Zen of Testing

mesutdurukal

Zen (I)

- derived from the Chinese word **"Chan"** (禪),
- meaning meditation
- emphasizes
  - direct experience,
  - mindfulness,
  - and the practice of meditation

as a path to enlightenment.


Let’s take an imaginary visit to a shrine
and experience the activities inside.

(I) The Fox (狐, Kitsune)

Messengers and Masters of Transformation

They symbolize

- Intelligence
- Adaptability
- Transformation

47 Ronin

Mizuki, The Witch of Lord Kira

https://japanagram.me/2021/08/31/japanagram-47-ronin/

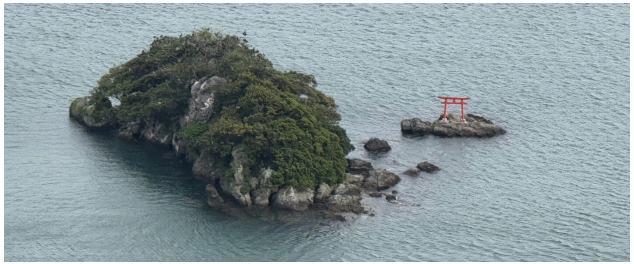
The Fox (狐, Kitsune)

Messengers and Masters of Transformation

- Messengers of Quality
- Verification and Human Interaction
- Transformation and Adaptability
- Mystery and Hidden Layers
- Guardians of Integrity

(II) Torii Gate (鳥居)

Marking a Transition



Torii Gate (鳥居)

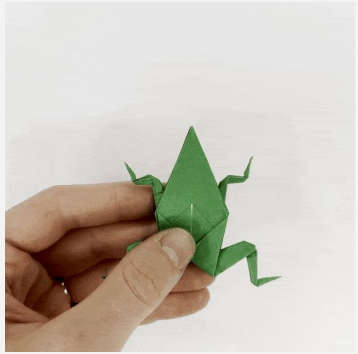
Marking a Transition

- Ensuring **readiness**
- Stops **bugs**
- **Quality** gate

(III) Origami (折り紙)

Complexity from Simplicity

The art of folding paper into intricate designs, starting from a single sheet.



Origami (折り紙)

Complexity from Simplicity

A small, well-crafted test case can uncover complex issues, just as origami transforms a simple sheet into something extraordinary.

- Build wisely, avoid **duplication**
- **Test design** is an art
- Leave something memorable: **Legacy** code!

(IV) Zen Gardens (枯山水)

Removing Noise

A minimalistic garden designed to create peace and focus, often representing natural elements like water and mountains through sand and rocks.

(IV) Zen Gardens (枯山水)

Removing Noise

Prepare and organize

- Test environment
- Management tools
- Workflows
- Bug reports
- Test flakiness

(V) Noodles (ラーメン)

Wishing a long life

- From the beginning, plan tests for long-term stability.
- Avoid technical debt.
- Maintainability: Treat test code properly.

(VI) Kintsugi (金継ぎ)

Embracing Imperfections

The art of repairing broken pottery with gold, highlighting the cracks rather than hiding them. It celebrates flaws as part of an object's history.

		Any Issues in the feature?	
		No	Yes
Test Report	PASS	No Problem	Silent Horror
	FAIL	False Alarm	Real Bug

