סיכום בניהול איכות תוכנה

**פרק 1: יסודות בדיקות תוכנה (K2)**

* מדוע הבדיקות נחוצות?
* בדיקות, מהן?
* שבעת עקרונות הבדיקה
* תהליך הבדיקה הבסיסי
* הפסיכולוגיה של בדיקות
* הקוד האתי

**1.1 מדוע הבדיקות נחוצות? (K2) מושגי מפתח**

* שגיאה (error)

*-גם: טעות (mistake)*

* פגם (defect)

*-גם: תקלה (fault)*

*-גם: באג bug))*

* כשל (failure)
* איכות (quality)
* סיכון (risk)

שגיאה (error)*-גם: טעות (mistake)*-פעולה אנושית המייצרת תוצאה שגוייה [בעקבות IEEE 610] ולפיכך גורמת ל...-פגם, תקלה, ובקיצור: באג

IEE: Institute of Electrical and Electronics Engineers-ארגון תקינה מוביל, בינלאומי (במקורו אמריקאי) תקן 610: מילון המונחים התקני להנדסת תוכנה

פגם (defect)*-גם: תקלה (fault)-גם: באג bug))*- ליקוי ברכיב או במערכת אשר עשוי לגרום להם להיכשל בביצוע התפקוד (function) הנדרש מהם;

ליקוי כזה יכול להיות למשל פקודה או הגדרת נתונים שאינן נכונות.

פגם, אם נתקלים בו תוך כדי ביצוע, עשוי לגרום לכשל (failure) של הרכיב או המערכת.

כשל (failure)-סטייה של רכיב או מערכת מהמסירה (delivery), מהשירות או מהתוצאות המצופים ממנו. [בעקבות Fenton]

איכות (quality)-המידה שבה רכיב, מערכת, או תהליך עומדים בדרישות שאופיינו ו/או בצרכים ובציפיות של משתמש/לקוח. [IEEE 610]

סיכון (risk)- גורם שתוצאותיו בעתיד עשויות להיות שליליות; לרוב מבוטא כהשפעה (impact) או כסבירות (likelihood)

**1.1.1 ההקשר: מערכות תוכנה**

התוכנה בכל מקום

שעון, מעלית, מכונית, מטוס, חללית...

דוגמאות לכשלי תוכנה בחיי יומיום

מחיר בכסף

מחיר בחיי אדם

**1.1.2 הגורמים לפגמים בתוכנה**

טעות לעולם חוזרת

הפגם יכול להיות בכל שרשרת הפיתוח

-דרישות לרמותיהן

-עיצוב

-כתיבת קוד

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.

* + **תנאי סביבה**

קרינה, מגנטיות, שדות אלקטרוניים , בציוד המשדר וקולט: רעשי רקע, חום (!), אבק, לחות

**1.1.4 בדיקות ואיכות**

דרישות

דרישות תפקודיות (פונקציונליות)

דרישות לא-תפקודיות

* אמינות (reliability), שימושיות (usability),   
  יעילות (efficiency), תחזוקתיות (maintainability),   
  ניידות (portability)

**דרישות לא-תפקודיות (לא פונקציונליות)**

**אמינות (reliability)**

היכולת של מוצר תוכנה לבצע את התפקידים (functions) הנדרשים,   
בתנאים נתונים ולאורך תקופת זמן שהוגדרה,   
או עבור פעולות (operations) שמספרן הוגדר. [ISO 9126]

**תקני ISO**

ISO: International Organization for Standardization-ארגון תקינה מוביל, בינלאומי (נוסד 1947) השם אינו ראשי תיבות... (מקורו במילה היוונית *isos*, שמשמעותה "שווה")

* + - * 1. תקן 9126: איכות מוצרי תוכנה – מאפיינים ותת-מאפיינים של איכות

ISO/IEC 9126-1:2001. Software Engineering – Software Product Quality –   
Part 1: Quality characteristics and sub-characteristics

פועל בשיתוף עם IEC (International Electrotechnical Commission)-הנציבות הבין-לאומית לאלקטרוטכניקה, המורכבת מנציגי מכוני תקנים לאומיים.

**שימושיות (usability)**

היכולת של התוכנה להיות מובנת, נלמדת, ברת-שימוש ומושכת עבור המשתמש, כאשר נעשה בה שימוש בתנאים מוגדרים. [ISO 9126]

**יעילות (efficiency)**

א) יכולתו של מוצר תוכנה לספק ביצועים מתאימים (appropriate), ביחס לכמות המשאבים בהם משתמשים בתנאים שהוגדרו. [ISO 9126]

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, התאמה לסביבה שהשתנתה. [ISO 9126]

**ניידות (portability)**

מידת הקלות שבה ניתן להעביר מוצר תוכנה מסביבה אחת של חומרה או תוכנה לסביבה אחרת. [ISO 9126]

**בדיקות ואיכות**

כשלא מוצאים פגמים...

