LECTURE 04.
TEST ATTRIBUTES. PART I

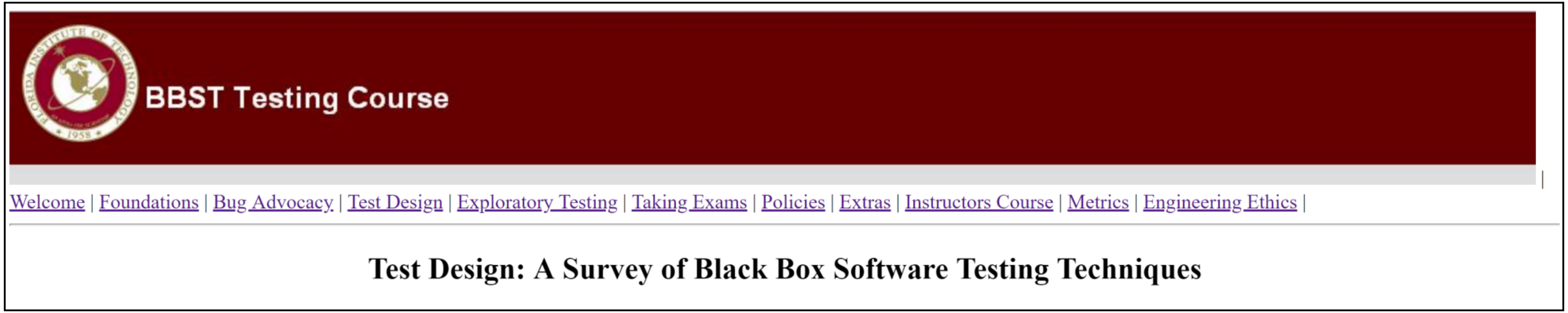
Test Design Techniques
[16 March 2022]

Elective Course, Spring Semester, 2021-2022

Camelia Chisăliţă-Creţu, Lecturer PhD
Babeş-Bolyai University

Acknowledgements

The elective course **Test Design Techniques** is based on the Test Design module available on the **BBST Testing Course** platform.



BBST Testing Course

Welcome | Foundations | Bug Advocacy | Test Design | Exploratory Testing | Taking Exams | Policies | Extras | Instructors Course | Metrics | Engineering Ethics |

Test Design: A Survey of Black Box Software Testing Techniques

The BBST Courses are created and developed by Cem Kaner, J.D., Ph.D.., Professor of Software Engineering at Florida Institute of Technology.

Contents

- Test Attributes
  - Definition. Types
- Test Attributes (first 10 attributes)
  1. Coverage
  2. Information value
  3. Easy to evaluate
  4. Value
  5. Credible
  6. Validity
  7. Power
  8. Representative
  9. Non-redundant
  10. Motivating

Test Attributes

- test differ in their **strengths**, but is harder to describe their **weaknesses**;
- types of attributes: **core** and **desired**;
- each technique has its own **core attributes**;
  - if a test does not have a given *core attribute* ==> bad individual for that technique;

- E.g.:
  1. **Risk-based tests** are meant to be ***powerful***, but this does means they are designed ***not*** to be ***credible***;
     - it would be good for it to be credible, but it's not mandatory (must have to) attribute;
     - if it's not powerful, then it's not a good risk-based test;
  2. **Scenario-based tests** are meant to be ***credible***, but this does means they are designed ***not*** to be ***powerful***;
     - it would be good for it to be powerful, but it's not mandatory (must have to) attribute;
     - if it's not credible, then it's not a good scenario-based test;

- **there are 18 core attributes that emphasize the *differences* among techniques.**

Test Attributes. Types

- Attributes of relevant (good) test cases:

1. Coverage
2. Information value
3. Easy to evaluate
4. Value
5. Credible
6. Validity
7. Power
8. Representative
9. Non-redundant
10. Motivating
11. Performable
12. Reusable
13. Maintainable
14. Supports troubleshooting
15. Appropriately complex
16. Accountable
17. Affordable
18. Opportunity Cost

- each test case has each of these attributes to some degree;

Test Attributes. Test Evaluation

- to evaluate a test, the tester should imagine possible tests that would have ***more of the attribute*** or ***less of it***;
  - Compared to those, where does the addressed test stand?

