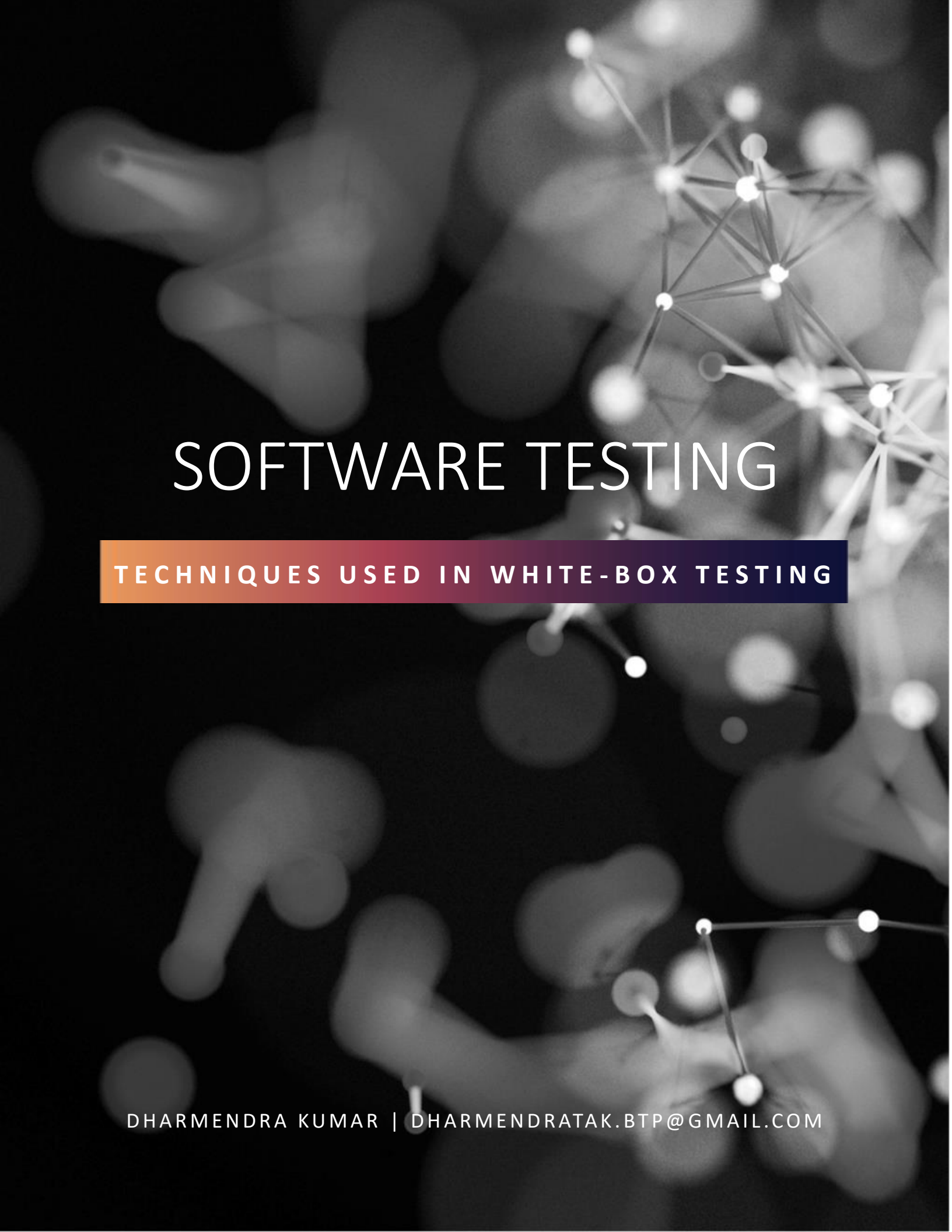

SOFTWARE TESTING

TECHNIQUES USED IN WHITE-BOX TESTING

DHARMENDRA KUMAR | DHARMENDRATAK.BTP@GMAIL.COM

TECHNIQUES USED IN WHITE-BOX TESTING

Techniques used in White-Box testing include:

1) API Testing
2) Test Coverage
3) Fault Injection
4) Mutation Testing
5) Static Testing

API TESTING

API testing involves testing APIs directly (in isolation) and as part of the end-to-end transactions exercised during integration testing. Beyond RESTful APIs, these transactions include multiple types of endpoints such as web services, ESBs, databases, mainframes, web UIs, and ERPs. API testing is performed on APIs that the development team produces as well as APIs that the team consumes within their application (including third-party APIs).

