FIND THE BUG!

A Game of Testing and Fixing Bugs in an IT Project

for 2-6 players
(playing time 30 min)

by Nicholas Hjelmberg
Nova Suecia Games
http://www.novasuecia.se
Version 1.3

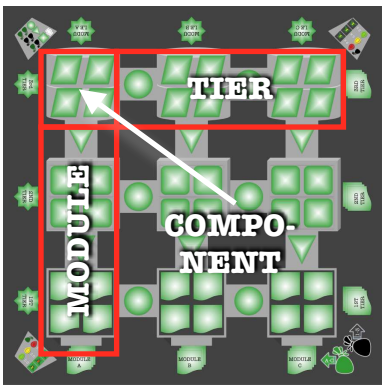
Internal

Welcome to the game Find the Bug! In this game, you take on the role of a test lead in a test project. The game takes you through 8 test phases, where you have to use your test skills to find as many bugs as possible.

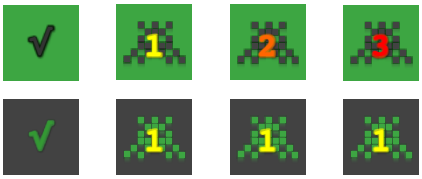
The top menu serves a table of contents that helps you navigate through the rules. The phases marked with an asterisk (*) are advanced phases, suitable when you have played several games and want additional challenges.

1. Plan: In this phase, you set up the game.
2. Analyze: In this phase, you gather information about the game conditions that will help you find bugs.
3. Test: In this phase, you execute system tests on the different components of the game board to find bugs.
4. Integration/Performance/Regression Test*: In this advanced phase, you execute tests covering several components.
5. Fix: In this phase, you reveal any unfound bugs.
6. Retest*: In this advanced phase, you execute automated tests to find new bugs after the fixes.
7. Report: In this phase, you count the number of found bugs and determine a winner.
8. Close: In this phase, you discuss the lessons learned from the game.

Game Parts



1 game board; 3 module columns (A-C) and 3 tier rows (1-3)



72 tiles; 36 green and 36 black

54 pawns; 9 pawns of 6 different colors

6 cloth bags; 3 green and 3 black

38 bugs; 2 green, 2 yellow, 2 red, 16 black and 16 white

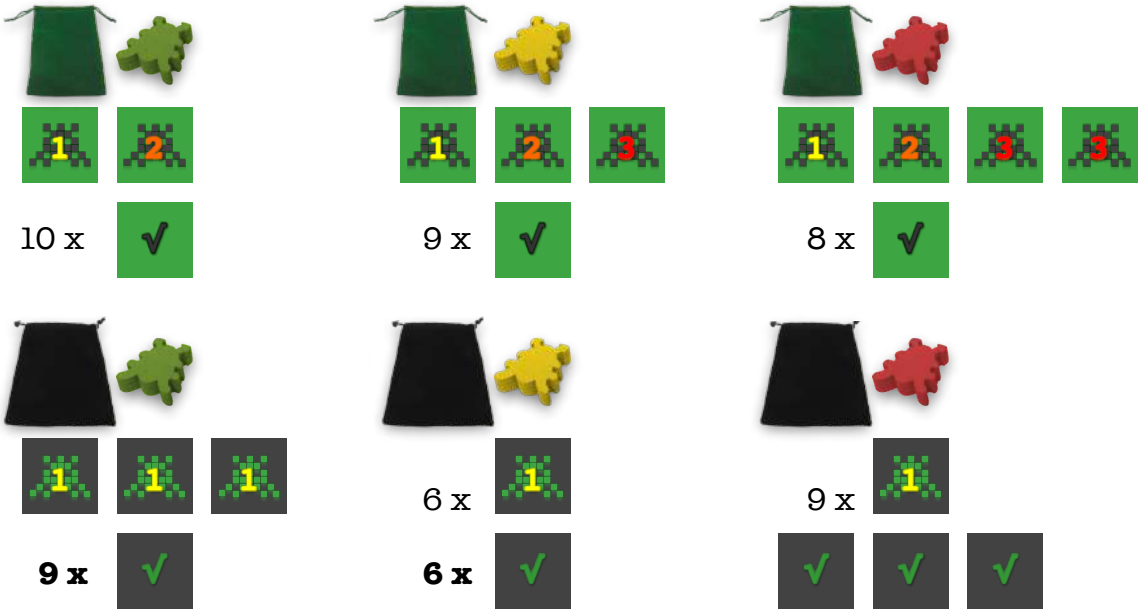
We hope that you will enjoy your game of Find the Bug!

