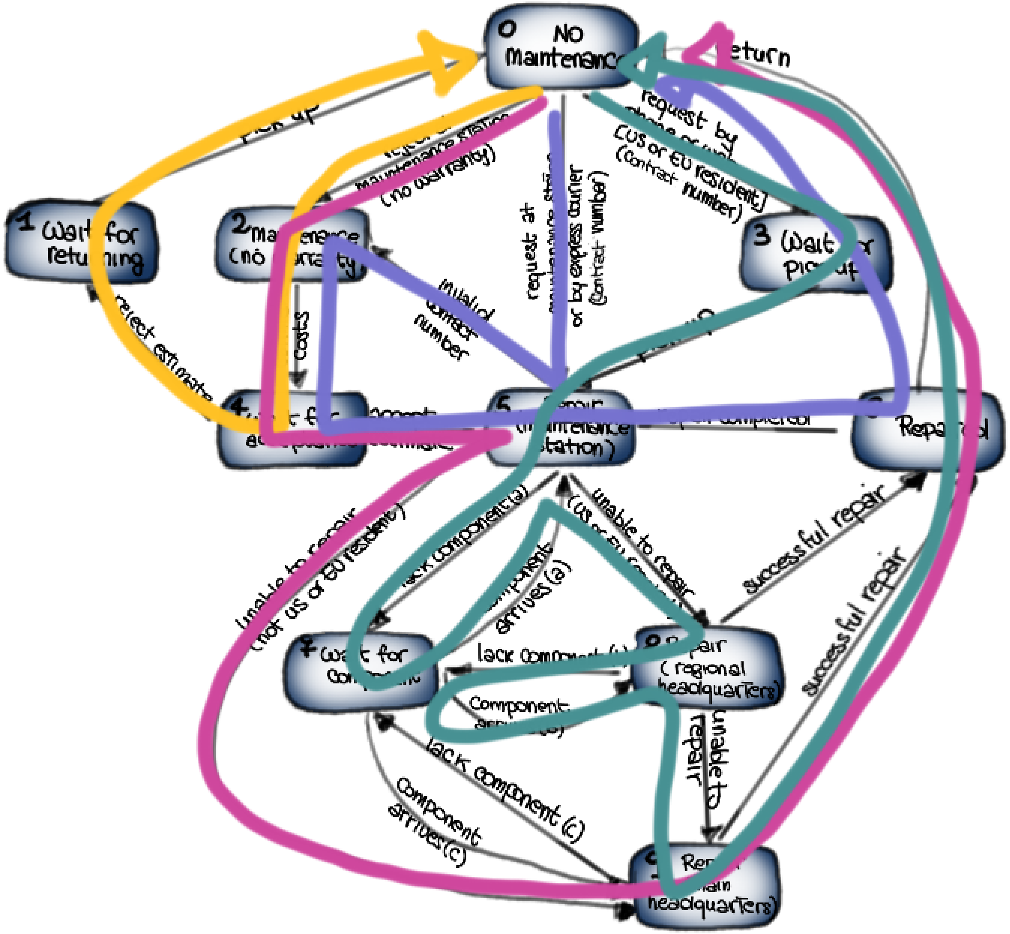
TO A SET OF TEST CASES



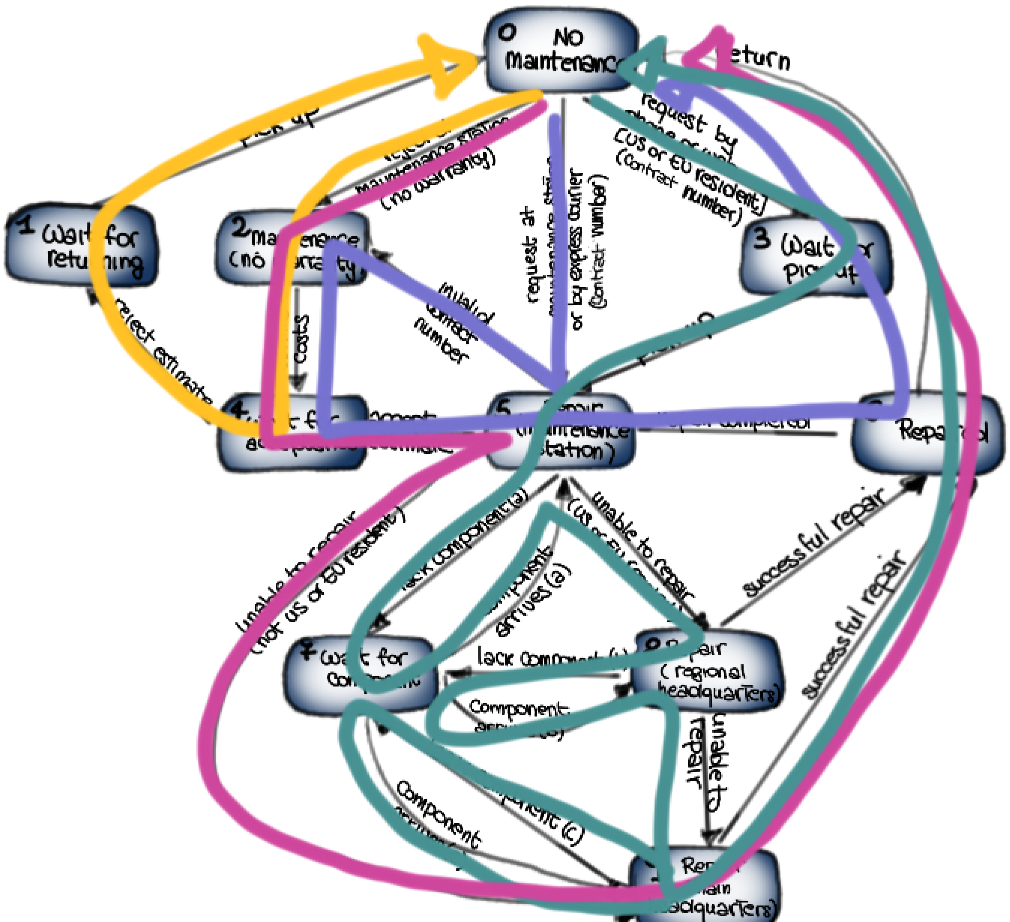
TO A SET OF TEST CASES