No.	Test Smell Name	Abbreviation	Definition	Ref.
01	**Abnormal UTF-Use**	**AUU**	Overriding the default behavior of the testing framework by test-suite.	[70]
02	**Anonymous Test**	**AT**	A test method with a meaningless and unclear method name.	[70]
03	**Assertion Roulette**	**AR**	A test method with multiple assertions without explanation messages.	[27]
04	**Assertionless**	**AL**	A test that is acting to assert data and functionality but does not.	[32]
05	**Assertionless Test**	**ALT**	A test that does not contain at least one valid assertion.	[70]
06	**Brittle Assertion**	**BA**	A test method that has assertions that check data input.	[48]
07	**Comments Only Test**	**COT**	A test that has been put into comments.	[70]
08	**Conditional Test Logic**	**CTL**	A test method that contains a conditional statement as a prerequisite to executing the test statement.	[65]
09	**Constructor Initialization**	**CI**	A test class that contains a constructor.	[65]
10	**Control Logic**	**ConL**	A test method that controls test data flow by methods such as debug or halt.	[70]
11	**Dead Field**	**DF**	When a class has a field that is never used by any test methods.	[45]
12	**Default Test**	**DT**	Default or an example test suite created by Android Studio.	[65]
13	**Dependent Test**	**DepT**	A test that only executes on the successful execution of other tests.	[87]
14	**Duplicate Assert**	**DA**	Occurs when a test method has the exact assertion multiple times within the same test method.	[65]
15	**Duplicated Code**	**DC**	A test method that has redundancy in the code.	[32]
16	**Eager Test**	**ET**	A test method that calls several methods of the object to be tested.	[27]
17	**Early Returning Test**	**ERT**	A test method that returns a value too early which may drop assertions.	[70]
18	**Empty Method Category**	**EMC**	A test method with an empty method category.	[70]
19	**Empty Shared-Fixture**	**ESF**	A test that defines a fixture with an empty body.	[70]
20	**Empty Test**	**EmT**	A test method that is empty or does not have executable statements.	[65]
21	**Empty Test-Method Category**	**ETMC**	A test method with an empty test method category.	[70]
22	**Exception Handling**	**EH**	Occurs when custom exception handling is utilized instead of using JUnit's exception handling feature.	[65]
23	**For Testers Only**	**FTO**	A production class that contains methods that are only used for test methods.	[27]
24	**General Fixture**	**GF**	This smell emerges when `setUp()` fixture creates many objects, and test methods only use a subset.	[27]
25	**Guarded Test**	**GT**	A test that has conditional branches like `ifTrue:aCode` or `ifFalse:aCode`.	[70]
26	**Ignored Test**	**IgT**	A test method that uses an ignore annotation which prevents the test method from running.	[65]
27	**Indented Test**	**InT**	A test method that contains a large number of decision points, loops, and conditional statements.	[32]
28	**Indirect Testing**	**IT**	A test that interacts with a corresponding class by using another class.	[27]
29	**Lack of Cohesion of Methods**	**LCM**	When test methods are grouped in one test class, but they are not cohesive.	[45]
30	**Lazy Test**	**LT**	Occurs when multiple test methods check the same method of production object.	[27]
31	**Likely Ineffective Object-Comparison**	**LIOC**	A test that performs a comparison between objects will never fail.	[70]
32	**Long Test**	**LoT**	A test with many statements.	[70]
33	**Magic Number Test**	**MNT**	A test method that contains undocumented numerical values.	[65]
34	**Max Instance Variables**	**MIV**	A test method that has a large fixture.	[70]
35	**Mixed Selectors**	**MS**	Violates test conventions by mixing up testing and non-testing methods.	[70]
36	**Mystery Guest**	**MG**	A test that uses external resources, such as a database, that contains test data.	[27]
37	**Obscure In-line Setup**	**OISS**	A test that has too much setup functionality in the test method.	[45]
38	**Overcommented Test**	**OCT**	A test with numerous comments.	[70]
39	**Overreferencing**	**OF**	A test that causes duplication by creating unnecessary dependencies.	[70]
40	**Proper Organization**	**PO**	Bad organization of methods	[70]
41	**Redundant Assertion**	**RA**	A test method that has an assertion statement that is permanently true or false.	[65]
42	**Redundant Print**	**RP**	A test method that has print statement.	[65]
43	**Resource Optimism**	**RO**	A test that make an assumption about the existence of external resources.	[27]
44	**Returning Assertion**	**RA**	A test method that has an assertion and returns a value.	[70]
45	**Rotten Green Tests**	**RT**	Occurs when intended assertions in a test are never executed.	[36]
46	**Sensitive Equality**	**SE**	Occurs when an assertion has an equality check by using the `toString` method.	[27]
47	**Sleepy Test**	**ST**	Occurs when a test method has an explicit wait.	[65]
48	**Teardown Only Test**	**TOT**	Exists when a test-suite is only specifying teardown.	[70]
49	**Test Code Duplication**	**TCD**	Occurs when code clones contained inside the test.	[27]
50	**Test Maverick**	**TM**	Exists when a test class has a test method with an implicit setup; however, the test methods are independent.	[45]
51	**Test Pollution**	**TP**	Test that introduces dependencies such as reading/writing a shared resource.	[47]
52	**Test Redundancy**	**TR**	Occurs when the removal of a test does not impact the effectiveness of the test suite.	[50]
53	**Test Run War**	**TRW**	A test that fails when more than one programmer runs them.	[27]
54	**Test-Class Name**	**TCN**	A test that has a class with a meaningless name.	[70]
55	**Test-Method Category Name**	**TMC**	A test method has a meaningless name.	[70]
56	**Transcripting Test**	**TT**	A test that is printing and logging to the console.	[70]
57	**TTCN-3 Smells**	**TTCN**	Collection of smells specific to TTCN-3 test suites.	[25]
58	**Unclassified Method Category**	**UMC**	A test method that is not organized by a method category.	[70]
59	**Under-the-carpet Assertion**	**UCA**	A test that has assertions in the comments.	[70]
60	**Under-the-carpet failing Assertion**	**UCFA**	A test method that has failing assertions in the comments.	[70]
61	**Unknown Test**	**UT**	A test method without an assertion statement and non-descriptive name.	[65]
62	**Unused Inputs**	**UI**	Inputs that are controlled by the test.	[48]
63	**Unused Shared-Fixture Variables**	**USFV**	Occurs when a piece of the fixture is never used.	[70]
64	**Unusual Test Order**	**UTO**	A test that is calling other tests explicitly.	[70]
65	**Vague Header Setup**	**VHS**	A field that is initialized in the class header but not explicitly defined in code.	[45]
66	**Verbose Test**	**VT**	Test code that is complex and not simple or clean.	[32]