בדיקות וסיכון

פגמים שמתקנים...

ופגמים שלא...

**1.1.5 שאלת מיליון הדולר (פשוטו כמשמעו): כמה לבדוק?...**

סיכון

היקף

אילוצים

מידע נדרש ודיווחו

לשחרר או לא? מי ממליץ ומי מחליט, ולמה

**אז לסיכום, מדוע הבדיקות נחוצות? (K2)**

כדי למצוא כשלים (failures) בתוכנה, כדי למצוא את סיבת הכשל, שהיא פגם (defect) בתוכנה-קרוי גם: תקלה (fault) ובקיצור: באג bug))...

לתקן את הפגם (או לא...)

כדי להבין את השגיאה (error)   
או הטעות (mistake) שהובילו לפגם, וכך למנוע פגמים דומים בעתיד ובנוסף: כדי להעריך את האיכות (quality) של התוכנה ולהעריך את הסיכון (risk) שהיא תיכשל ולא תפעל כמצופה.

**1.2 הבדיקות מהן? (K2) מושגי מפתח**

דרישות (requirements)

סקירה (review)

יעד הבדיקות (test objective)

בדיקות (testing)

מקרה בדיקה (test case)

ניפוי באגים (debugging)

**דרישות (requirements) (בעקבות IEEE 610)**

מצב או יכולת שהמשתמש זקוק להם כדי: לפתור בעיה, להשיג מטרה אשר חובה לעמוד בה או שנכפית על רכיב או מערכת כדי לעמוד ב...

חוזה, תקן (סטנדרט), אפיון, או כל מסמך פורמלי ומחייב אחר.

**סקירה (review)**

הערכת הסטטוס של מוצר או פרויקט כדי: לברר אם קיימים פערים מהתוצאות המתוכננות, להציע שיפורים.

דוגמאות:

סקירת הנהלה, סקירה לא רשמית, סקירה טכנית, ביקורת (inspection), דיון מודרך (walkthrough).

**בדיקות (testing)**

תהליך הכולל את כל פעילויות מחזור החיים סטטיות ודינמיות כאחת.

עוסק בתכנון, הכנה והערכה של מוצרי תוכנה ותוצרי עבודה הקשורים אליהם כדי לקבוע שהם עונים לדרישות שהוגדרו, להראות שהם מתאימים למטרה ו... למצוא פגמים

יעד הבדיקות (test objective)- סיבה או מטרה שלשמה מעצבים ומבצעים בדיקה.

**מקרה בדיקה (test case) [בעקבות IEEE 610]**

סט של: ערכי קלט, תנאי-קדם (preconditions) לביצוע, תוצאות צפויות, תנאים שלאחר הביצוע (postconditions)

שפותח למטרה או לתנאי בדיקה מסויימים, כגון ביצוע של נתיב מסויים בתכנית המחשב, או כדי לאמת עמידה בדרישה ספציפית.

**ניפוי באגים (debugging)**

התהליך של מציאת הסיבות לכשלים (failures) בתוכנה -כלומר מציאת הפגם (defect) = תקלה (fault) = באג (bug), שגרם לכשל.

ניתוח סיבות אלה והסרתן.

קרוי לרוב בשפה המדוברת דיבוג או דיבאג או דיבאגינג.

**תהליך הבדיקות**

לא רק הרצת בדיקות

מכלול פעילויות הבדיקה: תכנון ובקרה, בחירת תנאי הבדיקה, עיצוב וכתיבת מקרי בדיקה

ביצוע מקרי בדיקה (הרצת בדיקות), בחינת התוצאות, הערכת אמות המידה ליציאה (exit criteria)

דיווח על תהליך הבדיקה ועל המערכת הנבדקת, סיכום וסיום פעילויות סגירה לאחר ששלב הבדיקה (test phase) הושלם.

וגם: סקירת מסמכים (כולל קוד) וניתוח סטטי.

**הבדיקות מהן? יעדי הלימוד**

**יעדי הבדיקות (K1)**

* מציאת פגמים
* השגת אמון בהערכה של רמת האיכות
* השגת מידע הנחוץ לקבלת החלטות
* מניעת פגמים

**יעדי הבדיקות במחזור החיים**

* מניעה מוקדמת
* גילוי פגמים ברכיבים
* גילוי פגמים באינטגרציה ובמערכת
* בדיקות קבלה: המערכת עושה מה שצריך
* הערכת איכות
* בדיקות תחזוקה ותפעול

**בדיקות מול ניפוי באגים (debugging)**

בדיקות (דינמיות) מציגות כשלים הנגרמים על ידי... פגמים

ניפוי באגים (debugging) הוא מציאת הפגמים עצמם ותיקונם.

בדיקות חוזרות מאמתות התיקון ובקיצור: בודקים מבצעים בדיקות ומפתחים מבצעים דיבוג.

