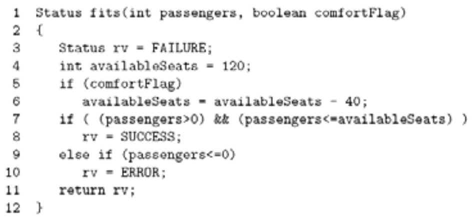
# White Box Testing fits() Example

Software Testing and Quality Assurance

Joseph Timoney

## Source code for fits()

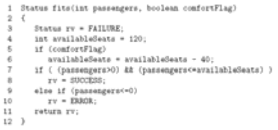


```
 1  Status fits(int passengers, boolean comfortFlag)
 2  {
 3     Status rv = FAILURE;
 4     int availableSeats = 120;
 5     if (comfortFlag)
 6        availableSeats = availableSeats - 40;
 7     if ( (passengers>0) && (passengers<=availableSeats) )
 8        rv = SUCCESS;
 9     else if (passengers<=0)
10        rv = ERROR;
11     return rv;
12  }
```

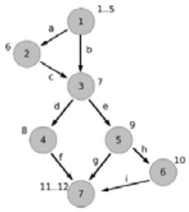
## Example- fits()

- Specification:
- A plane has 120 seats. A flight can accommodate up to that number of passengers. If the passengers require extra comfort, the effective number of seats is reduced by 40 to ensure that every passenger has an empty seat next to them. The method fits() indicates whether a particular number of passengers can be accommodated with or without extra comfort.

## CFG for fits()



```
 1  Status fits(int passengers, boolean comfortFlag)
 2  {
 3     Status rv = FAILURE;
 4     int availableSeats = 120;
 5     if (comfortFlag)
 6        availableSeats = availableSeats - 40;
 7     if ( (passengers>0) && (passengers<=availableSeats) )
 8        rv = SUCCESS;
 9     else if (passengers<=0)
10        rv = ERROR;
11     return rv;
12  }
```

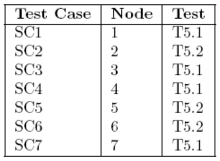


- Node 1: Lines 1–5
- Node 2: Line 6
- Node 3: Line 7
- Node 4: Line 8
- Node 5: Line 9
- Node 6: Line 10
- Node 7: Line 11–12

## Fits() specification

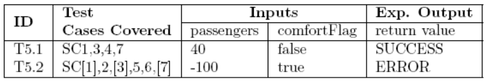
- **Status** fits(int passengers, boolean comfortFlag)
- **Inputs**
  - passengers: the number of passengers to be carried
  - comfortFlag: flag to indicate whether extra comfort is required
- **Outputs**
- return value:
  - SUCCESS if passengers≤120 and !comfortFlag
  - SUCCESS if passengers ≤ 80 and comfortFlag
  - FAILURE if passengers>120, or if passengers>80 and comfortFlag
  - ERROR if any inputs are invalid (e.g. passengers<1)
- Status is defined as follows:
  - enum Status { SUCCESS, FAILURE, ERROR };

## Test Cases for Statement Coverage for fits()



| Test Case | Node | Test |
|---|---|---|
| SC1 | 1 | T5.1 |
| SC2 | 2 | T5.2 |
| SC3 | 3 | T5.1 |
| SC4 | 4 | T5.1 |
| SC5 | 5 | T5.2 |
| SC6 | 6 | T5.2 |
| SC7 | 7 | T5.1 |

## Test Data for Statement Coverage for fits()



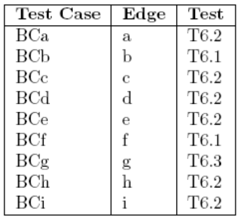
| ID | Test<br>Cases Covered | Inputs | | Exp. Output |
|---|---|---|---|---|
| | | passengers | comfortFlag | return value |
| T5.1 | SC1,3,4,7 | 40 | false | SUCCESS |
| T5.2 | SC[1],2,[3],5,6,[7] | -100 | true | ERROR |

## Path Coverage

Six paths can be identified in the CFG for fits()

