**ניהול איכות תוכנה**

**סילבוס פרק 1:**

**מבוא ופרק 1: יסודות בדיקת תוכנה**

**1.1 מדוע בדיקות נחוצות (K2) מושגי מפתח:**

* **שגיאה/טעות** – פעולה אנושית המייצרת תוצאה שגוייה ולפיכך גורמת ל- פגם/תקלה/באג. (IEEE – ארגון תקינה מוביל, בינלאומי שאחראי על התקן להנדסת תוכנה). 🡨
* **פגם/תקלה/באג** – ליקוי ברכיב או במערכת אשר עשוי לגרום להם להיכשל בביצוע התפקוד הנדרש מהם. ליקוי כזה יכול להיות למשל פקודה או הגדרת נתונים שאינן נכונות. כאשר נתקלים בפגם תוך כדי ביצוע עשוי להיגרם כשל של הרכיב או המערכת 🡨
* **כשל**- סטייה של רכיב או מערכת מהמסירה (delivery), מהשירות או מהתוצאות המצופים ממנו.
* **איכות**- המידה שבה רכיב, מערכת או תהליך עומדים בדרישות שאופיינו ו/או בצרכים ובצפיות של משתמש/ לקוח.
* **סיכון**- גורם שתוצאותיו בעתיד עשויות להיות שליליות, לרוב מבוטא כהשפעה (impact) או כסבירות (likelihood).
* **ההקשר: מערכות תוכנה-** התוכנה נמצאת בכל מקום למשל בשעון, מעלית , מכונית מטוס ... שכן, כשלי תוכנה בחיי היום יום עשויים להיות מלווים במחיר של כסף ואף במחיר של חיי אדם.
* **הגורמים לפגמים בתוכנה-**אדם יכול לשגות (לעשות טעות), ובכך לגרום לפגם (ליקוי, באג) בקוד התוכנה או במסמך. אם הפגם בקוד יבוצע, המערכת עלולה להימנע מפעולה שהיה עליה לבצע (או לבצע פעולה ממנה היה עליה להימנע), ולגרום לכשל (failure). פגמים בתוכנה, במערכות או במסמכים עלולים לגרום לכשלים, אם כי לא כל הפגמים אכן גורמים לכך בפועל. פגמים קורים משום שבני אדם עשויים לטעות ומשום שיש לחץ של זמן, קוד מורכב, תשתית מורכבת, טכנולוגיות משתנות ו/ או יחסי גומלין מרובים בתוך המערכת (system interactions).כשלים עשויים להיגרם גם על ידי תנאי הסביבה. למשל: קרינה, מגנטיות, שדות אלקטרוניים וזיהום יכולים לגרום לתקלות בקושחה (firmware), או להשפיע על ביצועי התוכנה על ידי שינוי בסביבה ובתנאים בהם החומרה פועלת.
  + - **בדיקות ואיכות**:
* **דרישות**
* **דרישות תפקודיות (פונקציונליות):**

דרישה שמפרטת פונקציונליות (תפקוד) שרכיב או מערכת חייבים לבצע. (הסבר בהמשך)

**בדיקות תפקודיות (פונקציונליות):**

בדיקות המבוססות על ניתוח מפרט הפונקציונליות של רכיב או מערכת.

* **דרישות לא- תפקודיות:**
  + **אמינות (reliability)-**  היכולת של מוצר/ תוכנה לבצע את התפקידים (function) הנדרשים, בתנאים נתונים ולאורך תקופת זמן שהוגדרה או עבור פעולות (operation) במספרן הוגדר.
  + **שימושיות (usability)-** היכולת של התוכנה להיות מובנת, נלמדת ברת-שימוש ומושכת עבור המשתמש, כאשר נעשה בה שימוש בתנאים מוגדרים.
  + **יעילות (efficiency)-** א) יכולתו של מוצר תוכנה לספק ביצועים מתאימים, ביחס לכמות המשאבים בהם משתמשים בתנאים שהוגדרו. ב) יכולתו של תהליך לייצר את התוצרים שהתכוונו אליהם, ביחס לכמות המשאבים שבהם נעשה שימוש
  + **תחזוקתיות (maintainability) –** מידת הקלות שבה ניתן לשנות מוצר/תוכנה לשם: תיקון פגמים, התאמה לדרישות חדשות, ביצוע קל יותר של תחזוקה עתידית והתאמה לסביבה שהשתנתה
  + **ניידות (portability)-** מידת הקלות שבה ניתן להעביר מוצר/תוכנה מסביבה אחת של חומר או תוכנה לסביבה אחרת.
  1. **הבדיקות מה הן? (K2) מושגי מפתח**
     + **דרישות (requirements)-** מצב או יכולת שהמשתמש זקוק להם כדי לפתור בעיה או על-מנת להשיג מטרה אשר חובה עלינו לעמוד בה (או שנכפית על רכיב או מערכת כדי לעמוד בחוזה, תקן, אפיון, או כל מסמך פורמלי אחר מחייב).
     + **סקירה (review) –** הערכת הסטטוס של מוצר או פרויקט כדי: לברר אם קיימים פערים מהתוצאות המתוכננות , להציע שיפורים (למשל: סקירת הנהלה, סקירה לא רשמית, סקירה טכנית, ביקורת ,דיון מודרך וכו')
     + **בדיקות (testing)-** תהליך הכולל את כל פעילויות מחזור החיים בין אם הם סטטיות ובין אם הם דינמיות. שלב הבדיקות עוסק בתכנון ובהערכה של מוצרי תוכנה ותוצרי עבודה הקשורים אליהם. כל זאת, על מנת לוודא שהבדיקות עונות לדרישות שהוגדרו ואף מתאימים להשגת המטרה וכמובן למציאת פגמים/כשלים/תקלות.
     + **יעד הבדיקות (test objective)-** סיבה או מטרה שלשמה מעצבים ומבצעים בדיקה.
     + **מקרה בדיקה (test case)-** אוסף של נקודות שלפיו אנו מבצעים את הבדיקה. כלומר, נבצע בדיקות ל- ערכי קלט, תנאי קדם ביצוע, תוצאות צפויות ותנאים שלאחר ביצוע.
     + **ניפוי באגים (debugging)-** תהליך של מציאת הסיבות לכשלים בתוכנה. כלומר מציאת הפגם/תקלה/באג שגרם לכשל. וניתוח הסיבות לכך והסרתן.

**שבע עקרונות הבדיקה**

* **עקרון 1-** בדיקות המצביעות על נוכחותם של פגמים , אבל לא מוכיחות שאין עוד פגמים(אם לא נמצאו זה לא אומר שלא קיימים)
* **עקרון 2-** בלתי אפשרי לבצע בדיקות ממצות, ולכן חשוב לדעת מה לבדוק, לפי סיכונים ותעדוף.
* **עקרון 3-** בדיקות מוקדמות(early testing)**,** ככל שנתחיל לבדוק בשלב מוקדם יותר, יותר טוב.
* **עקרון 4-** התקבצות פגמים (defect clustering) יש למקד את מאמץ הבדיקות באופן יחסי לצפיפות הפגמים הצפויה, ולצפיפות הפגמים הנצפת בפועל.
* **עקרון 5-** פרדוקס ההדברה (pesticide paradox). אם חוזרים על בדיקות זהות שוב ושוב, בסופו של דבר אותה קבוצת מקרי בדיקה לא תגלה עוד פגמים חדשים.
* **עקרון 6-** בדיקות הן תלויות הקשר (context dependent), למערכות שונות בודקים דברים שונים ומבצעים בדיקות אחרות.
* **עקרון 7-** אשליית העדר שגיאות (absence-of-errors fallacy)- אין טעם לגלות פגמים ולתקן אותם אם המערכת אינה שמישה ואינה מספקת את צרכי המשתמש, מצב בו לא נותרו פגמים ידועים אבל המערכת לא עומדת בדרישות ובצורך שלה.

**תהלכי הבדיקה היסודיים**

**מושגים:**

* **מדיניות בדיקות (test policy) –** מסמך המתאר את העקרונות, הגישה והמטרות העיקריות של הארגון בנוגע לבדיקות
* **בסיס בדיקות (test basis)-** כל המסמכים שמתוכם ניתן להסיק את הדרישות של רכיב או מערכת (מקרי בדיקה מבוססים על מסמך זה). בסיס הדרישות כולל בתוכו: דרישות הרכיבים, עיצוב מפורט וקוד.
* **תכנית בדיקות (test plan)-** מסמך המתאר את ההיקף, הגישה, המשאבים ולוח הזמנים של פעילויות בדיקה שבכוונתנו לעשותה.
* **מושא בדיקות (test object)-** הרכיב או המערכת שצריך לבדוק. מושא בדיקות מאופיינת בבדיקת רכיבים, תוכנות, תוכנות להמרה או הגירה של נתונים ומודלים של מסדי נתונים.
* **פריט בדיקה (test item)-** האלמנט המסוים שצריך לבדוק אותו. (\*לרוב יש מושא בדיקות אחד והרבה פריטי בדיקות).
* **בודקה, מכלול מרכבי הבדיקות (testware)-**  מרכיבים אשר נוצרים תוך כדי תהליך הבדיקות ונדרשים לשם תכנון, עיצוב וביצוע בדיקות כגון: תיעוד, קלט, תוצאות צפויות ועוד.
* **תנאי בדיקה (test condition)-** פריט או אירוע של רכיב או מערכת שעשוי להיות מאומת ע"י מקרה בדיקה אחת או יותר. למשל: פונקציה.
* **נתוני בדיקות (test data)-** נתונים אשר קיימים לפני שבדיקה מתבצעת ועשויים להשפיע או להיות מושפעים מהרכיב או המערכת הנבדקים (ערכים מספריים במחשבון).
* **נוהל בדיקות (test procedure)-** רצף של פעולות לשם ביצוע בדיקות
* **סדרת בדיקות (test suite)-**  סט של מקרי בדיקה רבים עבור רכיב או מערכת שנבדקים**.**
* **כיסוי בדיקות (test coverage)-**  המידה (באחוזים) בה פריט מסוים כוסה ע"י סדרת בדיקות.
* **אמות מידה ליציאה (exit criteria)-** סט של תנאים כללים או ספציפיים, הנדרשים בכדי לאשר באופן רשמי את סיומו של הליך.וע"י כך נמנע מצב בו משימה תחשב שהושלמה בעוד ישנם חלקים משמעותיים שלה שלא הסתיימו.

יישום הערכת אמות המידה היא: השוואת תוצאות הבדיקה לעומת אמות המידה שהוגדרו בתוכניות הבדיקה 🡨 ביצוע הערכה של הצורך בבדיקות נוספות 🡨 כתיבת דו"ח סיכום בדיקה

* **ביצוע בדיקות (test execution)-** תהליך של הרצות בדיקות על הרכיב או המערכת הנבדקים המייצר תוצאה בפועל (הבדיקה עצמה)**.**
* **רישום בדיקות (test log)-** רישום כרונולוגי של פרטים רלוונטיים בנוגע לביצוע הבדיקות.
* **אירוע, תקרית (incident)-** כל אירוע המתרחש ואף דורש חקירה.
* **בדיקות חוזרות (re-testing)-** בדיקות המריצות מקרי בדיקה שנכשלו בריצתם האחרונה במטרה לאמת את הצלחתן של פעולות מתקנות שננקטו.
* **בדיקות נסיגה (regression testing)-** בדיקות של תכנית מחשב אשר נבדקה קודם לכן, בעקבות שינויים שעברה, כדי לוודא שכתוצאה מאותם שינויים פגמים לא הוכנסו או התגלו באזורי התוכנה. כמו כן, בדיקות נסיגה מתבצעות לאחר שהתוכנה או סביבתה עברו שינוי.
* **דו"ח סיכום בדיקות (test summary report)-** מסמך המסכם פעילויות בדיקה ותוצאותיהן. בין היתר כולל גם הערכה של פריטי הבדיקה אל מול אמות המידה ליציאה.
* **נעקבות traceability** - היכולת לזהות פריטים הקשורים אחד לשני במסמכים ובמערכת, כגון דרישות שונות עם בדיקות משותפות.

**תכנון הבדיקות ובקרתן:**

* **תכנון בדיקות (test planning):** הגדרת יעדי הבדיקה ומפרט המתאר את פעילויות הבדיקה הנחוצות לשם עמידה ביעדים.
* **בקרת בדיקות (test control):** זוהי פעולה מתמשכת בה משווים את ההתקדמות בפועל אל מול התוכנית, מדווחים על חריגות, מנטרים פעולות בדיקה , מבצעים צעדי תיקון .

**ניתוח הבדיקות ועיצובן:**

1. **סקירת בסיס הבדיקות לרבות:** דרישות, רמת תקינות התוכנה, מורכבות רמת אבטחה ועוד. הסקירה מוגדרת באופן המשקף, את חשיבות התוכנה לבעלי העניין בה, את רמת הסיכון ואף מכילה דוחות ניתוח סיכונים, ארכיטקטורה, עיצוב ודרישות הממשק.

**2. הערכת הבדיקוּת (testability):** של בסיס הבדיקות ושל יעדי הבדיקות

**3. זיהוי ותיעדוף תנאי הבדיקות על סמך:** ניתוח פריטי הבדיקות (test items), המפרט ומבנה התוכנה והתנהגותה

**4. עיצוב ותיעדוף מקרי בדיקה כלליים (high level test cases)**

**5. זיהוי נתוני בדיקות הנחוצים לשם קיום:** תנאי הבדיקות ומקרי הבדיקה.

**6.** **עיצוב הגדרות סביבת הבדיקה** **וזיהוי התשתיות והכלים הנדרשים**

**7.** **יצירת נעקבות דו כיוונית:** בין בסיס הבדיקות לבין מקרי הבדיקה. כלומר, שלכל דרישה ודרישה קיימת בדיקה המכסה את הדרישה המתבקשת.

**יישום הבדיקות וביצוען:**

* **אפיון הליכי הבדיקה או התסריט באמצעות:** שילוב מקרי בדיקה בסדר מסוים, הכוללת כל מידע אחר הדרוש לביצוע הבדיקה.
* **הקמת הסביבה**.
* **הרצת הבדיקות.**

שלבי יישום הבדיקות וביצוען:

1. סיום הגדרת מקרי הבדיקה, יישומם ותיעדופם (כולל זיהוי נתוני הבדיקות)
2. פיתוח ותעדוף נהלי בדיקות תוך יצירת נתוני בדיקות כשהבדיקות מיועדות לאוטומציה. באמצעות רצף של פעולות לשם ביצוע בדיקות (test procedure).
3. אימות הקמה תקינה של סביבת הבדיקה
4. אימות ועדכון נֶעֱקָבוּת דו-כיוונית בין בסיס הבדיקות ומקרי הבדיקה
5. רישום תוצאות ביצוע הבדיקות. וכן תיעוד זהויות וגרסאות של התוכנה הנבדקת, כי הבדיקה והבודקה (מכלול מרכיבי הבדיקה).
6. השוואת התוצאות בפועל לתוצאות הצפויות.
7. דיווח על חריגות בתוך אירועים וניתוחן במטרה להגדיר את הגורם לחריגה. למשל: פגם בקוד.

**פעילויות סגירת בדיקה:**

* אוספות מידע מפעילויות בדיקה שהושלמו, ומאחדות יחד את: הניסיון שנצבר, מכלול מרכיבי הבדיקה והעובדות והמספרים.
* מתבצעות בנקודת אבן הדרך של פרויקט. למשל:אלפא בטא, שחרור ללקוח.

שלבי פעילויות סגירת בדיקה:

1. בדיקה: אילו תוצרים מתוכננים אכן נמסרו.
2. סגירת דוחות אירועים (incident reports). ורישום דוחות שנשארו פתוחים לשם תיעוד, תיקון, בקשות לשינויים.
3. תיעוד קבלת המערכת (acceptance).
4. ארגון מכלול פרטי הבדיקה, סביבת הבדיקה ותשתיות הבדיקה וגניזתם לשימוש חוזר
5. הפקת לקחים, במטרה להגדיר את השינויים הדרושים בגרסאות ופרויקטים עתידיים.
6. שימוש במידע שנאסף לשיפור בשלות הבדיקות.

