# LECTURE 06.
# TEST ATTRIBUTES. PART II

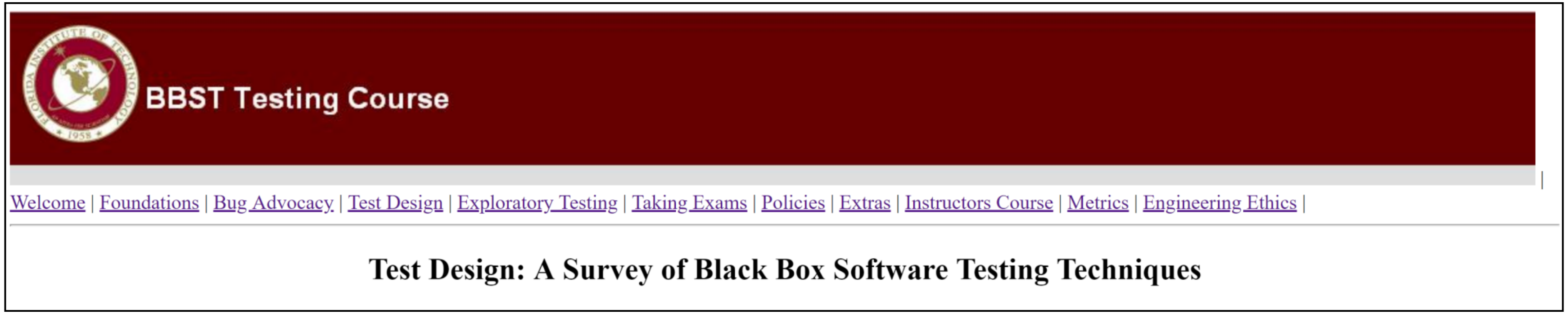
**Test Design Techniques**
**[30 March 2022]**

Elective Course, Spring Semester, 2021-2022

Camelia Chisăliţă-Creţu, Lecturer PhD
Babeş-Bolyai University

# Acknowledgements

The course Test Design Techniques is based on the Test Design course 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

- 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.**

## PART II

Performable. Reusable. Maintainable

Supports troubleshooting. Appropriately complex

Accountable. Affordable. Opportunity Cost

## Test Attributes. Performable

- **A test is performable if the tester can do the test as designed;**
- E.g.:
  - **Not performable:**
    - a manual test that requires the tester *to type lots of data without making mistakes* is **not very performable**;
    - a test that requires the tester *to do something at an exact time* **is not performable**.
  - **Performable:**
    - If the tester can improve performability by storing difficult-to-enter data in files that can be loaded into the test or by automating some pieces of the test that are hard to do by hand;
    - **regression tests** need to be **performable**;

- the tester needs to assess performability *for every test*, especially if he wants anyone else to perform/run the test.

## Test Attributes. Reusable

- **A test is reusable if it is easy and inexpensive to reuse it;**
- E.g.:
  - **Not Reusable:**
    - tests that are **not very performable** are **not easily reused**;
  - **Hard Reusable:**
    - a test can be **highly performable** *today* but **hard to reuse** because the program's design changes frequently ==> **reuse will require maintenance**;
    - **scenarios** are **hard to reuse**;
    - **regression tests** need to be **reusable**;

- **Reusability, performability and maintainability are tightly connected.**

## Test Attributes. Maintainable

- **A test is maintainable if it is easy to revise if the product interface changes;**
- E.g.:
  - **Good Maintainability:**
    - in *automated regression testing* good **maintainability is important**;
  - **Poor Maintainability:**
    - for *many exploratory tests* **maintainability is irrelevant**;
      - if the tester does not intend to reuse them he does not invest time in making them maintainable;

- a test that **lack maintainability** ==> **un-reusable** *quickly*.

# Test Attributes. Supports Troubleshooting

- **A test supports troubleshooting if it provides useful information for the debugging programmer;**
- E.g.:
  - **Low/Hard on Support to Troubleshoot:**
    - **long-sequence tests** must be *very carefully* designed to support troubleshooting. When a test fails after 10 hours of execution of a long sequence, *it can be very hard to figure out what went wrong, or when;*
      - programs often output event logs that provide diagnostic information about unusual or undesirable events; these illustrate ways that the software under test can make tests more **or less effective at supporting troubleshooting**;
  - **High/Easy on Support to Troubleshoot:**
    - **function tests** provides **excellent support for troubleshooting**.

# Test Attributes. Appropriately Complex

- **the design objective is that you should use more complex tests as a program gets *more stable*;**
- E.g.:
  - **Hard to achieve Complexity:**
    - **early testing**==> **complex tests are almost impossible to run**;
      - the tester waste time trying to run complex tests **before the program is stable enough to handle them**;
      - **tours** provides **low complexity**;
      - **function** tests provides **low complexity**;
      - **quick-tests** provides **low complexity**;
  - **Easy to achieve Complexity:**
    - **late testing** ==> the tester can finally run tests that realistically reflect the ways that experienced users will drive the program, to the **appropriate level of complexity**;
    - they provide more **power** and probably more **information value**;
      - **scenarios** provides **high complexity**.

# Test Attributes. Accountable

- **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;**
- E.g.:
  - **High Accountability:**
    - **accountability** is often *critical* for companies whose tests are audited or otherwise likely to be inspected by regulators or in court;
    - **accountability** can be *very costly*, and the cost of it can drive people to rely on **regression tests** (old, documented tests) rather than inventing new ones;
      - **session-based test management** is a popular method for **improving the accountability** of exploratory testing.

# Test Attributes. Affordability

- **the cost of a test** includes
  - the *time* and *effort* associated with it as well as its directly *financial costs*;
- **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;
- E.g.:
  - **High Affordability:**
    - a technique is **more affordable** if it is designed *to reveal better information at the same cost or equivalent information at a lower cost*;
      - **automated testing** done well is **more affordable than manual testing**;
  - **Poor Affordability:**
    - a technique is **less affordable** if it is *reveals the same or equivalent information at a higher cost*;
      - **automated user-level regression testing** is very expensive, with high maintenance costs ==> **poor affordability**.

# Test Attributes. Opportunity Cost

- there is an infinite number of potential tests ==> an infinite number of potential test-related task;
  - every test and every task has *opportunity costs*;
- **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;**
- E.g.: **Opportunity Cost Worthiness:**
    - a common kind of discussion is *whether* **achieving 5% more coverage** of a certain kind is worth the **opportunity cost**, i.e., a different set of tests or reports will never be started if the tester spends his resources this way;
- *question to ask: (when assessing the value of a test)*
  - **How much value he could get from another test activities if he did them instead?**
- *the answer represents*
  - **the missed value of not doing those activities** and
  - **the opportunity cost of the assessed test.**
- **==> the lower *missed value*, the higher *opportunity cost*.**

# Test Attributes. Part I

- Coverage measures the amount of testing of a given type that you have completed, compared to the population of possible tests of this type;
- 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.
- A test is easy to evaluate if the tester can determine easily and inexpensively whether the program passed or failed the test;
- A test has value if it reveals things that the clients want to know about the product/project;
- 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.
- A test is valid if the tester can be sure that the problems it reveals are *genuine problems*;
- A test is powerful if it is designed to be likely to expose a type of error.
- A test is representative if it is focused on actions or events most likely to be tried or encountered by real users;
- 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*;
- A test is motivating if the stakeholders will want to fix problems exposed by 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.