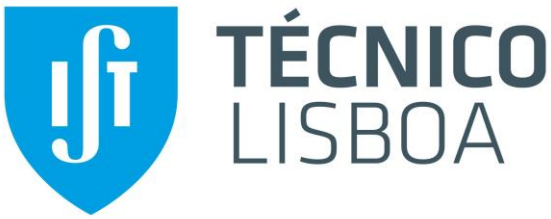


Testes e Validação de Software

Instituto Superior Técnico

Project Report

Group 27

Rui Peres 73831
Ricardo Abreu 76370
Rui Silva 77271

Habitat - Class Level Testing

Invariant Boundaries

1. Define the class invariant, responsibility-based assertions;

- Animals abstract states: allSameSpecies, allBelongZoo(habitat.zoo)
- area: positive integer type
- #animals: positive integer type
- maxAnimals: positive integer type

assert(area / #animals >= 4 **&&** #animals <= maxAnimals **&&** #animals >= 1 **&&** animals.allSameSpecies() **&&** animals.allBelongZoo(habitat.zoo)**)**

2. Develop on points and off points for each condition in the invariant using the 1x1 selection criteria of domain model;

Condition	On	Off
area / #animals >= 4	4	3
#animals <= maxAnimals	maxAnimals	maxAnimals + 1
#animals >= 1	1	0
animals.allSameSpecies()	[1]	[2]
animals.allBelongZoo(habitat.zoo)	[3]	[4]

[1] - $\forall\ animal: a1, a2 \mid a1.species = a2.species$
[2] - $\exists\ animal: a1, a2 \mid a1.species \neq a2.species$
[3] - $\forall\ animal: a1 \in habitat \mid a1.zoo = habitat.zoo$
[4] - $\exists\ animal: a1 \in habitat \mid a1.zoo \neq habitat.zoo$

3. Complete the test suite by developing in points for the variables not referenced in a condition and represent the results in a domain matrix

Constraint		**Test Cases**													
Variable	Condition		1	2	3	4	5	6	7	8	9	10	11	12	13
maxAnimals	>= 1	On	1												
		Off		0											
	Typical	In			10	20	50	39	67		567	15	26	64	17
area/#animals	>= 4	On			4										
		Off				3									
	Typical	In	10	✗			10	20	87	✗	10	11	12	13	14
#animals	<= maxAnimals	On					maxAnimals								
		Off						maxAnimals + 1							
	>= 1	On							1						
		Off								0					
	Typical	In	1	✗	10	7					2	3	4	5	6
animals	allSameSpecies()	On									aS				
		Off										nS			
	allBelongZoo(habitat.zoo)	On											aB		
		Off												nB	
	Typical	In	Z	Z	Z	Z	Z	Z	Z	Z					Z
Expected Result			✓	✗✗	✓	✗	✓	✗	✓	✗✗	✓	✗	✓	✗	✓