**הפסיכולוגיה של בדיקות מושגים:**

* **ניחוש שגיאות (error guessing )-** טכניקת עיצוב בדיקות שבה נעשה שימוש בניסיון של הבודק בכדי לצפות אלו פגמים עשויים להתקיים ברכיב או במערכת הנתונים לבדיקה. ובעצם לעצב בדיקות ספציפיות כדי לחשוף אותם.
* **עצמאות הבדיקות (independence of testing)-** הפרדת תחומי האחריות על מנת שהבדיקות יהיו אובייקטיביות.
* **דפוסי מחשב שונים למפתח ולבודק:** אופטימיות מול פסימיות , מהירות והספק אל מול איכות, שיקולים טכניים בקוד מול שיקולי נוחות ויעילות למשתמשוהתמקדות ברכיב המקודד מול מבט כללי על היישום ומטרותיו.
* **רמת אי-התלות (מהנמוכה לגבוהה):** כותב התוכנה בודק אותה 🡨 כותבי התוכנה בודקים זה את זה 🡨 צוות בדיקות או מומחה בדיקות מתוך החברה (תלוי/ בלתי תלוי) 🡨 צוות בדיקות מחוץ לחברה.

**הקוד האתי:**

* + **הציבור:** בודקי תוכנה מוסמכים יפעלו תמיד בהתאם לטובת האינטרס הציבורי
  + **לקוח ומעסיק:** בודקי תוכנה מוסמכים יפעלו לטובת הלקוח והמעסיק שלהם, ובהתאם לאינטרס הציבורי.
  + **מוצר:** בודקי תוכנה מוסמכים יבטיחו שהתוצרים שהם מייצרים ביחס למוצרים ולמערכות שהם בודקים, תואמים לאמות המידה המקצועיות הגבוהות ביותר שבאפשרותם.
  + **שיקול דעת:** בודקי תוכנה מוסמכים ישמרו בשיפוטם המקצועי על יושרה (integrity), עצמאות ואי-תלות.
  + **ניהול:** מנהלי בדיקות תוכנה וראשי צוות מוסמכים יהיו מחויבים לגישה אתית ביחס לניהול בדיקות תוכנה ויקדמו גישה אתית כזו.
  + **המקצוע:** בודקי תוכנה מוסמכים יקדמו את יושרת המקצוע והמוניטין שלו, בהתאם לאינטרס הציבורי.
  + **עמיתים:** בודקי תוכנה מוסמכים יהיו הוגנים כלפי עמיתיהם, יתמכו בהם, ויקדמו שיתוף פעולה עם מפתחי תוכנה.
  + **מחויבות עצמית:** בודקי תוכנה מוסמכים ימשיכו ללמוד את המקצוע במשך כל חייהם המקצועיים, ויקדמו גישה אתית לעיסוק במקצוע

**סילבוס פרק 2:**

**פרק 2 : בדיקות לאורך מחזור חיי התוכנה**

**מודלים לפיתוח תוכנה:**

* **אימות מול תיקוף-** קיים בלבול בין אימות(verification) מול תיקוף ((validation.

**אימות-** מוודא שעמדנו בכל הדרישות. כלומר: עשינו/בדקנו נכון.

**תיקוף ("ולידציה") -** מוודא שהדרישות עונות על הצרכים והציפיות של הלקוח. כלומר: עשינו/ בדקנו את הדבר הנכון

**אימות ותיקוף (V&V)-** מוודא שעשינו נכון את הדבר הנכון.

הערה: אימות ותיקוף מוגדרים ע"י בחינה ובדיקה, הבאת ראיות אובייקטיביות לכך שמולאו דרישות ספציפיות.

**מודל V (מודל פיתוח סדרתי):**

הגרסה המצויה של המודל בנויה על ארבע רמות בדיקה החופפות לארבעת שלבי הפיתוח: בדיקת רכיבים, בדיקות אינטגרציה, בדיקות מערכת, בדיקות קבלה.

**הגדרת המודל:** מודל V הינו מסגרת לתיאור פעילויות מחזור החיים של פיתוח תוכנה. החל מאפיון הדרישות, דרך כתיבתן בקוד, ועד לתחזוקה. המודל מדגים כיצד פעילויות בדיקה ניתנו לשילוב בתוך כל שלב של מחזור חיי פיתוח תוכנה.

**פיתוח בסבבים:**

* **פיתוח מחזורי בסבבים מצטברים-** מצב בו אנו מגדרים את הדרישות, עיצוב, בנייה ובדיקת המערכת בסדרת מחזורי פיתוח קצרים. למשל: בניית אב-טיפוס, פיתוח יישומים מהיר וכו'.
* **מודל מחזור חיים מבוסס סבבים מצטברים-** מודל שבו פרויקט מתחלק, לרוב, למספר גדול של איטרציות. שכן, מדובר בגרסה חלקית של המוצר הסופי שנמצא בפיתוח. ומאיטרציה לאיטרציה מתקרבים להשגת המטרה – המוצר הסופי עצמו.

פיתוח בסבבים מאפשר בדיקה של המערכת במספר רמות במהלך כל סבב. כאשר, כל תוספת יוצרת מערכת חלקית הולכת וגדלה, שגם אותה יש לבדוק. יש לציין כי, חשיבותן של בדיקות הנסיגה הולכת וגוברת עם כל סבב נוסף ואף ניתן לבצע אימות ותיקוף אחרי כל תוספת (סבב).

**בדיקות בתוך מודל מחזור החיים:** ייתכן שימוש במודלים אחרים. אולם, ישנם מאפיינים עליהם כדי להתבסס במטרה לבצע בדיקות בצורה טובה. ראשית, לכל פעילות פיתוח יש פעילות בדיקה מקבילה. שנית, לכל רמת בדיקה ישנם יעדים הייחודיים לרמה זו. כמו כן, עבור כל רמת בדיקה, ניתוח הבדיקות ועיצובן מתחילים במהלך פעילות הפיתוח המקבילה. ולבסוף, קיימת חשיבות רבה למעורבותם של הבודקים בסקירת המסמכים מיד עם הופעתה של הטיוטות הראשוניות.

הערה: ניתן לשלב יחד רמות בדיקה שונות או לארגן מחדש את הרמות, בהתאם לאופי הפרויקט או המערכת.

**רמות בדיקה:**

**הגדרה:** קבוצה של פעולות בדיקה אשר מאורגנות ומנוהלות יחד. רמת בדיקה, קשורה לחלוקת תחומי האחריות בפרויקט. לדוגמא: בדיקות יחידה, בדיקות אינטגרציה, בדיקות מערכת, בדיקות קבלה.

* בכל רמת בדיקה יש את המרכיבים הבאים:

1. יעדים כלליים
2. בסיס הבדיקות: המסמכים עליהם מסתמכים לשם הכנת הבדיקות ולהגדרת מקרי הבדיקה
3. מושא הבדיקה (test object) כלומר הדבר אותו בודקים
4. פגמים וכשלים אופייניים שיש לזהות (defects and failures)
5. גישות ותחומי אחריות ספציפיים.

**הערה:** יש להבדיל ולא להתבלבל בין רמות בדיקה לסוגי בדיקה. שכן, רמות בדיקה הן השלב בבדיקות (יחידה > אינטגרציה > מערכת > קבלה) סוגי בדיקה מתייחסים ליעד הבדיקה (תפקוד, ביצועים...) או לטכניקה.

**בדיקות רכיבים (component testing) :**

בדיקות רכבים נקראות גם בדיקות יחידה (unit testing) ומטרתם היא בדיקה של רכיבי תוכנה פרטניים.

* **תותב (stub)** - יישום בעל מטרה מיוחדת של רכיב תוכנה המשמש לפיתוח או בדיקת רכיב שקוראת לו או תלויה בו. ה- Stubs מחליף את הרכיב לו קוראים.
* **דרייבר (driver):**

בהקשר של בדיקות תוכנה- רכיב תוכנה או כלי בדיקות אשר מחליף רכיב שדואג לבקרה ו/או לקריאה של רכיב או מערכת. כלומר, הדרייבר קורא לרכיב, והרכיב קורא לתותב. בשימוש המוקבל הינו מנהל התקן המאפשר לתוכנית מחשב (לרוב מערכת הפעלה לתקשר עם חומרה כלשהי.

בהקשר של רתמת בדיקות: הינו כלי עזר אשר מחליף רכיב שאינו קיים וקורא לרכיב תוכנה שבבדיקה.

הסבר: דרייבר מדמה רכיב **ששולח אלינו** פלט, תותב מדמה רכיב **שאנחנו שולחים אליו** קלט.

* **רתמת הבדיקותtest harness) )-** סביבת בדיקות המורכבת מדרייברים ומתותבים הדרושים לביצוע של בדיקה

**הערה:** הדרייברים והתותבים מדמים חלקים של הסביבה הדרושים לבדיקה ושאינם זמינים.

**מהות בדיקות הרכיבים:**

* לחפש פגמים במודלי תוכנה, תוכנות, עצמים מחלקות וכד' אשר ניתנים לבדיקה באופן נפרד ומאמתים את תפקודם.
* לאפשר ביצוע בדיקות של הרכיבים בצורה מבודדת מהמערכת. בעזרת, רתמת בדיקות הכוללת תותבים ודרייברים וסימולטורים.

**בדיקות רכיבים לדוגמא:**

* התנהגות המשאבים-כגון חיפוש דליפת זיכרון.
* בדיקות חוסן (robustness)
* בדיקות מבניות (structural testing)- למשל, כיסוי החלטות
* מקרי הבדיקה נכתבים למשל, לפי: פרט של רכיב, עיצוב התוכנה, מודל הנתונים וכו' .

**פיתוח מובל בדיקות (test driven development):** קרו גם "בדיקות תחילה" גישה זו הינה גישה מחזורית ומבוססת סבבים. כאשר בכל סבב או מחזור יש: פיתוח מקרי בדיקה, בניה ואינטגרציה של קטעי קוד קצרים, ביצוע בדיקות רכיבים תוך תיקון בעיות. כל זאת, עד אשר הקוד עובר את הבדיקה בהצלחה.

**מושגים נוספים:**

* **סביבת בדיקה (test environment)-**  סביבה המכילה חומרה, מכשור, מדמים (סימולטורים), כלי תוכנה ואלמנטים תומכים נוספים הדרושים להרצות הבדיקה.
* **מכשור (instrumentation)-** הכנסת קוד נוסף לתוכנית. במטרה לאסוף מידע על התנהגות התוכנית במהלך ביצועה. למשל, כדי למדוד כיסוי קוד.
* **בדיקות חוסן (robustness testing)-** הדרגה שעד אליה רכיב או מערכת יכולים לתפקד נכון בנוכחות נתוני קלט בלתי-תקפים ("לא חוקיים") או תנאי עומס בסביבה.

**בדיקות אינטגרציה:**

**אינטגרציה (integration)-** תהליך של שילוב רכיבים או מערכות לתוך מכלולים גדולים יותר.

**בדיקות אינטגרציה (integration testing)-**  בודקות ממשקים בין רכיבים ומערכות, יחסי גומלין בין חלקיה השונים של המערכת.

דוגמאות לבדיקות אינטגרציה:

* בדיקות אינטגרציה של רכיבים- בודקות את יחסי הגומלין בין רכיבי תוכנה ומבוצעות לאחר בדיקות רכיבים.
* בדיקות אינטגרציה של המערכת- בודקות את יחסי הגומלין בין מערכות שונות, או בין חומרה לבין תוכנה. בדיקות אלה ניתנות לביצוע לאחר בדיקות המערכת.
* תהליכים עסקיים- עשוי להצריך שילוב (אינטגרציה) בין מערכות. וכאשר משלבים בין מערכות עשויים להיות להופיע כשלים שנובעים אך ורק מהאינטגרציה בעוד שכל תוכנה או מערכת בפני עצמה פועלים טוב(ללא כשלים).

**שיטות אינטגרציה:**

* **אינטגרציה "במכה אחת" (big bang)-** ככל שהיקף האינטגרציה רחב, קשה יותר לבודד פגמים. בידוד פגמים במצב זה הינו בעל סיכון מוגבר ודורש זמן יותר לאיתור תקלות לכן נבדוק "במכה אחת"
* **אינטגרציה בשלבים (incremental)- הבודקים מתרכזים באינטגרציה עצמה-** כלומר, נבדקת האינטגרציה עצמה בין שני מודלים ולא התפקוד של כל מודל בפני עצמו.
* **בדיקת מאפיינים תפקודיים ולא תפקודיים ספציפיים-** למשל, ביצועים.

**בדיקות מערכת (system testing) :**

* **בדיקות מערכת (system testing) –** תהליך של בדיקת מערכת שעברה כבר אינטגרציה כדי לאמת שהוא עונה לדרישות שפורטו.כלומר, בדיקת עבודת המערכת בכללותה.
* **בדיקות המערכת נוגעות להתנהגות המערכת או המוצר כמכלול**
* **היקף הבדיקות מוגדר במסמכים-** תכנית האב לבדיקות או תכנית הרמה

**הערה:** כאשר מבצעים בדיקות מערכת, על סביבת הבדיקה להתאים ככל הניתן לסביבה הסופית של המערכת. הבדיקות יכולת להתבסס על: סיכונים, מפרט דרישות, תהליכים עסקיים, מקרי שימוש, תיאורים מפורטים אחרים, אינטראקציות עם מערכת ההפעלה ומשאבי המערכת.

* על בדיקות מערכת לבחון דרישות תפקודיות ולא תפקודיות של המערכת וכן גם מאפיינים של איכות הנתונים
* על הבודקים להתמודד עם מצבים בהם תיעוד הדרישות מהמערכת הוא חלקי או לא קיים.
* בתור התחלה, בדיקות מערכת של דרישות תפקודיות עושות שימוש בטכניקות מבוססות- המפרט (קופסא שחורה) המתאימות ביותר להיבט המערכתי הנבדק. למשל, יצירת טבלת החלטה עבור צירופים של תופעות המתוארות בכללים העסקיים.
* לאחר מכן, אפשר להפעיל טכניקות מבוססות מבנה (קופסא לבנה). על מנת להעריך את מידת היסודיות של הבדיקות ביחס לאלמנט מבני. כגון: מבנה תפריט או ניווט בדף אינטרנט.
* במקרים רבים צוות בדיקות עצמאי הוא המבצע בדיקות מערכת.

**בסיס הבדיקות ומושאי בדיקה לדוגמא:**

* **בסיס הבדיקות (מסמכים)-** כוללים בתוכם ,בין היתר, את מפרט הדרישות של המערכת ושל התוכנה, את המפרט התפקודי ודוחות ניתוח סיכונים
* **מושאי בדיקה –** מאופיינים בבדיקת המערכת עצמה, בתיעוד הנלווה למערכת למשל, מדרכי המערכת, המשתמש וההפעלה ובתצורת המערכת ונתוני תצורה.

**בדיקות קבלה (acceptance testing):**

* **בדיקות קבלה (acceptance testing)-** בדיקות פורמליות בדגש על צרכי המשתמש, הדרישות והתהליכים הרשמיים. המתבצעות כדי לקבוע אם מערכת עונה או לא לאמות המידה לקבלה. ובכדי לאפשר למשתמש, ללקוח או לישות מוסמכת לקבוע אם לקבל את המערכת או לא

הערה: במקרים רבים בדיקות קבלה הן באחריות הלקוח המשתמשים המערכת. אם כי בעלי עניין אחרים עשויים להיות מעורבים אף הם.

**מטרות בדיקות הקבלה:**

* לבסס בטחון במערכת, בחלקים ממנה או במאפיינים לא תפקודיים מסוימים שלה.
* מציאת פגמים איננה עומדת במוקד בדיקות הקבלה.
* בדיקות קבלה יכול להיות כלי להערכת מוכנות המערכת לפריסה בשטח ולשימוש. אך אינן בהכרח מהוות את רמת הבדיקות הסופית.

**מועד בדיקות הקבלה:** בדיקות הקבלה עשויות להתבצע בזמנים שונים במחזור החיים.

**בדיקות קבלה ע"י המשתמש:**  בד"כ מאמתות את כשירות המערכת לשימוש ידי משתמש עסקי. בדיקה זו עשויה להתבצע ע"י: בודקי ספק התוכנה והלקוח. שניהם יחד/כל אחד לחוד/ בנוכחות אחד מהשניים.

**בדיקות (קבלה) תפעוליות:** קבלת המערכת ע"י מנהלי המערכת. כוללות, בין היתר, בדיקת גיבוי ושחזור, התאוששות מאסון, ניהול משתמשים, משימות תחזוקה, עומס נתונים ומשימות הגירה ובדיקות תקופתיות לפרצות אבטחה.

**בדיקות קבלה חוזיות ורגולטוריות:** בדיקות קבלה חוזיות מבוצעות לפי הקריטריונים לקבלה המופיעים בחוזה לייצור תוכנה מותאמת ללקוח. ובהתאם לרגולציה.

**בדיקות אלפא ובדיקות בטא (המכונות גם: בדיקות שטח):**

לעיתים קרובות, צוות הפיתוח של מוצרי מדף מעוניין לקבל משוב מלקוחות קיימים או פוטנציאליים לפני שמוצר התוכנה מופץ באופן מסחרי. לכן, הבדיקה מתחלקת לשתיים. בדיקות אלפא המתבצעות באתר הפיתוח, אך לא ע"י צוות הפיתוח. ובדיקות בטא, או בדיקות שדה, המתבצעות ע"י לקוחות או לקוחות פוטנציאליים במקומות בהם הלקוחות נמצאים. ארגונים עושים במונחים בדיקות במפעל ובדיקות באתר.

