Software Testing and Validation

MEIC

Autores:
Afonso Matos (103479)
Isabela de Ornelas (102703)
Tiago Deane (103811)

Group 28

2024/2025 – 2nd Semester, P4

Contents

1 Client class scope test cases

- **Test design strategy: Class Invariant Model**

Invariants Boundaries:

- $name.length() \leq 40$
- $1 \leq numberOfTerminals() \leq 9$
- $0 \leq points \leq 200$
- $numberOfFriends() \leq 5 * numberOfTerminals() - 3$

Constraint			Test Cases											
Variable	Condition		1	2	3	4	5	6	7	8	9	10	11	12
name.length()	<=40	ON	40											
		OFF		41										
	Typical	IN			0	1	2	3	4	5	6	7	8	9
numberOfTerminals()	>=1	ON			1									
		OFF				0								
	<=9	ON					9							
		OFF						10						
	Typical	IN	2	3					4	6	7	8	5	5
points	>=0	ON							0					
		OFF								-1				
	<=200	ON									200			
		OFF										201		
	Typical	IN	1	2	3	4	5	6					7	8
numberOfFriends()	<=5 * numberOfTerminals() - 3	ON											22	
		OFF												23
	Typical	IN	2	6	1	0	4	5	10	7	8	9		
Expected Result			Valid	Invalid	Valid	Invalid	Valid	Invalid	Valid	Invalid	Valid	Invalid	Valid	Invalid

Figure 1: Client Invariant Boundaries

Notes:

- In test case 3, numberOfFriends() is 1 because if it was 2 that would make it an On point $(5 * 1 - 3 = 2)$
- In test cases 11 and 12, numberOfTerminals() is 5 in order for numberOfFriends() = 22 to be an On point and 23 be an Off point

2 Terminal class scope test cases

- **Test design strategy: FSM Model**

Step 1: **State Model**

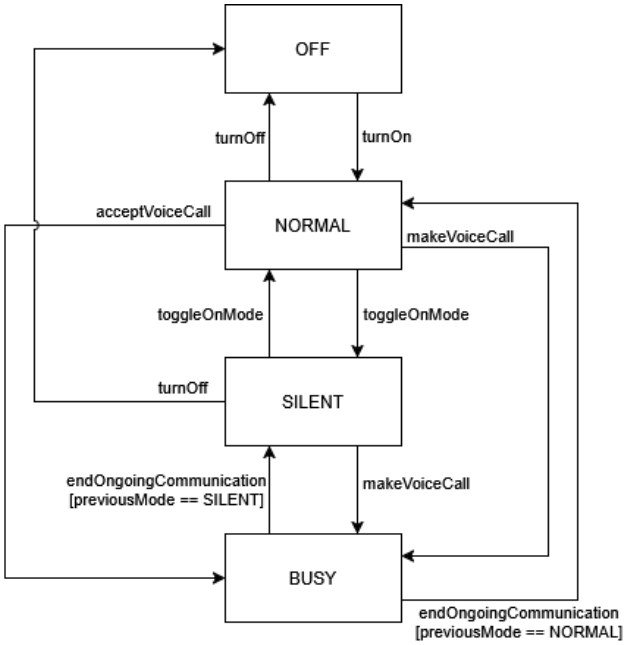

Figure 2: Terminal State Model

Step 2: **Full expansion of conditional transition variants**

State	Event	Condition	Next State
BUSY	endOngoingCommunication	Pre: [previousMode = NORMAL]	NORMAL
BUSY	endOngoingCommunication	Pre: [previousMode = SILENT]	SILENT

