**פרק 8 Sommerville :**

.04 מהן מטרות תהליכי הבדיקות?

.05 הסבר את המשפט של Dijkstra,1972 השתמש במונחי תיקוף ואימות:

בדיקות יכולות להראות את הימצאותן של הטעויות, אך לא את העדרן.

.06 מהם ההבדלים בין חקירת תוכנה לבדיקות תוכנה?

.07 מה ההבדל בין בדיקות שיחרור לבדיקות משתמש?

.08 מהן הבדיקות הקשורות בבדיקות הפיתוח?

.09 האם יש קשר בין תהליכי הבדיקות השונים? הסבר ותן דוגמאות.

.21 מהן שתי הגישות הרווחות לגבי טעויות אנוש?

.20 כיצד ניתן לבחור את מקרי הבדיקה. הסבר את המונחים בדיקות מבוססות הנחיות ובדיקות מבוססות חלוקה, לדוגמה

קבוצות שקילות.

.22 מה ההבדל בין בדיקות יחידה לבדיקות של רכיבים?

.23 מה החידוש שך TDD? מה האתגר?

24 . מתי נכון להשתמש באוטומציה בבדיקות?

25 . מה מייחד בדיקות של תוכנות מדף?

שאלה4: מהן מטרות תהליכי הבדיקות?

The testing process has two distinct goals:

1. To demonstrate to the developer and the customer that the software meets its

requirements. For custom software, this means that there should be at least one

test for every requirement in the requirements document. For generic software

products, it means that there should be tests for all of the system features, plus

combinations of these features, that will be incorporated in the product release.

2. To discover situations in which the behavior of the software is incorrect, undesirable,

or does not conform to its specification. These are a consequence of

software defects. Defect testing is concerned with rooting out undesirable system

behavior such as system crashes, unwanted interactions with other systems,

incorrect computations, and data corruption.

שאלה 5: הסבר את המשפט של Dijkstra,1972 השתמש במונחי תיקוף ואימות:

בדיקות יכולות להראות את הימצאותן של הטעויות, אך לא את העדרן.

בדיקות אינן מוכיחות כי התוכנה בנויה ללא פגמים כלומר ניתן לבדוק את התוכנה ולמצוא טעויות בצורה טכנית אבל לפעמים יש מחסור בשורות קוד באלגוריתם עצמו ולכן הוא אינו משיג את מטרתו ואת זה בדיקות לא יכולות לגלות אלא רק באמצעות בדיקות אימות ותיקוף אשר ייבדקו האם התוכנה השיגה את מטרתה והלקוח מרוצה והיא עונה על דרישותיו.

שאלה 6: מהם ההבדלים בין חקירת תוכנה לבדיקות תוכנה?

There are several differences:

1. During testing, errors can mask (hide) other errors. When an error leads to

unexpected outputs, you can never be sure if later output anomalies are due to

a new error or are side effects of the original error. Because inspection is a

static process, you don’t have to be concerned with interactions between

errors. Consequently, a single inspection session can discover many errors in a

system.

1. Incomplete versions of a system can be inspected without additional costs. If

a program is incomplete, then you need to develop specialized test harnesses

to test the parts that are available. This obviously adds to the system development

costs.

1. As well as searching for program defects, an inspection can also consider

broader quality attributes of a program, such as compliance with standards,

portability, and maintainability. You can look for inefficiencies, inappropriate

algorithms, and poor programming style that could make the system difficult to

maintain and update.

שאלה 7: מה ההבדל בין בדיקות שחרור לבדיקות משתמש?

User testing is essential, even when comprehensive system

and release testing have been carried out.

שאלה 8: מהן הבדיקות הקשורות בבדיקות הפיתוח?

During development, testing may be carried out at three levels of granularity:

1. Unit testing, where individual program units or object classes are tested. Unit

testing should focus on testing the functionality of objects or methods.

2. Component testing, where several individual units are integrated to create composite

components. Component testing should focus on testing component

interfaces.

3. System testing, where some or all of the components in a system are integrated

and the system is tested as a whole. System testing should focus on testing component

interactions.

שאלה 9: האם יש קשר בין תהליכי הבדיקות השונים? הסבר ותן דוגמאות.

two possible strategies here that can be effective in helping you choose

test cases. These are:

1. Partition testing, where you identify groups of inputs that have common characteristics

and should be processed in the same way. You should choose tests from

within each of these groups.

2. Guideline-based testing, where you use testing guidelines to choose test cases.

These guidelines reflect previous experience of the kinds of errors that programmers

often make when developing components.

שאלה 20: כיצד ניתן לבחור את מקרי הבדיקה. הסבר את המונחים בדיקות מבוססות הנחיות ובדיקות מבוססות חלוקה, לדוגמה

קבוצות שקילות.

מטרת עיצוב הבדיקה הוא לזהות תנאי בדיקה ומקרי בדיקה.

זהוי הבחנה קלאסית להבדיל את טכניקות הבדיקה לטכניקות הקופסא השחורה והלבנה. טכניקות קופסא שחורה (או טכניקת מבוססת מפרט) שואבות את תנאי הבדיקה ומקרי הבדיקה מבסיס הבדיקה – בדרך כלל מסמכי איפיון. הטכניקה מבצעת בדיקות פונקציונאליות ולא פונקציונאליות עבור רכיב או מערכת, מבלי להתייחס למבנה הפנימי שלהם. טכניקת קופסא לבנה (או טכניקת מבוססת מבנה) מתבססת על ניתוח המבנה הפנימי של הרכיב או המערכת.

שאלה 21: מהן שתי הגישות הרווחות לגבי טעויות אנוש?

Human errors made during the specification, design, and development stages may mean that faults are introduced into the system.

שאלה 22: מה ההבדל בין בדיקות יחידה לבדיקות של רכיבים?

Testing composite components should therefore focus on showing that the component

interface behaves according to its specification. You can assume that unit tests on the individual objects within the component have been completed.

שאלה 23: מה החידוש של TDD? מה האתגר?

An automated testing environment, such as the JUnit environment that supports Java program testing (Massol and Husted, 2003), is essential for TDD. testing tools can be extended to integrate some aspects of system testing with TDD. However, there is no evidence that TDD leads to poorer quality code.

שאלה 24: מתי נכון להשתמש באוטומציה בבדיקות?

Whenever possible, you should automate unit testing. In automated unit testing, you make use of a test automation framework (such as JUnit) to write and run your program tests.

שאלה 25: מה מייחד בדיקות של תוכנות מדף?

During system testing, reusable components that have been separately developed and off-the-shelf systems may be integrated with newly developed components. The complete system is then tested.