Author information
Name: Lucassen, Mario M.
Course: Software Engineering
Email: M.lucassen@student.fontys.nl

PAZURU
SYSTEM TEST
DOCUMENTATION

Document no: 5 Created: 06.06.2019
Version: 1.00 Last changes: 06.06.2019
File: Pazuru System Test Report.docx

Version history

Version	Date	Changes	Author[1]
0.01	06.06.2019	Copied contents of Pazuru System Test Documentation 1.01.	Lucassen, Mario M. M.lucassen@student.fontys.nl
1.00	06.06.2019	Completed test report .	Lucassen, Mario M. M.lucassen@student.fontys.nl

[1] Author's name with email address

Table of contents

Table 1. Global definitions.

Name	Description
Fontys	Fontys University of Applied Sciences.

1 Introduction

1.1 Purpose

This System Test Report is made to show the results of the test documentation.

The requirements in the *Pazuru Software Analysis Document* and the *Pazuru Sudoku Specification Document* contain the actions, rules and quality attributes the tests will cover. The actions will mostly be covered by Test Cases according to the Use Cases listed in both documents. The rules will be tested with unit tests to verify if the logic is correctly implemented. The quality attributes are not all testable by code, but performance measurements can be done to verify if it is within the set constraints for performance. These can also been seen in the form of component tests.

As documented in the Pazuru Software Analysis Document the goal is to have at least 80% test coverage, which will become clear in this document.

1.2 Scope

The scope of the system tests being made are what is listed in the requirements in the *Pazuru Software Analysis Document* and the *Pazuru Sudoku Specification Document*.

Scope includes
Requirements that are listed with an importance level of ‘Must’, ‘Should’ or ‘Could’.

Scope excludes
Due to time constraints, not all requirements will be tested. Requirements that are listed with an importance level of ‘Wont’ will not be included in the test coverage.

1.3 Author Profile

The author of the *Pazuru System Test Report* is Lucassen, Mario M, a Software Engineering student at Fontys. The product will be developed at the Fontys University of Applied Sciences, Rachelsmolen 1 in Eindhoven during the time allotted for the Software Engineering course.

Author information			
Name:	Lucassen, Mario M.	Course:	Software Engineering
Email:	M.lucassen@student.fontys.nl	Role:	Software Developer

2 Risk analysis

This chapter describes the product risk analysis. The time available for testing is limited thus not everything can be tested equally. So decisions have to be made. The aim is to distribute the test capacity as effectively and efficiently as possible over the entire test process.

2.1 Strategy

Each test goal will be evaluated by Impact, chance of failure, risk class and severity class. **Impact** is the impact the organization will have if the test goes wrong. Impact is measured within the range Low, Middle and High. **Chance of failure** is the chance of failure the test will have. This is based on the developers experience in implementing that feature. Chance of failure is measured within the range Low, Middle and High. **Risk class** is impact times the chance of failure where Low = 1, Middle = 2 and High = 3. **Severity class** is the risk class but scaled back to an enumeration of C = risk class =< 2, B = risk class =< 6 and A = risk class > 6.

2.2 Analysis

The risk analysis, based on the strategy described in the chapter above, is shown below. Not all quality attributes will be included in the analysis, only what is testable. Test goals covered in red are removed from the deliverable and thus will not be tested.

Pazuru Platform				
Test goal	**Impact**	**Chance of failure**	**Risk class**	**Severity class**
F1	High	Low	3	B
F2	High	Low	3	B
F3	High	Low	3	B
F4	High	Low	3	B
F5	Middle	Low	2	C
F6	Middle	Low	2	C
F7	Middle	Low	2	C
F8	Low	Low	1	C
F9	Low	Low	1	C
F10	Low	Low	1	C
R1	High	Low	3	C
R2	High	Low	3	C
R3	High	Low	3	C
Q1	High	Middle	6	B
Q3	High	Low	3	C

Sudoku Puzzle				
Test goal	**Impact**	**Chance of failure**	**Risk class**	**Severity class**
F1	High	Low	3	B
F2	High	Low	3	B
F3	High	Low	3	B
F4	High	Low	3	B
R1	High	Low	3	B
R2	High	Low	3	B
R3	High	Low	3	B
R4	High	Low	3	B

3 Testing strategy

In the previous chapter all test goals have been evaluated by Impact, Change of failure, Risk class and Severity class. All Actions and Rules must be tested by unit tests. The quality attributes are covered by integration tests because it will convey the cohesion of the system better.