Figure 3: Terminal Conditional Transitions Table

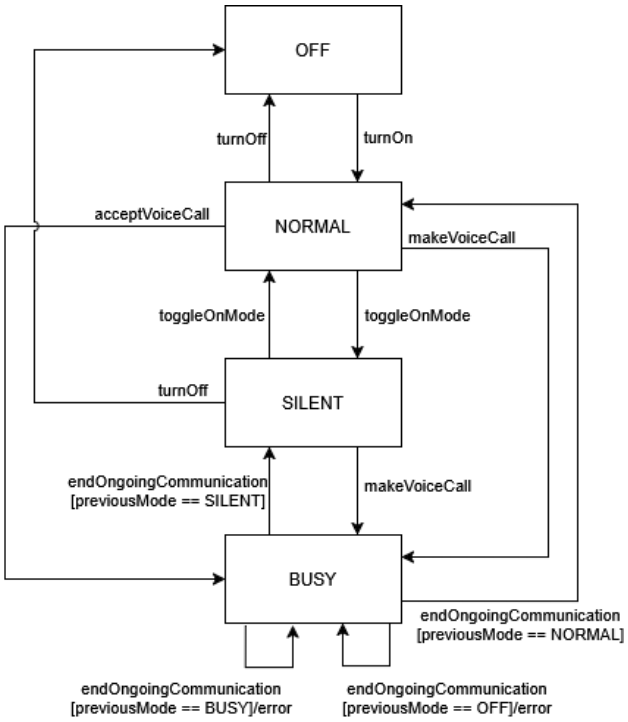

Figure 4: Updated Terminal State Model

Step 3: **Transition Tree**

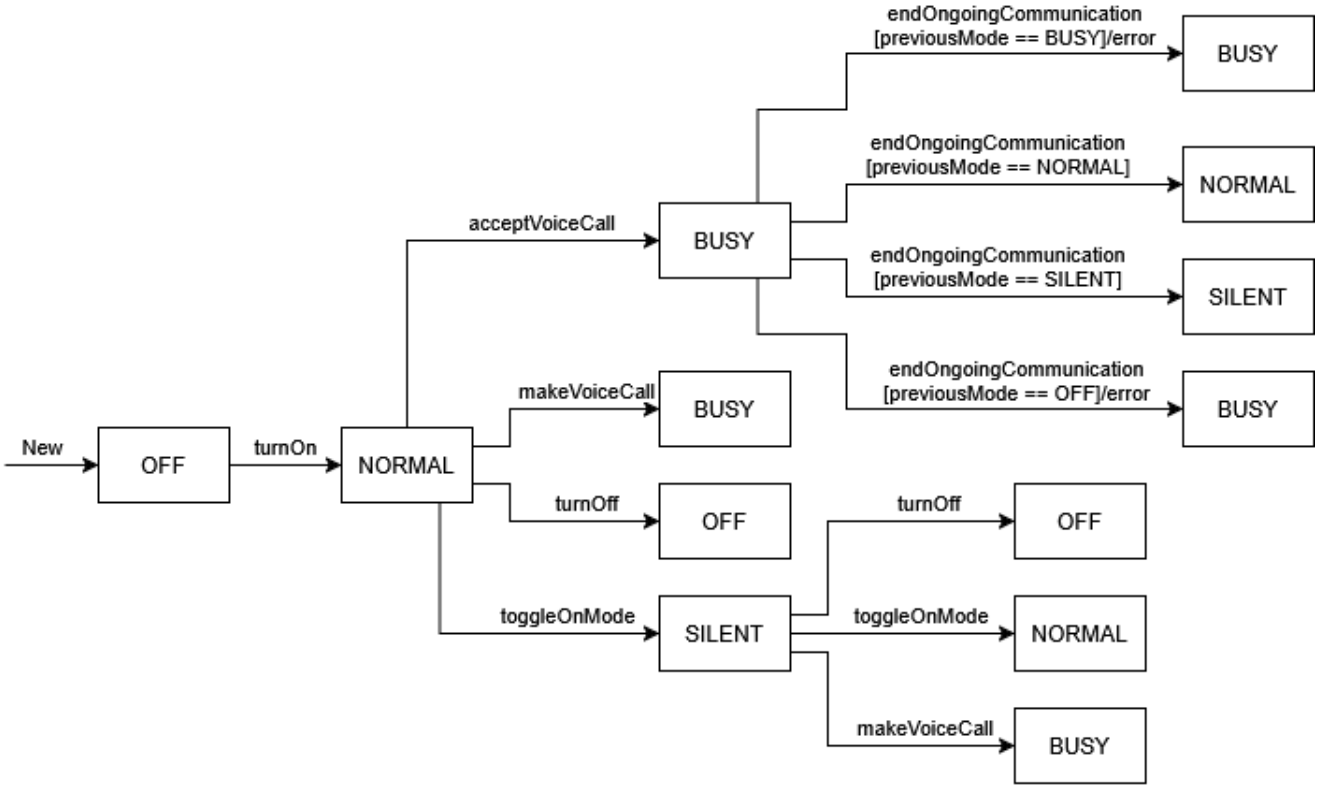

Figure 5: Terminal Transition Tree

Step 4: **Conformance Test Suite**

RUN	Test Run/Event Path				Expected Terminal State	Exception
	Level 1	Level 2	Level 3	Level 4		
1	new				OFF	-
2	new	turnOn			NORMAL	-
3	new	turnOn	turnOff		OFF	-
4	new	turnOn	toggleOnMode		SILENT	-
5	new	turnOn	toggleOnMode	turnOff	OFF	-
6	new	turnOn	toggleOnMode	toggleOnMode	NORMAL	-
7	new	turnOn	toggleOnMode	makeVoiceCall	BUSY	-
8	new	turnOn	acceptVoiceCall		BUSY	-
9	new	turnOn	acceptVoiceCall	endOngoingCommunication [previousMode == SILENT]	SILENT	-
10	new	turnOn	acceptVoiceCall	endOngoingCommunication [previousMode == NORMAL]	NORMAL	-
11	new	turnOn	acceptVoiceCall	endOngoingCommunication [previousMode == BUSY]/error	BUSY	YES
12	new	turnOn	acceptVoiceCall	endOngoingCommunication [previousMode == OFF]/error	BUSY	YES
13	new	turnOn	makeVoiceCall		BUSY	-

Figure 6: Terminal Conformance Test Suite

Step 5: **Test Data using Invariant Boundaries**

endOngoingCommunication in BUSY		
Condition	On point	Off point
[previousMode = NORMAL]	NORMAL	OFF, BUSY
[previousMode = SILENT]	SILENT	OFF, BUSY

Figure 7: Terminal Invariant Boundaries

Step 6: "Execute conformance test suite until all tests pass"

Step 7: **Sneak Path Test Suite**

Events	States			
	OFF	NORMAL	SILENT	BUSY
turnOn	Valid	PSP	PSP	PSP
turnOff	PSP	Valid	Valid	PSP
toggleOnMode	PSP	Valid	Valid	PSP
acceptVoiceCall	PSP	Valid	PSP	PSP
makeVoiceCall	PSP	Valid	Valid	PSP
endOngoingCommunication	PSP	PSP	PSP	Conditional