**בדיקות בעזרת מודלים של תוכנה:**

* **בבדיקות מבניות-** נשתמש במודל זרימת הבקרה, או במודל מבנה התפריט.
* **בבדיקות תפקודיות-** נשתמש במודל זרימת תהליך, מודל החלף מצבים או במפרט הכתוב בשפה פשוטה.
* **בבדיקות לא-תפקודיות -**  נשתמש במודל ביצועים, מודל שימושיות ומודל איומי אבטחה.

**בדיקות תפקודיות(פונקציונליות):**

תחילה נחזור על ההגדרה: בדיקות המבוססות על ניתוח מפרט הפונקציונליות של רכיב או מערכת. למשל באמצעות בדיקות קופסה שחורה.

הפעולות שעל מערכת, או תת-מערכת או רכיב לבצע, מתוארות בד"כ בתוצרים כגון מפרט דרישות, מקרי שימוש או מפרט תפקודי. אם כי לעיתים תיעוד כזה אינו קיים ולכן במקרה כזה הפעולות הינן "מה שהמערכת עושה"

**השימוש בבדיקות קופסה שחורה:**  בדיקות מבוססות מפרטים בכדי להפיק תנאי בדיקה ומקרי בדיקה מתפקודיות התוכנה או המערכת. בפועל, מדובר בבדיקות תפקודיות המעריכות את ההתנהגות החיצונית של התוכנה( לדוגמא: בטלוויזיה בדיקה שהלחצנים עובדים)

**הגדרות:**

* **בדיקות קופסא שחורה –** בדיקות בין שהן פונקציונליות או לא פונקציונליות, המתוכננות ומתבצעות ללא התייחסות למבנה הפנימי של רכיב או של מערכת.
* **בדיקות אבטחה-**  בדיקות אבטחה, שהן סוג של בדיקות תפקודיות, חוקרות תפקודים הנוגעים לגילו איומים, כמו למשל וירוסים שמקורם בגורמים חיצונים (למשל Firewall).
* **בדיקות ליכולת פעולה-הדדית-** מעריכות את יכול המוצר לשתף פעולה עם רכיב אחד או יותר עם מערכת אחת או יותר.

**בדיקות לא תפקודיות (לא פונקציונליות):**

**מושגים:**

* **בדיקות ביצועים (performance testing)-** תהליך בדיקות שנועד לקבוע את הביצועים של מוצר/ תוכנה (ראה גם בדיקות יעלות)
* **בדיקות עומס (load testing)-** סוג של בדיקות ביצועים, המבוצעת בכדי להעריך את ההתנהגות של רכיב או מערכת עם עומס הולך וגובר. למשל מספר המשתמשים בו זמנית. בדיקה זו נעשית כדי לקבוע איזה עומס יכול להיות מטופל בידי הרכיב או המערכת (ראה גם בדיקות מאמץ)
* **בדיקות מאמץ (stress testing)-** סוג של בדיקות ביצועים. המתבצעות כדי להעריך את התנהגות מערכת או רכיב כאשר העומסים: מגיעים לרמה המרבית.
* **בדיקות שימושיות (usability testing)-** בדיקות כדי לקבוע את המידה שבא מוצר/תוכנה הינו: מובן, קל ללמידה, קל להפעלה, אטרקטיבי, עומד בתנאים שפורטו.
* **בדיקות אמינות (reliability testing)-**  תהליך הבדיקות שנועד לקבוע את מידת האמינות של מוצר/תוכנה (ראה גם אמינות)**.**
* **בדיקות תחזוקתיות (maintainability testing)-** תהליך בדיקות שנועד לקבוע את מידת התחזוקתיות של מוצר/תוכנה
* **בדיקות ניידות (portability testing)-** תהליך הבדיקות שנועד לקבוע את מידת הניידות של מוצר תוכנה.

**הערה:** בדיקות לא פונקציונליות מעריכות ההתנהגות החיצונית של התוכנה וברוב המקרים עושות שימוש בטכניקות קופסא שחורה לעיצוב הבדיקות.

**לסיכום:**

* בדיקות לא תפקודיות אינן מוגבלות לסוגי הבדיקות שסקרנו- כל בדיקה לא-תפקודית נופלת במסגרתן.
* אפשר לבצע בדיקות לא –תפקודיות בכל רמות הבדיקה.
* המונח בדיקות לא-תפקודיות מתאר בדיקות הדרושות לצורך מדידת מאפיינים כמותיים רציפים של המערכת והתוכנה- כמו למשל זמני תגובה בבדיקות ביצועים.
* בדיקות אלה ניתנות לייחוס מול מודל איכות -כמו זה המוגדר במסגרת התקן "הנדסת תוכנה - איכות מוצר תוכנה"

**בדיקת ארכיטקטורה של התוכנה (בדיקות מבניות):**

**מושגים:**

* **בדיקות קופסה לבנה (white‑box testing)-** בדיקות המבוססת על ניתוח המבנה הפנימי של רכיב או מערכת. שמות נוספים הם בדיקות קופסא שקופה (clear-box testing)או בדיקות מבניות (structural testing)- כלומר בדיקות המבנה/הארכיטקטורה של התוכנה.
* **כיסוי קוד (code coverage)-** שיטת ניתוח אשר קובעת איזה חלקים מהתוכנה בוצעו בפועל (כוסו) על ידי סדרת בדיקות ואיזה לא בוצעו דוגמאות: כיסוי משפטים, כיסוי החלטות, כיסוי תנאים

**כיסוי קוד (code coverage)-** הכיסוי הוא המידה בה פריט מסוים (כגון: שורות קוד) נבדק בפועל ע"י סדרת בדיקות .

הערה: כיסוי קוד מוצג ע"י אחוז הפריטים שכוסו מתוך סך פריטים מסוג זה הקיימים בקוד הנבדק. כמו כן, במידה והכיסוי הינו פחות מ- 100%, יתכן צורך לתכנן בדיקות נוספות לשם הגדלת הכיסוי.

**עוד על כיסוי ובדיקות מבניות:**

* בכל רמות הבדיקה אפשר להשתמש בכלים כדי למדוד את כיסוי הקוד של אלמנטים, כמו פקודות או החלטות. בבדיקות רכיבים ואינטגרציה של רכיבים, אבל גם בבדיקות מערכת.
* אפשר לבסס בדיקות מבניות על ארכיטקטורת המערכת.
* גישות בדיקה מבניות מתאימות ליישום ברמות בדיקת המערכת, האינטגרציה של המערכת או הקבלה.

**בדיקות עקב שינוים: בדיקות חוזרות ובדיקות נסיגה:**

* **בדיקות חוזרות (re-testing)-** בדיקות המריצות מקרי בדיקה שנכשלו בריצתם האחרונה במטרה לאמת את הצלחתן של פעולות מתקנות שננקט. מיותר לציין, כי לאחר זיהוי פגם ותיקונו, יש לבדוק שוב את התוכנה על מנת לוודא שהפגם המקורי אכן הוסר בהצלחה. תהליך זה מכונה אישור (confirmation).

לעומת זאת, דיבאגינג הינו פעילות פיתוח ולא פעילות בדיקה.

* **בדיקות נסיגה (regression testing)-** בדיקות של תכנית מחשב אשר כבר נבדקה קודם לכן כדי לוודא שהתוצאה מאותם שינוים פגמים לא הוכנסו או התגלו באזורי התוכנה שלא השתנו. בדיקה זה מתבצעת לאחר שהתוכנה או סביבתה עברו שינוי.

**פגמים שנגרמו או נחשפו כתוצאה מן השינוי-** עלולים להימצא בתוכנה הנבדקת או ברכיב תוכנה אחר שקשור או שאינו קשור בתוכנה המפותחת.

**היקף הבדיקות נקבע על-פי הסיכוי להופעת פגמים חדשים עקב השינוי-**  ומתבצע כך שככל שקיים סיכוי גבוהה למציאת פגמים חדשים. כך נשקיע יותר מאמץ בבדיקות נסיגה.

**על הבדיקות להיות כאלה שניתן לחזור עליהן בדייקנות-** אם רוצים לעשות בהן שימוש חוזר לבדיקות אישור ובדיקות נסיגה. כמו כן , **ניתן לבצע בדיקות נסיגה בכל רמות הבדיקה** והן כוללות בדיקות תפקודיות, לא תפקודיות ומבניות.

**סדרות של בדיקות נסיגה מופעלות מספר רב של פעמים** ובד"כ עוברות התפתחות הדרגתית ומסיבה זו בדיקות נסיגה הן מועמדות מתאימות מאוד לאוטומציה.

**בדיקות תחזוקה:**

במקרים רבים מערכת תוכנה משמשת במשך שנים ואפילו עשורים. בזמן זה המערכת, נתוני התצורה שלה או סביבתה עוברים שינויים, תיקונים או הרחבות. לכן, תכנון מראש של גרסאות הוא חיוני לקיום בדיקות תחזוקה מוצלחות (יש להבחין בן גרסאות מתוכננות לבין תיקוני חירום). בדיקות תחזוקה מתבצעות על-גבי מערכת מתפקדת קיימת. כאשר, יוזמים אותן בעקבות שינויים, הגירה או סיום חיי התוכנה או המערכת.

**סוגי שינוים (תחזוקתיים):** שיפורים מתוכננים(למשל מבוססי גרסה), תיקונים או עדכוני חירום, שינוים בסביבה (שדרוג מערכת ההפעלה או בסיס הנתונים) שדרוג מתוכנן של מוצר-מדף מסחרי, טלאים (patch) לתיקון פרצות ותקלות שהתגלו. יש לציין, כי תיקון חירום הינו דחוף ומידי בעוד שטלאי הינו תיקון שבא לפתור בעיות שאינן קריטיות.

**עוד על בדיקות תחזוקה:** בנוסף לבדיקת השינויים, בדיקות תחזוקה כוללת בדיקות נסיגה של חלקי המערכת שנותרו ללא שינוי. כאמור, בדיקות התחזוקה תלויות בסיכון, בגודל המערכת ובהיקפה. כמו כן, בהתאם לשינויים ניתן לבצע בדיקות תחזוקה בכל רמות הבדיקה ועבור כל סוגי הבדיקות. בחינת ההשפעה של שינוים מהווה כלי המסייע בקבלת החלטה על היקף בדיקות הנסיגה הדרושות ואף לשמש בבחירת סדרת בדיקות הנסיגה.

**קשיים בבדיקות תחזוקה:** הקושי המרכזי הינו שמפתחי ובודקי המערכת כבר התחלפו. קושי נוסף, הוא היעדר בודקים בעלי ידע או מפרטים בלתי מעודכנים או חסרים.

**מושגים מהמילון:**

* **בדיקות תחזוקה-** בדיקת שינויים שנעשו במערכת מבצעית או בדיקת ההשפעה של שינוי בסביבת הפעולה של מערכת כזו.
* **ניתוח השפעה-** הערכה של השפעת שינוי על: שכבות של תיעוד הפיתוח, תיעוד ורכיבים בבדיקות. כל זאת, במטרה ליישם את אותו שינוי ביחס לדרישות שפורטו.

**הגירה (migration):**

* **בדיקות הגירה (migration testing), המרה (conversion)-** בדיקות של תוכנה המשמשת להמרת והעברת נתונים ממערכות קיימות לשם שימוש במערכות המחליפות אותן.
* **במסגרת בדיקות תחזוקה לפני הגירה יש לכלול בדיקות תפעוליות**  של הסביבה החדשה כמו גם לתוכנה המעדכנת.
* בדיקות הגירה (בדיקות הסבה) נחוצות גם במקרים בהם נתונים מיישום אחר יועברו אל המערכת המתחזקת.

**סיום חיי מערכת (end-of-life):**  בדיקות תחזוקה לסיום חיי מערכת עשויות לכלול; בדיקות להגירת נתונים או להעברתם לארכיון- במקרים בהם יש צורך לאחסנם לתקופה ממושכת.

**סילבוס פרק 3:**

**שיטות סטטיות:**

**שיטות סטטיות ותהליך הבדיקה:**

* **בדיקות דינמיות (dynamic testing)**- בדיקות הכרוכות בביצוע (הרצה) של התוכנה של רכיב או מערכת.
* **בדיקות סטטיות (static testing)-** בדיקות של מרכיב בפיתוח תוכנה, כגון: דרישות, עיצוב, קוד. באופן סטטי ובלי ביצוע בפועל; למשל: סקירות או ניתוח סטטי של קוד.

מסתמכות על; בחינה ידנית וניתוח אוטומטי (ניתוח סטטי) של מסמכים או של קוד (ללא הרצת הקוד!)

**מחיר הפגמים:**

* **ניתן לבדוק את תוצרי העבודה** כולל הקוד באמצעות סקירות אותם אפשר לקיים זמן רב לפני ביצוע הבדיקות הדינמיות.
* **מחיר הסרתם של פגמים שזוהו המהלך סקירות בשלבים המוקדמים במחזור החיים** (למשל, פגמים שנמצאו ברשימת הדרישות) נמוך בהרבה מזה של פגמים שיזוהו ע"י הרצת בדיקות על הקוד המתבצע.

**מה בודקים בסקירות ידניות?**

הפעילות הידנית העיקרית היא בחינת תוצר העבודה וכתיבת הערות. ניתן לבדוק, בין היתר: מפרט הדרישות, מפרט העיצוב, קוד, תכניות הבדיקה ומפרטי הבדיקה, מקרי הבדיקה ותסריטי הבדיקה, המדריכים למשתמש ודפי אינטרנט וכל מסמך הנוגע למוצר.

**יתרונות הסקירה:**

**בין יתרונות הסקירה נכללים:** גילוי ותיקון מוקדם של פגמים,שיפור פריון העבודה (פרודוקטיביות) של המעורבים בתהליך הפיתוח**,** צמצום משך הפיתוח**,** צמצום עלויות הבדיקה בכסף ובזמן**,** צמצום עלויות לאורך כל חיי המדף של התוכנה**,** הפחתת מספר הפגמים**,** שיפור בתקשורת בין חברי הצוות. בכוחן של סקירות לגלות השמטות, כמו למשל בדרישות, אשר הסיכוי למוצא אותן במהלך בדיקות דינמיות הוא קלוש.

**סיכום המבוא לבדיקות סטטיות בתהליך הבדיקה:**

* מטרה משותפת לסקירות, ניתוח סטטי ובדיקות דינמיות היא זיהוי פגמים. שכן, השיטות הן משלימות כאשר כל אחת מהן מותאמת לזיהוי יעיל של פגמים מסוגים שונים.
* בהשוואה לבדיקות דינמיות, השיטות הסטטיות מזהות את הפגמים הגורמים לכשלים, ולא רק את הכשלים עצמם.
* בין הפגמים הניתנים לזיהוי ביתר קלות בסקירות מאשר בבדיקות דינמיות ניתן למצוא: סטיות מהגדרות של תקנים, פגמים בדרישות, פגמים בעיצוב ותחזוקתיות (maintainability) בלתי מספקת, מפרטי ממשק שגויים.

**תהליך הסקירה:**

* **קשת הסקירות מגוונת ונעה בין:**
  + **הבלתי רשמיות:** מאופיינת בעדר הוראות כתובות
  + **שיטתיות ורשמיות יותר:** אותן מאפיינים; השתתפות הצוות, תיעוד תוצאות הסקירה ונהלים מתועדים לביצוע הסקירה.

**שיקולים בבחירת סוג הסקירה:**

* **מידת הרשמיות של תהליך הסקירה תלויה בגורמים כמו:** בשלות תהליך הפיתוח, עמידה בדרישות חוקיות או רגולטוריות ו/או צורך בהצגת נתיב ביקורת.
* **האופן בו מבצעים סקירה תלוי ביעדים המוסכמים של הסקירה:** למשל, מציאת פגמים, הבנת המערכת, למידה ע"י בודקים וחברי צוות חדשים ודיון וקבלת החלטות בקונצנזוס.

**מושגים בסקירה רשמית:**

* **סקירה רשמית (formal review)-**  סקירה המאופיינת בנהלים ודרישות מתועדים (ביקורת).
* **ביקורת (inspection)-** סוג של סקירת עמיתים, הנשען על בחינה חזותית של מסמכים כדי לגלות פגמים

**הערה:** ביקורת הינה טכניקת הסקירה הרשמית ביותר ולפיכך מתבססת תמיד על נוהל כתוב ומתועד.

**אמות מידה לכניסה (entry criteria):** סט של תנאים כללים ו/או ספציפיים, אשר נדרשים כדי להרשות לתהליך להתקדם עם מטרה מוגדרת (למשל, שלב הבדיקות). מטרתה של אמות המידה לכניסה היא למנוע ממשימה להתחיל כשאיננו מוכנים אליה.