Internal

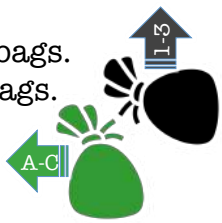
The project has been approved by the steering committee and you have been assigned a test lead. Please proceed with the **Test Plan**. Your first task is to identify your test team and prepare the test environment:

1. Select your **test team** (colored pawns):
   a. 2 test leads: 18 testers each, use 2 colors each
   b. 3 test leads: 12 testers each, use 2 colors each
   c. 4 test leads: 9 testers each
   d. 5 test leads: 7 testers each
   e. 6 test leads: 6 testers each

   (If you play without the advanced Retest* rules, select 1 tester less.)
2. Prepare the **test basis** (tiles and bugs):
   a. 3 piles with business documentation (green tiles)
      i) Low criticality: 2 bugs (1, 2) + 10 pass + 1 green bug
      ii) Medium criticality: 3 bugs (1, 2, 3) + 9 pass + 1 yellow bug
      iii) High criticality: 4 bugs (1, 2, 3, 3) + 8 pass + 1 red bug
   b. 3 piles with technical documentation (black tiles)
      i) Low complexity: 3 bugs (3*1) + 9 pass + 1 green bug
      ii) Medium complexity: 6 bugs (6*1) + 6 pass + 1 yellow bug
      iii) High complexity: 9 bugs (9*1) + 3 pass + 1 red bug



3. Prepare the **test repository** (cloth bags):
   a. Place each green pile in a green bag and shuffle the bags.
   b. Place each black pile in a black bag and shuffle the bags.
4. Prepare the **test environment** (game board):
   a. Place 1 green bag next to each of the 3 modules A-C.
   b. Place 1 black bag next to each of the 3 tiers 1-3.



You are now ready to **analyze** the test basis.

Internal

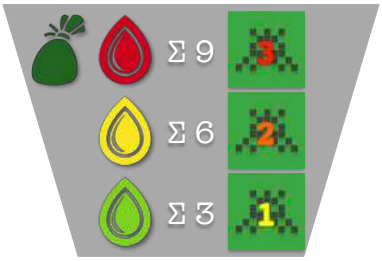
Is the test team and test basis ready? Good, then it's time to get started!

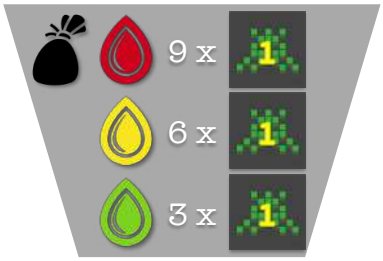
Starting with the most junior test lead and continuing clockwise, take turns to assign testers to tasks (placing pawns in empty spaces on the game board). When the last test lead in turn has assigned a tester, he or she starts the next round and the other test leads follow in counter clockwise order. Continue until no test lead has testers left.

There are several available tasks. The 1st task is **Analyze**, where you analyze the test basis in the test repository to assess business criticality of the modules and technical complexity of the tiers. The higher the business criticality, the greater the bug severity, and the higher the technical complexity, the greater the bug density.

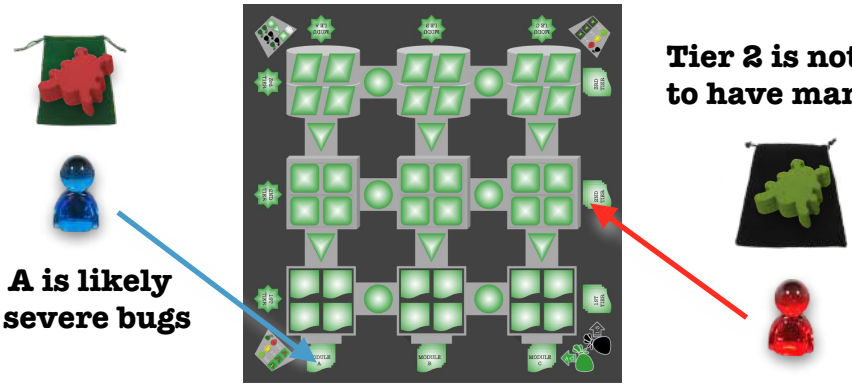
1. Assign a tester to any of the 6 **documents** (1 for each of the modules A-C and 1 for each of the tiers 1-3).
2. Take out the **bug** from the corresponding **bag** and look at it. Do not take out any tile.
3. Return the bug without showing it to the others.
4. Repeat step 2 and 3 for 1 more bag of your choice.