**1.3 שבעת עקרונות הבדיקה (K2)**

עקרון 1 – בדיקות מצביעות על נוכחותם של פגמים אבל לא מוכיחות שאין עוד פגמים...

עקרון 2 – בלתי אפשרי לבצע בדיקות ממצות ולכן חשוב לדעת מה לבדוק, לפי סיכונים ותיעדוף.

עקרון 3 – בדיקות מוקדמות (early testing) - כל המקדים הרי זה משובח..

עקרון 4 – התקבצות פגמים (defect clustering) ואם כבר, אז שאר חוקי הבאגיקה...

חוק רציפות הבאגים

חוק עומק הבאגים

חוק תחיית הבאגים

חוק נקמת הבאגים

עקרון 5 – פרדוקס ההדברה (pesticide paradox) מה תעדיפו – ג'וקים או נמלים?...

-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)-החיים זה לא משחק (או שכן...)

עקרון 7 – אשליית העדר שגיאות (absence-of-errors fallacy)-אין טעם לגלות פגמים ולתקנם אם המערכת הנבנית אינה שמישה ואינה מספקת את צרכי המשתמשים ואת הציפיות ממנה.

זהו מצב בו אמנם לא נותרו פגמים ידועים בעצם פעולת המערכת, אך היא אינה מספקת את צרכי המשתמשים, הלקוחות ו/או בעלי עניין אחרים ואת ציפיותיהם ממנה, ובפועל אינה שמישה כמצופה.

**1.4 תהליכי הבדיקה היסודיים**

* + מושגים ורקע
  + תכנון הבדיקות ובקרתן (K1)
  + ניתוח הבדיקות ועיצובן
  + יישום הבדיקות וביצוען (K1)
  + הערכת אמות מידה ליציאה, ודיווח
  + פעילויות סגירת בדיקה (K1)

**תהליכי הבדיקה היסודיים: מושגים א-יב**

* מדיניות בדיקות (test policy)-מסמך ברובד גבוה (high level) המתאר את:

העקרונות, הגישה, והמטרות העיקריות של הארגון בנוגע לבדיקות.

בסיס בדיקות (test basis) [בעקבות Tmap]-כל המסמכים שמתוכם ניתן להסיק את הדרישות של רכיב או מערכת. התיעוד עליו מקרי בדיקה מבוססים.

בסיס בדיקות מוקפא: כאשר מסמך יכול להשתנות רק דרך תהליך שינוי פורמלי.

* תכנית בדיקות (test plan)
  + 1. מסמך המתאר את ההיקף, הגישה, המשאבים ולוח הזמנים של פעילויות בדיקה שיש כוונה לעשותן. המסמך מגדיר בין היתר: פריטי בדיקה (test items), התכונות (features) שיש לבדקן משימות הבדיקה ומי יבצע כל משימה, מידת העצמאות של הבודק, סביבת הבדיקות, טכניקת עיצוב הבדיקה, אמות מידה (קריטריה) שישמשו לכניסה וליציאה והסיבה לבחירתן, כל סיכון הדורש הכנת תכנית מגירה (contingency planning) לטיפול בו.

למעשה זהו תיעוד של תהליך תכנון הבדיקות. [בעקבות IEEE 829]

מושא בדיקות (test object)

הרכיב או המערכת שיש לבדקם.

פריט בדיקה (test item)-האלמנט המסויים שיש לבדקו.

לרוב יש מושא בדיקות אחד, והרבה פריטי בדיקות.

IEEE 829-

* תקן לתיעוד בדיקות תוכנה

[IEEE 829] IEEE 829:1998. Standard for Software Test Documentation

בּוֹדְקָה, מכלול מרכיבי הבדיקות (testware) [בעקבות Fewster and Graham]

* + 1. מרכיבים אשר נוצרים תוך כדי תהליך הבדיקות ונדרשים לשם תכנון, עיצוב וביצוע בדיקות, כגון:

תיעוד, תסריטים (scripts), קלט (inputs), תוצאות צפויות, נהלים (פרוצדורות) של הקמת סביבת הבדיקות (set-up) ושל פירוקה (clear-up),קבצים, מסדי נתונים, סביבה, כל תוכנה או עזרים המשמשים בבדיקות.

תנאי בדיקה (test condition)

(התנאי לבדיקה הוא שיהיה מה לבדוק... מכאן ההגדרה)

פריט או אירוע של רכיב או מערכת שעשוי להיות מאומת על ידי מקרה בדיקה אחד או יותר, למשל:

פונקציה, תנועה (transaction), תכונה (feature), מאפיין איכות, אלמנט מבני.

נתוני בדיקות (test data)

נתונים אשר קיימים (למשל, במסד נתונים) לפני שהבדיקה מתבצעת ואשר משפיעים או מושפעים עלי ידי הרכיב או המערכת הנבדקים.