- **the best way to think about these attributes regarding a test is by comparison;**
- **E.g.:**
  - **Question:** *Why a specific test is powerful/credible/motivating/…?*
  - **The answer** should be organized around comparison with
    - **another test**, or
    - **a hypothetically modified version of the addressed test.**

Test Attributes. Coverage

- **Coverage measures the amount of testing of a given type that you have completed, compared to the population of possible tests of this type;**
- E.g.:
  - **High Coverage/Focus on Coverage:**
    - a test technique is **focused on coverage** if a designer using the technique could readily imagine a coverage measure related to the technique and would tend to create a set of tests that would have high coverage according to that measure;
    - **tours** are **high on coverage**;
    - **function testing** is **high on coverage**, i.e., test suites;
  - **Low Coverage:**
    - a test technique that is **not focused on coverage;**
    - **scenarios** provide **low coverage**;
    - **risk-based testing, e.g., quick-tests,** provides **low coverage**;
- ***no individual test* has much coverage, but *a group of tests* can have high coverage.**

Test Attributes. Information Value

- **The information value of a test reflects the extent to which the test will increase your knowledge (reduce “uncertainty”), whether the program passes or fails the test.**
- **this is one of the most important attribute**;
- E.g.:
  - **Poor Information Value:**
    - most **regression tests** have relatively **little information value**; they are more like demonstrations than like tests because no one expects them to expose many bugs;
    - **A “Pass” teaches the tester almost nothing !**
    - **scenarios** provide **low information value** **when they are** **reused**; **scenarios** provide *high information value* on the program design **only the first time they are run;**
      - ***high information value* is desired for scenarios;**
  - **High Information Value:**
    - when the tester designs tests so that **he will learn something of value** ***whether the program passes or fails the test***;
    - **tours** are **high on information value**.

Test Attributes. Easy to Evaluate

- **A test is easy to evaluate if the tester can determine easily and inexpensively whether the program passed or failed the test;**
- E.g.:
  - **Hard to Evaluate:**
    - **scenario tests** are *often* **hard to evaluate** because the test creates a *lot of output* that has to be inspected by a human;
    - ***easy to evaluate* is desired for scenarios;**
  - **Easy to Evaluate:**
    - **function tests** are *usually* **easy to evaluate** because the test creates a *small amount of output* that can be easily inspected by the tester;
    - **quick-tests** are *usually* **easy to evaluate** because the tester creates them while having little knowledge about the tested program, expecting to find usual mistakes.

Test Attributes. Value

- **A test has value if it reveals things that the clients want to know about the product/project;**
  - E.g.:
    - **Low-value:**
      - some companies treat ***corner cases*** as low value;
      - they consider the *extreme values so extreme* that they don’t care what happens if someone actually pushes the program to those limits;
    - **High-value:**
      - the company *Toys"R"Us* lost a lot of money because their website could not handle high pre-Xmas volume orders;
      - this was an ***extreme value*** that they would have wanted to know about, and that they probably spend a lot of money to study currently.

- **Tests have high value if they are designed to reveal things that are particularly valuable (relevant) to specific stakeholders.**

Test Attributes. Credible

- **A test is credible if**
  - **the stakeholders will believe that people will actually do the things that were done in the test, or**
  - **the events like the ones studied in the test are likely to happen.**

- **Credible vs Value**
  - when someone says *“no one would do that”* ==> **the test credibility is challenged;**
  - when someone says, *“I don’t care what would happen if someone did this”* **==> the test value is challenged.**

  - **value ===> credible**
  - **credible =≠=> value**

Test Attributes. Validity

- **A test is valid if the tester can be sure that the problems it reveals are *genuine problems*;**
  - E.g. **invalidity** test;
    - a failure that occurs only on a system that has insufficient memory, i.e., below the minimum-published requirements;
      - some companies will treat this as a problem if the program doesn’t fail gracefully;
      - others will reject the failure, and the test, as unreasonable.
  - **automated black-box regression testing** allows
    - to run old tests and compare with results got last time;
    - when the code is changed ==> many tests fails ==> **many false positives** ==> the test lack validity ==>

Validity is <u>not</u> a strength of automated black-box regression testing.

Test Attributes. Power

- **A test is powerful if it is designed to be likely to expose a type of error.**
  - A test can be powerful even if it doesn’t find a bug;
    - *question to ask:* **If the program has a bug of this type, will this test expose it?**
  - A test can be powerful with respect to some types of bugs but weak with respect to others;

- *comparison* between tests:
  - **A more powerful test is more likely to expose a type of bug than a test that is less powerful for bugs of that kind.**

Test Attributes. Representative

- **A test is representative if it is focused on actions or events most likely to be tried or encountered by real users;**

- **Credible vs Representative**
  - a test can be credible but unrepresentative:
    - a test that emulates a situation that arises ***0.05% of the time*** is **credible** but **not very representative;**
    - a test that emulates a situation that arises ***every day*** is **representative**.