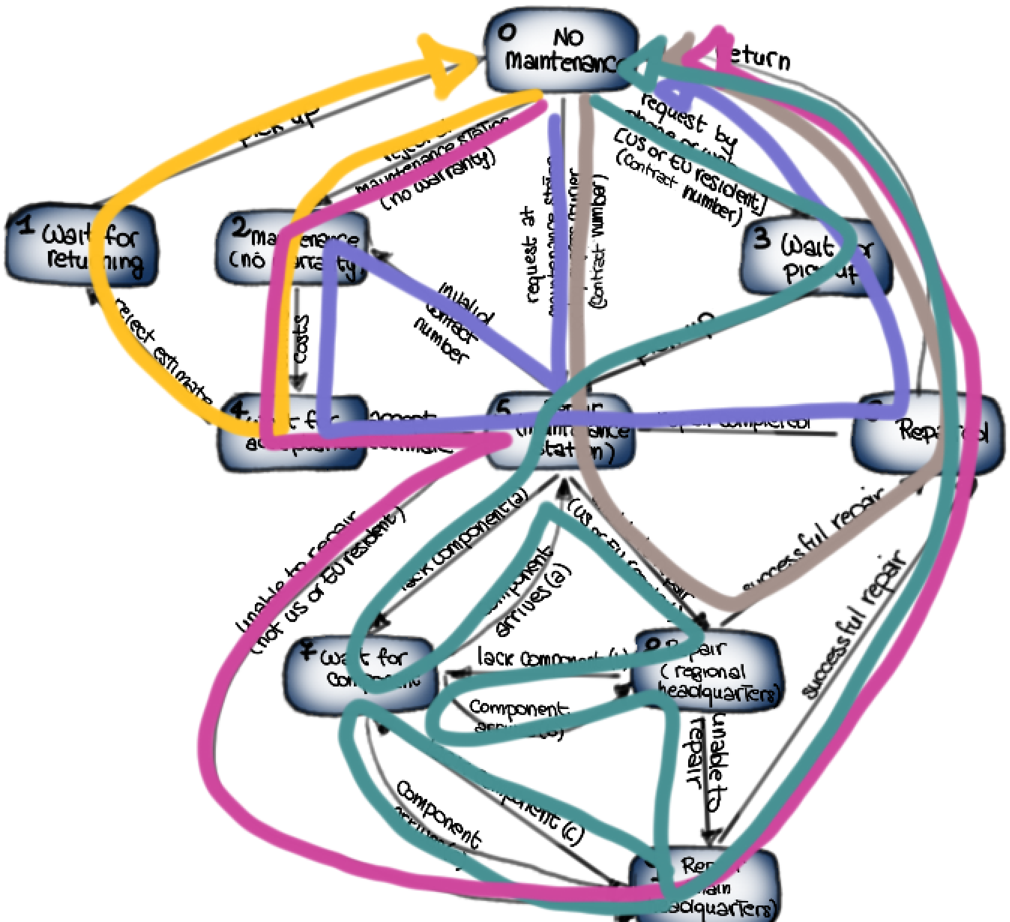
TO A SET OF TEST CASES



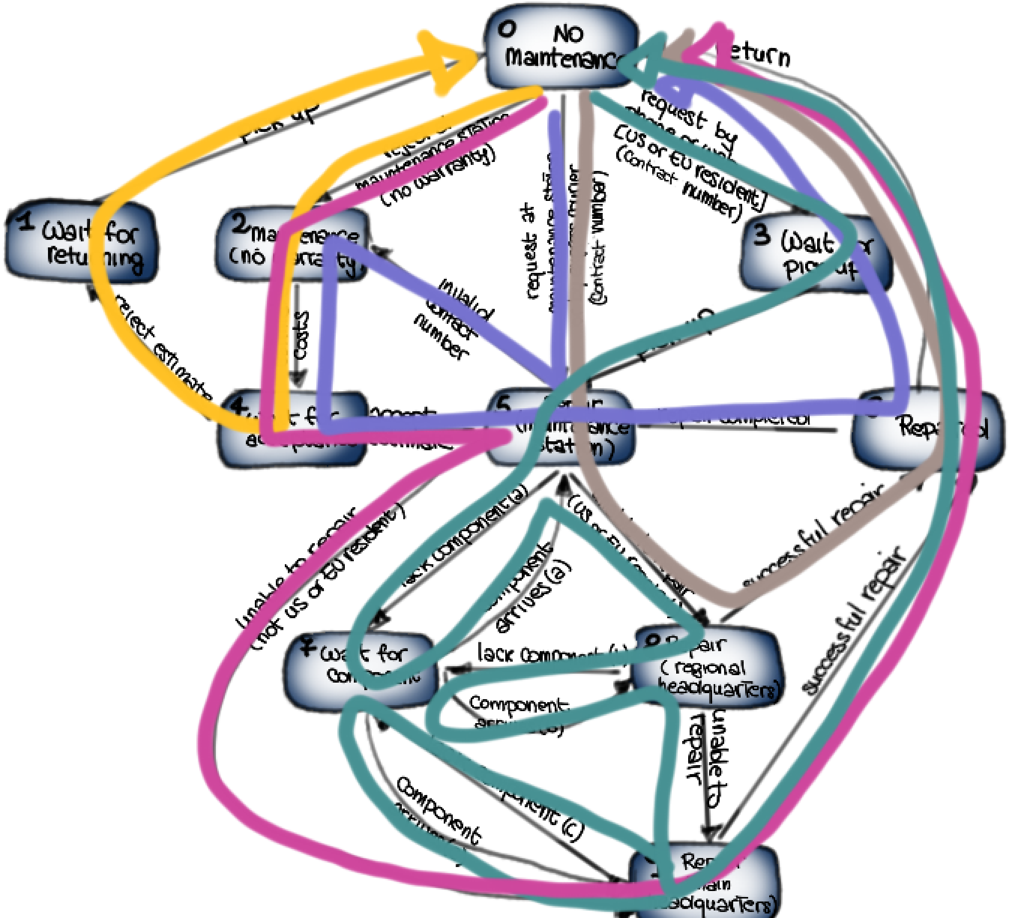
TO A SET OF TEST CASES



TO A SET OF TEST CASES