נוהל בדיקות (test procedure) [בעקבות IEEE 829]-רצף של פעולות לשם ביצוע בדיקות.

ידוע גם בתור תסריט בדיקות (test script) או תסריט בדיקות ידני.

סדרת בדיקות (test suite)

סט של מקרי בדיקה רבים עבור רכיב או מערכת שנבדקים תנַאי-אַחַר (post-condition) של בדיקה אחת משמש תכופות בתור תנאי-קדם (pre-condition) של הבדיקה הבאה.

כיסוי (coverage), *גם: כיסוי בדיקות (test coverage)* -המידה, המבוטאת באחוזים, בה פריט מסויים כוסה על ידי סדרת בדיקות.

שימושים עיקריים, שחשוב להבחין ביניהם: כיסוי קוד, כיסוי דרישות ע"י בדיקות, כיסוי דרישות ע"י קוד.

אמות מידה ליציאה (exit criteria) [בעקבות Gilb and Graham]

סט של תנאים כלליים (generic) ו/או ספציפיים: אשר נדרשים כדי לאשר רשמית את סיומו של תהליך

אשר מוסכמים עם בעלי העניין (stakeholders).

מטרת אמות המידה ליציאה היא למנוע מצב בו: משימה תיחשב כאילו הושלמה כאשר ישנם עדיין חלקים משמעותיים שלה שלא הסתיימו.

(וכן למנוע מצב בו ממשיכים במשימה לנצח, כי תמיד אפשר לעשות עוד)

אמות מידה ליציאה משמשות לשם: דיווח מולן, כדי לתכנן מתי לעצור את הבדיקות.

ביצוע בדיקות (test execution)

תהליך של הרצת בדיקה/ות על הרכיב או המערכת הנבדקים המייצר תוצאה/ות בפועל (actual result(s)).

רישום בדיקות (test log) [IEEE 829]-רישום כרונולוגי של פרטים רלוונטיים בנוגע לביצוע הבדיקות.

אירוע, תקרית (incident) [בעקבות IEEE 1008]-כל אירוע (event) המתרחש ואשר דורש חקירה.

IEEE 1008-תקן לבדיקות יחידה (Unit Testing)

[IEEE 1008] IEEE 1008:1993. Standard for Software Unit Testing

בדיקות יחידה (Component Testing = Unit Testing)-בדיקה של רכיבי תוכנה פרטיים (אינדיבידואליים). [בעקבות IEEE 610]

בדיקות חוזרות (re-testing) קרויות גם: *בדיקות אישור (confirmation 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 כבר נבדקה קודם לכן בעקבות שינויים שעברה כדי לוודא שכתוצאה מאותם שינויים פגמים לא הוכנסו או התגלו באזורי התוכנה אשר... **לא השתנו (!)**

מתבצעות לאחר שהתוכנה או סביבתה עברו שינוי.

דו"ח סיכום בדיקות (test summary report) [בעקבות IEEE 829]

מסמך המסכם פעילויות בדיקה ותוצאותיהן.

כולל גם הערכה של פריטי הבדיקה הקשורים לכך מול אמות המידה ליציאה (exit criteria)

**תהליכי הבדיקה היסודיים: רקע**

1. תכנון ובקרה

* בקרה מתחילה כאן אבל נמשכת לאורך התהליך

1. ניתוח ועיצוב
2. יישום וביצוע
3. הערכת אמות המידה ליציאה ודיווח
4. פעילויות סגירת הבדיקה

**1.4.1 תכנון הבדיקות ובקרתן**

* תכנון בדיקות (test planning): פעילות הגדרת יעדי הבדיקה ומפרט פעילויות הבדיקה הנחוצות לעמידה ביעדים
* בקרת בדיקות (test control): פעילות מתמשכת
  + 1. משווים התקדמות בפועל אל התוכנית
    2. מדווחים כולל חריגות
    3. מנטרים פעולות בדיקה
    4. מבצעים צעדי תיקון
* פעילויות תכנון ובקרה מוגדרות בפירוט בפרק 5

**1.4.2 ניתוח הבדיקות ועיצובן**

1. סקירת בסיס הבדיקות, לרבות:

- דרישות, רמת תקינות התוכנה (software integrity level)- רמת התאימות של התוכנה לקבוצת תכונות, שנבחרו על ידי בעלי העניין (stakeholders)(למשל: מורכבות המערכת, הערכת סיכונים, רמת בטיחות, רמת אבטחה, ביצועים רצויים, אמינות, או עלות) מוגדרות באופן המשקף את חשיבות התוכנה לבעלי העניין בה.

רמת הסיכון, דוחות ניתוח סיכונים (risk analysis), ארכיטקטורה, עיצוב, דרישות 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, מקרי הבדיקה.

6. עיצוב הגדרות סביבת הבדיקה וזיהוי התשתיות והכלים הנדרשים.