Test Smell Detection Tools: A Systematic Mapping Study - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/Definition-of-the-test-smells-detected-by-the-tools-in-our-dataset_tbl1_351278876 [accessed 1 Mar 2025]

(VII) The Ronin's Journey (浪人)

Exploratory Testing

In Japanese culture, a ronin (浪人) is a masterless samurai who journeys independently, relying on their skills, intuition, and adaptability to navigate the world.

Impact of Exploratory Testing (All compared to scripted testing)

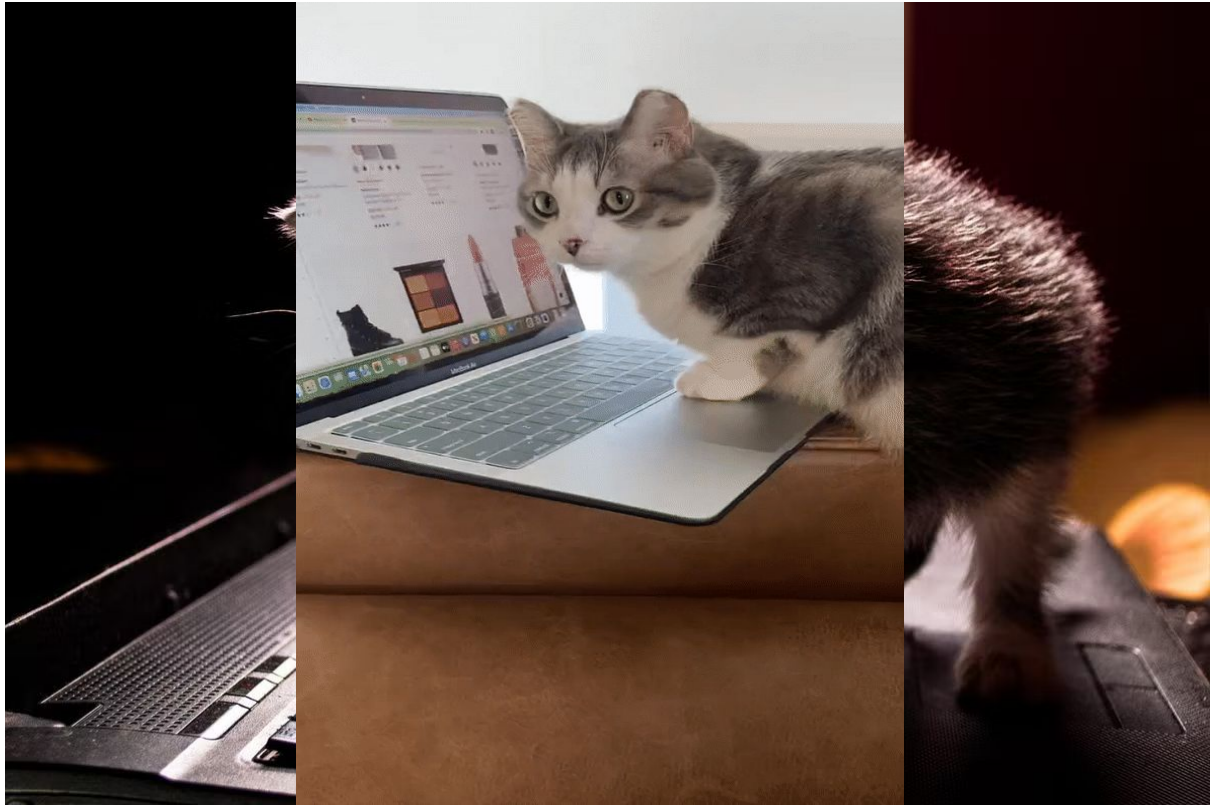
11% more defects (in average)

