# Why Are You NOT Performing Risk Analysis to Improve Your Testing?

A Risk Analysis Approach to Testing

SQGNE
March 11th, 2020

1

## Speaker Information

Bob Crews
bcrews@checkpointech.com
Office: 813-818-8324
Direct: 813-493-3678

- **CEO/Co-founder**
  Checkpoint Technologies, Inc.
- **President**
  TBQAA (Tampa Bay Quality Assurance Association)
- **Community Director**
  Vivit Board of Directors (Micro Focus' Independent Software User Group)
- **Co-leader**
  Florida Vivit Chapter

2

2

# Agenda

**What We'll Discuss**

Why you can't test everything

The value of performing Risk Analysis

How to improve your testing with Risk Analysis

3

"You can't test a program completely"

4

## Three Reasons Complete Testing is Impossible

| 1 | 2 | 3 |
|---|---|---|
| The domain of possible inputs is *too large* to test | Too many possible paths through the program to test | User Interface issues are *too complex* |

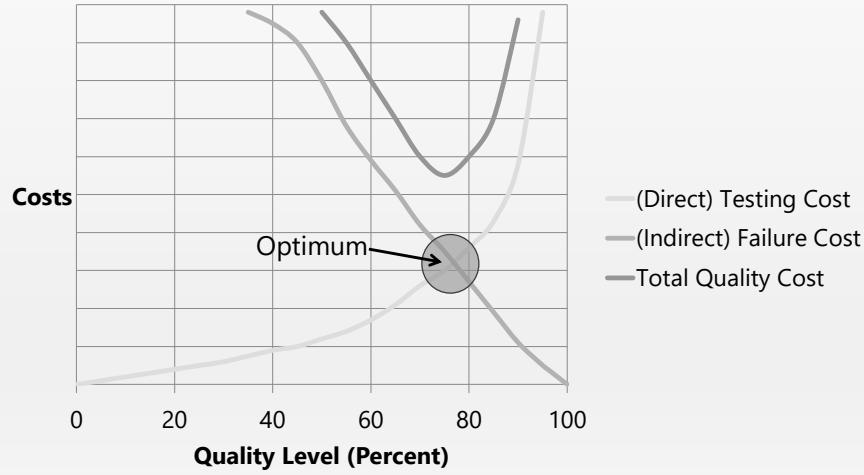
The Point: there are ***numerous*** scenarios and conditions which must be validated

"Testing Computer Software" by Cem Kaner, Jack Falk, and Hung Nguyen

5

GOAL

## Optimize the Testing Effort




6

“Risk comes from not knowing what you’re doing.”

—WARREN BUFFET

7

## Risk

Potential loss to an organization

## Risk Exposure

Measure of the probability of an event times the loss

## Risk Management

Process to identify, quantify, respond to and control risk

## Risk Acceptance

Amount of risk acceptable to the project

## Risk Appetite

Amount of loss an organization is willing to accept for a given risk

## Risk Capacity

*Maximum* amount of loss an organization is willing to accept for a given risk

8

## Risk Identification

Discovery of risks *before* they occur

## Threat

Something capable of exploiting a vulnerability

## Vulnerability

Flaw that may be exploited by a threat

## Inherent Risk

Risk in the *absence* of action

## Residual Risk

Remaining risk *after* the response

## Risk Mitigation

Action to *reduce* threats

## Control

Anything that reduces risk

9

10

## ASSET

Or object of the protection efforts, can be a system component, data, requirement, test or even a complete system

## IMPACT or CRITICALITY

On the organization, were the risk to be realized, can be monetary, reputation, or breach of a law, regulation, or contract

## PROBABILITY is the LIKELIHOOD

That a given event will be triggered

## EXPOSURE

Represents the number of users impacted and/or the "importance" of the users impacted