7. יצירת נֶעֱקָבוּת דו-כיוונית   
(bi-directional traceability)   
בין בסיס הבדיקות לבין מקרי הבדיקה

מכיוון הדרישות:

דרישה א -> בדיקה 1, 2

דרישה ב -> TBD

דרישה ג -> בדיקה 2, 4

מכיוון הבדיקות:

בדיקה 1 <- דרישה א

בדיקה 2 <- דרישה א, ג

בדיקה 3 <- מיותרת?

בדיקה 4 <- דרישה ג

**1.4.3 יישום הבדיקות וביצוען**

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

1. סיום הגדרת מקרי הבדיקה, יישומם ותיעדופם (כולל זיהוי נתוני הבדיקות)

2. פיתוח ותיעדוף נהלי בדיקות תוך יצירת נתוני בדיקות כשהבדיקות מיועדות לאוטומציה:  
הכנת רתמת בדיקות (test harness) ותסריטים אוטומטיים, יצירת סדרות בדיקה מנהלי הבדיקות לשם ביצוע יעיל.

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 כלי הבדיקה, הבּוֹדְקָה (מכלול מרכיבי הבדיקה).

6. השוואת התוצאות בפועל לתוצאות הצפויות.

7. דיווח על חריגות (discrepancies) בתור אירועים (incidents)וניתוחן במטרה להגדיר את הגורם לחריגה. למשל: פגם בקוד, פגם בנתוני הבדיקות, פגם במסמך הבדיקה, טעות באופן ביצוע הבדיקה.

8. חזרה על פעילות בדיקה כתוצאה מפעולה שננקטה עבור כל חריגה, למשל:

ביצוע חוזר של בדיקה שנכשלה במטרה לאשר את התיקון (בדיקות אישור), ביצוע בדיקה מתוקנת (אם רלוונטי), ביצוע בדיקות נסיגה במטרה להבטיח ש...לא הוחדרו פגמים באזורים בקוד שלא השתנו, תיקון הפגם לא חשף פגמים אחרים.

**1.4.4 הערכת אמות מידה ליציאה (**exit criteria**), ודיווח**

1. השוואת תוצאות הבדיקה לעומת אמות המידה ליציאה שהוגדרו בתוכניות הבדיקה.

2. ביצוע הערכה של הצורך בבדיקות נוספות או בשינוי אמות המידה ליציאה...

3. כתיבת דו"ח סיכום בדיקה (test summary report)עבור בעלי העניין (stakeholders).

**1.4.5 פעילויות סגירת בדיקה**

פעילויות סגירת בדיקה אוספות מידע מפעילויות בדיקה שהושלמו, ומאחדות יחד את הניסיון שנצבר

מכלול מרכיבי הבדיקה (הבּוֹדְקָה)העובדות והמספרים.

מתבצעות בנקודות אבן הדרך של פרויקט (milestones). למשל: אלפא, בטא, שחרור ללקוח...

השלמת גרסת תחזוקה (maintenance release)

כאשר משלימים פרויקט בדיקה...או כאשר מבטלים אותו.

1. בדיקה: אילו תוצרים מתוכננים אכן נמסרו

2. סגירת דוחות אירועים (incident reports) ורישום דוחות שנשארו פתוחים לשם תיעוד, תיקון עתידי, בקשות לשינויים.

3. תיעוד קבלת המערכת (acceptance)

4. ארגון מכלול פריטי הבדיקה, סביבת הבדיקה ותשתית הבדיקה וגניזתם לשימוש חוזר בעתיד.

5. הפקת לקחים במטרה להגדיר את השינויים הדרושים בגרסאות ובפרויקטים עתידיים.

6. שימוש במידע שנאסף לשיפור בשלות הבדיקות (test maturity)

**1.5 הפסיכולוגיה של בדיקות (K2) – מושגים**

ניחוש שגיאות (error guessing)-טכניקת עיצוב בדיקות שבה נעשה שימוש בניסיון של הבודק,

כדי לצפות אלו פגמים עשויים להתקיים ברכיב או במערכת הנתונים לבדיקה(כתוצאה מטעויות שנעשו)ולעצב בדיקות ספציפיות כדי לחשוף אותם.

עצמאות הבדיקות (independence of testing) [בעקבות DO-178b]

-גם: *אי-תלות (independence)*, הפרדת תחומי האחריות...

באופן המעודד השגת בדיקות אובייקטיביות.

תקן DO-178B (1992): שיקולי תוכנה בהסמכת מערכות וציוד טיסה.

מסמך התקינה המחייב באיכות תוכנה בתחום הטיסה.

**1.5 הפסיכולוגיה של בדיקות**

דפוסי מחשבה שונים למפתח ולבודק, אופטימיות מול פסימיות, מהירות והספק מול איכות, שיקולים טכניים בקוד מול שיקולי נוחות ויעילות למשתמש, התמקדות 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 בדיקה מתוך החברה.

