LECTURE 11.
TESTER-BASED TECHNIQUES

Test Design Techniques

[11 May 2022]

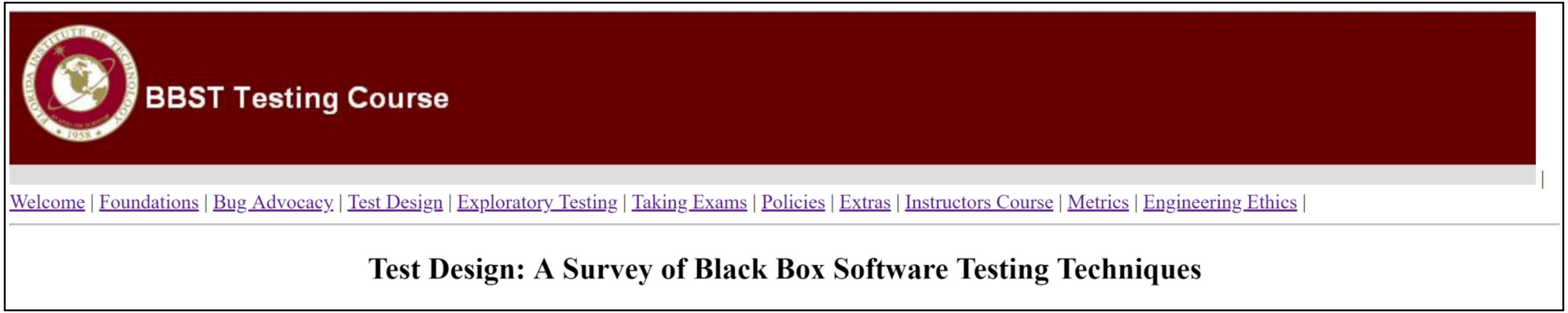
Elective Course, Spring Semester, 2021-2022

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

- Last lecture…
  - **Evaluation-based techniques**
  - **Desired results techniques**
- Terminology
- **Tester-based techniques**
  1. User testing;
  2. Alpha testing;
  3. Beta testing;
  4. Bug bashes;
  5. Subject-matter expert testing;
  6. Paired testing;
  7. Eat your own dog food;
  8. Localization testing.

TDTs Taxonomy

- The main test design techniques are:
  - **Black-box approach:**
    - **Coverage-based techniques;**
    - **Risk-based techniques;**
    - **Activity-based techniques;**
    - **Tester-based techniques;**
    - **Evaluation-based techniques;**
    - **Desired result techniques;**
  - **White-box approach:**
    - **Glass-box techniques.**

Test Case. Attributes

- **A test case** is
  - a question you ask the program. [BBST2010]
  - we are more interested in the *informational goal*, i.e., to gain information; e.g., whether the program will pass or fail the test.
- Attributes of relevant (good) test cases:

  - **Power**
  - **Valid**
  - **Value**
  - **Credible**
  - **Representative**
  - **Non-redundant**
  - **Motivating**
  - **Performable**
  - **Reusable**
  - **Maintainable**
  - **Information value**
  - **Coverage**
  - **Easy to evaluate**
  - **Supports troubleshooting**
  - **Appropriately complex**
  - **Accountable**
  - **Affordable**
  - **Opportunity Cost**

- **A test case has each of these attributes to some degree.**

Tester-based Techniques

- **A tester-based technique considers**
  - **the person that performs the actual testing.**

- there’s a mystique in designing a technique around the type of person who tests;
  - however, what that person will actually do may have little to do with what someone imagine will happen;
- **it allows to identify what activities would be especially useful for these users and by that to gain additional knowledge of the program and its weaknesses.**

Tester-based Techniques. Focus

Tester-based techniques focus on <u>who</u> does the testing.

- E.g.: **user testing** is focused on testing by people who would normally use the product.

Tester-based Techniques

- **Tester-based Techniques:**
  - User testing;
  - Alpha testing;
  - Beta testing;
  - Bug bashes;
  - Subject-matter expert testing;
  - Paired testing;
  - Eat your own dog food;
  - Localization testing.

User Testing. Definition

- **User testing** allows
  - to test the product by the **types of people** who would **typically use your product**;

- it is applied:
  - at any time during development;
  - at developers' site or at the client's site;
  - in carefully directed exercises or at the user's discretion.
- some types of user testing, e.g., **task analysis**, are more like *joint exploration* (involving at least one **user** and at least one member of the company's **testing team**) than like testing by one person.

Testers: Users (ideally, representative of the product market) or people who the company treats as surrogates for users.

Alpha Testing. Definition

- **Alpha testing** allows
  - to test the product by the software development group (programmers and/or testers) on the **developers'** site.

- ***alpha*** is a milestone with different meanings at different companies;
  - it might start immediately after the first feature is finished, e.g., in Extreme Programming, or not until all features are "complete" (coded but probably not yet working).

- generally, the typical alpha period includes:
  - the program is **stable** and **complete** enough for some level of **functional testing**;
  - but not yet stable enough for the ***beta*** milestone;

Testers: Typically programmers and in-house testers who work closely with the programmers.