10


11

# A

Risks are defined before all else

All assets **evolve** from the risks

This is true risk-based approach

Must start very early


12

13

# B

## Risks are defined <u>after</u> creation of assets

You then map assets to risks and adjust accordingly

13

# C

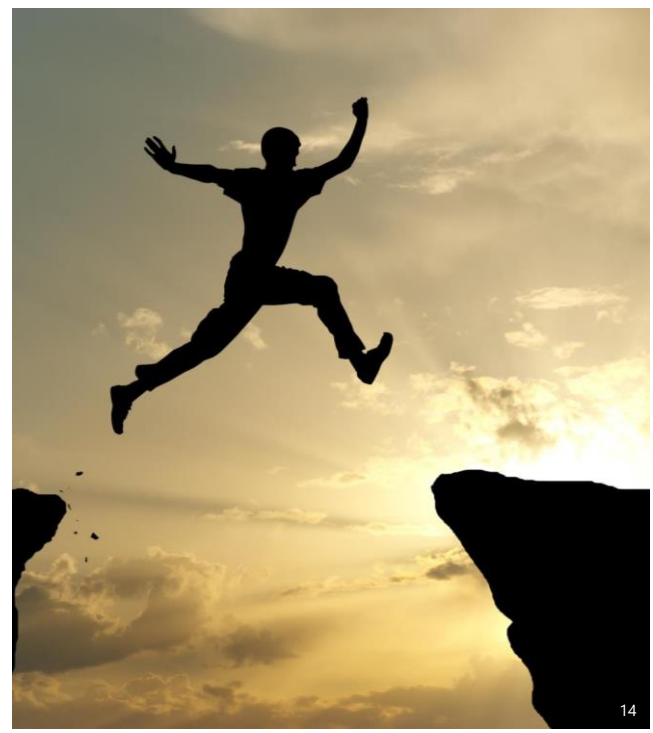
## Risks are "implied" by assets failing

Perform **risk analysis**

Assign risk scores based upon IMPACT and PROBABILITY

Ask "What is the business impact if this fails?"

Ask "What is the probability of this failing?"



14

14

## Benefits of Risk Analysis

| 1 | 2 | 3 | 4 |
|---|---|---|---|
| Running tests in risk order | Allocating test effort based on risk | Allows organizations to make smarter release decisions | If schedule requires, drop test in reverse risk order |
| Find scary stuff first | Pick the right tests | Release when risk of delay balances risk of dissatisfaction | Give up tests you worry about the least |

15

## Develop Risk Analysis Process

Formalize the process!

1. **Create Risk Profile**
   - Define numeric ratings with detailed descriptions (more granularity the better)
   - Develop assessment questionnaire
2. **Assign risk scores to granular assets**
   - Discussed more in a moment
3. **Compile risk assessment database**
   - Improves risk assessment process
   - Helps management plan development projects
4. **Revise risk profile as appropriate**

16

16

## Assigning Risk Scores

**1**

Assemble your list of assets (requirements or tests)

**2**

For each asset, determine the impact if the risk eventuates

**3**

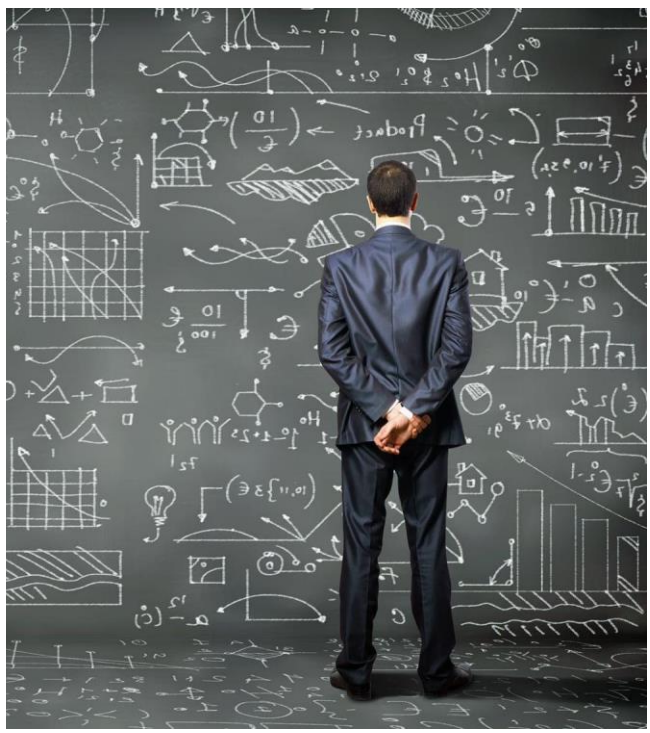
For each asset, determine the likelihood the risk will eventuate

**4**

Calculate the Risk Score: a combination of the risk impact & risk likelihood & (perhaps) weight

17

17



18

18

# Risk Analysis

## IMPACT

Loss of life?
Loss of revenue?
Inconvenience?
Exposure/frequency?

