Ontology Summit 2013: Ontology Evaluation Across the Ontology Lifecyle

Track B: Extrinsic Aspects of Ontology Evaluation

Black Box Testing Paradigm in the Lifecycle

TCPC #014168-PA

Mary Balboni, Doug Toppin, Thanh-Van Tran
Intelligence and Information Systems (IIS)
Raytheon Dulles

Black Box Testing Paradigm in the Lifecycle

- **Where does Black Box Testing apply in the Lifecycle?**
- **How does it apply to large database systems?**
- **How do we validate using Black Box Testing?**
- **How well can we measure the completeness and goodness of this testing?**
- **Modern Databases and Black Box Behavior / Functionality Testing**
- **Some Black Box Tests**
- **What are reusable methods that could be used for ontologies?** (assuming the paradigm that ontologies exist in the black box similar to how DBs exist in a black box)
- **What are any security concerns?**

Black Box Testing Paradigm in the Lifecycle

White Box Testing
aka
Structural or Glass Box Testing
Verification : are we building the software right?

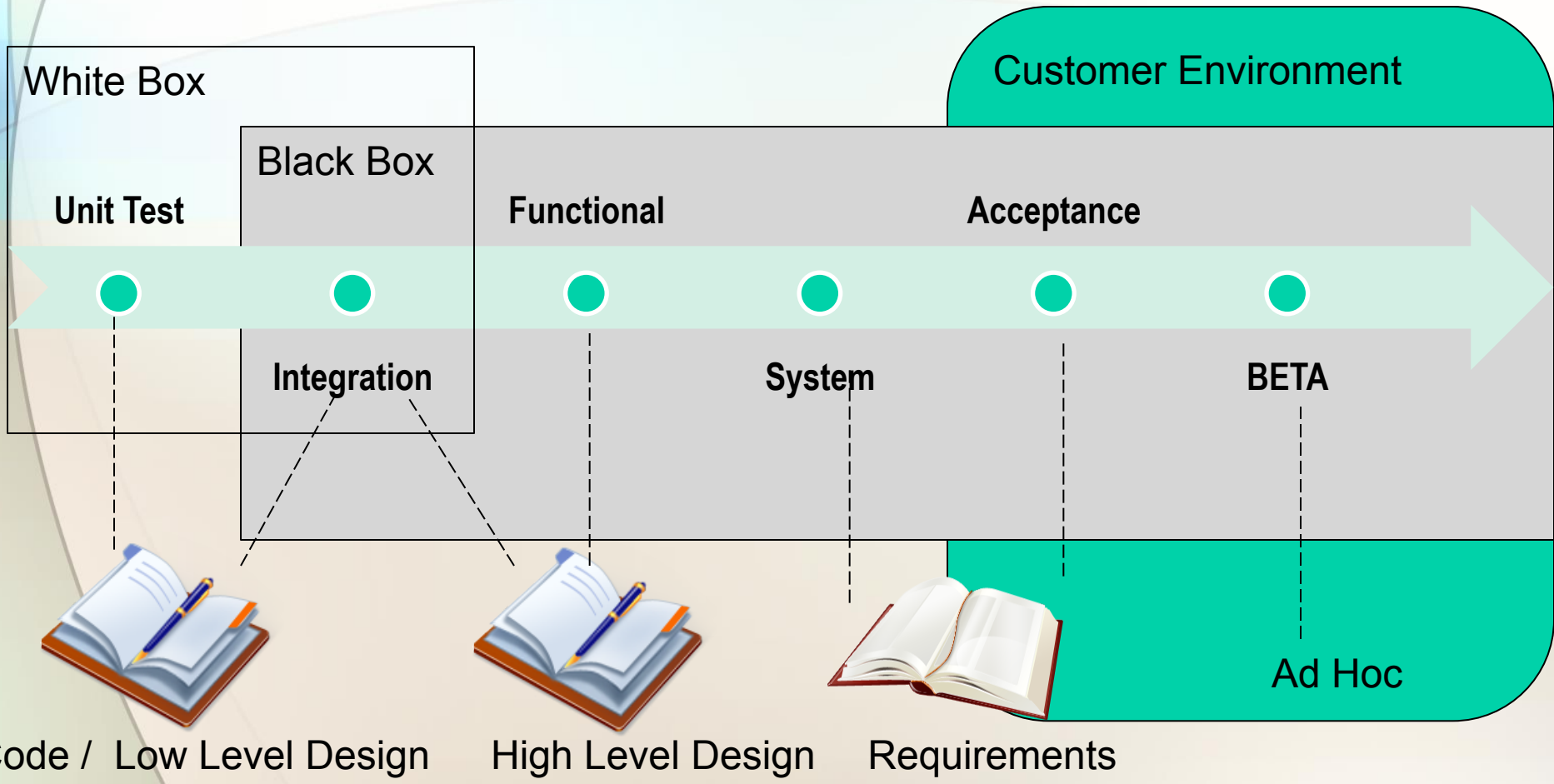
- Code Modules
- Statements
- Branches
- Paths
- Conditions

Black Box Testing
aka
Functional or Behavioral Testing
Validation : are we building the right software?

