ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

DRisk
Unitary Test Report

Version 1.0

Revision History

Date	Version	Description	Author
2015-01-13	0.01	Initial Draft	Oswaldo Bayona

1. Introduction

1.1 Purpose of this document

This documents presents the test case of the unitary test of the most critical parts of the project, and the results of the tests.

1.2 Document organization

The document is organized as follows:

- Section 1, *Introduction,* describes the purpose, the intended audience of this document and the methodology to do the test.
- Section 2, *Test Plan and results,* covers the execution results of the test plan and includes a list of all the test cases and test logs produced by the testing effort

1.3 Intended Audience

This document is intended for all the stakeholders of the DRisk project and their customers, and for anybody who wants to know how we tested the project.

1.4 Methodology

To do the unitary test we used the coverture of instructions (path's coverture), and calculate the cyclomatic complexity.

2. Test Cases and results:

The next functions were choose to be tested because are the most critical and important.

2.1 Test joinPlayer function:

This function joins one player to the Match, and do a broadcast to notify the players of the match

Function joinPlayer (idMatch, nickPlayer,sockets)

- idMatch: the id of the Match
- nickPlayer: the id of the player that will joined
- sockects: a list of the sockets objects to the player of the ga,e

Cyclomatic complexity = 2

Test cases:

To do this test we had to create a match so that the idMatch is 1, we used the nickPlayer “Oswaldo“.

ID	Input			Output	Coverture
	idMatch	nickPlayer	sockets	player	
DR-01	1	“Rodrigo“	Object	“Rodrigo“	Path 1
DR-02	1	“Oswaldo“	Object	Error: the player already exist.	Path 2

Test log:

```
joinPlayer
  ✓ shuld return the player when he is not joined in the match


1 passing (7ms)
```

2.2 Test function selectTerritory.validateMove function:

This function is called in the stage of select territory and it occurs when the player clicks over a territory. This function validate if the selected territory is valid to do de selection.

function validateMove(idTerritory, idPlayer)

-idTerritory: The id of the selected territory.

-idPlayer: The id of the player who perform the action.

return: True if the territory is valid else return False

Cyclomatic complexity = 2

Test cases:
To do this test the player with id “Oswaldo” creates a match and the player “Jorge” joins to this match.
The test is with the pre-load world map.

DR-04: Oswaldo clicks in Peru, Peru is a territory without owner.
DR-05: Jorge clicks in Peru, the Owner of Peru is Oswaldo.

ID	Input		Output	Coverture
	idTerritory	idPlayer	Value	
DR-04	“Peru“	“Oswaldo“	TRUE	Path1
DR-05	“Peru“	“Jorge“	FALSE	Path2

Test log:

```
selectTerritory
  validateMove
    ✓ Return true is the territory does not have owner


1 passing (8ms)
```

2.3 Test function changeCards.doUpdateMap

This function is called when a player changes three Risk cards, specifically when the server reply comes.

function doUpdateMap(flag, cardsTraced, idPlayer, numSoldier)

-flag: a Boolean variable, it means that the player did a change of three cards i.e. the player did not jump the stage.

- cardsTraced: the cards that were changed
- idPlayer: the id of the player who perform the action
- numSoldier: the number of soldiers received by the cards

Cyclomatic complexity = 3

Test Cases:

ID	Inputs					Outputs	Coverage
	flag	cardsTraced	idPlayer	numSodier	Turn		
DR-06	F	[ ]	Eloy	0	Eloy	No effect	Path1
DR-07	T	[card1, card2, card3]	Eloy	X	Eloy	Notification to the player	Path2
DR-08	T	[c1, c2, c3]	Eloy	X	Other	No effect	Path3

Test log:

```
changeCards
  doUpdateMap
    ✓ This function is called when a player changes three Risk cards


1 passing (11ms)
```

2.4: Test function calculateSoldiersByCards
This function calculate the number of soldiers that a player must receive when he change three Risk cards.

function calculateSoldiersByCards(timesCardTraced)
- timesCardTraced: the times that a player has changed cards.

Cyclomatic complexity = 3

ID	Input	Output	Coverage
	timesCardTraced	numSoldiers	
DR-09	3	8	Path1
DR-10	6	15	Path2
DR-11	7	20	Path3

Test Log:

```
calculateSoldiersByCards
  ✓ Calculate the number of soldiers that a player must receive


1 passing (11ms)
```

2.5: Test function ServerSelectTerritory.validateChangeStage
This function is in the server and it verifies if the stage of select territories must ends.

function validateChangeStage(listNumSoldierPlayer)
- listNumSoldierPlayer: it is a collection of the number of soldier of each player of the match

Return: NextState: is the next state of the match, it can be “SelectTerritory“ or “Reforce”
Cyclomatic complexity = 3

Test Cases:

To do this test, create a match with tree players, and begin to select territories.

ID	Input	Output	Coverage
	listNumSoldierPlayer	NextState	
DR-12	[0,0,0]	“Reforce“	Path1
DR-13	[0,0,1]	“SelectTerritory“	Path2
DR-14	[1, 1, 2]	“SelectTerritory“	Path3

Test Log:

```
ServerSelectTerritory
  validateChangeStage
    ✓ Validate whether the stage have to change to Reforce


1 passing (10ms)
```