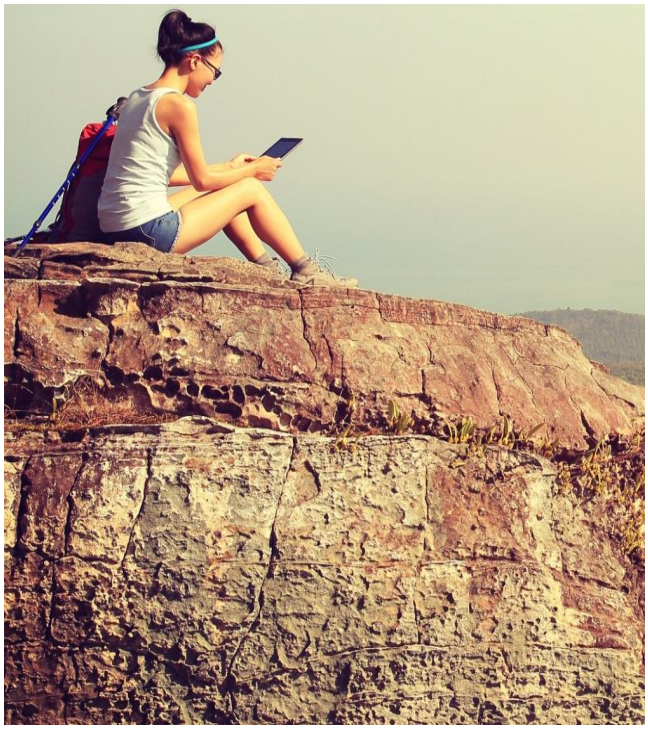
## PROBABILITY

Is it *new* functionality or new technology?
Is it *existing* functionality? Has it been tested before?
Is it *mature* functionality?

## WEIGHT(optional)

Additional factor(s) taken into account and factored into calculation to more adequately determine risk score

18



Knowing definitions previously discussed will enable you to better analyze, define, and communicate the risk.

19

19

# Generic Risks

**(Partial List)**

Be careful of lists!
*"Checklists can inadvertently mix perspectives, which can interfere with the prioritization process."*
Robin Goldsmith

- ***New*** technology (PROBABILITY)
- ***Changed*** (PROBABILITY)
- ***Strategic*** (IMPACT)
- ***Critical*** (IMPACT)
- ***History of defects*** (PROBABILITY)
- ***Recent failure*** (PROBABILITY)

William E. Perry "A Standard for Testing Application Software" 1992 Auerbach Publishers, Boston, MA
Robin Goldsmith "Early and Effective: The Perks of Risk-Based Testing" STP Magazine July 2006

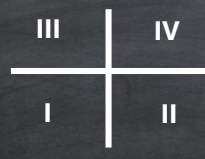
20

# Risk Score Computation Options

➢**Add criteria scores**
- ***Risk Score = (Impact * Weight) + (Probability * Weight)***

➢**Multiply criteria scores**
- ***Risk Score = (Impact * Weight) * (Probability * Weight)***

➢ **Score plotting**
- ***Risk score = Plot Impact score & Probability score on Risk Analysis chart***




21



## Score Plotting Procedure

Three steps to scoring an application:

| 1 | 2 | 3 |
|---|---|---|
| Determine the *Impact* of failure | Calculate the *Probability* score | Plot the scores on the Risk Analysis chart |

Use results to focus test effort:

| 1st | 2nd | 3rd | 4th |
|---|---|---|---|
| Focus on components in Quadrant IV | Focus on components in Quadrant III | Focus on components in Quadrant II | Focus on components in Quadrant I |

22


When would you consider a risk acceptable?
23

23


Besides risk, what other criteria might impact priority?
24

24

# Risk vs Priority

Risk may not always dictate priority (and vice versa)

- Target dates
- Available, acceptable workarounds
- Management
- Customers

Is Risk "acceptable"?

25

## Software Quality Factors

**Attributes of software**

**Needed for trade off decisions**

**Should be included in software requirements**

**Should be included in the test plan**

**Lack of the needed factors cause customer dissatisfaction**


26

# Software Quality Factors

| Quality Categories | Quality Factors | Broad Objectives |
|---|---|---|
| Product Operation | Correctness<br>Reliability<br>Efficiency<br>Integrity<br>Usability | Does it do what the customer wants?<br>Does it do it accurately all of the time?<br>Does it quickly solve the intended problem?<br>Is it secure?<br>Can I run it? |
| Product Revision | Maintainability<br>Testability<br>Flexibility | Can it be fixed?<br>Can it be tested?<br>Can it be changed? |
| Product Transition | Portability<br>Reusability<br>Interoperability | Can it be used on another machine?<br>Can parts of it be reused?<br>Can it interface with another system? |