Beta Testing. Definition

- **Beta testing** allows
  - to test the product by the persons outside the development team **on their own site**.

- typically:
  - the **tests do not need to be approved by the development team**;
  - the testers run their own tests, **at their own pace**, by looking at whatever they consider important to check.

Testers: Typically people external to the company (or at least external to the development group). Representatives of the market or owners of market-relevant equipment.

Beta Testing. Details

- **types of beta testing:**
  - **typical case:** external users run almost-finished software on their own computers;
    - this testing starts at the ***beta*** milestone;
  - **design beta:** user representatives or subject matter experts assess the software's design;
  - **marketing beta:** pre-release to potential large customers, typically later and more stable than at ***beta*** milestone;
  - **compatibility beta:** external users test the product’s compatibility with their software or hardware, typically because they have software or hardware that the development group doesn’t have;
    - this starts as soon as the software can be tested for **compatibility** because adapting the software can be difficult under these circumstances.

- **different beta tests have different goals the tester needs to run them differently.**

Bug Bashes. Definition

- **Bug bashes** considers
  - product testing by anyone who is available, e.g., secretaries, programmers, tech support, managers;
- **it represent a testing event inside a company**, e.g., for an afternoon while having a pizza and drinks;
- it provides usually a very superficial look at the product;
- typically, **it lasts a half-day** and is done when **the software is close to being ready to release**;
- **Note:** some testers do not endorsing it; some companies have found it useful for various reasons, others have not:
  - *often an ineffective replacement for exploratory testing*;
  - *often seen as more effective by non-testing managers* than by the testers.

Testers: Typically employee non-testers or testers who aren't assigned to test this product.

Subject-Matter Expert. Definition

- **Subject-matter expert** are
  - people who have special knowledge of the subject matter of the program;
- E.g.: for a product accounting program:
  - an accountant, a person that understands how financial matters are processed at that product owner;
- **he can provide valuable design suggestions and bug reports;**
- BUT, **he can demand changes that are impossible for the established schedule or low priority to the stakeholders;**
  - the development team need to temper his input with some product development wisdom.

Subject-Matter Expert Testing. Definition

- **Subject-matter expert testing** considers
  - product testing by an expert on some issues addressed by the software, and request feedback (bugs, criticisms, and compliments).
- the expert
  - may or may not be someone you would expect to use the product ---> **his value is his knowledge, not his representativeness of product market or his skill as a tester**;
  - may pair with a tester or programmer, i.e., **he serves as a live oracle**, and this results in knowledge gaining and a new level of training as a side effect of the testing process.

Testers: Someone who is seen as highly knowledgeable about the product category or its risks.

Paired Testing. Definition

- **Paired testing** considers
  - product testing by two people that have different role in product development, e.g., **developer**, **tester**;
- the people:
  - may or may not watch the same screen;
  - may share one computer and trade control of it while they test;
  - may do separate and coordinated tasks;
  - may test on their own machines, with dual-monitor systems (one placed for easy reading by the other tester) so that each tester can easily see what's on the other's screen.
- collaboration might involve:
  - tester reading (specs, bug reports, etc.) or writing up a bug report while the other executes tests;
  - one tester might protect the other's time by dealing with all the visitors, e.g., manager nagging for status report.

Testers: Two people (testers and/or programmers) on the project team, testing together.

Eating Your Own Dog Food. Definition

- **Eating your own dog food** involves
  - using the self developed product to do real work;
- E.g.:
  - a company that uses and relies on pre-release versions of its own software;
- advantages:
  - it often yields **more critical design feedback** (by following the exercised stories) **than beta testing;**
  - it often provides **a harsher and more credible real-world readiness assessment of the software than beta** or **formal in-house testing;**
- **Note:** This type of testing can miss ways that other organizations will use the software:
  - **it might provide false reassurance about the quality of the software.**

Testers: In-house users who do real work with the software.

Localization Testing. Definition

- **Localization testing** involves
  - testing the product by **people from that culture** or who are **fluent in that language** (probably a native speakers);
- approaches:
  - **coverage-based testing:**
    - the focus is on the possibility for a program to be run in a different languages, countries, cultures;
    - it is used to test against a **list of localization-related changes and risks;**
  - **tester-based testing:**
    - the focus is on who is doing testing;
    - it is used when **the software is adapted to another culture or language;**
- the people are regarded as **subject matter experts** who can speak authoritatively about the appropriateness of the localization.

Testers: People from (or deeply familiar with) the target culture.

Next…

- **Lecture 12 + Lab 06 : Wednesday, 18 May,**
    - **TDTP presentation, hours 09:00-12:00, Room A320, 3rd floor, Campus.**
    - **Team scheduling will be available soon.**

References

- **[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.
- **[Whittacker2002]** Whittaker, J.A. (2002). How to Break Software. Addison Wesley.
- **[Jorgensen2003]** Jorgensen, A.A. (2003), Testing with hostile data streams, ACM SIGSOFT Software Engineering Notes, 28(2), http://cs.fit.edu/media/TechnicalReports/cs-2003-03.pdf
- **[Bach2006]** Bach, J. (2006), Heuristic Test Strategy Model, Version 5.7, https://www.satisfice.com/tools/htsm.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