Legend

Z: allSame & allBelong

aS: allSame

nS: notAllSame

aB: allBelong

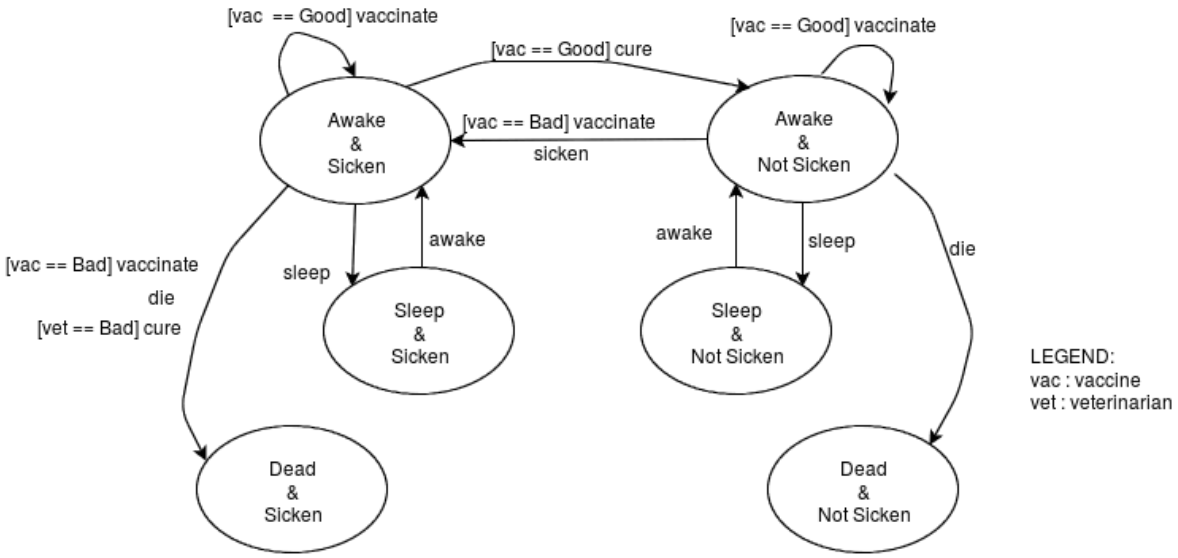
Animal - Class level Testing

Modal Class Test

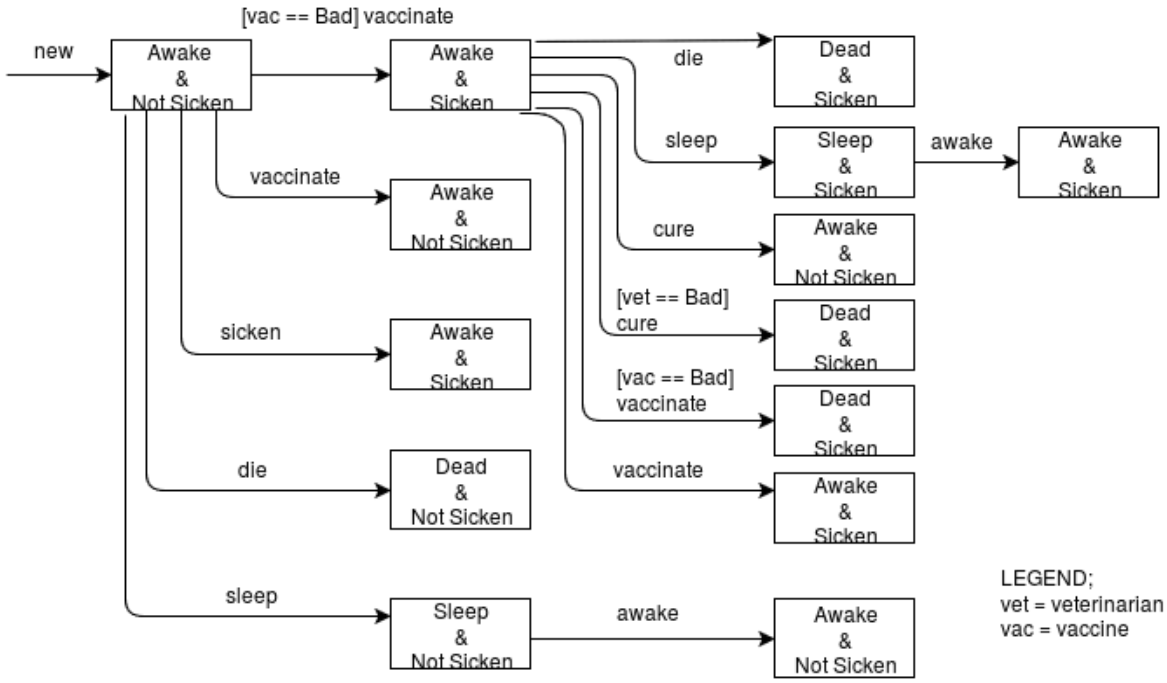
1. Develop FREE state model for CUT.

In the following diagram we consider GOOD and BAD on our guards so that it facilitates the reading.

- Vet == GOOD : veterinarian knows species
- Vet == BAD : veterinarian doesn't know species
- Vac == GOOD : vaccine appropriate to species
- Vac == BAD : vaccine not appropriate to species



2. Elaborate the transition tree with a full expansion of conditional transition variants



3. Generate Conditional Tests

State	Message	Condition	Next State
Awake ∧ ~Sick	vaccinate	Pre: Appropriate to species	Awake ∧ ~Sick
Awake ∧ ~Sick	vaccinate	Pre: Not appropriate to species	Awake ∧ Sick
Awake ∧ Sick	vaccinate	Pre: Appropriate to species	Awake ∧ Sick
Awake ∧ Sick	vaccinate	Pre: Not appropriate to species	Dead ∧ Sick
Awake ∧ Sick	cure	Pre: Knows how to cure species	Awake ∧ ~Sick
Awake ∧ Sick	cure	Pre: Doesn't know how to cure species	Dead ∧ Sick