**אמות מידה ליציאה (exit criteria):**  סט של תנאים כללים ו/או ספציפיים, המוסכמים עם בעלי העניין, אשר נדרשים כדי לאשר רשמית את סיומו של תהליך. מטרתה של אמות המידה ליציאה היא למנוע מצב בו משימה תחשב כהושלמה, בעוד יש חלקים משמעותיים שלה שלא הסתיימו. ובנוסף, בכדי לתכן מתי לעצור את הבדיקות.

**פעילויות במסגרת סקירה רשמית:**

1. **תכנון:** הגדרת אמות המידה של הסקירה 🡨 בחירת הצוות 🡨 חלוקת תפקידים 🡨 הגדרת אמות המידה לכניסה ויציאה (עבור סקירות רשמיות יותר כמו ביקורת) 🡨 בחירת המסמכים או חלקי המסמכים שייסקרו 🡨 בדיקת אמות המידה לכניסה
2. **התנעה (kick-off):** הפצת מסמכים 🡨 הסברה למשתתפים בעניין היעדים(התהליך, המסמכים)
3. **הכנה אישית:** הכנה לישיבת הסקירה ע"י סקירת המסמכים 🡨 הכנת רשימה של פגמים פוטנציאליים (שאלות והערות).
4. **בחינה/ הערכה/ רישום של תוצאות הישיבה:** דיון כולל או רישום של תוצאות מתועדות או פרוטוקול 🡨 רישום של פגמים (המלצות לטיפול והחלטות לגבי הפגמים) 🡨 בחינה, הערכה ורישום של בעיות העולות במהלך הפגישות
5. **תיקונים:** תיקון פגמים שנמצאו 🡨 בסקירות רשמיות (רישום הסטטוס המעודכן של הפגמים)
6. **מעקב:** ווידוא שהפגמים טופלו 🡨 איסוף מדדים (מטריקות) 🡨 בסקירות רשמיות: בדיקת אמות המידה ליציאה.

**מדדים (מטריקות):**

**מדד (metric)-** כולל בתוכו את;סוג המדידה (מה מודדים) , השיטה הננקטת לשם ביצוע המדידות וקנה(scale) המידה של מדידות.

**דוגמאות למדדים:** רישום באגים, מידת הביצוע של מקרי בדיקה, מידת ההצלחה של מקרי בדיקה, כלל מקרי הבדיקה לפי תוצאותיהם: טרם בוצע, עבר בהצלחה, נכשל וכו'.

**תפקידים ותחומי אחריות:**

* **מנהל:** מחליט על ביצוע סקירות, מקצה זמן בלו"ז הפרויקט ובודק שיעדי הסקירה הושגו.
* **מתאם (moderator):** האדם המוביל את הסקירה של המסמך או סט המסמכים, כולל: תכנון הסקירה, ניהול הישיבה, ומעקב לאחר הישיבה.
* **מחבר (author):** הכותב או האדם הנושא באחריות העיקרית למסמכים הנסקרים.
* **סוקרים:** אנשי צוות בעלי רקע טכני או עסקי מתאים (מכונים גם בודקים או מבקרים) אשר לאחר ההכנה הדרושה, מזהים ומתארים ממצאים (למשל, פגמים) במוצר הנסקר. יש לבחור את הסוקרים באופן שייצגו נקודות מבט ותפקידים שונים בתהליך הסקירה, ועליהם להשתתף בישיבות הסקירה.
* **רשם, כתבן (scribe, recorder):** מתעד את כל הבעיות והנקודות הפתוחות שהועלו במהלך הישיבה.

**רשימות-ביקורת (checklists) ונקודות מבט שונות:** האפקטיביות של סקירות ויעילותן משתפרות מתוך התבוננות במוצרי תוכנה או תוצרי עבור נלווים מנקודות מבט שונות (שימוש ברשימת ביקורת). שכן, גיוון של בעלי תפקיד עשוי לסייע לחשיפת בעיות שלא התגלו עד כה.

**סוגי סקירות:**

מוצר תוכנה או תוצר עבודה נלווה (מסמך) עשוי להיות נתון ליותר מסקירה אחת. יש לציין כי ניתן לעשות שימוש ביותר מסוג סקירה אחת. כאשר סדר ביצוע הסקירות אינו מוחלט אלא נתון לשינויים.

**סקירה לא רשמית:**

בסקירה לא רשמית אין תהליך רשמי. כלומר, היא יכולה להתבצע ע"י מוביל טכני הסוקר את העיצוב והקוד או במסגרת תכנות בזוגות ולכן מידת התועלת תלויה בסוקרים. כמו כן, יתכן תיעוד של הממצאים. מטרתה העיקרית של סקירה זו היא השגת תועלת מסוימת בעלות נמוכה (זמן, משאבים).

**דיון מודרך:**

בדיון מודרך הפגישה מתבצעת בהנחיית המחבר. למשל, לשם תיאור תרחישים או הרצת הקוד "על יבש". המפגשים הינם ללא סיום מוגדר ואינם מחייבים מסקנות וסיכומים. כמו כן, אפשר (אך לא חובה): להכין את הסוקרים טרם הישיבה, להכין דוח סקירה ואף למנות רשם (שאיננו המחבר). אופיו של דיון מודרך יכול לנוע בין בלתי-רשמי למדי לבין רשמי מאד. מטרתו העיקרית של דיון מודרך היא למידה, הבנה ומציאת פגמים.

**סקירה טכנית:** הינו ,תהליך גילוי פגמים מוגדר ומתועד הכולל עמיתים ומומחים טכניים. סקירה טכנית יכולה להתבצע עם או בלי השתתפות ההנהלה. בנוסף, רצוי שהסקירה תתבצע בהנחיית מתאם שאינו המחבר. ואף תכלול דוח סקירה הכולל: רשימת ממצאים, הכרעה בשאלה אם התוכנה עומדת בדרישות והמלצות בנוגע לממצאים. בפועל אופי הסקירה יכול לנוע בין בלתי-רשמי למדי לבין רשמי מאוד. מטרותיה העיקריות של הסקירה: דיון, הערכת חלופת, מציאת פגמים, פתרון בעיות טכניות, קבלת החלטות ועוד.

**ביקורת:**

**הגדרה:** סוג של סקירת עמיתים הנשען על בחינה חזותית של מסמכים כדי לגלות פגמים. כגון: הפרת תקני פיתוח ואי התאמה לרמות תיעוד גבוהות יותר. הביקורת מסתמכת על טכניקת הסקירה הרשמית ביותר ולפיכך מבוססת תמיד על נוהל כתוב ומתועד. כמו כן, הביקורת היא בהנחיית מתאם ולרוב מתבצעת כבחינת עמיתים, כל אחד לפי תפקידו ואף כוללת איסוף מדדים.

הביקורת הינו תהליך רשמי המבוסס על כללים ורשימות- ביקורת. כוללת בתוכה את אמות המידה לכניסה ויציאה לצורך אישור מוצר התוכנה. היות ותהליך זה הוא רשמי יש צורך בכתיבת דוח ביקורת הכולל ממצאים, מעקב רשמי ואף אפשרות למינוי קורא (reader). מטרה עיקרית: מציאת פגמים.

**סקירת עמיתים:** אפשר לבצע דיון מודרך , סקירות טכניות וביקורת בתוך קבוצת עמיתים (בדרג דומה בארגון)

**גורמים התורמים להצלחת סקירות:**

* קיום יעדים ברורים ומגדרים מראש ומעורבות של אנשים המתאמים ליעדי הסקירה שהוגדרו. כמו למשל הבודקים שנתפשים כמי שמביאים ערך מוסף להצלחת הסקירה.
* הפגמים המתגלים מתוארים בצורה עניינית ומתקבלים בברכה.
* הסקירה מתבצעת באווירה של אמון.
* טכניקות הסקירה מותאמות להשגת היעדים וכן, לסוג ולרמה של תוצרי העבודה, וליכולת הסוקרים
* נעשה שימוש לפי הצורך ברשימות בקרה או בחלוקת תפקידים.
* מתבצעת הדרכה בשיטות הסקירה הנתונות במיוחד במקרה של סקירה רשמית (ביקורת).
* ההנהלה תומכת בתהליך סקירה תקין (הקצאת זמן בלו"ז לפעילויות הסקירה).
* קיים דגש על למידה ועל שיפור תהליכים

**ניתוח סטטי באמצעות כלים:**

מטרת הניתוח הסטטי הוא לאתר פגמים בקוד מקור ובמודלים של תוכנה. שכן, הניתוח הסטטי מתבצע ללא הרצה בפועל של התוכנה שהכלי בודק. ניתוח סטטי מסוגל לגלות פגמים שקשה לחשוף בבדיקות דינמיות. ובדומה לסקירות חושף פגמים ולא כשלים. כלי הניתוח הסטטי מנתחים את קוד התוכנית למשל, באמצעות, זרימת בקרה וזרימת נתונים.

**מושגים בניתוח סטטי:**

* **ניתוח סטטי (static analysis)-** ניתוח של תוצרים (כתובים) הקשורים בפיתוח תוכנההנעשים ללא ביצוע או הרצה שלהם. למשל, דרישות או קוד. כמו כן, ניתוח סטטי נעשה לרוב בעזרת כלי תומך.
* **זרימת בקרה (control flow)-**  רצף של אירועים או נתיבים בריצה של רכיב או מערכת.
* **סיבוכיות (complexity)-**  הדרגה שבה עיצוב ו/או מבנה פנימי של רכיב או מערכת קשים: להבנה, לתחזוקה ולאימות
* **קומפיילר, מהדר (compiler)-**  כלי תוכנה אשר מתרגם לשפת מכונה תכניות שנכתבו בשפות מסדר גבוה.
* **מורכבות ציקלומטית (cyclomatic complexity)-** המספר המרבי של רכיבי התוכנה. הכוללים בתוכם נתיבים (paths) לינאריים ועצמאיים בתוכנה מסוימת. מורכבות ציקלומטית ניתנת לחישוב בנוסחה: L – N + 2P כאשר,

L = מספר הקצוות/קישורים בגרף (הקווים)

N = מספר הצמתים (nodes) בגרף (הנקודות)

P = מספר החלקים המנותקים בגרף

**התועלת בניתוח סטטי:**

* גילוי מוקדם של פגמים עוד בטרם ביצוע בדיקות.
* התראה מוקדמת על היבטים חשודים בקוד או בעיצוב- אשר מתקבלת על בסיס חישוב מדדים. למשל, חישוב סיבוכיות.
* איתור פגמים (ולא כשלים!) שקשה למצוא באמצעות בדיקות דינמיות.
* גילוי תלויות וחוסר עקביות במודלים של התוכנה כמו למשל קישורים (link).
* שיפור תחזוקתיות הקוד והעיצוב (design).
* מניעת פגמים (אם מחלקת הפיתוח מיישמת לקחים שנלמדו מהסקירה).

**פגמים אופייניים הנחשפים ע"י כלים לניתוח סטטי:** שימוש במשתנה בעל ערך שאינו מוגדר, ממשקים לא אחידים בין מודולים (modules) ורכיבים (components), משתנים שאינם בשימוש או הצהרה לא תקינה על משתנים, קוד מת (dead code), לוגיקה חסרה או שגויה (פוטנציאל ללולאות אינסופיות), מבנים מורכבים יתר על המידה, סטיות מנהלי התכנות התקני, פרצות אבטחה, ושגיאות תחביר (syntax) בקוד ובמודלים**.**

**שימוש בכלים בידי מפתחים ומעצבים:**

* כלים לניתוח סטטי בדרך כלל משמשים את המפתחים
* מעצבים (designers) עושים שימוש בכלים אלה בעת יצירת מודלים.
* כלי ניתוח סטטי עשויים לייצר כמות בלתי מבוטלת של התראות -אותן יש לנהל באופן נאות על מנת להפיק את המרב מן השימוש בכלים.
* ישנם מהדרים (compilers) התומכים באופן חלקי בניתוח סטטי, כולל גם חישוב מדדים.

**סילבוס פרק 4:**

**טכניקות לעיצוב בדיקות:**

**תהליך פיתוח הבדיקות**:

* **תהליך פיתוח הבדיקות המתואר בפרק זה ניתן לביצוע באופנים שונים**, אשר נעים בין תהליכים בלתי-רשמיים לגמרי הכוללים מעט תיעוד אם בכלל לבין תהליכים רשמיים ביותר, כמתואר בהמשך.
* **דרגת הרשמיות תלויה בהקשר של הבדיקות** כגון: בשלות תהליכי הבדיקות והפיתוח, אילוצי זמן, דרישות אבטחה ורגולציה והאנשים המעורבים.

**תנאי הבדיקות (test conditions):** במהלך ניתוח הבדיקות, מנתחים את מסמכי בסיס הבדיקות על מנת להחליט מה יש לבדוק. כלומר, מזהים את תנאי הבדיקות. תנאי בדיקה מוגדר בתור פריט או אירוע שניתן לאמת באמצעות מקרה בדיקה אחד או יותר. למשל: פונקציה, תנועה, תוכנה מאפיין, אלמנט מבני וכו' .

**נֶעֱקָבוּת (traceability) וגישת בדיקות (test approach)**:

**עקיבות traceability** - היכולת לזהות פריטים הקשורים אחד לשני במסמכים ובמערכת, כגון דרישות שונות עם בדיקות משותפות.

* יצירת נֶעֱקָבוּת מתנאי הבדיקה חזרה אל המפרטים והדרישות – כלומר, יצירת מצב בו ניתן לזהות פריט או אירוע במסמכים ובמערכת. מצב זה מאפשר ניתוח השפעה אפקטיבי כאשר הדרישות משתנות ואף קביעת רמת כיסוי הדרישות (המושגות ע"י סדרה של בדיקות).
* בעת ניתוח בדיקות מיישמים את פרטי גישת הבדיקות- ע"י בחירת טכניקות עיצוב בהן נשתמש.
* הבחירה מתבססת, בין השאר, על הסיכונים שזוהו.

**הערה:** מקרי הבדיקה (אוסף של נקודות שלפיו אנו מבצעים את הבדיקה) ונתוני הבדיקה (נתונים אשר קיימים לפי שהבדיקה מתבצעת ) מיוצרים ומאופיינים במהלך עיצוב הבדיקות.

**התוצאות הצפויות (expected results):**

יש להגדיר, כחלק ממפרט מקרה הבדיקה, את התוצאות הצפויות. התוצאות צריכות לכלול: פלטים, שינוים בנתונים ובמצבים וכל השלכות אחרות של הבדיקה. שכן, אם התוצאות הצפויות לא הוגדרו, יתכן מצב בו תוצאה שגויה, תתפרש כתוצאה תקינה. או להפך תוצאה תקינה תתפרש כשגויה. לכן, רצוי מאד להגיד את התוצאות הצפויות לפני ביצוע הבדיקה. למשל, אם נחשב 3+2 התוצאה הצפויה הינה 5 ויש להגדירה מראש!

**נוהל בדיקות (test procedure)**

תזכורת, יישום הבדיקות- שילוב מקרי בדיקה בסדר מסוים, הכוללת כל מידע אחר הדרוש לביצוע הבדיקה.

במהלך יישום הבדיקות מקרי הבדיקה עוברים: פיתוח, יישום, תעדוף ואיגוד למפרט נוהל בדיקות. נוהל הבדיקות, מגדיר את רצף הפעולות לביצוע הבדיקה. יש לציין כי, כאשר מריצים בדיקות באמצעות כלי לביצוע בדיקות, רצף הפעולות מפורט בתסריט בדיקות שהוא נוהל בדיקות אוטומטי. במילים אחרות, נוהל בדיקה מוגדר בשלב יישום הבדיקות ומגדיר את סדר הפעולות של הבדיקות.

**לוח הזמנים לביצוע הבדיקות**:

בהמשך, מתוך נהלי הבדיקות השונים ותסריטי הבדיקות האוטומטיות, יוצרים את לוח הזמנים לביצוע הבדיקות, אשר מגדיר את הסדר לפיו יבוצעו נהלי הבדיקות. כמו כן, הלו"ז לביצוע לוקח בחשבון גורמים כגון: בדיקות נסיגה, תלות טכנית, ולוגיות.