The risk analysis has been divided into specific tests that will help reach each test goal as shown in the tables below. Test goals covered in red are removed from the deliverable and thus will not be tested.

Pazuru Platform						
Test goal	**Severity class**	**Unit test**	**Component test**	**Integration test**	**System test manual**	**System test Automatic**
F1	B				●●	
F2	B				●●	
F3	B	●●	●		●	
F4	B	●●	●		●	
F5	C			●	●	
F6	C			●	●	
F7	C	●	●	●	●	
F8	C				●	
F9	C			●	●	
F10	C			●	●	
R1	C			●	●	
R2	C			●●	●	
R3	C	●	●	●	●	
Q1	B			●●●	●	
Q3	C		●	●●	●	

Sudoku Puzzle						
Test goal	**Severity class**	**UT**	**CT**	**IT**	**ST Manual**	**ST Automatic**
F1	B			●●	●	
F2	B			●●	●	
F3	B	●●	●	●	●	
F4	B	●●	●	●	●	
R1	B	●●			●	
R2	B	●●			●	
R3	B	●●			●	
R4	B	●●			●	

●	Limited dynamic test
●●	Average dynamic test
●●●	Heavy dynamic test

4 Logical Test Cases

In this chapter the logical test cases can be found in the tables below. The purpose of this chapter is to give an overview of the test cases.

Pazuru Platform	
ID	**Description**
TC01	Selecting a puzzle.
TC02	Generating a puzzle.
TC03	Solving a puzzle.
TC04	Executing a move.
TC05	Verifying a puzzle state.
TC06	Describing a puzzle.
TC07	Show solved puzzles.
TC08	Filter solved puzzles.
TC09	A puzzle must have at least 1 puzzle rule.

Sudoku Puzzle	
ID	**Description**
TC10	Add number to the grid.
TC11	Remove a non-seeded number from the grid.
TC12	Generating a Sudoku puzzle.
TC13	Solving a Sudoku puzzle.
TC14	Verify that rule “Each row can only contain each number from 1 to 9 once.” is correct.
TC15	Verify that rule “Each column can only contain each number from 1 to 9 once.” is correct.
TC16	Verify that rule “Each 3x3 sub-grid can only contain each number from 1 to 9 once.” is correct.
TC17	Verify that a generated puzzle only has one unique solution.

TC10 to TC13 is actually included in TC01 to TC04 so will be taken out of the Physical Test Cases. They remain listed so that the intent of testing these can still be done if needed.

5 Physical Test Cases

This chapter contains the logical test cases with an extended description.