4. Generate Conformance Test Suite

Test case	Test Run / Event Path				Expected Terminal State	Exception
	Level 1	Level 2	Level 3	Level 4		
1	new				Awake & Not Sicken	-
2	new	vaccinate			Awake & Not Sicken	-
3	new	vaccinate(BAD)			Awake & Sicken	-
4	new	sicken			Awake & Sicken	-
5	new	sleep			Sleep & Not Sicken	-
6	new	die			Dead & Not Sicken	-
7	new	vaccinate(BAD)	die		Dead & Sicken	-
8	new	vaccinate(BAD)	sleep		Sleep & Sicken	-
9	new	vaccinate(BAD)	cure		Awake & Not Sicken	-
10	new	vaccinate(BAD)	cure(BAD)		Dead & Sicken	-
11	new	vaccinate(BAD)	vaccinate		Awake & Sicken	-
12	new	vaccinate(BAD)	vaccinate(BAD)		Dead & Sicken	-
13	new	sleep	awake		Awake & Not Sicken	-
14	new	vaccinate(BAD)	sleep	awake	Awake & Sicken	-

5. Develop test data for each path using Invariant Boundaries pattern for events, messages and actions

Invariant Boundaries for vaccinate		
	On Point	**Off Point**
Vaccine appropriate to species	Appropriate to species	Not appropriate to species

Invariant Boundaries for cure		
	On Point	**Off Point**
Veterinarian knows how to cure species	Knows how to cure	Doesn't know how to cure

6. Develop a sneak path test suite. Add all forbidden transitions in all states and define the expected exception

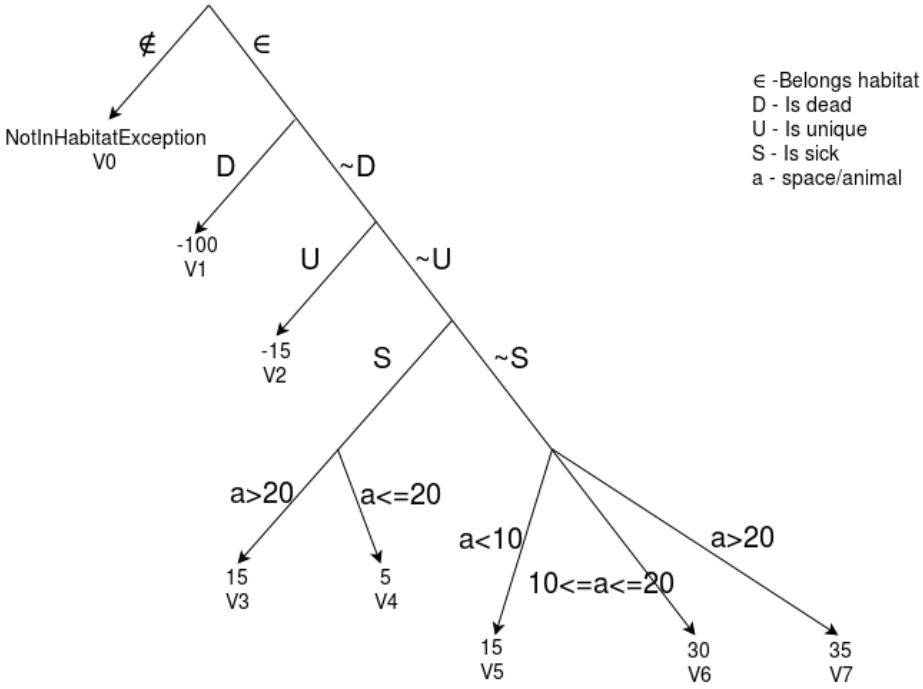
Events	**States**					
	Awake ∧ Sick	Awake ∧ ~Sick	Sleep ∧ Sick	Sleep ∧ ~Sick	Dead ∧ Sick	Dead ∧ ~Sick
awake	PSP	PSP	➳	➳	PSP	PSP
cure	➳	PSP	PSP	PSP	PSP	PSP
vaccinate	➳	➳	PSP	PSP	PSP	PSP
sleep	➳	➳	PSP	PSP	PSP	PSP
die	➳	➳	PSP	PSP	PSP	PSP
sicken	PSP	➳	PSP	PSP	PSP	PSP

LEGEND:
➳: Valid Transition
PSP : Possible Sneak Path

getSatisfaction() - Method Level Tests

Combinational Functional Test

Decision tree:



Decision table:

	Variant							
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Condition								
Belongs habitat	F	T	T	T	T	T	T	T
Is dead	DC	T	F	F	F	F	F	F
Is alone	DC	DC	T	F	F	F	F	F
Is sick	DC	DC	DC	T	T	F	F	F
area/animal < 10	DC	DC	DC	DC	DC	T	DC	F
area/animal > 20	DC	DC	DC	T	F	DC	T	F
Action								
Satisfaction	**-**	-100	-15	15	5	15	35	30
NotInHabitatException	✓	x	x	x	x	x	x	x

getSatisfaction() Domain Matrix (for each variant):