  - **representative ===> credible**
  - **credible $\not\Longrightarrow$ representative**

Test Attributes. Non-redundant

- **two tests can be *similar* in fundamental ways;**
- E.g.:
  - two tests might be focused on the same risk;
  - two tests might rely on the same data or on values that are only trivially different;

- **A test technique is focused on non-redundancy if it selects *one test from a group of similar ones* and treats that *test as a representative of the larger group*;**
  - E.g.: ***Equivalence class analysis, Domain testing*** - are techniques focused on *non-redundancy*.
    - the tester does not waste time on running similar tests; he tests with one representative of an equivalence class.

Test Attributes. Motivating

- **A test is motivating if the stakeholders will want to fix problems exposed by this test;**
- E.g.:
  - **Motivating:**
    - a problem might be serious enough or potentially embarrassing enough that the company will want to fix it even if it is **not credible** (unlikely to ever arise in practice);
  - **Not motivating:**
    - a problem might be **credible** and **valuable** (the company is glad to know about it), but *the company* ***doesn’t think it is important enough to fix***; perhaps it documents the bug instead to facilitate later bug fixing.

Test Attributes. Other Attributes

- A test is performable if the tester can do the test as designed;
- A test is reusable if it is easy and inexpensive to reuse it;
- A test is maintainable if it is easy to revise in the product interface changes;
- A test supports troubleshooting if it provides useful information for the debugging programmer;
- the design objective is that the tester should use more complex tests as a program gets *more stable*;
- A test is accountable if the tester can explain what he did, justify why he did it, and provide that he actually conducted the test;
- A test affordability is concerned with:
  - the absolute cost of testing in this way and
  - whether the tester could find this information more cheaply, more efficiently;
- The opportunity cost of a test refers to what the tester could have done *instead*, if he hadn't spent his resources running this test;

Test Attributes. Conclusions

- Good test design involves developing tests that
  - can help the tester **to satisfy the information objectives** for the project (or this part of it);
  - **address the things that the tester intends to test** in ways that can reveal the information that he wants to find out about them;
  - **are achievable within their constraints**;
  - include the support materials (code, documentation, etc.) the tester will need for the level of reuse he considers appropriate;
  - **are optimized for the qualities**, e.g. power, **most important for his purposes**.

Test Attributes. Take away

No food has all the vitamins. We have to eat different types of food to achieve nutritional completeness.

No technique will fill all of the tester’s needs. The tester needs to use many techniques, designing each test in a way that makes a given design problem seem easy and straightforward.

No technique is good for everything. A good testing strategy combines many techniques, that have complementary strengths.

References

- **[Kaner2003]** Cem Kaner, An introduction to scenario testing, http://www.kaner.com/pdfs/ScenarioIntroVer4.pdf, 2003.
- **[BBST2011]** BBST – Test Design, Cem Kaner, http://www.testingeducation.org/BBST/testdesign/BBSTTestDesign2011pfinal.pdf.
- **[BBST2010]** BBST – Fundamentals of Testing, Cem Kaner, http://www.testingeducation.org/BBST/foundations/BBSTFoundationsNov2010.pdf.
- **[KanerBachPettichord2001]** Kaner, C., Bach, J., & Pettichord, B. (2001). Lessons Learned in Software Testing: Chapter 3: Test Techniques, http://media.techtarget.com/searchSoftwareQuality/downloads/Lessons_Learned_in_SW_testingCh3.pdf .
- **[Kaner2000]** Kaner, C., Falk, J., & Nguyen, H.Q. (2nd Edition, 2000b), Bug Taxonomy (Appendix) in Testing Computer Software, Wiley, http://www.logigear.com/logi_media_dir/Documents/whitepapers/Common_Software_Errors.pdf
- **[Bach1999]** Bach, J. (1999), Heuristic risk-based testing, Software Testing & Quality Engineering, http://www.satisfice.com/articles/hrbt.pdf
- **[Agruss2000]** Agruss, C. (2000), Software installation testing: How to automate tests for smooth system installation, Software Testing & Quality Engineering, 2 (4), http://www.stickyminds.com/getfile.asp?ot=XML&id=5001&fn=Smzr1XDD1806filelistfilename1%2Epdf
- **[Kahn1967]** Kahn, H. (1967), The use of scenarios, in Kahn, Herman & Wiener, Anthony (1967), The Year 2000: A Framework for Speculation on the Next Thirty-Three Years, pp. 262-264. https://www.hudson.org/research/2214-the-use-of-scenarios
- **[Carroll1999]** Carroll, J.M. (1999), Five reasons for scenario-based design, Proceedings of the 32nd Hawaii International Conference on System Sciences, http://www.massey.ac.nz/~hryu/157.757/Scenario.pdf.