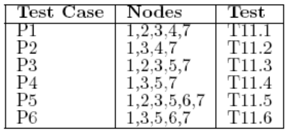
- Node 1–2–3–4–7
- Node 1–3–4–7
- Node 1–2–3–5–7
- Node 1–3–5–7
- Node 1–2–3–5–6–7
- Node 1–3–5–6–7

## Test Cases for Branch Coverage for fits()



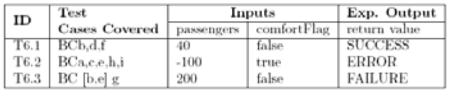
| Test Case | Edge | Test |
|---|---|---|
| BCa | a | T6.2 |
| BCb | b | T6.1 |
| BCc | c | T6.2 |
| BCd | d | T6.2 |
| BCe | e | T6.2 |
| BCf | f | T6.1 |
| BCg | g | T6.3 |
| BCh | h | T6.2 |
| BCi | i | T6.2 |

## Test Cases for Path Coverage for fits()



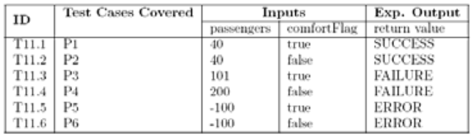
| Test Case | Nodes | Test |
|---|---|---|
| P1 | 1,2,3,4,7 | T11.1 |
| P2 | 1,3,4,7 | T11.2 |
| P3 | 1,2,3,5,7 | T11.3 |
| P4 | 1,3,5,7 | T11.4 |
| P5 | 1,2,3,5,6,7 | T11.5 |
| P6 | 1,3,5,6,7 | T11.6 |

## Test Data for Branch Coverage for fits()

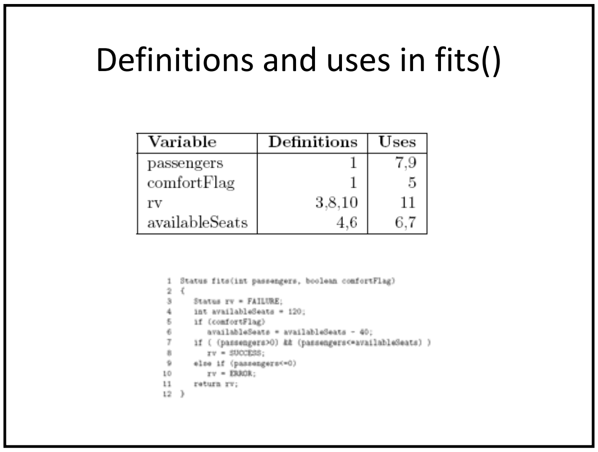


| ID | Test<br>Cases Covered | Inputs | | Exp. Output |
|---|---|---|---|---|
| | | passengers | comfortFlag | return value |
| T6.1 | BCb,d,f | 40 | false | SUCCESS |
| T6.2 | BCa,c,e,h,i | -100 | true | ERROR |
| T6.3 | BC [b,e] g | 200 | false | FAILURE |

## Test Data for Path Coverage for fits()



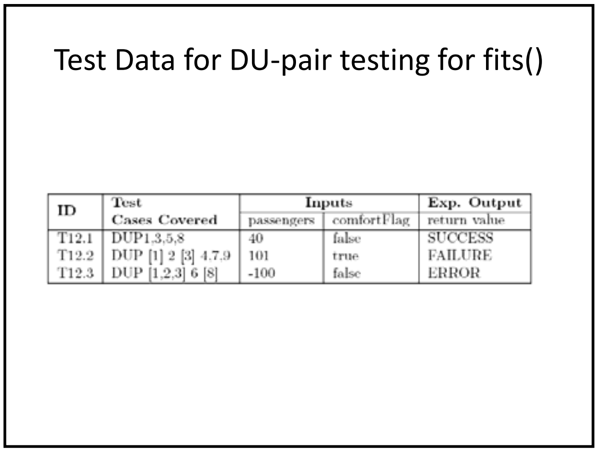
| ID | Test Cases Covered | Inputs | | Exp. Output |
|---|---|---|---|---|
| | | passengers | comfortFlag | return value |
| T11.1 | P1 | 40 | true | SUCCESS |
| T11.2 | P2 | 40 | false | SUCCESS |
| T11.3 | P3 | 101 | true | FAILURE |
| T11.4 | P4 | 200 | false | FAILURE |
| T11.5 | P5 | -100 | true | ERROR |
| T11.6 | P6 | -100 | false | ERROR |