PSP = Possible Sneak Path

Figure 8: Terminal Sneak Path Table

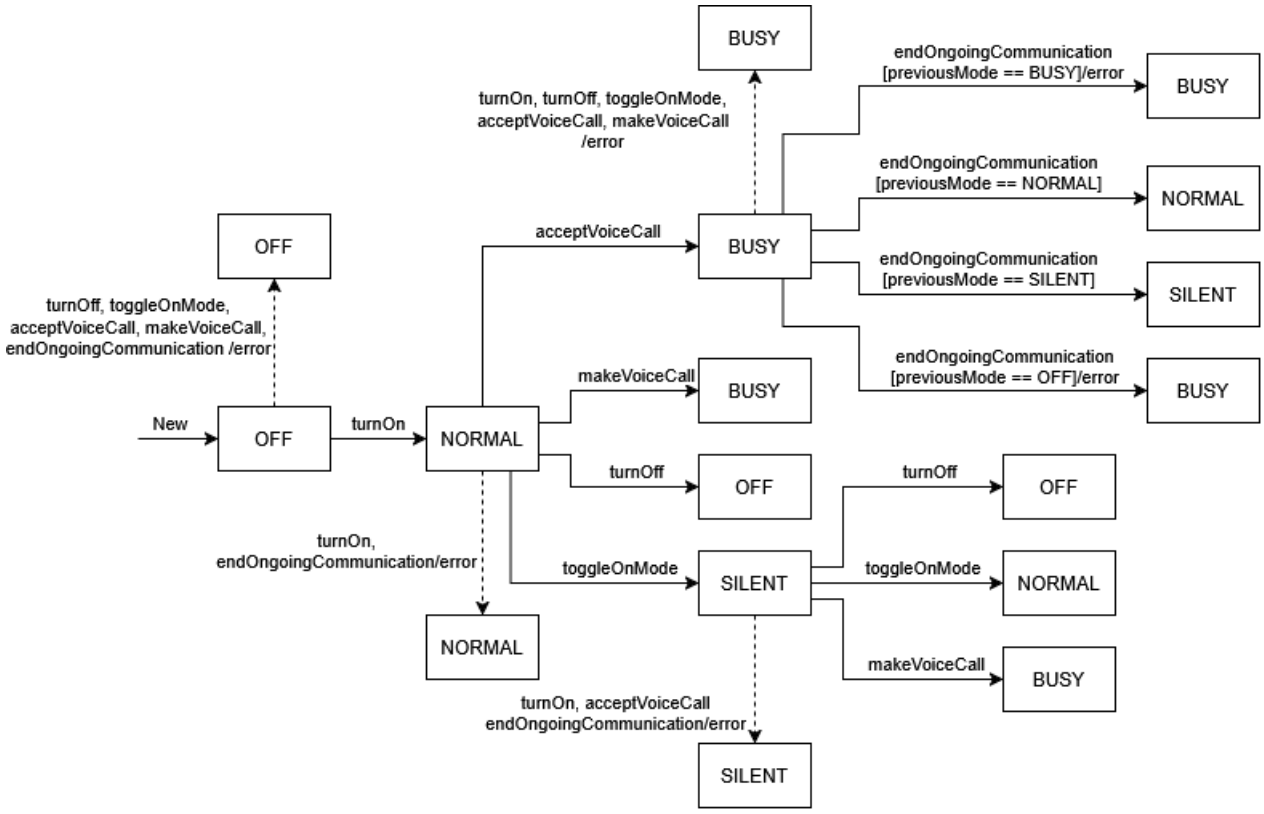

Figure 9: Terminal Transition Tree With Sneak Paths

RUN	Test Run/Event Path				Expected Terminal State	Exception
	Level 1	Level 2	Level 3	Level 4		
14	new	turnOff			OFF	Yes
15	new	toggleOnMode			OFF	Yes
16	new	acceptVoiceCall			OFF	Yes
17	new	makeVoiceCall			OFF	Yes
18	new	endOngoingCommunication			OFF	Yes
19	new	turnOn	turnOn		ON	Yes
20	new	turnOn	endOngoingCommunication		ON	Yes
21	new	turnOn	toggleOnMode	turnOn	SILENT	Yes
22	new	turnOn	toggleOnMode	acceptVoiceCall	SILENT	Yes
23	new	turnOn	toggleOnMode	endOngoingCommunication	SILENT	Yes
24	new	turnOn	acceptVoiceCall	turnOn	BUSY	Yes
25	new	turnOn	acceptVoiceCall	turnOff	BUSY	Yes
26	new	turnOn	acceptVoiceCall	toggleOnMode	BUSY	Yes
27	new	turnOn	acceptVoiceCall	acceptVoiceCall	BUSY	Yes
28	new	turnOn	acceptVoiceCall	makeVoiceCall	BUSY	Yes

Figure 10: Terminal Sneak Path Test Suite

3 computeCost() method scope test cases

- **Test design strategy: Combinational Function**

Decision Tree:

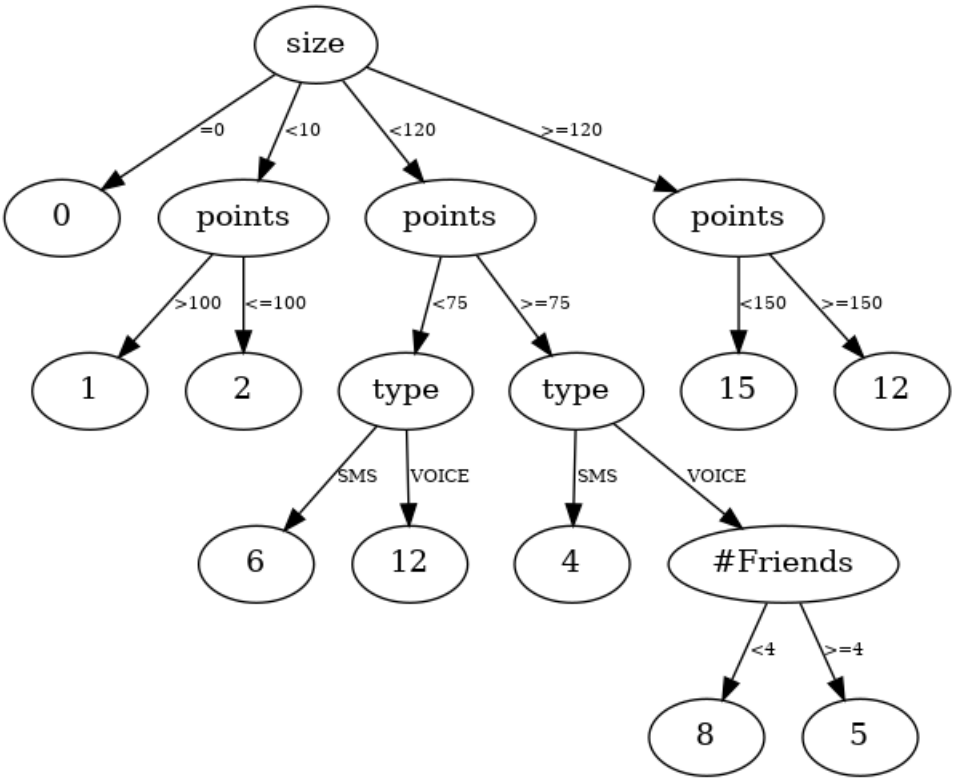

Figure 11: computeCost() Decision Tree

Variables:

Variable	Condition	Return Value
v0	size = 0	0
v1	size < 10 ∧ points > 100	1
v2	size < 10 ∧ points <= 100	2
v3	size < 120 ∧ points < 75 ∧ type = SMS	6
v4	size < 120 ∧ points < 75 ∧ type = VOICE	12
v5	size < 120 ∧ points >= 75 ∧ type = SMS	4
v6	size < 120 ∧ points >= 75 ∧ type = VOICE ∧ #Friends < 4	8
v7	size < 120 ∧ points >= 75 ∧ type = VOICE ∧ #Friends >= 4	5
v8	size >= 120 ∧ points < 150	15
v9	size >= 120 ∧ points >= 150	12

Figure 12: computeCost() Decision Tree Variables

Domain testing for each variable:

Constraint			Test Cases		
Variable	Condition		1	-	-
size	=0	ON	0		
		OFF		1	
		OFF			-1
points	IN		110	120	130
type	IN		SMS	VOICE	SMS
#Friends	IN		0	1	2
Expected Result			0	v1	Impossible

Figure 13: computeCost() Decision Tree variable v0

Constraint			Test Cases			
Variable	Condition		-	2	-	3
size	<10	ON	10			
		OFF		9		
	IN				1	2
points	>100	ON			100	
		OFF				101
	IN		110	120		
type	IN		SMS	VOICE	SMS	VOICE
#Friends	IN		0	1	2	3
Expected Result			v5	1	v2	1

Figure 14: computeCost() Decision Tree variable v1

Constraint			Test Cases			
Variable	Condition		-	4	5	-
size	<10	ON	10			
		OFF		9		
	IN				1	2
points	<=100	ON			100	
		OFF				101
	IN		20	30		
type	IN		SMS	VOICE	SMS	VOICE
#Friends	IN		0	1	2	3
Expected Result			v3	2	2	v1

Figure 15: computeCost() Decision Tree variable v2

Constraint			Test Cases					
Variable	Condition		-	6	-	7	8	-
size	< 120	ON	120					
		OFF		119				
	IN				110	100	80	90
points	< 75	ON			75			
		OFF				74		
	IN		10	30			40	20
type	= SMS	ON					SMS	
		OFF						VOICE
	IN		SMS	SMS	SMS	SMS		
#Friends	IN		0	1	2	3	4	5
Expected Result			v8	6	v5	6	6	v4

Figure 16: computeCost() Decision Tree variable v3

Constraint			Test Cases					
Variable	Condition		-	9	-	10	11	-
size	< 120	ON	120					
		OFF		119				
	IN				110	100	90	80
points	< 75	ON			75			
		OFF				74		
	IN		70	60			20	40
type	= VOICE	ON					VOICE	
		OFF						SMS
	IN		VOICE	VOICE	VOICE	VOICE		
#Friends	IN		0	1	2	3	4	5
Expected Result			v8	12	v6	12	12	v3

Figure 17: computeCost() Decision Tree variable v4

Constraint			Test Cases					
Variable	Condition		-	12	13	-	14	-
size	< 120	ON	120					
		OFF		119				
	IN				110	100	15	80
points	>= 75	ON			75			
		OFF				74		
	IN		80	85			110	100
type	= SMS	ON					SMS	
		OFF						VOICE
	IN		SMS	SMS	SMS	SMS		
#Friends	IN		0	1	2	3	5	4
Expected Result			v8	4	4	v3	4	v7