תלוי, בלתי תלוי, צוות בדיקות מחוץ לחברה

**1.5 הפסיכולוגיה של בדיקות**

קביעת יעדי בדיקות ברורים, היחס לבודקים, יחסי אנוש, תקשורת חיובית.

**1.5 הפסיכולוגיה של בדיקות: קווים מנחים**

מטרה משותפת ושיתוף פעולה, לא עימות.

דיווח ניטראלי בלי טון ביקורתי, רצוי להדגיש תחילה את הצדדים החיוביים.

אמפתיה -לשים עצמך בנעלי הצד השני, מה היית אתה מרגיש לו קיבלת ביקורת מעין זו?

מה היה מקל עליך לקבל הביקורת?

תקשורת ברורה, וידוא הבנה- לוודא שהצד השני הבין את מה שנאמר, וכן להיפך.

סיכום כתוב (לאחר שיחה והסכמות!) עוזר מאד.

**1.6 הקוד האתי (K2)**

* *המקרה של Therac-25*
* לפחות 3 נפטרים
* עשרות נפגעים
* לאורך שנתיים (!) 1985 - 1987

(כל התביעות הוסדרו מחוץ לבית המשפט)

* עלות לא ידועה

**1.6 הקוד האתי**

* + דוגמא: בניית צינור להובלת דם
  + מעורבות בבדיקות תוכנה מאפשרת לאנשים גישה למידע חסוי.
  + למערכות תוכנה השלכות משמעותיות על הציבור, טובתו ואף חייו.
  + קוד אתי הוא הכרחי
    1. בין השאר על מנת להבטיח שהמידע לא ישמש לצרכים בלתי ראויים
    2. או שהסתרתו תפגע בטובת הציבור, הלקוח והמעסיק של הבודק

**1.6 הקוד האתי של ISTQB**

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 שבאפשר

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.

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

ALM: Application Life-Cycle Management

2.1 מודלים לפיתוח תוכנה

2.2 רמות בדיקה

2.3 סוגי בדיקות

2.4 בדיקות תחזוקה (Maintenance Testing)

**2.1 מודלים לפיתוח תוכנה**

* 1. רקע ומושגים
  2. מודל V (מודל פיתוח סדרתי) (K2)
  3. פיתוח בסבבים (K2)
  4. בדיקות בתוך מודל מחזור החיים (K2)
  5. רקע: בדיקות אינן מתקיימות בסביבה מבודדת
     1. פעילויות הבדיקה קשורות לפעילויות פיתוח התוכנה יש להתאים את גישת הבדיקה למודל מחזור הפיתוח.

**מודלים לפיתוח תוכנה : אימות מול תיקוף**

קיים בלבול בין אימות (verification) מול תיקוף ((validation:

אימות מוודא שעמדנו בכל הדרישות עשינו נכון, בדקנו נכון.

תיקוף מוודא שהדרישות עונות על הצרכים והציפיות של הלקוח עשינו את הדבר הנכון, בדקנו את הדבר נכון.

אימות ותיקוף (V&V): מוודא שעשינו נכון את הדבר הנכון, במילה תיקוף לא משתמשים לרוב, אלא אומרים "ולידציה".

**מודלים לפיתוח תוכנה : אימות**

אימות (verification):

מוודא שעמדנו בכל הדרישות, עשינו נכון, בדקנו נכון.

הגדרה: אישור על ידי בחינה ובדיקה ודרך הבאת ראיות אובייקטיביות לכך שמולאו דרישות ספציפיות. [ISO 9000]

**מודלים לפיתוח תוכנה : תיקוף ((validation**

תיקוף מוודא שהדרישות עונות על הצרכים והציפיות של הלקוח, עשינו את הדבר הנכון, בדקנו את הדבר נכון.

הגדרה: אישור על ידי בחינה ובדיקה ודרך הבאת ראיות אובייקטיביות לכך שמולאו הדרישות לשם שימוש או יישום מיועדים וספציפיים.   
ISO 9000]]

**2.1.1 מודל V (מודל פיתוח סדרתי)**

הגרסה המצויה של המודל בנויה על ארבע רמות בדיקה חופפות לארבעת שלבי הפיתוח

1. בדיקות רכיבים (component testing)
2. בדיקות אינטגרציה (integration testing)
3. בדיקות מערכת (system testing)
4. בדיקות קבלה (acceptance testing)

**מודלים לפיתוח תוכנה : הגדרת מודל V**

מודל V (V-model)מסגרת לתיאור פעילויות מחזור החיים של פיתוח תוכנה:

מאיפיון הדרישותrequirements specification) ) (דרך כתיבתן בקוד, הבדיקות ברמות השונות)

ועד לתחזוקה.

מודל V מדגים כיצד פעילויות בדיקה ניתנות לשילוב בתוך כל שלב של מחזור חיי פיתוח תוכנה.