Document





The bugs in the green bags indicate the bug severity (total value of bugs) and the bugs in the black bags indicate the bug density (total number of bugs).



Module A is likely to have severe bugs

Tier 2 is not likely to have many bugs

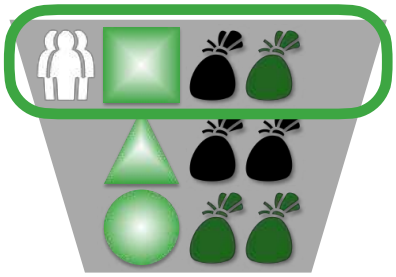
You will now know where to focus your test effort. Use this knowledge when you assign your testers to the 2nd available task: **Test**.

Internal

Test Cases

Are you done assigning testers to Analyze? Then you may consider the 2nd available task: **System Test**.

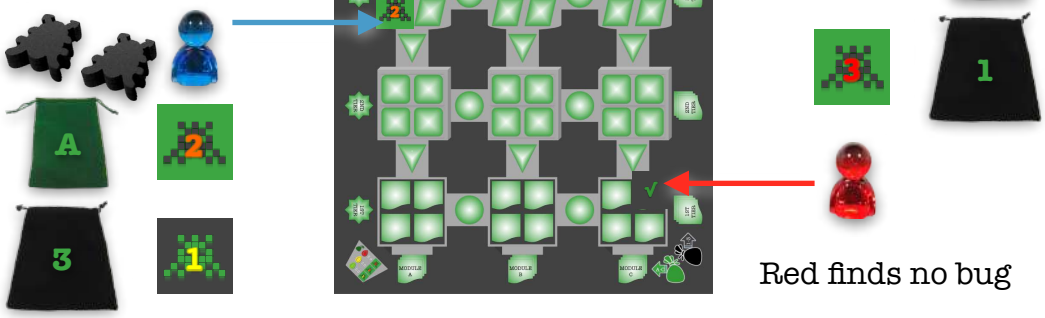
1. Assign a tester to any of the 36 square shaped **test cases** in the test environment. Each test case corresponds to 1 module and 1 tier.
2. Draw 1 **tile** from the corresponding **green module bag** and 1 **tile** from the corresponding **black tier bag**. Do not take out the bug. A tile shows either a bug or a pass. (Ignore the back fix symbol for now.)



   a. If both tiles contain a bug, the test **fails**. Place the tiles on the test case with the green bug tile at the top. Take as many **black bugs** as the value of the green bug tile.
   b. Else, the test **passes**. Place the tiles on the test case with a pass tile at the top and take no black bug.

Black Bug

Blue finds a bug with severity 2



Red finds no bug

Do you want further challenges? Then consider assigning your testers to any of the 3 advanced test tasks on the next pages:

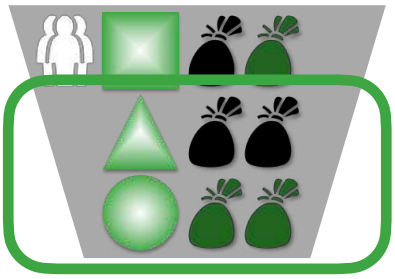
- 3rd Task: **Integration Test***
- 4th Task: **Performance Test***
- 5th Task: **Regression Test***

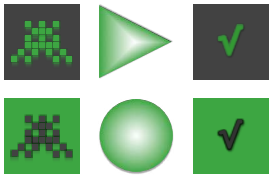
Internal

Performance Test Case

Integration Test Case

1. Assign a tester to any of the 6 circular shaped **performance test cases** or any of the 6 triangular shaped **integration test cases**. Each performance test case corresponds to 2 modules and each integration test case corresponds to 2 tiers.
2. Draw 1 tile from each of the 2 corresponding bags, i.e. **2 tiles** from **green module bags** for a performance test and **2 tiles** from **black tier bags** for an integration test.
3. Place the tiles in empty test cases adjacent to the performance or integration test cases. Never place tiles in the performance or integration test case itself.