## Definitions and uses in fits()

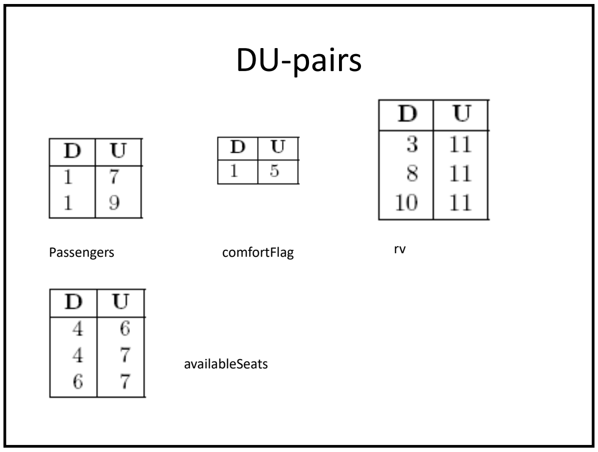
| Variable | Definitions | Uses |
|---|---|---|
| passengers | 1 | 7,9 |
| comfortFlag | 1 | 5 |
| rv | 3,8,10 | 11 |
| availableSeats | 4,6 | 6,7 |

```
 1 Status fits(int passengers, boolean comfortFlag)
 2 {
 3    Status rv = FAILURE;
 4    int availableSeats = 120;
 5    if (comfortFlag)
 6       availableSeats = availableSeats - 40;
 7    if ( (passengers>0) && (passengers<=availableSeats) )
 8       rv = SUCCESS;
 9    else if (passengers<=0)
10       rv = ERROR;
11    return rv;
12 }
```



## Test Data for DU-pair testing for fits()

| ID | Test<br>Cases Covered | Inputs | | Exp. Output |
|---|---|---|---|---|
| | | passengers | comfortFlag | return value |
| T12.1 | DUP1,3,5,8 | 40 | false | SUCCESS |
| T12.2 | DUP [1] 2 [3] 4.7,9 | 101 | true | FAILURE |
| T12.3 | DUP [1,2,3] 6 [8] | -100 | false | ERROR |