**סוגי טכניקות לעיצוב בדיקות**: המטרה של טכניקה לעיצוב בדיקות היא לקבוע את; תנאי הבדיקה, מקרי הבדיקה ונתוני הבדיקה.

**מושגים:**

**קופסה שחורה:** מקובל להבחין בין טכניקות קופסה שחורה וקופסה לבנה. טכניקת קופסה שחורה המכונות גם טכניקות מבוססות- מפרט, הן דרך להפיק ולבחור תנאי בדיקה, מקרי בדיקה או נתוני בדיקה בעקבות ניתוח של בסיס הבדיקה. כמו כן, שיטה זו כוללת בדיקות תפקודיות ושאינן תפקודיות. מיותר לציין, כי מעצם הגדרתן, בדיקות קופסה שחורה אינן משתמשות במידע המתייחס למבנה הפנימי של הרכיב או של המערכת הנבדקת.

**קופסה לבנה:** טכניקות קופסה לבנה המכונות גם טכניקות מבניות או מבוססות-מבנה, מבוססות על ניתוח מבנה הרכיב או המערכת. כמו כן, אפשר לשלב בדיקות קופסה לבנה או קופסה שחורה עם טכניקות מבוססות-ניסיון. על מנת לנצל את ניסיונם של המפתחים, הבודקים והמשתמשים בקבלת ההחלטה מה לבדוק.

**הערה:**  בקורס נתייחס לטכניקות מבוססות-מפרט כאל טכניקות קופסה שחורה, ואל טכניקות עיצוב מבוססות-מבנה כאל טכניקות קופסה לבנה.

**טכניקות מבוססות מפרט= קופסה שחורה:**

**הגדרה:** נוהל להפקה או בחירה של מקרי בדיקה, המבוסס על ניתוח המפרט של רכיב או מערכת. בין שהוא פונקציונלי או לא ללא התייחסות למבנה הפנימי שלהם.

**מאפייני בדיקות מבוססות-מפרט(קופסא שחורה):** המאפיינים הנפוצים הם: מודלים רשמיים או לא-רשמיים. המשמשים לאפיון הבעיה אותה יש לפתור או לאפיון התוכנה או רכיביה. כמו כן, ניתן לגזור מקרי בדיקה באופן שיטתי ממודלים אלה.

**מאפייני בדיקות מבוססות-מבנה(קופסא לבנה):** המאפיינים נפוצים הם: מידע המתייחס לאופן בו בנויה התוכנה משמש לגזירת מקרי בדיקה (למשל,, קוד ומידע מפורט אודות עיצוב). כמו כן, ניתן למדוד את היקף כיסוי התוכנה עבור מקרי בדיקה קיימים ואף ניתן לגזור מקרי בדיקה נוספים באופן שיטתי על-מנת להגדיל את הכיסוי.

**מאפייני בדיקות מבוססות-ניסיון:** המאפיינים הנפוצים הם: הידע והניסיון של אנשים המשמש לגזיר מקרי הבדיקה. מקור מידע אחד הוא הידע של הבודקים, המפתחים המשתמשים ובעלי עניין אחרים לגבי התוכנה, אופן השימוש בה וסביבה בה היא תופעל. מקור מידע נוסף הוא ידע אודות פגמים אפשריים ומקום הימצאותם.

* **אפשר לשלב בדיקות קופסה לבנה או שחורה עם טכניקות מבוססות-ניסיון**

**חלוקת שקילות:**

* **מחלקת שקילות (equivalence partition)-** חלק מתחום של קלט או פלט, שעבורו ההתנהגות של רכיב או מערכת אמורה להיות זהה, (עבור כל פריט) על בסיס המפרט.
* **חלוקת שקילות, חלוקה למחלקות שקילות-** טכניקת עיצוב בדיקות קופסה שחורה אשר בה מקרי בדיקה מעוצבים לשם ביצוע מדגם מייצג מתוך מחלקת השקילות. או במילים אחרות מקרי בדיקה שמחסה לפחות פעם אחת כל מחלקת שקילות.

במסגרת חלוקת שקילות, הקלט מחולק לקבוצות לפי הדמיון הצפוי בהתנהגות המערכת, עבור ערכי קלט מכל קבוצה.כמו כן, מחלקות שקילות קיימות עבור נתונים תקפים, כלומר, ערכים קבילים. וכן עבור נתונים לא תקפים, כלומר ערכים שיש לדחות.

באופן כללי, אפשר להגדיר מחלקות עבור: פלט, ערכים פנימיים, ערכים תלוי זמן, פרמטרים של ממשק וכו' . בנוסף, חלוקת שקילות מתאימה ליישום בכל רמות הבדיקה ומאפשרת להשיג יעדי כיסוי של קלט ופלט.

**ניתוח ערכי גבול:**

* **ערך גבול (boundary value)-**  ערך קלט או פלט הנמצא בקצה של מחלקת שקילות או במרחק המצטבר בקטל ביותר מאחד מצדי הקצה. למשל, בטוח מספרים שלמים בין 0-100ערכי הגבול הם: 0 ו- 100 (כדאי לבדוק גם: 1-, 0, 1, 99, 100, 101)
* **ניתוח ערכי גבול (boundary value analysis)-**  טכניקת עיצוב בדיקות קופסא שחורה אשר בה מקרי בדיקה מעוצבים על בסיס ערכי גבול. ראוי לציין, כי סביר יותר לגלות התנהגות לא תקינה בקצה של כל מחלקת שקילות מאשר בתוכה. לכן, הקצוות הם אזורים שצפוי שיכילו פגמים.
* **ערכי מינימום ומקסימום של מחלקה הינם:**
  + ערך גבול עבור מחלקה תקפה הינו ערך גבול תקף
  + ערך הגבול של מחלקה בלתי-תקפה הוא ערך גבול לא תקף

**הערה:** ניתן לעצב בדיקות כך שיכסו ערכי גבול תקפים ולא תקפים ולכן, כאשר מעצבים מקרי בדיקה, יש לבחור בדיקה עבור כל ערך גבול.

**יישום טכניקת ערכי הגבול:** ניתוח ערכי גבול מתאים ליישום בכל רמות הבדיקה. שכן, טכניקה זו קלה יחסית ליישום ויכולות גילוי הפגמים שלה היא גבוהה. יש לציין כי, מפרטים מפורטים עשויים לסייע בבחירת הקצוות המעניינים. ניתן לומר, כי הטכניקה נחשבת להרחבה של חלוקת שקילות או טכניקות קופסה שחורה אחרות ( טכניקה זו ניתנת ליישום עבור קלט מסך מן המשתמש ).

**בדיקות טבלת החלטה**:

* **טבלת החלטות (decision table)-** טבלה המראה צירופים של קלט ו/או מניעים עם הפלט ו/או הפעולות (האפקטים) הקשורים בהם. כמו כן, ניתן להשתמש בה לעיצוב מקרי בדיקה.
* **בדיקות טבלאות החלטה (decision table testing)-** טכניקת קופסה שחורה של עיצוב בדיקות, שבה מקרי בדיקה מעוצבים כדי לבצע את צירופי הקלט ו/או המניעים/הגורמים המוצגים בטבלת החלטות.

טבלאות החלטה הן שיטה טובה, לקבל תמונה של דרישות המערכת המכילות תנאים לוגיים ולתעד את עיצובה הפנימי של המערכת. בנוסף, ניתן להשתמש בהן לרשום כללים עסקיים מורכבים אשר המערכת נדרשת ליישם.

**צירופים בוליאניים**:

* כאשר יוצרים טבלאות החלטה, מנתחים את המפרט ומזהים את תנאי המערכת ופעולותיה.
* תנאי הפלט והפעולות בדרך כלל מיוצגים באופן בוליאני. כלומר, הם יכולים להיות במצב נכון (true) או לא-נכון (false).
* טבלת ההחלטה מכילה את התנאים המחוללים שהם לרוב צירופים של נכון ולא-נכון עבור כל ערכי הקלט, ופעולות המערכת הנובעות מכל צירוף כזה.

**טבלת החלטה:**

* כל עמודה בטבלה תואמת לכלל עסקי המגדיר צרוף ייחודי של תנאים, אשר כתוצאה מהם מתבצעות הפעולות הקשורות לכלל העסקי הנתון.
* הכיסוי התקני המקובל לשימוש עם טבלאות החלטה דורש הכנה של לפחות בדיקה אחת לכל עמודה בטבלה. ולרוב משמעות הדבר היא כיסוי כל הצירופים האפשריים של תנאים מחוללים.
* החוזק של בדיקות טבלת החלטה נעוץ בכך שהן יוצרות צירופי תנאים אשר אחרת ייתכן שלא היו נבדקים במהלך הבדיקות.
* טכניקה זו ניתנת ליישום בכל מצב בו פעולה של התוכנה תלויה במספר החלטות לוגיות.

**בדיקות הֶחְלֵף מצבים**:

* **הֶחְלֵף מצבים, מעבר מצבים (state transition testing)-** הֶחְלֵף (מעבר) בין שני מצבים של רכיב או מערכת.
* **בדיקות הֶחְלֵף מצבים-** טכניקת בדיקת קופסה שחורה, שבה מקרי בדיקה מעוצבים כדי להריץ חילופי מצבים תקפים ובלתי-תקפים.

כמו כן, בדיקות החלף מצבים משמשות רבות את תעשיית התוכנה ואת המיכון הטכני.

**הערה:** מערכת עשויה להציג תגובה המשתנה בהתאם לתנאים הנוכחים, או להיסטוריה קודמת. במקרה כזה אפשר להציג את היבט המערכת בתרשים החלף מצבים.

**היתכנות וכיסוי**:

* **תרשים הֶחְלֵף מצבים מאפשר לבודק לצפות בתוכנה במונחים של:** מצבים אפשריים במערכת, המעברים בין המצבים, הקלט או האירועים המחוללים שינויי מצב (מעברים) והפעולות הנובעות מן המעברים האמורים.
* **במערכת או בעצם הנבדק ישנו מספר סופי של מצבים אפשריים**, והם נבדלים וברי-זיהוי.
* **טבלת מצבים מציגה את היחס בין המצבים לקלט** –עשויה להבליט מעברים מסוימים שאינם אפשריים או אסורים.
* **אפשר לעצב בדיקות אשר תכסינה:** רצף טיפוסי של מצבים או את כל המצבים, רצף מסוים של מעברים או את כל המעברים, או שתבדוקנה מעברים בלתי תקפים.

**בדיקות מקרי שימוש:**  מקרי שימוש(use case) מתאר, את פעולות הגומלין בין השחקנים (משתמשים/מערכות) אשר מייצרים תוצאה בעלת ערך למשתמש במערכת או ללקוח. כמו כן, אפשר לתאר מקרי שימוש ברמה המופשטת (ללא התייחסות לצד הטכני) או ברמת המערכת (מקרה שימוש של המערכת ברמה התפקודית שלה).

**תנאים ותרחישים:** לכל מקרה שימוש יש תנאים מוקדמים שצריכים להתקיים על מנת שמקרה השימוש יתבצע כמתוכנן. שכן, כל מקרה שימוש מסתיים בתנאי אחרי-ביצוע (post-condition) שהם התוצאות הנצפות ומצבה הסופי של המערכת. בד"כ למקרה שימוש יש: תרחיש ראשי שהוא התרחיש שקורה בד"כ, ותרחישים חלופיים.

**שימושים במקרי שימוש:** מקרי שימוש מתארים את "זרימת התהליך" במערכת בהתבסס על השימוש הצפוי בפועל, ולכן מקרי הבדיקה הנגזרים מהם יעלים ביותר בחשיפת פגמים. כמו כן, מקרי שימוש יעלים גם בעת עיצוב בדיקות קבלה (בהשתתפות הלקוח או המשתמש). בנוסף לכך, מקרי שימוש מסייעים בחשיפת פגמי אינטגרציה הנובעים מפעולות גומלין או בהפרעות בקשר בין רכיבים שונים אשר אינם נראים בבדיקת רכיבים בודדים.

**הערה:** עיצוב בדיקות על בסיס מקרי שימוש ניתן לשילוב עם טכניקות בדיקה מבוססות-מפרט אחרות. כמו כן, **ממקרה שימוש נגזרות לרוב סדרת מקרי בדיקה.**

**טכניקות מבססות מבנה = קופסה לבנה:**

כאמור, בדיקות קופסה לבנה הן בדיקות המבססות על ניתוח המבנה הפנימי של רכיב או מערכת. והן מבוססות על כך, שהבודקים מכירים מבנים מוגדרים ברמת הקוד או המערכת כגון:

* רמת הרכיב (component)- משפטים, החלטות, הסתעפויות ואף מסלולים (ברמת התוכנה)
* רמת האינטגרציה- נקרא גם עץ קריאה. כלומר, תרשים שבו מודלים קוראים למודלים אחרים.
* רמת המערכת- המבנה עשוי להיות מבנה תפריט, תהליך עסקי או מבנה דף אינטרנט.
* כיסוי קוד- שיטות ניתוח אשר קובעות איזה חלקים מהתוכנה בוצעו בפועל (כוסו) ע"י סדרת בדיקות, ואיזה לא בוצעו (כיסוי משפטים, החלטות, תנאים וכו').

**2 רמות כיסוי קוד בסיסיות:**

קיימות שתי טכניקות מבוססות-מבנה לכיסוי קוד, אשר מבוססות על משפטים, החלטות והסתעפויות.

**בדיקות משפטים וכיסוי משפטים:** מדבר על אחוז המשפטים שבוצעו בפועל ע"י סדרת בדיקות, מתוך כלל המשפטים (פקודות) בקוד. לפיכך, טכניקות המשפטים מגדירה מקרי בדיקה, שיריצו פקודות מסוימות על מנת להגדיל את כיסוי המשפטים. ובאופן כללי, כיסוי משפטים, נקבע על-פי חלוקה של מספר הפקודות שהורצו ע"י מקרי הבדיקה, בסך-כל הפקודות הניתנות להרצה בקוד הנבדק.

**בדיקות החלטות וכיסוי החלטות:** כיסוי החלטות הנקרא גם כיסוי הסתעפויות, הינו הערכת אחוז תוצאות של החלטות אשר נבחנו ע"י סדרת בדיקות (למשל:T/F של פקודות IF). טכניקות בדיקות ההחלטות מגדירה מקרי בדיקה שיגרמו להחלטות ספציפיות בנקודות הסתעפות בקוד. באופן כללי, כיסוי החלטות נקבע ע"פ מספר תוצאות ההחלטות המכוסות ע"י מקרי הבדיקה, חלקי סך-כך תוצאות ההחלטות האפשריות בקוד הנבדק. שכן, בדיקות החלטות היא צורה של בדיקות תרשים זרימת בקרה. משום שהיא עוקבת אחר מסלול מסוים של בקרה דרך נקודות החלטה.

**לכיסוי החלטות יש עוצמה רבה יותר מאשר לכיסוי משפטים. כיסוי החלטות של 100% מבטיח כיסוי של 100% משפטים. אך ההיפך אינו נכון.**

**טכניקות מבוססות-ניסיון:**

**טכניקות מבוססות-ניסיון experience-based test design technique)):**

**בדיקות מבוססות-ניסיון-** בבדיקות מבוססות-ניסיון מגדירים בדיקות המבוססות על: מיומנות הבודקים, האינטואיציה שלהם וניסיונם עם יישומים וטכנולוגיות דומים. טכניקות מבוססות-ניסיון עשויות להיות יעילות בזיהוי בדיקות מיוחדות שקשה להגיע אליהם באמצעות טכניקות רשמיות אך יש לציין, כי משתמשים בבדיקות אלה כדי להשלים את הבדיקות השיטתיות.

**ניחוש טעויות (error guessing)**- ניחוש טעויות היא טכניקה מבוססת-ניסיון נפוצה. כאשר בד"כ הניחוש מתבצע ע"י ניסיונו של הבודק. כמו כן, קיימת גישה מובנית לטכניקת ניחוש הטעויות לפיה, מכינים רשימה של פגמים אפשריים, ולאחר מכן מעצבים בדיקות שמכוונות למציאת הפגמים הללו. גישה שיטתית זו מכונה **התקפה מוכוונת ליקויים (fault attack)**. את רשימות הפגמים האפשריים מבססים על-פי הניסיון, נתונים קיימים לגבי פגמים וליקוים ואף על בסיס הידע הקיים על סיבות לכשלים בתוכנה.

**בדיקות חוקרותexploratory testing) )-** בבדיקות חוקרות מתקיימים בו זמנית, עיצוב, רישום ולמידה כל זאת על בסיס אמנת הבדיקה. בין היתר, בדיקות חוקרות כוללות יעדי בדיקה המבוצעות במסגרת לוחות זמנים מוגדרים. יש לציין, כי גישה זו יעילה במיוחד במצבים בהם יש מפרט מצומצם או בלתי מספיק, אילוצי זמן חריפים, או כאשר יש צורך בהרחבה או השלמה לבדיקות הרשמיות יותר.

הערה: גישה זו יכולה לשמש כביקורת על תהליך הבדיקה, על מנת להעלות את הסיכוי במרבית הפגמים החמורים יימצאו.