A quality factor represents the behavioral characteristic of a system.
Examples: correctness, reliability, efficiency, testability, portability, …

27

# Software Quality Factors

| Factors | Definition |
|---|---|
| **Correctness** | **Extent to which a program satisfies its specifications and fulfills the user's mission objective.** |
| **Reliability** | **Extent to which a program can be expected to performs its intended function with required precision.** |
| **Efficiency** | **The amount of computing resources and code required by a program to perform a function.** |
| **Integrity** | **Extent to which access to software or data by unauthorized persons can be controlled.** |
| **Usability** | **Effort required learning, operating, preparing input, and interpreting output of a program.** |
| **Maintainability** | **Effort required locating and fixing an error in an operational program.** |
| **Testability** | **Effort required testing a program to ensure that it performs its intended function.** |
| **Flexibility** | **Effort required to modify an operational program.** |
| **Portability** | **Effort required to transfer software from one configuration to another.** |
| **Reusability** | **Extent to which a program can be used in other applications - related to the packaging and scope of the functions that programs perform.** |
| **Interoperability** | **Effort required to couple on system with another.** |

28

# Software Quality Criteria

1. Access audit: Ease with which software and data can be checked for compliance with standards or other requirements.
2. Access control: Provisions for control and protection of the software and data.
3. Accuracy: Precision of computations and output.
4. Communication commonality: Degree to which standard protocols and interfaces are used.
5. Completeness: Degree to which a full implementation of the required functionalities has been achieved.
6. Communicativeness: Ease with which inputs and outputs can be assimilated
7. Conciseness: Compactness of the source code, in terms of lines of code.
8. Consistency: Use of uniform design and implementation techniques and notation throughout a project.
9. Data commonality: Use of standard data representations.
10. Error tolerance: Degree to which continuity of operation is ensured under adverse conditions.
11. Execution efficiency: Run time efficiency of the software.
12. Expandability: Degree to which storage requirements or software functions can be expanded.

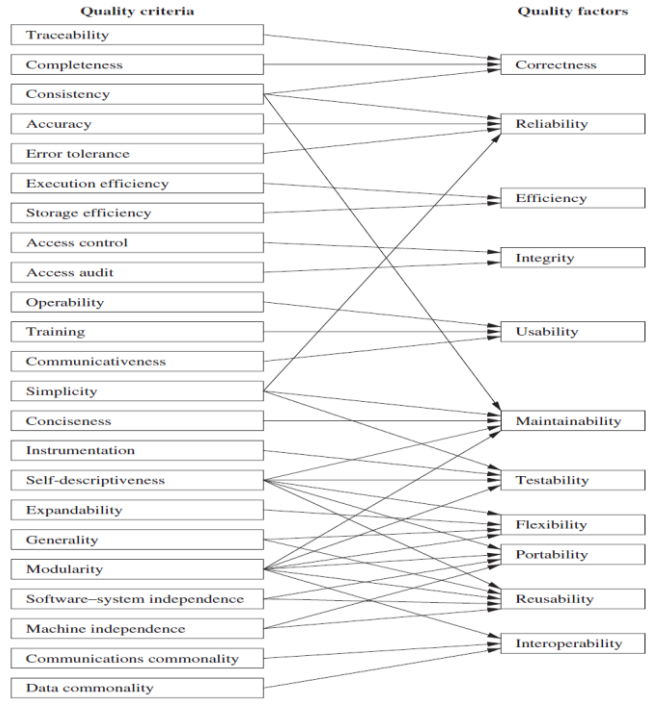
29

# Software Quality Criteria

13. Generality: Breadth of the potential application of software components.
14. Hardware independence: Degree to which software is dependent on the underlying hardware.
15. Instrumentation: Degree to which the software provides for measurement of its use or identification of errors.
16. Modularity: Provision of highly independent modules.
17. Operability: Ease of operation of the software.
18. Self-documentation: Provision of in-line documentation that explains implementation of components.
19. Simplicity: Ease with which the software can be understood.
20. Software system independence: Degree to which the software is independent of its software environment—nonstandard language constructs, operating system, libraries, database management system, etc.
21. Software efficiency: Run time storage requirements of the software.
22. Traceability: Ability to link software components to requirements.
23. Training: Ease with which new users can use the system.