29% more evident defects (such as missing button)

33% more complex defects (more user actions to cause a failure)

https://www.researchgate.net/publication/4279015_Defect_Detection_Efficiency_Test_Case_Based_vs_Exploratory_Testing

Experiments: Shoe Test (James Bach)



(VIII) Bonsai Trees (浪人)

Exploratory Testing

- The art of cultivating miniature trees from an early stage.
- This practice mirrors Shift Left Testing

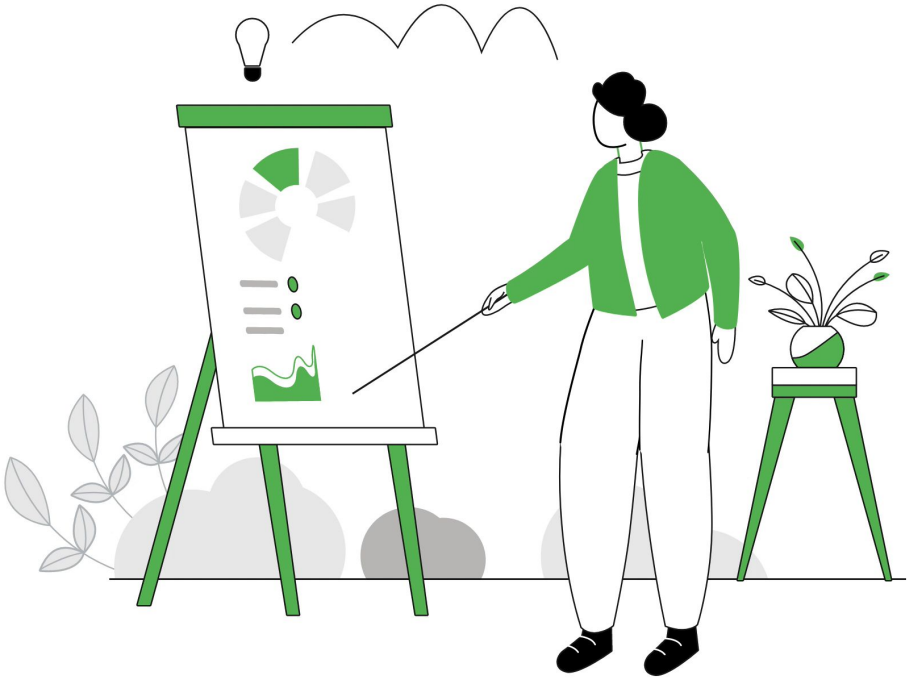
(IX) Calligraphy (書道)

Documentation

- writing characters with brush and ink
- not just about the final product, but also about
  - effort
  - precision
  - and intent
- that go into each stroke.