API testing is used to determine whether APIs return the correct response (in the expected format) for a broad range of feasible requests, react properly to edge cases such as failures and unexpected/extreme inputs, deliver responses in an acceptable amount of time, and respond securely to potential security attacks. Service virtualization is used in conjunction with API testing to isolate the services under test as well as

expand test environment access by simulating APIs/services that are not accessible for testing.

API testing commonly includes testing REST APIs or SOAP web services with JSON or XML message payloads being sent over HTTP, HTTPS, JMS, and MQ. It can also include message formats such as SWIFT, FIX, EDI and similar fixed-length formats, CSV, ISO 8583 and Protocol Buffers being sent over transports/protocols such as TCP/IP, ISO 8583, MQTT, FIX, RMI, SMTP, TIBCO Rendezvous, and FIX.

Software used:

Name	Vendor
SoapUI	SmartBear Software
Postman API Platform	Postman(software)
SOAtest	Parasoft
Swagger	SmartBear Software
Katalon Studio	Katalon
Step CI	Step CI

TEST COVERAGE

In computer science, test coverage is a percentage measure of the degree to which the source code of a program is executed when a particular test suite is run. A program with high test coverage has more of its source code executed during testing, which suggests it has a lower chance of containing undetected software bugs compared to a program with low test coverage. Many different metrics can be used to calculate test coverage. Some of the most basic are the percentage of program subroutines and the percentage of program statements called during execution of the test suite.

Test coverage was among the first methods invented for systematic software testing. The first published reference was by Miller and Maloney in Communications of the ACM, in 1963.

FAULT INJECTION

In computer science, fault injection is a testing technique for understanding how computing systems behave when stressed in unusual ways. This can be achieved using physical- or software-based means, or using a hybrid approach. Widely studied physical fault injections include the application of high voltages, extreme temperatures and electromagnetic pulses on electronic components, such as computer memory and central processing units. By exposing components to conditions beyond their intended operating limits, computing systems can be coerced into mis-executing instructions and corrupting critical data. It is often used with stress testing and is widely considered to be an important part of developing robust software. Robustness testing (also known as syntax testing, fuzzing or fuzz testing) is a type of fault injection commonly used to test for vulnerabilities in communication interfaces such as protocols, command line parameters, or APIs.

MUTATION TESTING

Mutation testing is based on two hypotheses. The first is the competent programmer hypothesis. This hypothesis states that competent programmers write programs that are close to being correct. "Close" is intended to be based on behavior, not syntax. The second hypothesis is called the coupling effect. The coupling effect asserts that simple faults can cascade or couple to form other emergent faults. Subtle and important faults are also revealed by higher-order mutants, which further support the coupling effect. Higher-order mutants are enabled by creating mutants with more than one mutation. Mutation testing is done by selecting a set of mutation operators and then applying them to the source program one at a time for each applicable piece of the source code. The result of applying one mutation operator to the program is called a mutant. If the test suite is able to detect the change (i.e. one of the tests fails), then the mutant is said to be killed.

STATIC TESTING

Static Testing is a type of a Software Testing method which is performed to check the defects in software without actually executing the code of the software application. Whereas in Dynamic Testing checks, the code is executed to detect the defects. Static testing is performed in early stage of development to avoid errors as it is easier to find sources of failures and it can be fixed easily. The errors that cannot be found using Dynamic Testing, can be easily found by Static Testing.

Static Testing Techniques: There are mainly two type techniques used in Static Testing:

Review	**Static Analysis**
Informal	Data Flow
Walkthrough	Control Flow
Peer Review	Cyclomatic Complexity
Inspection	


Thank
You