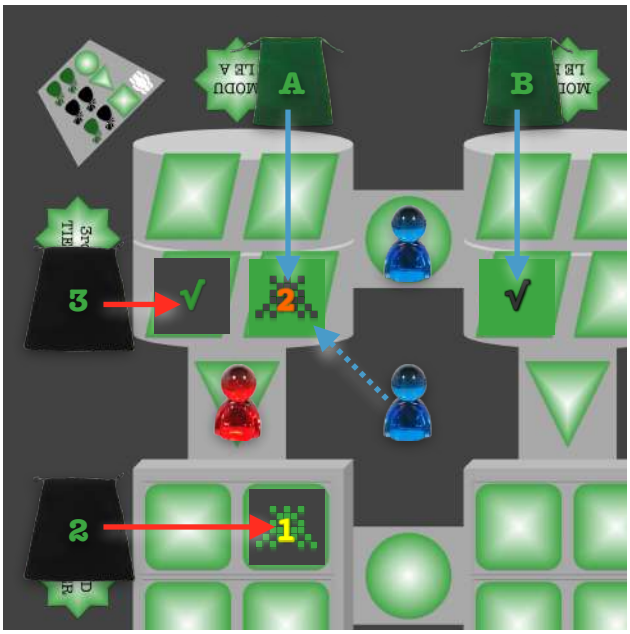


If there are not enough empty adjacent test cases, you cannot assign testers to that performance or integration test case.

Blue draws 1 green bug from the Module A bag and 1 green pass from the Module B bag

Red draws 1 black pass from the Tier 3 bag and 1 black bug from the Tier 2 bag

Blue may later place a tester on the test case with a green severity 2 bug and draw 1 black tile. (Red may not do this since her integration tester cannot have placed the green severity 2 bug.)



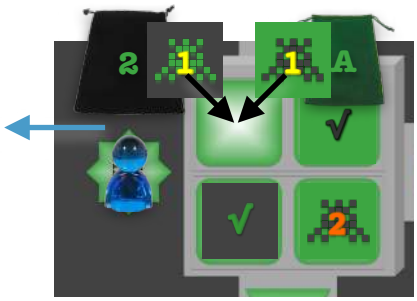
A performance or integration test only places 1 tile in each test case. To place a 2nd tile, you must in a later turn assign a 2nd tester to 1 of the test cases. Only the test lead whose tester drew the 1st tile may assign a 2nd tester to the test case and draw the 2nd tile. (You can tell from the placement and color of the 1st tile which of the adjacent testers that drew it.) As usual, 2 bug tiles are needed to earn bugs.

Internal

Regression Test Case

1. Assign a tester to any of the 6 cog shaped **regression test cases** to automate a regression test case. Each regression test corresponds to 1 entire module or 1 entire tier.
2. Do not draw any tiles now. You will do it in the **Retest* phase**.
3. Do not assign testers to regression test cases too early. If a bug is later found in the module or the tier, the tester's work is lost. The tester is **returned** from the test case to the test lead and may be used in a later turn. Another tester may now be assigned to the regression test case.

Blue has assigned a tester to regression test of Tier 2. A bug is later found in Component A2 and Blue must take back the tester from the test case.



PLAN | ANALYZE | TEST | IT/PT/RT* | FIX | RETEST* | REPORT | CLOSE

Internal

Have you assigned all your testers to tasks? Then it is time to **fix** the bugs. The fix team takes care of this so no testers are used.