Visualize

Use Dashboards



(X) Kaizen (改善)

Continuous Improvement

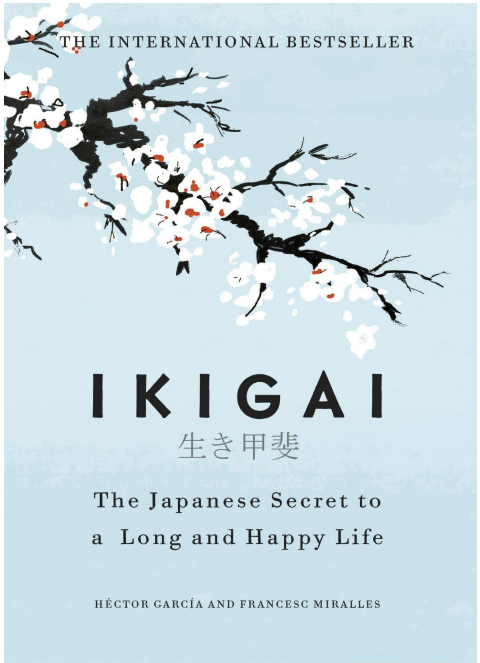
Continuously improve:

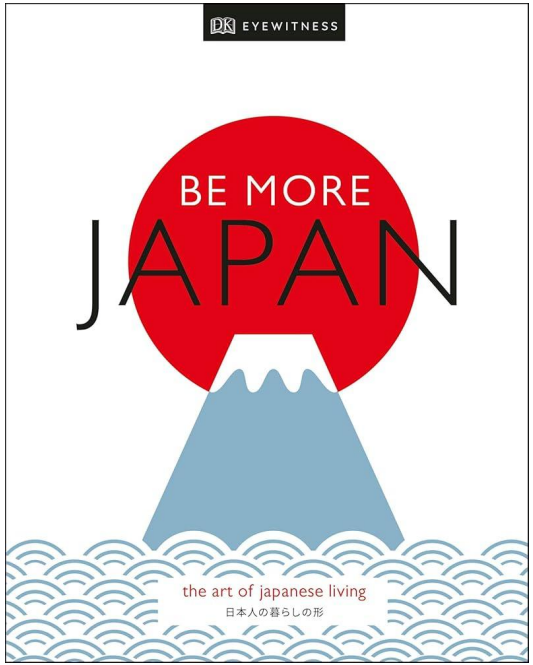
- Usability
- Maintainability
- Robustness
- Bug management
- Requirement/Feature Efficiency
- ..

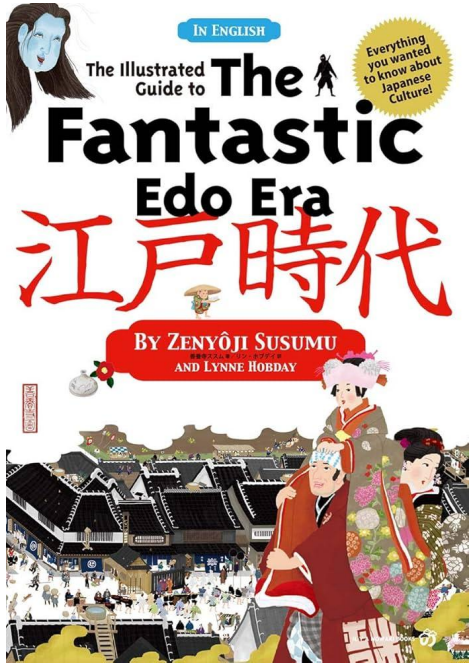

Anyone who can list 10 analogies?

Summary

1. Messenger like a Fox
2. Quality gate, like a Torii gate
3. Design like Origami
4. Prepare like Zen gardens
5. Longevity like Noodles
6. Remove Test Smells like Kintsugi
7. Exploratory Test like a Ronin
8. Shift-Left like growing a Bonsai
9. Document like Calligraphy
10. Continuous Improvement like Kaizen


THE INTERNATIONAL BESTSELLER
IKIGAI
生き甲斐
The Japanese Secret to
a Long and Happy Life
HÉCTOR GARCÍA AND FRANCESC MIRALLES


DK EYEWITNESS
BE MORE
JAPAN
the art of japanese living
日本人の暮らしの形


IN ENGLISH
The Illustrated Guide to The
Fantastic
Edo Era
江戸時代
Everything you wanted to know about Japanese Culture!
BY ZENYÔJI SUSUMU
AND LYNNE HOBDAY

mesutdurukal