Pazuru Platform		
ID	**Description**	**Result**
TC01	**Selecting a puzzle.** Select Sudoku puzzle in select puzzle selection on the home page. The expected result is that Sudoku is the selected puzzle in the puzzle select.	Successfully selected Sudoku.
TC02	**Generating a puzzle.** Generate a Sudoku puzzle by having selected Sudoku in the puzzle selection and clicking on the generate button. The expected result is that a Sudoku grid is shown on the page.	Successfully generated a Sudoku puzzle.
TC03	**Solving a puzzle.** Generate a Sudoku puzzle by having selected Sudoku in the puzzle selection and clicking on the generate button. Solve a Sudoku puzzle by clicking on the solve button. The expected result is that the Sudoku grid is solved.	Successfully solved the current puzzle.
TC04	**Executing a move.** Generate a Sudoku puzzle by having selected Sudoku in the puzzle selection and clicking on the generate button. In the first non seeded cell press the number 5. The expected result is that the number 5 is added on the grid on the selected cell.	Successfully added the number 5 to the puzzle grid on the selected cell.
TC05	**Verifying a puzzle state.** Generate a Sudoku puzzle by having selected Sudoku in the puzzle selection and clicking on the generate button. Fill the first row with pressing the number 5 on each cell. Click on the verify puzzle state button. The expected result is that the puzzle state is verified according to the sudoku puzzle rules and should only have one '5' green cell in the top row.	Successfully verified the current puzzle state and only one '5' is colored green in the top row.
TC06	**Describing a puzzle.** Select Sudoku puzzle in select puzzle selection on the home page. Click on the describe puzzle button. The expected result is that the Sudoku description is shown with the rules.	Successfully described the Sudoku puzzle by clicking on the 'Describe' button.
TC07	**Show solved puzzles.** Navigate from the home page to the previously solved puzzles page. The expected result is that the previously solved puzzles page is open and the previously solved puzzles can be seen.	Successfully navigated to the previously solved puzzles page.
TC08	**Filter solved puzzles.** Navigate from the home page to the previously solved puzzles page. Select Sudoku in the puzzle filter selection. Click on the filter button. The expected result is that the previously solved puzzles list has been filtered to only show the Sudoku puzzles.	Successfully filtered the previously solved puzzles to show Sudoku results only.
TC09	**A puzzle must have at least 1 puzzle rule.** Select Sudoku puzzle in select puzzle selection on the home page. Click on the describe puzzle button. The expected result is that the Sudoku description is shown with at least 1 rule.	Successfully showed the Sudoku puzzle description with several rules.

Sudoku Puzzle		
ID	**Description**	**Result**
TC14	**Verify that rule "Each row can only contain each number from 1 to 9 once." is correct.** Generate a Sudoku puzzle by having selected Sudoku in the puzzle selection and clicking on the generate button and on the first row fill it with only 9's. Click on verify puzzle state. The expected result is that only one cell is colored green with the number 9 in the top row.	Successfully verified the current puzzle state and only one '9' is colored green in the top row.
TC15	**Verify that rule "Each column can only contain each number from 1 to 9 once." is correct.** Generate a Sudoku puzzle by having selected Sudoku in the puzzle selection. Click on the generate button. In the most left column fill it with only 1's. Click on verify puzzle state. The expected result is that only one cell is colored green with the number '1' in the most left column.	Successfully verified the current puzzle state and only one '1' is colored green in the most left column.
TC16	**Verify that rule "Each 3x3 sub-grid can only contain each number from 1 to 9 once." is correct.** Generate a Sudoku puzzle by having selected Sudoku in the puzzle selection. Click on the generate button. In the left top corner 3 by 3 box of cells, fill it with 8's. Click on verify puzzle state. The expected result is that only one cell is colored green with the number '8' in the left top corner 3 by 3 box.	Successfully verified the current puzzle state and only one '8' is colored green in the left top corner 3 by 3 box.
TC17	**Verify that a generated puzzle only has one unique solution.** Brute force every number from 1 to 9 in every cell and check if there is only 1 solution for a generated puzzle. The expected result is that there is only 1 unique solution.	By applying Algorithm X in the Sudoku puzzle generating code this goal is also covered, because it checks for uniqueness.

6 Test coverage matrix

This chapter contains the test coverage matrix.