- Requirements
- Functionality
- Does it behave as expected in all system conditions

Black Box Testing Paradigm in the Lifecycle

- Where does Black Box Testing apply in the Lifecycle?



Black Box Testing Paradigm in the Lifecycle

- **How does it apply to large database systems?**
  - Knowing what data you put in and what the expected results should be
  - **Not** testing if the database is storing the input in the appropriate tables in the database
  - **Database testing is the act of validating the contents, schema, and functionality within a database**
  - If caching is involved testing must include cache control/ expiration tests

Black Box Testing Paradigm in the Lifecycle

- **How do we validate using Black Box Testing?**
  - Test using a replica of the original database (sandboxes)
  - Create database tests based on either rebuilding the existing database or starting with a new creation of the database and related schema
  - Identify Test Data
  - Develop Tests based on System requirements
  - Once the tests are ready, execute and check the results

Black Box Testing Paradigm in the Lifecycle

- **How do we validate using Black Box Testing?**
  - INCLUDE tests for error conditions and degraded mode operations
  - INCLUDE load and capacity testing, this is critical for systems that provide services to a larger enterprise
  - USE HELPFUL TOOL: Facilitated in an SOA environment by tools such as SOAPUI which can discover WSDLs (xml) and provide a test platform
  - If multiple subsystems accessing the database then concurrency and sharing testing should be included

Black Box Testing Paradigm in the Lifecycle

- **How Well Can We Measure the Completeness and Goodness of this Testing?**
  - Predicting Latent Defects left in system after system testing is complete (tools like SEER, SWEEP, based on Rayleigh-like curve[1])
  - Industry standard is in the range:
    - 30 to 85 defects per 1K of code during development
    - ½ to 3 defects per 1K of code latent in the delivered product

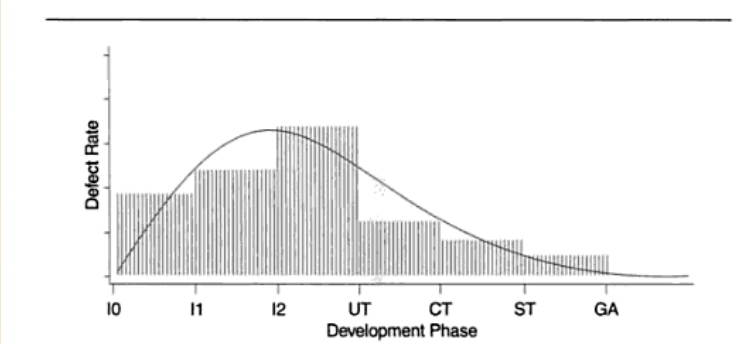


FIGURE 7.2
Rayleigh Model

[1] **Metrics and Models in Software Quality Engineering**
By Stephen H. Kan

Black Box Testing Paradigm in the Lifecycle

- **How Well Can We Measure the Completeness and Goodness of this Testing?**
  - Defect Injection / Defect Seeding Technique
    - How effective are the testers and what kinds of things they miss?
    - The Process of adding known defects to the existing ones is called Defect Seeding. The Idea is while detecting the known bugs unknown bugs might be detected
    - Sample Scenario:

1. Go find your friendly local development team
2. Ask them politely to insert 10-100 bugs throughout the code.
3. Test as normal
4. Go back to your friendly local development team and show them what you found
5. Compare lists
6. Ask your friendly local development team to fix all the bugs they inserted before shipping, please.

Black Box Testing Paradigm in the Lifecycle

- **Modern Databases and Black Box Behavior / Functionality Testing**
  - Modern DBs (NoSQL, document style, Hadoop) may not have established best practices nor has institutional knowledge been developed thus increasing a security risk factor
  - Security related aspects may be dependent on externally configured factors such as access to host/files

Black Box Testing Paradigm in the Lifecycle

- **Black Box Tests**
  - Validate Data Quality by Testing Data Rules
    - Equivalence Partitioning - Suppose system asks for "a number between 100 and 999 inclusive". This gives three equivalence classes of input:
      - less that 100 ; 100 to 999 ; greater than 999
      - We thus test the system against characteristic values from each equivalence class. Example: 50 (invalid), 500 (valid), 1500 (invalid).
    - Boundary Analysis - Arises from the fact that most program fail at input boundaries. Suppose system asks for "a number between 100 and 999 inclusive".
      - The boundaries are 100 and 999. We therefore test for values:

Black Box Testing Paradigm in the Lifecycle

- **Black Box Tests**
  - Validate Data Quality by Testing Data Rules
    - Column Domain – the Flavor column has allowable values of Chocolate, Vanilla, and Strawberry.
    - Column Default - the default value is Strawberry.
    - Value Existence – the value of StartDate must be less than EndDate when EndDate is provided
    - Size Rules - a code in a column must always be two characters in length or a value in a VARCHAR column must be at least five characters in length
  - Automated tests can run continuously in the background checking data consistency

Black Box Testing Paradigm in the Lifecycle

- **Black Box Tests**
  - Validate Data Integrity using Create Read Update and Delete (CRUD) logic
    - C: Create – When user ‘Save’ any new transaction, ‘Create’ operation is performed.
    - R: Retrieve – When user ‘Search’ or ‘View’ any saved transaction, ‘Retrieve’ operation is performed.
    - U: Update – when user ‘Edit’ or ‘Modify’ an existing record, the ‘Update’ operation of DB is performed.
    - D: Delete – when user ‘Remove’ any record from the system, ‘Delete’ operation of DB is performed.

Black Box Testing Paradigm in the Lifecycle

- **Black Box Tests**
  - Validate Relationships – Referential Integrity (RI) aka Business Logic (constraints, triggers, procedures)
    - 'Referential Integrity', relational constraints, triggers and stored procedures. So, using these and many other features offered by DBs, developers implement the business logic on DB level. Tester must ensure that the implemented business logic is correct and works accurately.
    - Would not go into database and check values of triggers, procedures, views or tables – which would be white box texting – but would test by expected results from business logic/rules

Black Box Testing Paradigm in the Lifecycle

- **Black Box Tests**
  - Validate Performance Characteristics of Database Access and Object / Relational (O/R) Mapping
    - User's actions upon the DB should be executed within needed/required time ranges
    - Load / Stress Testing – testing under heavy loads, repetition of certain actions, inputs beyond the limits
    - DB impacts in this area
      - Configuration of memory and processing resources of the computer running the DB
      - Impacted by DB maintenance, defrags that increase efficiency in accessing data
      - Timing of transaction log backups that interrupt database activity

Black Box Testing Paradigm in the Lifecycle

- **Black Box Tests**
  - Ensure ACID Properties of Database Transactions
    - ACID properties of DB Transactions refer to the ‘Atomicity’, ‘Consistency’, ‘Isolation’ and ‘Durability’. Proper testing of these four properties must be done during the DB testing activity. This area demands more rigorous, thorough and keen testing when the database is distributed.
      - Atomicity: a series of DB operations either All occur or Nothing occurs
      - Consistency: ensuring that any transaction will bring the DB from one valid state to another, database follows defined rules
      - Isolation: Ensuring that the concurrent execution of transactions results in a system state that could have been obtained if transactions are executed serially
      - Durability: once a transaction is committed it will remain so despite crashes, errors or loss of power

Black Box Testing Paradigm in the Lifecycle

- **What are reusable methods that could be used for ontologies?** (assuming the paradigm that ontologies exist in the black box similar to how DBs exist in a black box)
  - **Defining Requirements for Each of the Following**
  - **Validating Each of the Following**

    Data Rules

    Data Referential Integrity

    Relationships

    Performance Characteristics

    Properties of Transactions

Black Box Testing Paradigm in the Lifecycle

- **What are any security concerns?**
- Black Box testing alone is *not* a suitable alternative for security activities throughout the software development life cycle.
- Black Box testing is very effective in validating system design assumptions, discovering vulnerabilities and implementation issues that may lead to security vulnerabilities
- Black box tests can;
  - help development and security personnel identify implementation errors that were not discovered during code reviews, unit tests, or security white box tests
  - Discover potential security issues resulting from boundary conditions that were difficult to identify and understand during the design and implementation phases
  - Uncover security issues resulting from incorrect product builds (e.g., old or missing modules/files)
  - Detect security issues that arise as a result of interaction with underlying environment (e.g., improper configuration files, unhardened OS and applications)

Black Box Testing Paradigm in the Lifecycle

- **Testing for Security**
- **Fuzzing:** random character generation – testing by injection random data at the interfaces
- **Syntax testing**: tests based on the syntactic spec of an applications input values
- **Exploratory Testing and Fault Injection**: useful to perform tests without having specific expectations of outcome – helps spot anomalies
- **Data Analysis**: process of trying to understand a program's internals by examining the data it generates – may go beyond observations and go into influencing the program's behavior as well = from security POV, test is to see whether an attacker could do the same thing
- **Monitoring Program Behaviour: referred to as Observability** – to examine the behavior of the program under test and asking whether this observed behavior is symptomatic of a vulnerability in the system

Bios of Authors

- **Mary Balboni –** Over 30 years in engineering, worked through many lifecycles of large systems
- **Doug Toppin** – Over 30 years in engineering, with recent experience with large scale backend commercial / industry internet provider
- **Thanh-Van Tran** – Over 25 years in Database and software engineering, Certified DBA, experience with large scale database systems