**בחירת טכניקות הבדיקה:**

הבחירה באיזו טכניקת בדיקה להשתמש תלויה במספר גורמים בהם: סוגי המערכת, תקנים רגולטוריים, דרישות של הלקוח או דרישות חוזיות, דרגת הסיכון וסוג הסיכון, מטרות הבדיקה והתיעוד הקיים, הידע של הבודקים, זמן ותקציב, מחזור חיי הפיתוח, מודלים של מקרי שימוש, וניסיון קודם עם סוגי הפגמים שנמצאו.

כאשר מגדרים מקרי בדיקה, לרוב משתמשים בשילוב של טכניקות בדיקה שונות כגון:

* טכניקות מונחות-תהליך (process driven)
* טכניקות מונחות-כללים (rule driven)
* טכניקות מונחות-נתונים (data driven)

כל זאת על מנת להבטיח כיסוי נאות של העצם הנבדק.

**סילבוס פרק 5:**

**ניהול הבדיקות**

**ארגון הבדיקות:**

**בודק (Taster)-** איש מקצוע מיומן המעורב בבדיקות של רכיב או מערכת

**מנהל בדיקות (test manager)-** אדם האחראי לניהול פרויקט של פעולות בדיקה ומשאביהן ולהערכה של מושא הבדיקות. מנהל הבדיקות הינו אדם אשר מכוון, מבקר, מנהל, מתכנן ומסדיר את הערכת מושא הבדיקות

**מוביל בדיקות (test leader)-** לפי הגדרה מוביל בדיקות זהה למנהל בדיקות אך בפועל זה לא בהכרח ככה. זאת מכיוון שמוביל הבדיקות עשוי להיות המוביל הטכני של הבדיקות, מבלי לבצע את פונקציות הניהול השונות.

**ארגון הבדיקות ועצמאותן:**

ניתן לייעל את איתור הפגמים ע"י בדיקות וסקרים אם ארגון הבדיקות נהנה מעצמאות. להלן טווח האפשריות החל בתלות מוחלטת ועד לעצמאות מרבית:

* אין בודקים עצמאיים. כלומר, המפתחים עצמם בודקים את הקוד שכתבו (תלות מוחלטת)
* בודקים עצמאיים בתוך צוותי הפיתוח
* צוות או קבוצת בדיקות עצמאיים בתוך ארגון המדווחים למנהל הפרויקט או להנהלה הבכירה.
* בודקים עצמאיים מהיחידה העסקית או מתוך קהילת המשתמשים.
* מומחי בדיקות עצמאיים לסוגי בדיקות מסוימות למשל: בודקי שימושיות, אבטחה, הסמכה
* בודקים עצמאיים חיצונים לארגון או העובדים באמצעות מיקור חוץ

עצמאות הבדיקות בפרויקטים קריטיים:

* במקרה של פרויקט מורכב או קריטי מבחינה בטיחותית, עדיף בד"כ להסתמך על מספר רמות בדיקה, כאשר חלקן או כולן מבוצעות ע"י בודקים עצמאיים.
* צוות הפיתוח יכול להשתתף בבדיקות, בפרט ברמות הנמוכות יותר, אך העדר האובייקטיביות שלהם מגביל בד"כ את האפקטיביות שלהם.
* אפשר להסמיך את הבודקים העצמאיים לדרוש ולהגדיר נוהל בדיקות וכללי בדיקה. אולם, על הבודקים לקחת על עצמם תפקידים כאלה רק כאשר הוסמכו במפורט לעשות זאת ע"י ההנהלה.

**יתרונות וחסרות הבדיקה העצמאית:**

**יתרונות:** בודקים עצמאיים אינם מוטים, ולכן יכולים לזהות פגמים אחרים ומסוגים שונים מאלו בודקים לא עצמאיים מזהים. בנוסף לכך בודקים עצמאיים יכולים לאמת הנחות שהונחו המהלך אפיון ויישום המערכת.

**חסרונות:** הבודקים מבודדים מצוות הפיתוח( כאשר מקפידים על אי תלות מוחלטת). כמו כן, מפתחים עלולים לאבד את תחושת האחריות והמחויבות לאיכות. לא רק אלא גם, בודקים עצמאיים עלולים להצטייר כצוואר בקבוק או להיות מואשמים בעיכובים בשחרור הגרסה.

**בודקים אפשריים:** משימות בדיקה יכולות להתבצע ע"י: מי שהגדרת תפקידו היא "בודק" או ע"י בעלי תפקיד אחר. למשל: מנהל פרויקט, מפתח, מומחה עסקי או אנשי רשת ותשתית של מערכי מחשוב.

**משימותיהם של מנהל/מוביל בדיקות ושל בודק:**

הפעילות והמטלות המבוצעות בידי שני נושאי תפקידים אלה תלויות בהקשר הפרויקט והמוצר, באנשים הממלאים את התפקידים והארגון עצמו.

**מוביל/מנהל בדיקות (test leader)-** מכונה לעיתים מנהל בדיקות או מתאם בדיקות. את התפקיד הנ"ל יכול למלא, בין היתר, מנהל פרויקט, מנהל פיתוח, מנהל הבטחת איכות או מנהל קבוצת בדיקות. יש לציין, כי עבור פרויקטים גדולים ייתכנו שני תפקידים נפרדים (מוביל בדיקות ומנהל בדיקות) שכן, מוביל/מנהל הבדיקות יהיה אחראי על תכנון, פיקוח ובקרה של פעילויות הבדיקה והמשימות.

משימות אופייניות של מוביל/מנהל בדיקות:

בין משימותיו האופייניות של מוביל/מנהל בדיקות אפשר למנות:

* תיאום אסטרטגיית הבדיקות ותוכנית הבדיקות
* כתיבה או סקירה של אסטרטגיית בדיקות עבור הפרויקט-ושל מדיניות בדיקות עבור הארגון
* הוספת היבט הבדיקות לפעילויות אחרות כמו למשל האינטגרציה
* תכנון הבדיקות-בחירת גישות בדיקה, הערכת זמן, מאמץ ועלות הבדיקות הקצאת משאבים.
* הובלת אפיון, הכנה, יישום וביצוע הבדיקות- מעקב אחר תוצאות הבדיקות ואמות המידה ליציאה.
* התאמת התכנון לתוצאות הבדיקות ולהתקדמותן
* יישום ניהול תצורה נאות של מכלול מרכיבי הבדיקות (בודקה) לצורך נעקבות
* הגדרת מדדים מתאימים למדידת התקדמות הבדיקות- ולהערכת איכות הבדיקות והמוצר
* החלטה אילו בדיקות תעבורנה אוטומציה- עד איזו רמה וכיצד

משימות אופייניות של בודק:

* סקירת תכניות הבדיקה והשתתפות בכתיבתן
* ניתוח, סקירה והערכה של דרישות המשתמש, המפרטים והמודלים (מבחינת הבדיקות שלהן)
* יצרת מפרטי בדיקות
* הגדרת ויישום סביבת הבדיקות
* השגה ו/או הכנת נתוני הבדיקות
* יישום בדיקות בכל רמות הבדיקה- ביצוע בדיקות, תיעודן ואיסוף תוצאותיהם ורישומן, הערכת התוצאות, תיעוד סטיות מן התוצאות הצפויות.
* שימוש בכלי ניהול בדיקות ובכלי מעקב לפי הצורך
* אוטומציה של בדיקות (לעיתים בתמיכת מפתח או מומחה לאוטומציה)
* מדידת ביצועים של רכיבים ושל מערכות
* סקירת בדיקות שפותחו ע"י אחרים

בנוסף, האנשים העובדים על ניתוח בדיקות, עיצוב בדיקות, בדיקות ספציפיות או אוטומצית בדיקות עשויים להיות מומחים לתחומים אלה הבאים מחוץ לצוות הקבוע. כמו כן, בהתחשב ברמת הבדיקות והסיכונים הקשורים במוצר ובפרויקט הבודק ישמור על מידה כלשהי של עצמאות. ובד"כ בודקים ברמת הרכיב או האינטגרציה יהיו מפתחים. בודקים ברמת בדיקות הקבלה יהיו מומחים עסקיים ומשתמשים ובודקים ברמת בדיקות קבלה תפעוליות יהיו מפעילים.

**תכנון הבדיקות והערכת היקף העבודה:**

* אסטרטגיית בדיקות- תיאור כללי מגבוה (high-level) של רמות הבדיקה, והבדיקות הכלולות ברמות אלה שיש לבצען עבור ארגון או פרויקט.
* גישה לבדיקות (test approach)- היישום של אסטרטגיית בדיקות לפרויקט ספציפי כוללת, בין היתר, את ההחלטות שנתקבלו בעקבות מטרת הפרויקט (בדיקות הפרויקט), נקודת התחלה ביחס לתהליך של הבדיקות, טכניקות עיצוב הבדיקות שיש ליישם, אמות מידה ליציאה וסוגי בדיקות שיש לבצע.

**תכנון הבדיקות:**

את תכנון הבדיקות ניתן לתעד באמצעות מספר דרכים. ניתן לתעדן בתוכניות אב לבדיקות, ובתוכניות בדיקה נפרדות לכל רמת בדיקה כגון: בדיקות מערכת ו/או בדיקות קבלה.

גורמים המשפעים על תכנון הבדיקות: התכנון מושפע ממדיניות הבדיקות של הארגון, היקף הבדיקות, היעדים, הסיכונים, האילוצים, מידת הקריטיות, מידת הבדיקות זמינות ומשאבים. כתוצאה מכך, ומהתקדמות הפרויקט ותכנון הבדיקות מתקבל מידע רב אותו ניתן להוסיף לתוכנית הבדיקות.

פעילות מתמשכת ומשוב: תכנון בדיקות הינו פעילות מתמשכת המתבצעת במשך כל התהליכים והפעילויות של מחזור חיי המוצר. המשוב מפעילויות הבדיקה משמש לזיהוי שינוי בסיכונים שבמוצר, כך שניתן להתאים את התכנון לשינויים.

פעילויות תכנון בדיקות: פעילויות תכנון הבדיקות של מערכת או חלק ממנה כוללת בין היתר, בחינת ההיקף והסיכונים וזיהוי יעדי הבדיקות, הגדרת גישת הבדיקות הכללית הכוללת בתוכנה את הגדרת רמות הבדיקה ואמות המידה לכניסה ויציאה, אינטגרציה ותיאום פעילויות הבדיקה עם פעילויות מחזור חיי התוכנה(הגדרת דרישות, רכישה ואספקה, פיתוח, תפעול ותחזוקה). כמו כן, פעילויות תכנון בדיקות כוללת בתוכה את קבלת ההחלטות שיקבעו מה לבדוק, מי יבצע איך לבצע וכיצד להעריך את התוצאות. כמו גם, כוללת את ניתוח הבדיקות ועיצובן, יישום, ביצוע והערכת הבדיקות (באמצעות קביעת לו"ז וסדר ביצוע). ובנוסף לכך, כוללת גם את מסמכי הבדיקות, בחירת מדדים למעקב ובקרה של הכנת הבדיקות וביצוען, פתרון פגמים וסיכום. ואף הגדרת רמת הפירוט הנדרשת בתיאור נוהל הבדיקות (כך שההכנה והביצוע של הבדיקות יהיו ניתנים לשחזור.

אמות מידה לכניסה: אמות המידה לכניסה מגדירות את הנקודה בה מתחילים לבדוק למשל את תחילת רמת הבדיקה או הנקודה בה סדרה של בדיקות מוכנה לביצוע. בד"כ אמות המידה לכניסה מכסות את: זמינות סביבת הבדיקות ומוכנותה, מוכנות כלי הבדיקות בסביבת הבדיקה, זמינותו של קוד מוכן לבדיקות ואת זמינות נתוני הבדיקות.

אמות מידה ליציאה: אמות המידה ליציאה מגדירות את הנקודה בה מפסיקים לבדוק, למשל בסוף רמת הבדיקה או כאשר סדרת בדיקות מסוימת השיגה מטרה מוגדרת. בד"כ אמות המידה ליציאה מכסות את הנק' הבאות: מדדים ליסודיות הבדיקות כגון כיסוי קוד, כיסוי תפקודיות וכיסוי סיכונים. הערכות של צפיפות הפגמים ומדדי אמינות, עלות, סיכונים שנותרו כתוצאה מפגמים שלא תוקנו או חוסר בכיסוי בדיקות באזורים מסוימים וכמובן את לוחות הזמנים המבוססים על זמן הגעה לשוק.

הערכת היקף עבודת הבדיקות:

שתים מן הגישות להערכת ההשקעה הדרושה לביצוע הבדיקות שבסופן ניתן יהיה לזהות משאבים נדרשים ולהקצתם, ולנסח לו"ז וסדר ביצוע הינן:

* גישה מבוססת מדדים: הערכת ההשקעה שתידרש על-פי מדדים של פרויקטים קודמים או דומים או על-פי ערכים אופייניים.
* גישה מבוססת מומחה: הערכת היקף המשימות לפי הערכות של בעלי המשימות או המומחים.

מאמץ הבדיקות עשוי להיות תלוי במספר גורמים: מאפייני מוצר(איכות בסיס הבדיקות, גודל המוצר, מורכבות וכו') מאפייני תהליך הפיתוח (יציבות הארגון, מיומנות הצוות ואילוצי זמן), תוצאות הבדיקות (מספר הפגמים)

אסטרטגיית הבדיקות, גישה לבדיקות: הגישה לבדיקות הינה יישום של אסטרטגיית הבדיקות, עבור פרויקט מסוים. את גישת הבדיקות מגדירים ומלטשים במסגרת תכניות הבדיקות ועיצוב הבדיקות. בד"כ גישת הבדיקות כוללת החלטות שהתקבלו על בסיס מטרת הפרויקט והבדיקות, והערכת הסיכונים. גישת הבדיקות הינה נקודת התחלה לתכנון תהליך הבדיקות (בחירת טכניקות עיצוב הבדיקות וסוגי הבדיקות שיושמו, הגדרת אמות מידה לכניסה וליציאה).

הגישה הנבחרת, היא תלוית הקשר ועשויה להביא בחשבון: סיכונים, סכנות ובטיחות, משאבים ומיומנויות זמינים, טכנולוגיה, אופי המערכת, יעדי הבדיקות, תקנות ורגולציה ועוד.

גישות אופייניות:

* גישות אנליטיות, כגון בדיקות מבוססות סיכונים (מסיכון הגבוה לנמוך)
* גישות מבוססות מודל, כגון בדיקות אקראיות (ע"ב מידע סטטיסטי, אודות שיעורי הכשל או אודות השימוש)
* גישות שיטתיות כגון: גישה מבוססת כשל, גישות מבוססות ניסיון, גישות מבוססות רשימת ביקורת, וגישות מבוססות מאפייני איכות.
* גישות תואמות תהליך או תקן כגון: אלה המוכתבות ע"י תקנים תעשייתיים או ע"י שיטות זריזות למיניהן
* גישות דינמיות והאוריסטיות כגון: בדיקות חוקרות
* גישות ייעוציות כגון אלא בהן כיסוי הבדיקות מונע בעיקר ע"י ייעוץ והדרכה של מומחים חיצונים מהתחום העסקי או הטכנולוגי שאינם שייכים לצוות הבדיקות
* גישות המתמקדות בהפחתת הסיכון לתופעות נסיגה כגון אלה הכוללות שימוש חוזר במקרי בדיקה קיימים, אוטומציה של בדיקות נסיגה תפקודיות, סדרת בדיקות סטנדרטיות.

הערה: ניתן לשלב בין הגישות השונות למשל גישה דינמית שהיא גם מבוססת סיכונים.

**מעקב ובקרה של התקדמות הבדיקות**

**מושגים:**

**צפיפות פגמים (defect density)-** מספר הפגמים המזוהים ברכיב או מערכת. מחולק בגודל הרכיב או המערכת. כאשר הגודל מתבטא במונחי מדידה תקניים כמו מספר שורות קוד, מספר מחלקות או מספר נקודות פונקציה

**שיעור כשלים (failure rate)-** שיעור מספר הכשלים מקטגוריה נתונה, ביחס ליחדת מדידה נתונה. למשל כשלים ליחידת זמן, כשלים למספר תנועות, כשלים למספר הרצות מחשב.

**ניטור בדיקות (test monitoring)-** מטרת של ניהול בדיקות העוסקת בפעילויות הקשורות בבדיקות סטטוס פרויקט הבדיקות. שבמסגרתה מכינים דוחות המשווים את המצב הנוכחי בפועל לעומת מה שתוכנן.

**בקרת בדיקות (test control)-** מטרה של ניהול בדיקות העוסקת בפיתוח ויישום סט של פעולות מתקנות בכדי להחזיר פרויקט בדיקות למסלולו (כאשר הניטור מראה סטייה ממה שתוכנן).