**2.1.2 פיתוח בסבבים**

פיתוח מחזורי בסבבים מצטברים(iterative-incremental development)

הגדרת הדרישות, עיצוב, בניה ובדיקת המערכת בסדרת מחזורי פיתוח קצרים

למשל: בניית אב-טיפוס (prototyping),פיתוח יישומים מהיר – RAD (Rapid Application Development) ומודלים לפיתוח זריז (agile) [יורחב בהמשך].

**מודלים לפיתוח תוכנה : מושגים**

מודל מחזור חיים מבוסס סבבים מצטברים (iterative‑incremental development model).

מחזור חיי פיתוח שבו פרויקט נחלק לרוב למספר גדול של איטרציות (סבבים).

איטרציה (חִיזְרוּר, סבב) היא סבב פיתוח מלא שתוצאתו שחרור גירסה (release; פנימי או חיצוני) של מוצר בר-הרצה.

מדובר בגירסה חלקית של המוצר הסופי שנמצא בפיתוח אשר גדלה מאיטרציה לאיטרציה עד שהיא הופכת למוצר הסופי עצמו.

**2.1.2 פיתוח בסבבים (ב)**

המערכת ניתנת לבדיקה במספר רמות במהלך כל סבב (iteration), כל תוספת יוצרת מערכת חלקית הולכת וגדלה, שגם אותה יש לבדוק, חשיבות בדיקות הנסיגה גוברת עם כל סבב נוסף, ניתן לבצע אימות ותיקוף אחרי כל תוספת.

**תוספת: מודלים לפיתוח זריז (agile)**

פיתוח זריז, אג'ילי, אג'ייל (agile), שיטה שנהיית נפוצה יותר ויותר, בסילבוס ובבחינה כמעט אינה נזכרת, אבל חשוב להכיר, המונח אג'ייל כולל בעצם קשת רחבה של שיטות:

בדיקות אג'ייל במובן המקורי של המונח (הסבר), מעטות יחסית, בדיקות שהן בעצם מודל V המפותח בסבבים ,רוב המקרים, במגוון אפשרויות.

**2.1.3. בדיקות בתוך מודל מחזור החיים (K2)**

ייתכנו מודלים שונים. מאפיינים לרמת בדיקות טובה:

לכל פעילות פיתוח יש פעילות בדיקה מקבילה, לכל רמת בדיקה ישנם יעדים הייחודיים לרמה זו

עבור כל רמת בדיקה ,ניתוח הבדיקות ועיצובן מתחילים במהלך פעילות הפיתוח המקבילה,

הבודקים מעורבים בסקירת מסמכים מייד עם הופעת הטיוטות הראשונות.

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

דוגמא: אינטגרציה של מוצר מדף עם מערכת שבפיתוח, אינטגרציה של מערכות קיימות, "מערכת של מערכות" (system of systems).

**2.2 רמות בדיקה**

יש להבדיל ולא להתבלבל בין רמות בדיקה לסוגי בדיקה!

רמות בדיקה הן השלב בבדיקות (יחידה > אינטגרציה > מערכת > קבלה)

סוגי בדיקה מתייחסים ליעד הבדיקה (תפקוד, ביצועים...) או לטכניקה

* 1. בדיקות רכיבים (בדיקות יחידה)
  2. בדיקות אינטגרציה
  3. בדיקות מערכת
  4. בדיקות קבלה.

**מושגים**

* רמת בדיקה (test level)
  + 1. קבוצה של פעולות בדיקה אשר מאורגנות ומנוהלות יחד, רמת בדיקה קשורה לחלוקת תחומי האחריות (responsibilities) בפרויקט.
    2. דוגמאות לרמות בדיקה הן: בדיקות יחידה, בדיקות אינטגרציה, בדיקות מערכת ,בדיקות קבלה.
       - 1. [בעקבות Tmap]

**2.2 רמות בדיקה: רקע**

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.

**2.2.1 בדיקות רכיבים (component testing) (K2)**

נקראות גם (בעצם לרוב) בדיקות יחידה (unit testing),

בדיקה של רכיבי תוכנה פרטיים (אינדיבידואליים). [בעקבות IEEE 610]

תותב (stub): שלד או יישום-למטרה-מסויימת של רכיב תוכנה המשמש לפיתוח או בדיקת רכיב אחר אשר קורא לו (לתותב) או תלוי בו באופן אחר.

התותב מחליף את הרכיב אליו קוראים. [After IEEE 610]

דרייבר (driver) [הגדרה מפורטת יותר בהמשך]

בשימוש המקובל: מנהל התקן ,בהקשר של רתמת בדיקות: עזר אשר מחליף רכיב שאינו קיים ואשר קורא לרכיב תוכנה שבבדיקה.

רתמת הבדיקותtest harness) ): סביבת בדיקות המורכבת מדרייברים ומתותבים (stubs) הדרושים לביצוע של בדיקה.