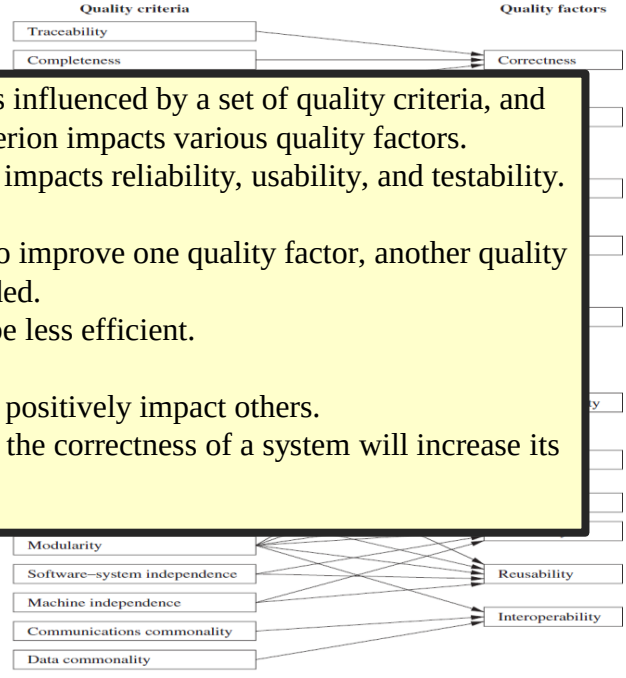
30

# Map Software Quality Factors & Quality Criteria




31

# Map




Each quality factor is influenced by a set of quality criteria, and the same quality criterion impacts various quality factors.
Example: Simplicity impacts reliability, usability, and testability.

If an effort is made to improve one quality factor, another quality factor may be degraded.
Portable code may be less efficient.

Some quality factors positively impact others.
An effort to improve the correctness of a system will increase its reliability.

32


33

# Component Scoring Procedure

Impact of Failure

| Rating | Description |
|---|---|
| 0 = | No impact |
| 1 = | Minor impact |
| 2 = | Minor impact, but some inconvenience |
| 3 = | Minor impact, some customers notice problem |
| 4 = | Moderate impact, little monetary loss |
| 5 = | Moderate impact, little monetary loss, workarounds needed |
| 6 = | Moderate impact, little monetary loss, workarounds needed, customers notice |
| 7 = | Moderate impact, significant monetary loss, workarounds needed, customers notice |
| 8 = | Major impact, major loss, no workarounds available, customers notice |
| 9 = | Major impact, major loss, no workarounds available, customers notice, recovery difficult |
| 10 = | Major impact, major loss, no workarounds available, customers notice, company-wide processing halted |

34

34

# Component Scoring Procedure

Likelihood of Failure

- **Complexity** **Weight of 3**
- **Frequency of use** **Weight of 2**
- **New functionality** **Weight of 1**

**Rate Components on each of the three factors:**

**High (5); Medium (3); Low (1)**

- **((C x 3) + (F x 2) + N)/3 = Probability of Failure**

35

# Component Scoring Procedure

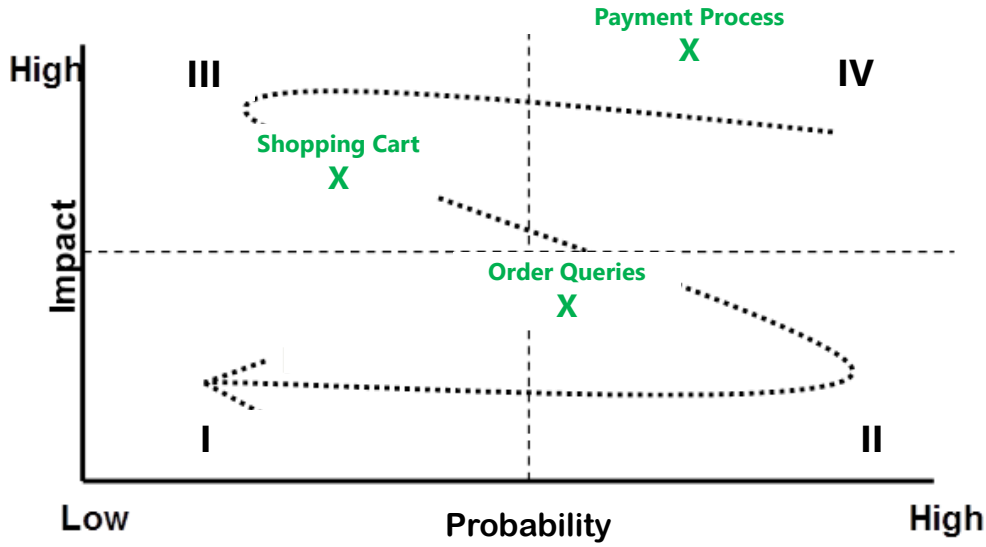
((C x 3) + (F x 2) + N)/3 = Probability of Failure