Variable	Condition	V0	Test Cases	
			1	2
animal	Belongs habitat	ON	T	
		OFF		F
		In		
	Is dead	ON		
		OFF		
		In	F	F
	Is unique	ON		
		OFF		
		In	F	F
	Is sick	ON		
		OFF		
		In	F	F
Area/animal	area/animal < 10	ON		
		OFF		
	area/animal > 20	ON		
		OFF		
	Typical	In	4	5
Expected			V5	NotInHabitatException

Variable	Condition	V1	Test Cases	
			1	2
animal	Belongs habitat	ON		
		OFF		
		In	T	T
	Is dead	ON	F	
		OFF		T
		In		
	Is unique	ON		
		OFF		
		In	F	F
	Is sick	ON		
		OFF		
		In	F	F
Area/animal	area/animal < 10	ON		
		OFF		
	area/animal > 20	ON		
		OFF		
	Typical	In	14	7
Expected			V6	-100

Variable	Condition	V2	Test Cases	
			1	2
animal	Belongs habitat	ON		
		OFF		
		In	T	T
	Is dead	ON		
		OFF		
		In	F	F
	Is unique	ON	F	
		OFF		T
		In		
	Is sick	ON		
		OFF		
		In	F	F
Area/animal	area/animal < 10	ON		
		OFF		
	area/animal > 20	ON		
		OFF		
	Typical	In	1231	47
Expected			V7	-15

Variable	Condition	V3	Test Cases	
			1	2
animal	Belongs habitat	ON		
		OFF		
		In	T	T
	Is dead	ON		
		OFF		
		In	F	F
	Is unique	ON		
		OFF		
		In	F	F
	Is sick	ON		T
		OFF		
		In	T	
Area/animal	area/animal < 10	ON		
		OFF		
	area/animal > 20	ON	20	
		OFF		21
	Typical	In		
Expected			V4	15

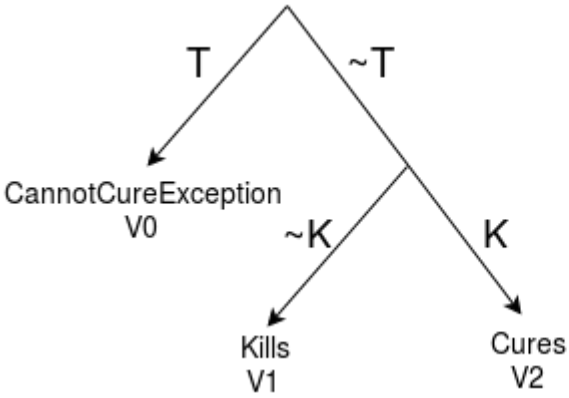
Variable	Condition	V6	Test Cases			
			1	2	3	4
animal	Belongs habitat	ON				
		OFF				
		In	T	T	T	T
	Is dead	ON				
		OFF				
		In	F	F	F	F
	Is unique	ON				
		OFF				
		In	F	F	F	F
	Is sick	ON	F			
		OFF				
		In		F	F	F
Area/animal	area/animal < 10	ON	10			
		OFF		9		
	area/animal > 20	ON			20	
		OFF				21
	Typical	In				
Expected Result			30	V5	30	V7

Variable	Condition	V7	Test Cases	
			1	2
animal	Belongs habitat	ON		
		OFF		
		In	T	T
	Is dead	ON		
		OFF		
		In	F	F
	Is unique	ON		
		OFF		
		In	F	F
	Is sick	ON	F	
		OFF		
		In		F
Area/animal	area/animal < 10	ON		
		OFF		
	area/animal > 20	ON	20	
		OFF		21
	Typical	In		
Expected Result			V6	35

cure() - Method Level Tests

Combinational Functional Test

Decision tree:



T - Veterinarian Is tired
K - Veterinarian knows species

Decision table:

	Variant		
	0	**1**	**2**
Condition			
Tired	T	F	F
Knows species	DC	F	T
Action			
CannotCureAnimalException	✓	x	x

cure() Domain Matrix (for each variant):

Variable	Condition	V0	Test Cases	
			1	2
Veterinarian	Is tired	ON	F	
		OFF		T
		In		
	Knows species	ON		
		OFF		
		In	T	T
Expected:				
	CannotCureAnimalException		V2	✓

Variable	Condition	V1	Test Cases	
			1	2
Veterinarian	Is tired	ON		
		OFF		
		In	F	F
	Knows species	ON	T	
		OFF		F
		In		
Expected:				
	CannotCureAnimalException		V2	x

Variable	Condition	V2	Test Cases	
			1	2
Veterinarian	Is tired	ON		
		OFF		
		In	F	F
	Knows species	ON	T	
		OFF		F
		In		
Expected:				
	CannotCureAnimalException		x	V1