TO A SET OF TEST CASES



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

TC2: Ø, 5, 2, 4, 5, 6, Ø

TC3: Ø, 2, 4, 1, Ø

TC4: Ø, 4, 5, 9, 6, Ø

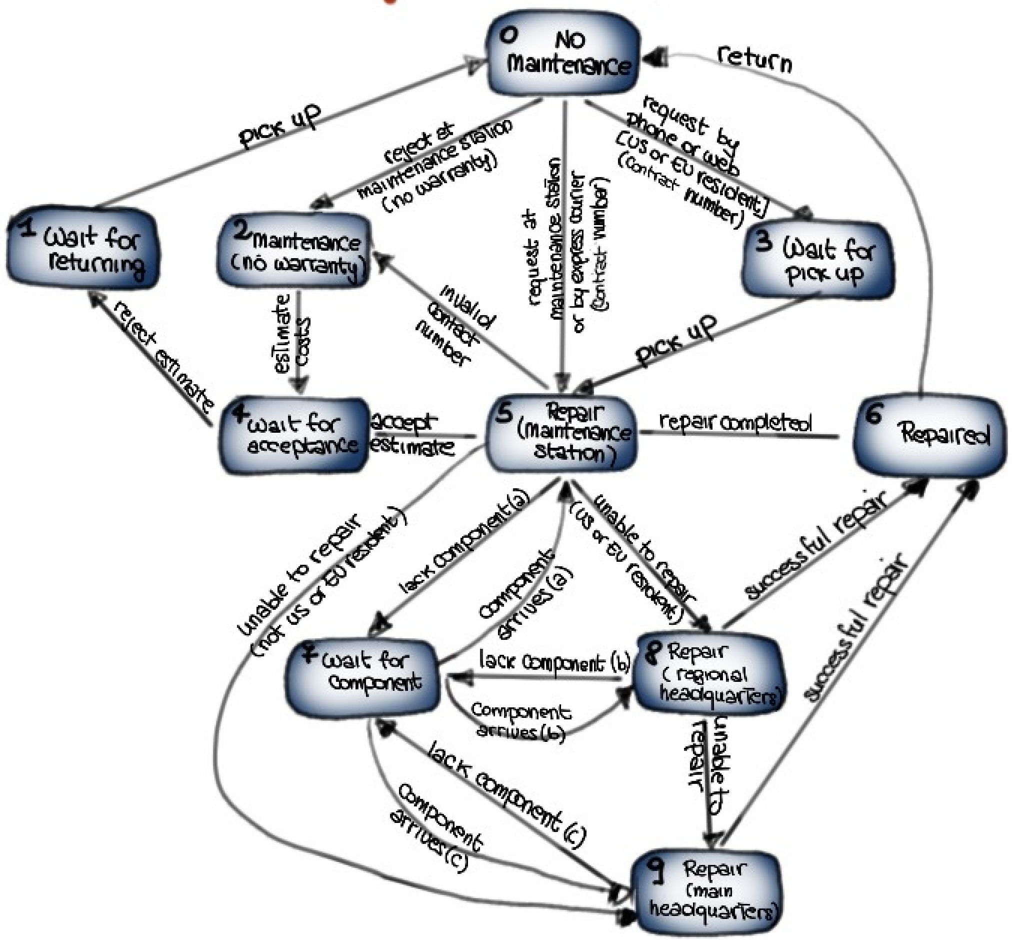
TC5: Ø, 5, 6, Ø

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

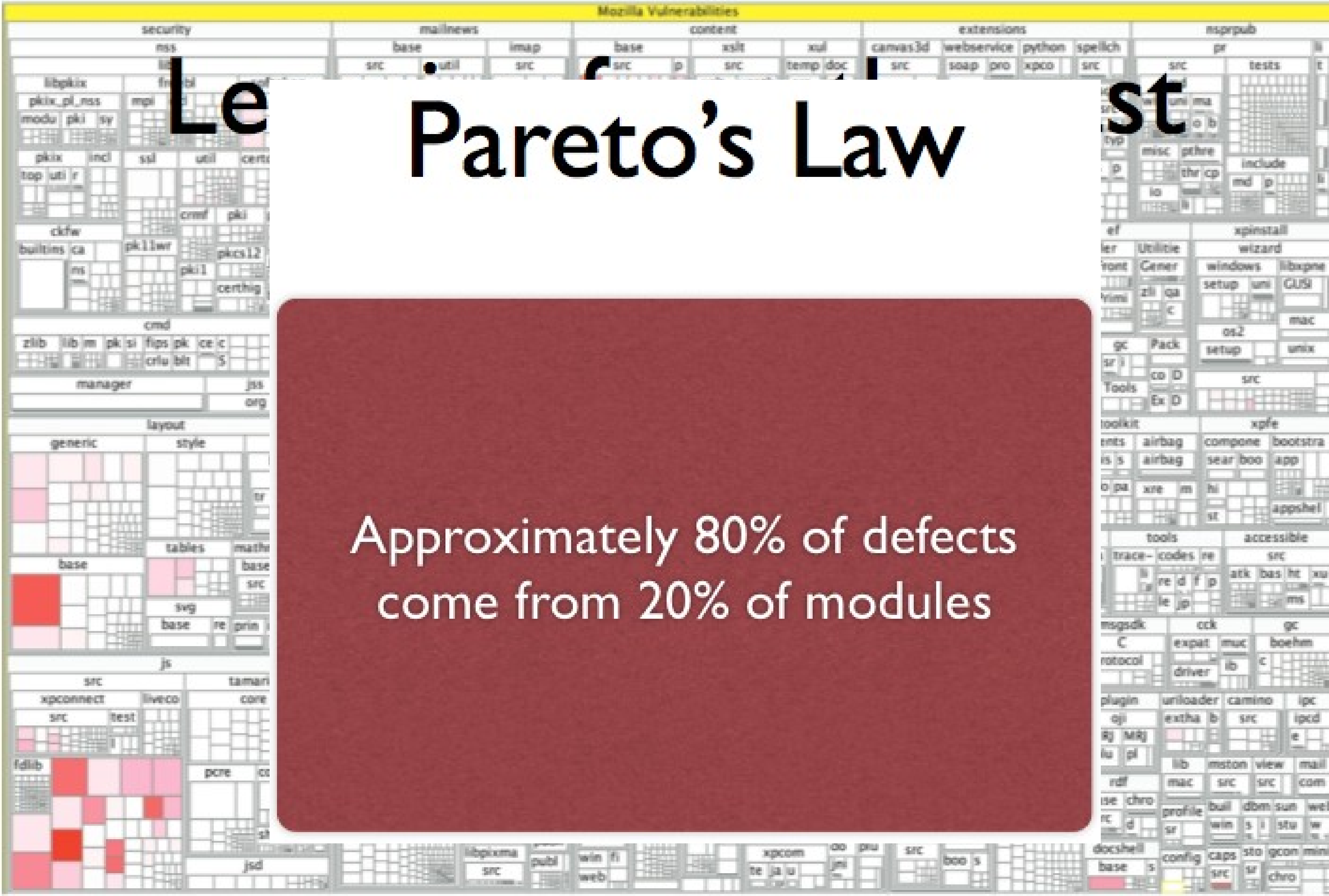
Abstraction is key

Many other approaches

- decision tables
- flow graphs
- historical models

...

Historical models



A SYSTEMATIC FUNCTIONAL-TESTING APPROACH