**דו"ח סיכום בדיקות (test summary report)-** מסמך המסכם פעילויות בדיקה ותוצאותיהן. כולל גם הערכה של פריטי הבדיקה הקשורים לכך- מול אמות המידה ליציאה.

**מעקב אחר התקדמות הבדיקות**

מטרת המעקב אחר בדיקות היא לספק משוב ולתת תמונת מצב ברורה של פעילויות הבדיקה. כמו כן, אפשר לאסוף את המידע אחריו עוקבים באופן ידני או אוטומטי ואף להשתמש בו בכדי למדוד את אמות המידה ליציאה כמו למשל רמת כיסוי, או לשם הערכה של קצב ההתקדמות ביחס ללו"ז ולתקציב המתוכננים.

מדדים להתקדמות הבדיקות:

* אחוז עבודת ההכנה של מקרי בדיקה שבוצע בפועל (או אחוז מקרי הבדיקה)
* אחוז עבודת ההכנה של סביבת העבודה שבוצע בפועל
* ביצוע מקרי בדיקה (למשל מספרי מקרי הבדיקה שהורצו/לא הורצו או עברו/נכשלו)
* מידע אודות פגמים (למשל, צפיפות פגמים, פגמים שהתגלו, שיעורי כשלים וכו')
* שיעורי הכיסוי ע"י הבדיקות בתחומים הבאים: דרישות, סיכונים וקוד
* תחושת הביטחון הסובייקטיבית של הבודקים באיכות המוצר
* מדדים הקשורים במועדיהן של אבני דרך בבדיקות
* עלויות בדיקה (כולל עלות אל מול תועלת או בהרצת הבדיקה הבאה)

דיווח על בדיקות

דיווח על בדיקות מתייחס לסיכום המידע אודות מאמץ הבדיקות כולל: מה אירע במהלך תקופת הבדיקות ניתוח של מידע ומדדים אשר יתמוך בהמלצות ובהחלטות לגבי פעולות בהמשך כגון: הערכת כמות הפגמים הנותרים, התועלת הכללית של המשך הבדיקות, סיכונים שעדיין קיימים ודרגת האמון בתוכנה הנבדקת.

תקן התיעוד והמדדים

מתייחס למבנה והסעיפים של דו"ח סיכום הבדיקות. את המדדים יש לאסוף במהלך רמת הבדיקה ובסופה על מנת להעריך את התאמתם של יעדי הבדיקות לרמת הבדיקה הנדונה, את התאמת גישות הבדיקה שננקטו ואת יעילות הבדיקות ביחס ליעדיהן.

בקרת בדיקות

בקרת בדיקות מתארת פעולות הנחיה או תיקון שננקטות כתוצאה ממידע או מדדים שנאספו ושדווחו. פעולות אלה עשויות לכלל כל פעילות שהיא ואף עשויות להשפיע על כל פעילות או משנה אחרת שמחזור חיי התוכנה

דוגמאות לפעולות בקרת בדיקות:

* קבלת החלטות המבוססות על מידע שהתקבל מניטור הבדיקות
* תיעדוף מחשב של בדיקות בעקבות התממשות סיכון צפוי
* שינוי בלו"ז או בסדר ביצוע הבדיקות
* הגדרת אמת מידה לכניסה הדורשת שמפתחי יבצע בדיקה חוזרת של תיקונים (בדיקות אישור) לפני קבלתם אל תוך הגרסה

**ניהול תצורה ( configuration management )**

נקראת גם בקרת גרסאות. שיטה האוכפת הנחיות ופיקוח, טכנית ואדמיניסטרטיבית לגבי: זיהוי ותיעוד של מאפיינים תפקודיים ופיזיים של פריט תצורה, בקרת שינויים של מאפיינים אלו, תיעוד ודיווח של תהליך השינוי וסטטוס יישומו, אימות התאמה עם דרישות שפורטו.

ניהול תצורה מצד הבדיקות:

באופן כללי, ניהול תצורה מיועד לבסס ולשמור את השלמות של מוצרי התוכנה או המערכת. מצד הביקורת, ניהול תצורה קשור להבטחת הנק' הבאות: יש זיהוי חד-משמעי לכל פריט במכלול מרכיבי הבדיקות, כאשר עבור על פריט יש בקרת גרסה ומעקב אחר שינויים וכל הפריטים מקושרים זה לזה או למושאי בדיקה באופן שמאפשר נעקבות לכל אורך תהליך הבדיקות. לכל המסמכים ופריטי התוכנה המזוהים יש הפניה מזהה חד-משמעית במסמכי הבדיקות.

ניהול תצורה מצד הבודק: מצד הבודק ניהול תצורה מסייע לזיהוי ודאי ושחזור של פריט נבדק, מסמכי בדיקה, רתמת הבדיקה והבדיקה עצמה

לסיכום, במהלך תכנון הבדיקות יש לבחור, לתעד וליישם את התשתית הנדרשת עבור ניהול התצורה ואת התהליכים והכלים של ניהול תצורה.

**סיכון ובדיקות**

סיכון מוגדר כסיכוי להתרחשות אירוע, סכנה, איום או מצב שיובילו להשלכות בלתי רצויות או לבעיות אפשריות. ובמילים אחרות רמת הסיכון נקבעת בהתאם לסבירות שאירוע לא רצוי יתרחש ולהשפעה (הנזק) שתגרם כתוצאה מהאירוע.

**מושגים:**

**סיכון (risk)-** גורם שתוצאותיו בעתיד עשויות להיות שליליות. לרוב מבוטא כהשפעה או כסבירות

**סיכון פרויקט (project risk)-** סיכון הקשור ישירות לניהול ובקרת פרויקט (הבדיקות) למשל, מחסור באנשי צוות, קווי-סיום קשוחים, דרישות משתנות וכד'.

**סיכוני מוצר (product risk)-** גישה לבדיקות המיועדת להפחית את רמת סיכוני המוצר וליידע את בעלי העניין על מצב הסיכונים, ושימוש ברמות הסיכון כדי להנחות את תהליך הבדיקות.

סיכוני פרויקט:

סיכוני פרויקט נוגעים ליכולת הפרויקט לעמוד ביעדיו למשל:

* גורמים ארגוניים- חוסרים במיומנות, בהכשרה ובאנשי צוות, בעיות של פוליטיקה ארגונית, יחס לא הולם כלפי הבדיקות או צפיות לא הולמות.
* סיכונים טכניים/בעיות טכניות- קושי בהגדרת הדרישות הנכונות, היקף אי-העמידה בדרישות כתוצאה מאילוצים קיימים, סביבת העבודה אינה מוכנה בזמן, איכות נמוכה של: העיצוב, הקוד, נתוני התצורה נתוני הבדיקות והבדיקות עצמן.
* סיכוני צד שלישי- בעיות מצד ספקים (אי עמידה בהתחייבות או בעיות חוזיות)

באופן כללי, כאשר מנהל הבדיקות מנתח, מנהל וממתן את הסיכונים הללו הוא פועל ע"פ העקרונות המקובלים לניהול פרויקט, ובהתאם לתקן תיעוד בדיקות תוכנה

סיכוני מוצר דוגמאות:

* אספקת תוכנה שנוטה להיכשל
* האפשרות שתוכנה או חומרה עלולים לגרום נזק לאדם או לגוף עסקי
* רמה נמוכה של תכונות תוכנה (למשל, תפקודיות, אמינות, שימושיות וביצועים)
* רמה נמוכה של איכות ושלמות הנתונים (בעיות בהגירה, המרה והעברה של נתונים)
* תוכנה שאינה מבצעת את הפעולות שלשמן נועדה.

ההבדל בין סיכוני פרויקט ומוצר: סיכוני פרויקט נוגעים ליכולת לעמוד ביעדים כנדרש כגון יעדי עמידה בזמנים וביעדי איכות (שהינם בעצם סיכוני מוצר) ואילו סיכוני מוצר נוגעים למוצר "נטו" .

סיכונים ובדיקות: שימוש בסיכונים עוזר להחליט היכן להתחיל לבדוק והיכן לבדוק יותר כל זאת במטרה להפחית את הסיכוי להתרחשותה של תופעה שלילית ואת היקף השפעתה. כמו כן, סיכוני מוצר הם סוג מיוחד של סיכון שאחראי על הצלחתו של הפרויקט (בפועל). שכן, הבדיקות כפעילות של בקרת סיכונים מספקות משוב לגבי הסיכון נותר -באמצעות מדידת האפקטיביות של הסרת פגמים קריטיים ושל מימוש תכנית חירום.

**גישת בדיקות מבוססת סיכונים:** גישת בדיקות מבוססת-סיכונים מאפשרת לנקוט פעולות יזומות להפחתת רמת הסיכון במוצר כבר בשלביו המוקדמים של הפרויקט. גישה זו כרוכה בזיהוי סיכוני המוצר ושימוש בהם לשם הנחיית: תכנון הבדיקות, בקרתן, מפרטיהן, הכנתן וביצוען.

שימוש בסיכונים שזוהו: בגישה מבוססת-סיכונים אפשר להשתמש בסיכונים שזוהו כדי לבחור את טכניקות הבדיקה, לקבוע את היקף הבדיקות, לתעדף את הבדיקות ואף בכדי לבחון פעילויות שאינן בדיקות שעשויות להפחית סיכונים (למשל ארגון הכשרה למעצבים חסרי ניסיון).

לסיכום, בדיקות מבוססות-סיכון מסתמכות על הידע והתובנות של בעלי העניין, כדי לזהות סיכונים ולקבוע את רמות הבדיקה הנדרשות להתמודדות איתם. כל זאת בשביל להבטיח את הפחתת הסיכוי לכשל המוצר. כאמור, פעולות ניהול סיכונים מספקות גישה מסודרת שבאמצעות אפשר להעריך מה עשוי להשתבש ואף מאפשרות להחליט באילו סיכונים חשוב לטפל וליישם פעולות לטיפול בסיכונים אלה.

**ניהול אירועים**

**מושגים:**

**ניהול אירועים (incident management)-** התהליך של הכרה, חקירה, נקיטת פעולה וסילוק של אירועים. כמו כן, ניהול אירועים כרוך ברישום אירועים, סיווגם וזיהוי השפעתם.

**רישום אירועים (incident logging)-** תיעוד הפרטים של כל אירוע שהתרחש.

**דו"ח אירועים (incident report)-** מסמך המדווח על כל אירוע שהתרחש (למשל: בזמן בדיקות ואשר דורש חקירה)

רקע/דגשים בניהול אירועים:

* יש לרשום כאירוע כל אי-התאמה בין תוצאות צפויות לצואות בפועל. מאחר ואחד היעדים של בדיקות הוא למצוא בפגמים.
* כל אירוע חייב להיחקר, שכן הוא עלול להתגלות ככשל שמקורו בפגם.
* צריך להגדיר את הפעולות שיש לבצע כשקורה אירוע או מתגלה פגם.
* יש לעקוב אחר אירועים ופגמים מרגע הגילוי והסיווג ועד לתיקון (ולאישור שהפתרון תקין).
* על הארגון לקבוע תהליך לניהול אירועים וכללים לסיווגם על מנת לנהל את האירועים עד להשלמתם.

מתי ועל מה לדווח? תמיד, ועל הכל. כמו כן, ניתן לדווח על אירועים במהלך שלב הפיתוח, הסקירה, הבדיקות או השימוש במוצר תוכנה. אפשר לדווח על בעיות בקוד או במערכת שכבר פועלת או בכל סוג של תיעוד כגון: הדרישות מסמכי הפיתוח , מסמכי הבדיקות ומידע למשתמש כגון "עזרה או הוראות ההתקנה.

יעדי דוחות האירועים:

* לספק משוב למפתחים ולקבוצות אחרות לגבי הבעיה, שיאפשר להם זיהוי, בידוד ותיקון לפי הצורך
* לספק למובילי הבדיקות אמצעי מעקב אחרי איכות המערכת הנבדקת ואחרי התקדמות הבדיקות
* להעלות רעיונות לשיפור תהליך הבדיקות

פרטי דו"ח אירוע:

הפרטים המופעים בדוח אירוע עשויים לכלול:

* תאריך, שם המחבר, והארגון אליו הוא שייך
* תוצאות צפויות ותוצאות בפועל
* פרטים מזהים של: פריט הבדיקות/פריט התצורה וסביבת הבדיקות
* השלב-במחזור חיי התוכנה או המערכת-בו נצפה האירוע
* תיאור של האירוע
* היקף או דרגת ההשפעה על אינטרסים של בעלי עניין
* חומרת ההשפעה על המערכת
* דחיפות או סדר עדיפות לתיקון
* מצב האירוע (פתוח/דחוי/כפול , ממתין לתיקון מתוקן וכו')
* מסקנות המלצות ואישורים
* בעיות כלליות (אזור אחר שעשוי להיות מושפע משינו הקוד כתוצאה מתקון הבעיה שדווחה)
* היסטורית שינוים
* הפניות, כולל זיהוי מדויק של מקרה הבדיקה אשר חשף את הבעיה

**סילבוס פרק 6:**

**כלים תומכי בדיקות**

**סוגים של תומכי בדיקות**

כלי בדיקה יכולים לשמש לפעילות תומכת בדיקות (אחת או יותר). בין הכלים הללו נמנים:

* כלים המשמשים באופי ישיר לבדיקות, כמו למשל: כליי ביצוע בדיקות, כלים ליצירת נתוני בדיקה, כלים להשוואת תוצאות.
* כלים המסייעים בניהול התהליך כמו אלה המשמשים לניהול של הבדיקות, התוצאות הנתונים, האירועים הפגמים וכו' ולדיווח ולמעקב אחר ביצוע הבדיקות
* כלים המשמשים לחקירה (למשל מעקב אחרי יישום שתחת בדיקה)

מטרות הכלים תומכי הבדיקות: מטרות כלים אלו, בין היתר, הינן שיפור יעלות פעילויות הבדיקה ע"י אוטומציה של פעולות החוזרות על עצמן, או שדורשות משאבים רבים כאשר מבצעים אותם ידנית, אוטומציה של פעילויות שאינן ניתנות לביצוע ידני, או תמיכה בפעילויות בדיקה ידניות כמו תכנון בדיקות, עיצוב בדיקות ואף דיווח על בדיקות וניטור, ואף שיפור אמינות הבדיקות.

מסגרת בדיקה (test frameworks):

הערה: השימוש במונח מסגרות בדיקה נפוץ בתחום בדיקות התוכנה בשלוש משמעויות שונות לפחות, בקורס נעסוק רק ב2 אלו:

* ספריות בדיקה הניתנות לשימוש חוזר או להרחבה ואף כאלה שניתן להשתמש בהן לבניית כלי בדיקה (מכונות גם רתמת בדיקות)
* סוג העיצוב של אוטומציית הבדיקות (מונחית נתונים, מונחית מילות מפתח)

סיווג כלי בדיקה: קיימים כלי בדיקה רבים התומכים בהיבטים שונים של בדיקות. את הכלים ניתן לסווג לפי קריטריונים כגון: מטרה, מסחרי/חינמי/קוד פתוח/תוכנה שיתופית, טכנולוגיה ועוד. כמו כן, ישנם כלים אשר תומכים בברור בפעילות אחת ואילו אחרים תומכים ביותר מאחת. כלים מספק יחיד הם כלים אשר תוכננו לעבוד יחד ואף לעיתים משווקים כמערכת משולבת אחת. בנוסף, ישנם כלים שאופי התמיכה שלהם מתאים יותר למפתחים. למשל: כלים המופעלים במהלך בדיקות רכיבים או בדיקות אינטגרציה של רכיבים. כלים אלו מסומנים באות "**מ**" ברשימה שלהלן.

כלים פולשניים ואפקט הגשושית: כלים מסוימים הינם פולשניים או במילת אחרות עשויים להשפיע על תוצאות הבדיקה. לדוגמא: מתקבלת תוצאה שונה כתוצאה מהפקודות הנוספות של הכלי.

**אפקט הגשושית (probe effect)­-** הינו אפקט שלילי שנוצר כתוצאה מכלי בדיקה אשר מבצע פעולה כלשהי על רכיב או מערכת. וכך בעצם האפקט עשוי להשפיע על, תוצאות הבדיקה ובמקרים מסוימים לעוות אותה משמעותית, ואף להאט את הביצועים (עקב הפעלת כלי בדיקת בדיקות ביצועים).

הערה: ניתן לעשות שימוש בכלי ניהול בדיקות בכל פעילויות הבדיקה ולכל אורך מחזור חיי התוכנה.