Figure 18: computeCost() Decision Tree variable v5

Constraint			Test Cases							
Variable	Condition		-	15	16	-	17	-	-	18
size	< 120	ON	120							
		OFF		119						
	IN				110	100	95	15	80	40
points	>=75	ON			75					
		OFF				74				
	IN		180	85			90	110	100	80
type	=VOICE	ON					VOICE			
		OFF						SMS		
	IN		VOICE	VOICE	VOICE	VOICE			VOICE	VOICE
#Friends	<4	ON							4	
		OFF								3
	IN		0	1	2	0	1	2		
Expected Result			v9	8	8	v4	8	v5	v7	8

Figure 19: computeCost() Decision Tree variable v6

Constraint			Test Cases							
Variable	Condition		-	19	20	-	21	-	22	-
size	< 120	ON	120							
		OFF		119						
	IN				110	100	95	15	80	40
points	>=75	ON			75					
		OFF				74				
	IN		180	85			90	110	100	80
type	=VOICE	ON					VOICE			
		OFF						SMS		
	IN		VOICE	VOICE	VOICE	VOICE			VOICE	VOICE
#Friends	>=4	ON							4	
		OFF								3
	IN		5	6	7	8	9	10		
Expected Result			v9	5	5	v4	5	v5	5	v6

Figure 20: computeCost() Decision Tree variable v7

Constraint			Test Cases			
Variable	Condition		23	-	-	24
size	>=120	ON	120			
		OFF		119		
	IN				130	140
points	< 150	ON			150	
		OFF				149
	IN		10	85		
type	IN		SMS	SMS	VOICE	VOICE
#Friends	IN		0	1	2	3
Expected Result			15	v5	v9	15

Figure 21: computeCost() Decision Tree variable v8

Constraint			Test Cases			
Variable	Condition		25	-	26	-
size	>=120	ON	120			
		OFF		119		
	IN				130	140
points	>=150	ON			150	
		OFF				149
	IN		180	170		
type	IN		VOICE	SMS	VOICE	SMS
#Friends	IN		0	1	2	3
Expected Result			12	v5	12	v8

Figure 22: computeCost() Decision Tree variable v9

4 removeTerminal() method scope test cases

- **Test design strategy: Category-Partition**

Functions:

Functions:	IN	OUT
remove terminal if possible	t, Terminal list, Friend list	list, bool
throw exception if would put Client in invalid state	t, Terminal list, Friend list	exception

Figure 23: removeTerminal() functions

Category-Choice table:

Parameter	Category	Choices
Terminal t	invalid	null, t.balance() < 0
	t in list	t1
	t not in list	tx
Terminal list	empty	{}
	full	{t1,...,t9}
	holding	{t1}, {t1,...,t8}, {t1,...,tn}, 1 < n < 8
friends list	empty	{}
	full	{f1,...,fm}, m = 5 * numberOfTerminals() - 3
	holding	{f1}, {f1,...,fm}, m <= 5 * (numberOfTerminals() - 1) - 3
	special case	{f1,...,fm}, 5 * (numberOfTerminals() - 1) - 3 < m < 5 * numberOfTerminals() - 3

Figure 24: removeTerminal() Category Choice Table

Constraints:

- If Terminal t1 is in Terminal list, then Terminal list can't be empty
- A null Terminal precludes the same response for all other values
- A Terminal with negative balance precludes the same response for all other values

Test suite:

	Input		Output			
TC	Terminal t	Terminal list	Friend list	Terminal list	bool	exception
1	null	{t1,..,tn}, 1 <n <8	{f1}	{t1,..,tn}, 1 <n <8	FALSE	-
2	t =>t.balance() < 0	{t1,..,tn}, 1 <n <8	{f1,f2}	{t1,..,tn}, 1 <n <8	FALSE	-
3	t1	{t1,t2,..,t9}	{}	{t2,..,t9}	TRUE	-
4	t1	{t1,..,t9}	{f1,..,f42}	{t1,..,t9}	-	YES
5	t1	{t1,t2,..,t9}	{f1}	{t2,..,t8}	TRUE	-
6	t1	{t1,t2,..,t9}	{f1,..,f10}	{t2,..,t8}	TRUE	-
7	t1	{t1,..,t9}	{f1,..,f38}	{t1,..,t9}	-	YES
8	t1	{t1}	{f1}	{t1}	-	YES
9	t1	{t1,t2,..,t6}	{f1,..,f4}	{t2,..,t6}	TRUE	-
10	t1	{t1,..,t6}	{f1,..,23}	{t1,..,t6}	-	YES
11	tx	{t1,..,t9}	{f1,..,f4}	{t1,..,t9}	FALSE	-
12	tx	{t1}	{}	{t1}	FALSE	-
13	tx	{t1,..,t8}	{f1,..,f10}	{t1,..,t8}	FALSE	-
14	tx	{t1,t2,..,t4}	{f1,..,f17}	{t1,t2,..,t4}	FALSE	-