Fix Tiles

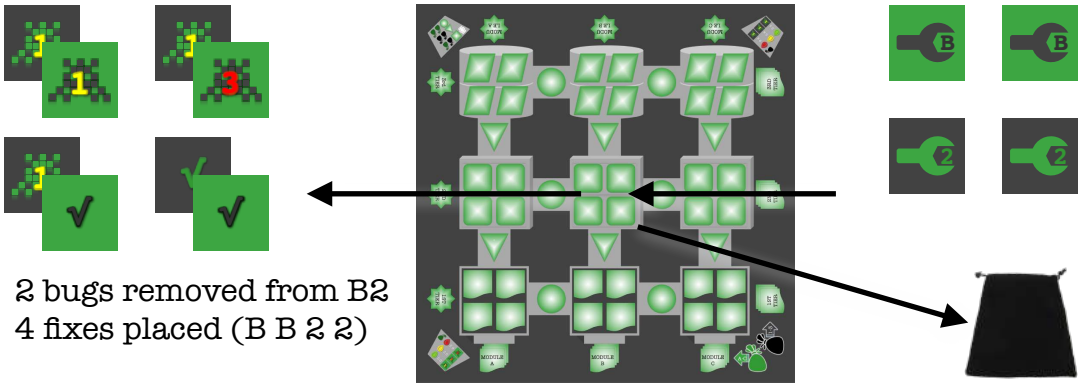
1. Place the remaining tiles on the remaining test cases so that all bags are empty and all test cases have 1 green tile and 1 black tile.
   a. If more than 1 component has bugs not found by any tester, the test team fails and all test leads **lose**.
   b. Else, proceed to **fix** the bugs in the next step.
2. Remove all **pass tiles** from the test cases and flip them so that the letters and numbers appear. They will be used as fix tiles.
3. For each test case with a **bug tile**, place **1 pair of fix tiles** there instead, one with the module letter and one with the tier number.

For each bug in the test environment, there should now be 2 fix tiles.

Internal

Have all bugs been fixed? Then it is time to run the automated regression tests that you created in the Regression Test*.

1. Place the fix tiles in an empty bag.



2 bugs removed from B2
4 fixes placed (B B 2 2)

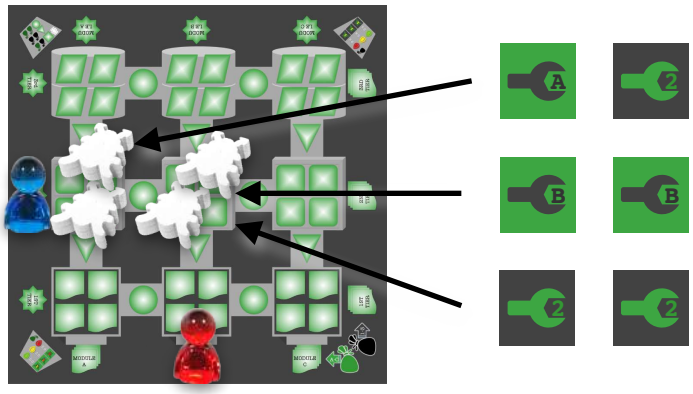
2. Draw **6 fix tiles** from the bag and ignore the rest.
3. Place **2 white bugs** on each component where at least 2 fix tiles intersect (but never more than 2, even if there are several intersections).
4. Each tester assigned to a **regression test case** may take the white bugs in the corresponding module or tier. If 2 testers claim the same bugs, they take 1 each.
5. Again, if more than 1 component has unfound bugs, the test team failed and all test leads lose.

White Bug

Fix tiles A, B and 2 are drawn, resulting in bugs in A2 and B2.

Blue takes both bugs from component A2.

Blue and Red takes 1 bug each from component B2.

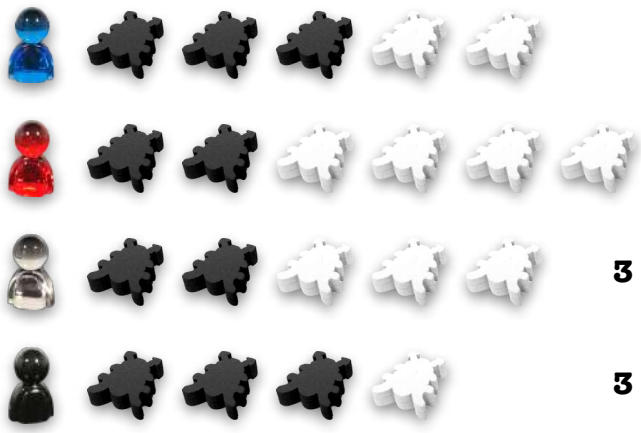


Congratulations, you have completed the test! Please proceed to **Report**.