**כלי ניהול בדיקות (test management tools)**- מרבים מכלים אלה קיים ממש לכמה יכולות בין היתר, ביצוע בדיקות, מעקב אחר פגמים, ניהול דרישות. כל זאת יחד עם תמיכה בניתוח כמותי ואף דיווח על מושאי הבדיקה.

**כלי ניהול דרישות (requirements management)**- בכלים אלה מפורטת הדרישות מהתוכנה, וכן התכונות של הדרישות כוללות את סדר העדיפויות של כל דרישה. שכן, כלים אלה יכולים לסייע בזיהוי דרישות לא עקביות או חסרות.

**כלי ניהול אירועים (incident management)**- לרוב קרויים "כלי ניהול באגים". בכלים אלה שומרים ומנהלים דוחות על אירועים כגון: פגמים, כשלים, בקשות לשינוים וכו'. כלים אלה אף מסייעים בניהול "מחזור החיים" של אירועים ובחלקם קיימת תמיכה בניתוח סטטי.

**כלי ניהול תצורה (configuration management)**- אלה אינם כלי בדיקות בהגדרה. אבל הם הכרחיים לשם שמירה וניהול גרסאות של הבודקה והתוכנות הנלוות (במיוחד כאשר בודקים על יותר מסביבת תוכנה/חומרה אחת). ובאופן כללי, כלים אלה חיוניים לשמירת איכות התוכנה.

**כלים לתמיכה בבדיקות סטטיות:** הינם כלי בדיקות סטטיות המספקים שיטה יעילה מבחינה כלכלית לגילוי פגמים בשלבים המוקדמים של הפיתוח. כל זאת ע"י ניתוחים האפשריים עוד לפי הרצת הקוד כגון: ניתוח מסמכים, ניתוח קוד בטרם הרצתו וכו'.

**כלי סקירה( review):** כלים אלה תומכים בתהליכי הסקירה, רשימות ביקורת והנחיות סקירה. הם משמשים לשמירה והעברת הערות סקירה בין אנשי הצוות (ולדווח על פגמים). יש לציין, כי אפשר להיעזר בהם גם לתמיכה בסקירה מקוונת.

**כלי ניתוח סטטי (static analysis) "מ" –** כלים אלה מסייעים למפתחים (ולבודקים) למצוא פגמים לפני ביצוע בדיקות דינמיות (לפני הרצת הקוד). זאת ע"י תמיכה באכיפת תקנים לכתיבת קוד כמו למשל כתיבת קוד מאובטח או ניתוח מבנים ותלויות. כלים אלה עשויים לתרום לתכנון הבדיקות או לניתוח סיכונים בעזרת המדדים שהם מספקים ביחס לקוד.

**כלי מידול (modeling) "מ" –** גם הם בעיקר בשימוש המפתחים, הם משמשים לאימות מודלים של תוכנה (למשל, טבלאות קשרים של מסד-נתונים) וזאת ע"י גילוי פגמים וחוסר עקביות. כמו כן, כלים אלה עשויים לסייע ביצרת מקרי בדיקה המבוססים על המודל.

**כלי עיצוב בדיקות (test design)**- כלים אלה משמים לייצור: קלט בדיקות, קוד בדיקה שניתן להרצה ו/או אורקלים לבדיקות. זאת, מתוך הדרישות, הממשקים הגרפיים, מודלי העיצוב או מן הקוד.

**כלים להכנת נתוני בדיקה (test‑data preparation)**- כלים הלו מבצעים שינוים ועדכונים של מסד-נתונים, קבצים ותשדורת נתונים על-מנת להגדיר נתוני בדיקות שישמשו במהלך ביצוע הבדיקות. מאחר והנתונים המיוצרים בעזרת הכלי הם אנונימיים מובטח שלא נוצרות בעיות של אבטחת מידע.

**כלים לתמיכה בהרצת בדיקות וברישום:**

**כלי הרצת בדיקות (test execution)-** כלים אלה מאפשרי הרצה אוטומטית או חצי-אוטומטית של בדיקות. בד"כ כלים אלה מפיקים יומן בדיקות עבור כל הרצת בדיקה וניתן להשתמש בחלק מכלים אלה גם להקלטה של מהלך בדיקה (כדי להריצה שוב בעתיד). הדבר מאפשר אוטומטיזציה של בדיקות גם ללא כישורי פיתוח קוד. כמו כן, כלי הרצת בדיקות תומכים בשינוי פרמטרים של ערכי נתונים בבדיקות והתאמות אחרות של בדיקות. כל זאת באמצעות שפת תסריטים (סקריפט).

**מערכת/ רתמה (harness) לבדיקות יחידה "מ"-**  נמצאת בשימוש המפתחים האחראים על בדיקות יחיד לקוד שלהם. מערכת לבדיקות יחידה ו/או רתמת בדיקות מסייעת בבדיקת רכיבי מערכת או חלקים ממנה, ע"י סימולציה של סביבת ההרצה של מושא הבדיקה. הסימולציה מושגת באמצעות אובייקטים מדומים כגון: תותבים או דרייברים.

**כלי השוואה (test comparator)**- כלי ההשוואה משווים קבצים, מסדי נותנים או תוצאות בדיקות. כלי הרצת בדיקות בד"כ כוללים כלי השוואה דינמיים, אך השוואה לאחר ההרצה ניתנת לביצוע ע"י כלי השוואה נפרד. כמו כן, כלי להשוואת בדיקות עשוי להשתמש באורקל בדיקות במיוחד במקרה שהוא אוטומטי.

**כלים למדידת כיסוי (coverage) "מ"-**  כלים אלה מודדים את אחוז הקוד אשר נדבק ע"י סידרת בדיקות באמצעים פולשניים או בלתי פולשניים. כמו כן, האחזור מחושב ביחס לסוגי כיסוי שונים כגון: כיסוי משפטים, כיסוי הסתעפויות או החלטות, כיסוי תנאים וכן קריאת מודל או פונקציה. (נמצא בשימוש המפתחים)

**כלים לבדיקות אבטחה (security)**- כלים אלה משמשים להערכת מאפייני האבטחה של תוכנה. כולל הערכת יכול התוכנה להגן על: סודיות מידע, שלמות מידע, אימות זהות, הרשאות, זמינות וכו' (כלים אלו מתמקדים לרוב בהיבט התוכנה).

**כלים לתמיכה בביצועים ובמעקב**

**כלי ניתוח דינמיים (dynamic analysis) "מ" –** כלי ניתוח דינמיים מגלים פגמים שנחשפים רק שהתוכנה רצה, למשל: תלויות זמן, דליפות זיכרון ועוד. הכלים הללו בד"כ משמשים בבדיקות רכיבים וכן כאשר בודקים תוכנת ביניים.

**כלי דיבוג (ניפוי באגים, debugging ) "מ"**- משמשים את המפתחים למציאת הבאג שגרם לתקלה.

**כלי בדיקות ביצועים / עומסים / מאמץ :**

כלים לבדיקות ביצועים מנטרים ומדווחים על התנהגות המערכת בעת הדמיה של מגוון תנאי שימוש וזאת למשל במונחים של מספר המשתמשים בו זמנית, דפוסי עליה ברמת הפעילות וכן תדירות והאחוז היחסי של התנועות. כמו ההדמיה של העומס מתקבלת באמצעות יצרת משתמשים מדומים במבצעים סדרה נבחרת של תנועות ע"י מספר מחשבי בדיקה ("מחוללי עומסים").

**כלי לבדיקת ביצועים (performance testing tool)**- כלי שנועד לתמוך בבדיקות ביצועים, בדרך כלל יש לו שתי יכולות עיקריות. האחת היא יצרת עומס והשנייה היא מדידת תנועה. יצירת העומס יכולה לדמות ריבוי משתמשים או נפחים גדולים של נתוני קלט. כמו כן, כלי בדיקות ביצועים מספקים לרוב דוחות המבוססים על רישומי הבדיקה וגרפים של העומס כנגד התגובה.

**כלי בדיקות עומסים (load testing tool)-** כלי המיועד לתמוך בבדיקות עומסים. אשר יכול לדמות עומס גובר והולך כמו למשל מס' המשתמשים בו זמנית ו/או מספר התנועות במהלך פרק זמן מוגדר.

**כלי בדיקת מאמץ (stress testing tool)-** כלי התומך בבדיקות מאמץ, בדיקות המתבצעות כדי להעריך את התנהגות מערכת או רכיב כשהעומסים מגיעים לרמה המרבית הצפויה או שפורטה, או מעבר לרמה זו, או עם זמינות מופחתת של משאבים כגון גישה לזיכרון או לשרתים.

**כלי ניטור (monitoring)**- מוודאים שימוש במשאבי המערכת. כלומר מנתחים באופן רצוף את נתוני השימוש ומייצרים דוחות על נתונים אלה. בנוסף, הכלים גם מתריעים על תקלות אפשריות בשירותי המערכת.

**הערכת איכות הנתונים-** נתונים הם המוקד של פרויקטים מסוימים כמו למשל, פרויקטים להמרת נתונים או הגירתם, או מחסני נתונים. מאפייני נתונים אלו שונים בנפחם וברמת החשיבות שלהם. לכן, בהקשרים כאלה יש צורך בכלים להערכת איכות הנתונים, כלים שיכולים לסקור ולאמת את המרת הנתונים בהתאם לכללי ההגירה וכך להבטיח את שלמות ובטיחות הנתונים. יש לציין כי כלים אחרים קיימים לשם בדיקות שימושיות.

**שימוש אפקטיבי בכלים: תועלת וסיכונים**

תועלת וסיכונים אפשריים בכל הכלים לתמיכה בבדיקות. מיותר לציין, כי קניה של כלי לא מבטיחה הצלחה בשימוש בו. ובנוסף עבור כל סוג כלי נדרשת השקעת מאמץ נוסף.

**תועלת אפשרית משימוש בכלים:**

* צמצום המטלות המונוטוניות(חוזרות על עצמן)- למשל, הרצת בדיקות נסיגה, הקלדה חוזרת של אותם נתונים וכו'.
* עקביות וחזרתיות טובות יותר – למשל, בדיקות המבוצעות עם כלי באותו סדר ובאותה תדירות, בדיקות המופקות מן הדרישות
* הערכה נטולת פניות- למשל, מדידות סטטיות וכיסוי.
* גישה נוחה למידע אודות בדיקות או תהליך הבדיקה- למשל, סטטיסטיקות וגרפים של התקדמות בדיקות, שיעורי אירועים וביצועים.

**סיכונים אפשריים בשימוש בכלים:**

* ציפיות לא מציאותיות מן הכלי- כולל פונקציונליות ונוחות השימוש
* אומדן-חסר של משאבים – הנדרשים להכנסתו הראשונות של הכלי כגון: זמן, עלות ומאמץ ואף הכשרה ותלות במומחים חיצוניים.
* אומדן-חסר של הזמן והמאמץ הדרושים להשגת תועלת משמעותית ומתמשכת מן הכלי- כולל הצורך בשינויים בתהליך הבדיקה ובשיפור מתמיד של אופן השימוש בכלי.
* אומדן-חסר של המאמץ הנדרש לתחזק את מה שהכלי מיצר
* הסתמכות יתר על הכלי
* הזנחת בקרת הגרסאות של נכסי הבדיקה הקשורים לכלי
* הזנחת קישוריות ובעיות ביכולת חיבור ובפעולה הדדית בין כלים חיוניים- למשל בין, כלי ניהול דרישות, כלי בקרת גרסאות, כלי ניהול אירועים, כלי מעקב אחר פגמים ועוד.
* הסיכון שספק של כלי מסוים יעלם מהשוק
* היענות נמוכה של הספק לפניות
* בכלים חינמיים או כלים שמקורם הוא קוד-פתוח קיים סיכון של השעיה

**שיקולים מיוחדים עבור סוגי כלים מסוימים:**

כלי ניהול בדיקות- כלי בדיקות צריכים להתממשק עם כלים אחרים, וכן עם גיליונות אלקטרונים כדי שניתן יהיה להפיק מהם מידע שימושי. במיוחד חשובה היכולת להוציא ולקבל נתונים בפורמט אחיד ומקובל כמו: אקסל, וורד וכו'.

כלי ניתוח סטטי- ניתן להיעזר בכלי ניתוח סטטי על מנת לאכוף תקנים לכתיבת קוד. אולם אם מיישמים אותם על קוד קיים עלולים לקבל כמות גדולה מאוד של התראות. אומנם ההתראות אינן מונעות את תרגום הקוד לתוכנה הניתנת להרצה, אך מומלץ לטפל בהן על מנת להקל על תחזוקת הקוד בעתיד. גישה אפשרית היא יישום הדרגתי של כלי הניתוח -ע"י סינון חלק מההתראות.

כלי ביצוע בדיקות- כלי ביצוע בדיקות מריצים את מושאי הבדיקות באמצעות תסריטי בדיקות אוטומציה. אולם, סוג זה של כלי דורש לעיתים קרובות מאמץ ניכר על מנת להפיק ממנו תועלת משמעותית.

**לכידת בדיקות ומגבלותיה:** לכידת בדיקות ע"י הקלטת הפעולות של בודק ידי נראית "מושכת" אולם, גישה זו אינה מתאימה בפרויקטים שדורשים כמות גדולה של תסריטים אוטומטים. שכן, תסריט מוקלט חוזר על מקרה בדיקה יחיד ו"לינארי" (מסלול אחד בקוד, עם נתונים ופעולות מסוימים מאוד) סוג זה של תסריט עלול להיות בלתי יציב. זאת מכיוון, שבזמן הרצתו מתרחשים במערכת אירועים לא צפויים. ובנוסף הוא קשה לתחזוקה ומחייב שינויים תקופתיים.

**גישת בדיקות מונחית-נתונים (data driven testing)**: גישה זו מפרידה את קלט הבדיקות (הנתונים), ומשתמשת בתסריט בדיקות כללי יותר המסוגל לקרוא את נתוני הקלט ולבצע את אותו תסריט בדיקות עבור ערכי קלט שונים. כמו כן, בודקים אשר אינם מכירים את שפת התסריטים יכולים ליצור נתוני בדיקה עבור תסריטים מוגדרים מראש. טכניקה נוספת, היא ייצור נתונים בשעת ההרצה באמצעות אלגוריתם, כדי שהרצות חוזרות של הבדיקה ישתמשו באותם מספרי משתמש (לצורך חזרתיות).

**גישה מונחית מילות-מפתח (keyword‑driven testing)**- בגישה הגיליון האלקטרוני מכיל, מילות-מפתח המתארות פעולות שיש לנקוט בהן ונתוני בדיקה. כמו כן, הבודקים יכולים להגדיר בדיקות באמצעות מילות מפתח אותן ניתן להתאים ליישום הנבדק וזאת גם אם אינם מכירים שפת תסריטים (סקריפטים).

הערות: מומחיות טכנית בשפת תסריטים דרושה בכל הגישות הן מצד הבודקים והן מצד המומחים לאוטומציה. בנוסף, וללא קשר לטכניקת התסריטים בה משתמשים יש צורך לשמור את התוצאות הצפויות עבור כל בדיקה לצורך השוואה בזמן הרצה.

**הכנסת כלי לארגון**

**בין השיקולים העיקריים בבחירת כלי עבור ארגון ניתן למנות:**

* הערכת בשלות הארגון- נקודות חוזק וחולשה שלו וזיהוי הזדמנויות לשיפור התהליך.
* הערכת הכלי- מול דרישות ברורות וקריטריונים אובייקטיבים
* הערכת האפקטיביות של הכלי
* זיהוי השינויים הדרושים לשם שימוש אפקטיבי
* הערכה של הספק- היכולת לספק הכשרה ותמיכה
* זיהוי דרישות פנים ארגוניות להכשרה בשימוש הכלי- כמו למשל, אימון חונכות
* הערכת יחסי עלות-תועלת

**תכנית ניסוי (פיילוט) להכנסת כלי חדש:** הכנסת הכל לארגון מתחילה בתכנית ניסוי עם היעדים הבאים: למידת פרטים רבים יותר אודות הכל, הערכת מידת התאמתו של הכלי לנהוג בחברה ולתהליכים הקיימים, קבלת החלטה לגבי שיטות מוסכמות (שימוש, ניהול אחסון וכו') וניסיון להעריך אם התועלת הצפויה תושג בעלות סבירה.

**הגורמים להטמעה מוצלחת של כלי חדש:**

* פריסה הדרגתית של השימוש בכלי על פני שאר הארגון
* התאמת תהליכים ושיפורם על מנת שיתאימו לשימוש בכלי
* הכשרה, תרגול, אימון וחונכות למשתמשים חדשים.
* הגדרת הנחיות לשימוש בכלי
* איסוף מידע אודות השימוש בפועל
* ניטור השימוש בכלי והתועלת ממנו
* תמיכה בצוות הבדיקות המשתמשים בכלי
* הפקת לקחים מפעילות כל הצוותים