| Risk Factor/ Module | Complexity | Frequency of Use | New | Probability Score | Impact Score |
|---|---|---|---|---|---|
| Weight of Risk | 3 | 2 | 1 | *WC+WF+WN/3* | 1 to 10 |
| **Shopping Cart** Module: Select Items | 1 | 2 | 3 | **10/3 = 3.33** | **7** |
| **Payment Process** Module: Credit Card Payment | 5 | 1 | 3 | **20/3 = 6.77** | **10** |
| **Order Queries** Shipping Query | 3 | 3 | 1 | **16/3 = 5.1** | **4** |

36

36

# Component Scoring Procedure




37

37

## Rapid Risk Scoring Process

1. Get team or designated group together
2. Each individual gets card with scores of: 1 (Very Low), 2 (Low), 3 (Medium), 4 ( High), and 5 (Very High). Can go more granular with wider range
3. Describe entity with same type of description for each. (Is it new? What's frequency? Exposure? Has it been a problem in the past?)
4. Give each person 5 seconds to hold up score card for Impact…for Probability
5. Average scores of members (for Impact…for Probability)
6. Compute risk score – add, multiply, or plot

38

# Rapid Risk Exercise: Plot Scores

**IMPACT** if functionality fails:
1. Low
2. Medium
3. High
4. Very High

**PROBABILITY**, or LIKELIHOOD, functionality may fail:
1. Low
2. Medium
3. High
4. Very High

**A. Login process**
1. Can successfully login with valid username & password
2. No customer service phone # displayed until login
3. Mature functionality
4. Performed > 12,000/day

**B. Product search**
1. Provides detailed product info and competitive comparison
2. Can call into customer service
3. New functionality/technology
4. Performed 8,000 – 9,000 per day

**C. Order checkout**
1. Includes process and accept payment
2. Can call customer service
3. Mature functionality but history of failures
4. Performed 3,000 – 6,000 per day

**D. Product return**
1. Includes process payment refund
2. Can call customer service
3. Mature functionality, stable history
4. Performed 100 - 500 per day

39



# Summary

**You can't test EVERYTHING! But you CAN focus on riskiest.**

**There's Risk Analysis and there's Risk-Based Development**

**Know Risk Terminology (Helps define, analyze, and communicate risk)**

**Understand what's important to customer (Generic Risks and Quality Factors)**

**Detailed Risk Analysis Approach**

1. *Assemble* assets
2. For each…*determine* IMPACT & PROBABILITY (can use optional WEIGHT)
3. For each…*calculate* RISK score
4. *Plot* on Risk Analysis Chart (four quadrants)

**Rapid Risk Scoring Agile Approach**

1. *Gather* team
2. *Describe* asset
3. *Determine* IMPACT & PROBABILITY
4. *Calculate* RISK Score
5. *Plot*

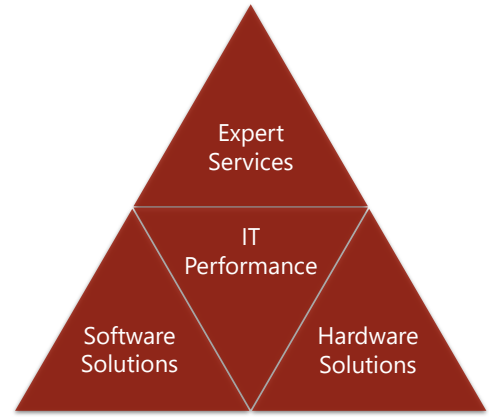
# Questions?

41

checkpoint TECHNOLOGIES
Software Quality. Assured.

## About Checkpoint Technologies

- ✓ Incorporated in January, 2003
- ✓ QA and QC expertise focused on functional, performance and application security testing
- ✓ Micro Focus Software Platinum Partner, Authorized Software Support Partner & Certified Training Partner
- ✓ Atlassian Solution Partners
- ✓ Also partners with Mobile Labs, Kobiton, and Tricentis (QASymphony)
- ✓ QAI Worldwide Training Partner




42

**Contact me anytime!**

## Bob Crews

Office: 813-818-8324
Direct: 813-493-3678

## Social Media

LinkedIn: linkedin.com/in/bob-crews-checkpointech
LinkedIn: linkedin.com/company/checkpoint-technologies
Twitter: @BobCrews_CPTech

43


44