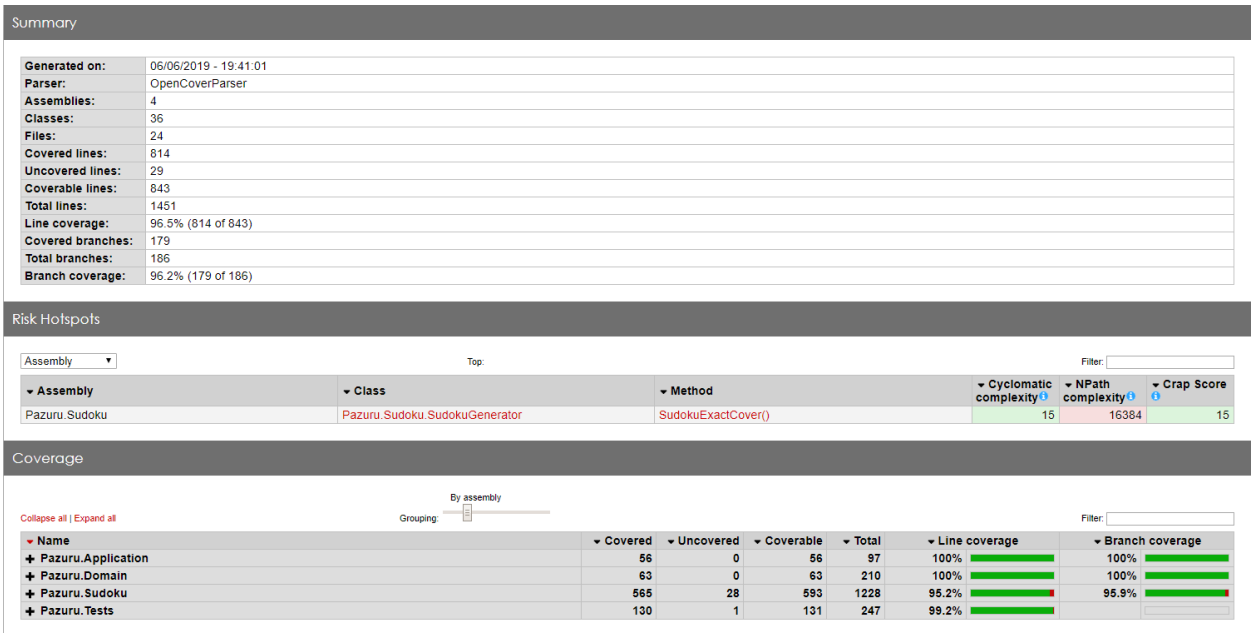
Pazuru Platform													
	TC-01	TC-02	TC-03	TC-04	TC-05	TC-06	TC-07	TC-08	TC-09	TC-14	TC-15	TC-16	TC-17
F1	X	X	X	X		X			X				
F3			X										
F4			X										
F5				X	X								
F7					X					X	X	X	
F8						X			X				
F9							X	X					
F10								X					

Sudoku Puzzle													
	TC-01	TC-02	TC-03	TC-04	TC-05	TC-06	TC-07	TC-08	TC-09	TC-14	TC-15	TC-16	TC-17
F1				X									
F2													
F3			X										
F4		X											
R1			X							X			
R2			X								X		
R3			X									X	
R4			X										X

7 Code coverage

This chapter contains the classes which the system should be tested for to reach the 80% code coverage goal.

By testing a Sudoku puzzle by first generating and solving it, at least 80% of the code will be covered by this. Because the main codebase is about puzzle solving and generating and the Sudoku implementation is literally an implementation of the abstracted Puzzle, PuzzleRule, PuzzleMove, PuzzleGenerator, PuzzleSolver and PuzzleState.



Summary

Generated on:	06/06/2019 - 19:41:01
Parser:	OpenCoverParser
Assemblies:	4
Classes:	36
Files:	24
Covered lines:	814
Uncovered lines:	29
Coverable lines:	843
Total lines:	1451
Line coverage:	96.5% (814 of 843)
Covered branches:	179
Total branches:	186
Branch coverage:	96.2% (179 of 186)

Risk Hotspots

Assembly | Top: | Filter:

Assembly	Class	Method	Cyclomatic complexity	NPath complexity	Crap Score
Pazuru.Sudoku	Pazuru.Sudoku.SudokuGenerator	SudokuExactCover()	15	16384	15

Coverage

Collapse all | Expand all | Grouping: By assembly | Filter:

Name	Covered	Uncovered	Coverable	Total	Line coverage	Branch coverage
+ Pazuru.Application	56	0	56	97	100%	100%
+ Pazuru.Domain	63	0	63	210	100%	100%
+ Pazuru.Sudoku	565	28	593	1228	95.2%	95.9%
+ Pazuru.Tests	130	1	131	247	99.2%	

8 Static code analysis

This chapter contains the static code analysis generated by SonarQube for C#.