Figure 25: removeTerminal() Method Scope Tests

5 Client class scope TestNG test cases

- Chosen SUCCESS test cases: 1, 5, 7 and 11
- Chosen FAILURE test cases: 2, 4, 10 and 12

Note: the following tests are on the file TestClient.java

```
Afonso Matos, 12 minutes ago | 1 author (Afonso Matos)
@Test
public class TestClient {
    private Terminal baseTerminal;

    @BeforeMethod private void setup() {
        baseTerminal = new Terminal("1234");
    }

    /**
     * Chosen SUCCESS test cases: 1, 5, 7 and 11
     **/

    @DataProvider
    public Object[][] computeDataForValidClient() {
        return new Object[][] {
            // name, numTerminals, points, numFriends
            { "A".repeat(40), 2, 1, 2 },  // ON: name length = 40
            { "12", 9, 5, 4 },            // ON: numTerminals = 9
            { "1234", 4, 0, 10 },         // ON: points = 0
            { "12345678", 5, 7, 22 }      // ON: numFriends = 5 * numTerminals - 3
        };
    }

    @Test(dataProvider = "computeDataForValidClient")
    public void testValidClient(String name, int numTerminals, int points, int numFriends) {
        // Arrange
        Client client;

        // Act
        client = new Client(name, taxNumber:12345, baseTerminal);

        client.updateName(name);
        client.updatePoints(points);
        Terminal terminal;
        for (int i = 1; i < numTerminals; i++) {
            terminal = new Terminal("T" + i);
            client.addTerminal(terminal);
            terminal.setClient(client);
        }

        for (int i = 0; i < numFriends; i++) {
            client.addFriend(new Client("F" + i, taxNumber:1, new Terminal("TF" + i)));
        }

        // Assert
        assertEquals(client.getName(), name);
        assertEquals(client.getPoints(), points);
        assertEquals(client.numberOfTerminals(), numTerminals);
        assertEquals(client.numberOfFriends(), numFriends);
    }
```

Figure 26: Chosen SUCCESS test cases

```java
    /**
     * Chosen FAILURE test cases: 2, 4, 10 and 12
     **/

    @Test // Test case 2
    public void testClientWithNameTooLong() {
        // Arrange
        String name = "A".repeat(41); // Off point

        // Act + Assert
        assertThrows(throwableClass:IllegalArgumentException.class, () -> {
            new Client(name, taxNumber:12345, baseTerminal);
        });
        assertThrows(throwableClass:InvalidOperationException.class, () -> {
            (new Client(name:"A", taxNumber:12345, baseTerminal)).updateName(name);
        });
    }

    @Test // Test case 4
    public void testClientWithNoTerminals() {
        // Arrange
        String name = "A";
        Client client = new Client(name, taxNumber:12345, baseTerminal);

        // Act + Assert
        assertThrows(throwableClass:InvalidOperationException.class, () -> {
            client.removeTerminal(baseTerminal); // numTerminals = 0 is Off point
        });
    }

    @Test // Test case 10
    public void testClientWithTooManyPoints() {
        // Arrange
        Client client = new Client(name:"1234567", taxNumber:12345, baseTerminal);

        // Act + Assert
        assertThrows(throwableClass:InvalidOperationException.class, () -> {
            client.updatePoints(p:201); // Off point
        });
    }
```

```java
    @Test // Test case 12
    public void testClientWithTooManyFriends() {
        // Arrange
        Client client = new Client(name:"123456789", taxNumber:12345, baseTerminal);
        int numTerminals = 5;
        int numFriends = 23; // Off point with numTerminals = 5
        Terminal  terminal ;
        for (int i = 1; i < numTerminals; i++) {
            terminal = new Terminal("TF" + i);
            client.addTerminal(terminal);

        }

        // Act + Assert
        assertThrows(throwableClass:InvalidOperationException.class, () -> {
            for (int i = 0; i < numFriends; i++) { // Off point: 23 > 5*5 - 3 = 22
                client.addFriend(new Client("F" + i, taxNumber:1, new Terminal("TF" + i)));
            }
        });
    }
}
```

Figure 27: Chosen FAILURE test cases