## DU-pairs

| D | U |
|---|---|
| 1 | 7 |
| 1 | 9 |

Passengers

| D | U |
|---|---|
| 1 | 5 |

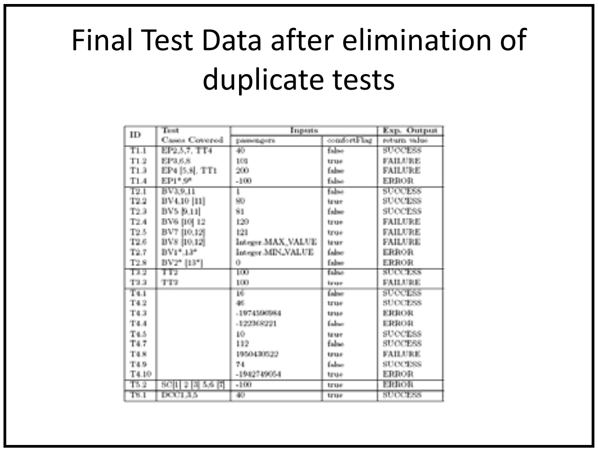
comfortFlag

| D | U |
|---|---|
| 3 | 11 |
| 8 | 11 |
| 10 | 11 |

rv

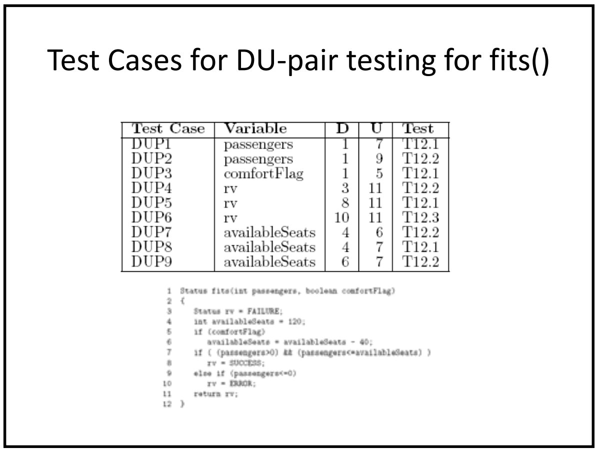
| D | U |
|---|---|
| 4 | 6 |
| 4 | 7 |
| 6 | 7 |

availableSeats



## Final Test Data after elimination of duplicate tests

| ID | Test<br>Cases Covered | Inputs | | Exp. Output |
|---|---|---|---|---|
| | | passengers | comfortFlag | return value |
| T1.1 | EP2,5,7, TT4 | 40 | false | SUCCESS |
| T1.2 | EP3,6,8 | 101 | true | FAILURE |
| T1.3 | EP4 [5,8], TT1 | 200 | false | FAILURE |
| T1.4 | EP1*,9* | -100 | false | ERROR |
| T2.1 | BV3,9,11 | 1 | false | SUCCESS |
| T2.2 | BV4,10 [11] | 80 | true | SUCCESS |
| T2.3 | BV5 [9,11] | 81 | false | SUCCESS |
| T2.4 | BV6 [10] 12 | 120 | true | FAILURE |
| T2.5 | BV7 [10,12] | 121 | true | FAILURE |
| T2.6 | BV8 [10,12] | Integer.MAX_VALUE | true | FAILURE |
| T2.7 | BV1*,13* | Integer.MIN_VALUE | false | ERROR |
| T2.8 | BV2* [13*] | 0 | false | ERROR |
| T3.2 | TT2 | 100 | false | SUCCESS |
| T3.3 | TT3 | 100 | true | FAILURE |
| T4.1 | | 16 | false | SUCCESS |
| T4.2 | | 46 | true | SUCCESS |
| T4.3 | | -1974596984 | true | ERROR |
| T4.4 | | -122368221 | false | ERROR |
| T4.5 | | 10 | true | SUCCESS |
| T4.7 | | 112 | false | SUCCESS |
| T4.8 | | 1950430522 | true | FAILURE |
| T4.9 | | 74 | false | SUCCESS |
| T4.10 | | -1942749054 | true | ERROR |
| T5.2 | SC[1] 2 [3] 5,6 [7] | -100 | true | ERROR |
| T8.1 | DCC1,3,5 | 40 | true | SUCCESS |



## Test Cases for DU-pair testing for fits()

| Test Case | Variable | D | U | Test |
|---|---|---|---|---|
| DUP1 | passengers | 1 | 7 | T12.1 |
| DUP2 | passengers | 1 | 9 | T12.2 |
| DUP3 | comfortFlag | 1 | 5 | T12.1 |
| DUP4 | rv | 3 | 11 | T12.2 |
| DUP5 | rv | 8 | 11 | T12.1 |
| DUP6 | rv | 10 | 11 | T12.3 |
| DUP7 | availableSeats | 4 | 6 | T12.2 |
| DUP8 | availableSeats | 4 | 7 | T12.1 |
| DUP9 | availableSeats | 6 | 7 | T12.2 |

```
 1 Status fits(int passengers, boolean comfortFlag)
 2 {
 3    Status rv = FAILURE;
 4    int availableSeats = 120;
 5    if (comfortFlag)
 6       availableSeats = availableSeats - 40;
 7    if ( (passengers>0) && (passengers<=availableSeats) )
 8       rv = SUCCESS;
 9    else if (passengers<=0)
10       rv = ERROR;
11    return rv;
12 }
```