Internal

Please **report** the severity and number of bugs found per test lead. Black bugs are worth 1 point and white bugs are worth 1/2. The test lead with the most points wins.

In the event of a tie, the test lead with the smallest difference between the number of black bugs and the number of white bugs is the winner.



4 (1)

4 (2)

3 1/2 (1)

3 1/2 (2)

Blue wins with 4 points and only 3-2=1 in difference between black and white bugs.

Similarly, White is ahead of Black with 3 1/2 points and difference 3-2=1.

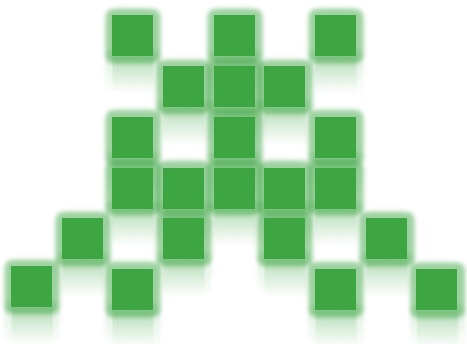
PLAN | ANALYZE | TEST | IT/PT/RT* | FIX | RETEST* | REPORT | CLOSE

Internal

You may now **close** the test. Please remember to continuously improve your way of working. Could you have spent less or more time on any task? Did you test any component too little or too much? Do discuss your strategies with your colleagues and learn from each other.

On the next pages you will find a summary of the test concepts covered in Find the Bug for further discussions.

We thank you for your effort and look forward to work with you again!



GLOSSARY

Term	Test concept	Game concept
Acceptance criteria	Criteria for component to be accepted	Bugs in more than 1 component not accepted
Ad hoc testing	Testing without preparation	Testers assigned to test without analysis
Business criticality	A measure of impact of defects	The severity of bugs in a bag, indicated by a colored bug (red is highest etc.)
Component	Item that can be tested in isolation	A specific module in a specific tier
Defect (bug)	Flaw in component causing system failure	Both module and tier tile has a bug
Defect density	Defects in relation to component size	Numbers/severity of bugs in component
False-fail	A defect is reported although no exists	If only 1 tile has a bug, no bug is found
Module	Group of items tested together	1 module consists of 3 tiers
Integration Test	Testing to find defects in the interactions between integrated components.	Test covering 2 modules
Pass-fail	A defect is not reported although one exists	A bug that is never detected
Performance Test	Testing to determine the performance of a software product.	Test covering 2 tiers
Project	Set of tasks to undertake an objective	The game from start to end
Regression test	Test of previously tested system after changes	The retest after fixes, when new bugs may appear
Risk-based testing	Test based on impact & probability of defects	Test in modules & tiers based on the color of the bug
Severity	Degree of impact that a bug has	Module bugs have a value between 1 and 3
System test	Testing of the behavior of the whole system	Test across all the modules and tiers

GLOSSARY

Term	Test concept	Game concept
Technical complexity	A measure of probability of defects	The number of bugs in a bag, indicated by a colored bug (red is highest etc.)
Test automation	Use of software to support test tasks, usually requires mature system	Test of entire module or tier, possible when free from bugs
Test basis	Documents on which test is based (e.g design)	Bags with a number of bug/no bug tiles
Test case	Set of input and expected output	1-4 squares in each component to place testers on
Test environment	Hardware and software for conducting test	The game board
Test lead	Person managing test tasks	The players
Test process	Planning, analysis, design, execution, reporting and closing of test	The tasks Plan, Analyse, Test, Fix, Retest, Report and Close
Tester	Person executing test tasks	The pawns placed by the players
Tier	Level of system; a 3-tier system contains presentation, application and data	1 tier covers 3 modules

Credits

Game design and art:	Nicholas Hjelmberg
Production:	The Game Crafter
Game testers:	Juan Garcia, Hans Larsson, Eva Nordström, Lotta Östergren, colleagues at SQS Sweden
Special thanks:	My wife Su-San Oh for her patience, my SQS colleague Hans Larsson for inspiration in the test area and my game colleague Nerdfest Games for advice on how to bring testing to a board game