הערה לנוסח העברי: הדרייברים והתותבים מדמים חלקים של הסביבה הדרושים לבדיקה ושאינם זמינים, למשל כי טרם פותחו.

**בסיס הבדיקות ומושאי הבדיקה**

* + בסיס הבדיקות
    1. דרישות הרכיבים
    2. עיצוב מפורט
    3. קוד
  + מושאי בדיקה אופייניים
    1. רכיבים
    2. תוכנות
    3. תוכנות להמרה או הגירה של נתונים   
       (migration programs / data conversion)
    4. מודולים של מסדי נתונים |

**מהות בדיקות הרכיבים**

מחפשים פגמים ב: מודולי תוכנה, תוכנות, עצמים, מחלקות וכדומה אשר ניתנים לבדיקה באופן נפרד

ומאמתים את תפקודם.

ניתנות לביצוע במבודד מהמערכת בעזרת: רתמת בדיקות הכוללת תותבים ודרייברים (drivers), סימולטור.

**בדיקות רכיבים לדוגמא**

* התנהגות המשאבים (resource behavior)
  + 1. כגון חיפוש דליפות זיכרון (memory leaks)
* בדיקות חוסן (robustness)
* בדיקות מבניות (structural testing)
  + 1. למשל: כיסוי החלטות (decision coverage)
* מקרי הבדיקה נכתבים, למשל, לפי:
  + 1. מפרט של רכיב
    2. עיצוב התוכנה
    3. מודל הנתונים.

**עוד על בדיקות רכיבים**

* קיימת גישה לקוד הנבדק
* סביבת פיתוח
  + 1. כולל דיבאגר - כלי לניפוי באגים (debugging tool)
    2. או אף מערכת לבדיקות יחידה (unit test framework)
* לרוב, המתכנת שכתב את הקוד הוא המבצע של בדיקות הרכיבים.
* לרוב, פגמים מתוקנים מיד עם זיהוים ללא ניהול פורמאלי של הבאגים.

**פיתוח מובל בדיקות (test driven development)**

* קרוי גם "בדיקות תחילה" (test-first approach)
* גישה זו היא מחזורית (iterative) ומבוססת-סבבים.
  + 1. בכל סבב (או מחזור) יש: פיתוח מקרי בדיקה, בניה ואינטגרציה של קטעי קוד קצרים, ביצוע בדיקות רכיבים תוך תיקון בעיות, חזרה על התהליך עד שהקוד עובר את הבדיקה.

**עוד מושגים**

* דרייבר (driver)

רכיב תוכנה או כלי בדיקות אשר מחליף רכיב שדואג לבקרה ו/או לקריאה של רכיב או מערכת. [בעקבות Tmap]

כלומר, הדרייבר קורא לרכיב, והרכיב קורא לתותב.

הערה חשובה: זוהי ההגדרה בהקשר של בדיקות בלבד!

בדרך כלל הכוונה פשוט למנהל התקן, המאפשר לתוכנית מחשב (לרוב מערכת ההפעלה) לתקשר עם חומרה כלשהי.

* סביבת בדיקה (test environment)

סביבה המכילה חומרה, מכשור (instrumentation), מדמים (סימולטורים), כלי תוכנה, ואלמנטים תומכים נוספים הדרושים להרצת הבדיקה.

[בעקבות IEEE 610]

* 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. [IEEE 610]

ר' גם סובלנות לליקויים, סובלנות לשגיאות.

* דרישה פונקציונלית (תפקודית, functional requirement)

דרישה שמפרטת פונקציונליות (תפקוד) שרכיב או מערכת חייבים לבצע. [IEEE 610]

* בדיקות פונקציונליות (תפקודיות, functional requirements)

בדיקות המבוססות על ניתוח מפרט הפונקציונליות (תפקודיות) של רכיב או מערכת.

ר' גם *בדיקות קופסה שחורה*

* דרישה לא-פונקציונלית/תפקודית (non functional requirements)

דרישה שאינה קשורה לפונקציונליות (תפקודיות), אלא לתכונות כמו אמינות, יעילות, שימושיות, תחזוקתיות וניידות.

**2.2.2 בדיקות אינטגרציה**

* אינטגרציה (integration)

התהליך של שילוב רכיבים או מערכות לתוך מכלולים גדולים יותר.

בדיקות אינטגרציה (integration testing)בודקות ממשקים בין רכיבים ומערכות, יחסי גומלין בין חלקיה השונים של מערכת, למשל: מערכת ההפעלה, מערכת הקבצים והחומרה.

**בסיס הבדיקות**

* 1. עיצוב התוכנה והמערכת
  2. ארכיטקטורה
  3. סדר פעולות (workflow)
  4. מקרי שימוש (use cases).

**מושאי בדיקה אופייניים**

* + תת-מערכות
  + יישום מסדי נתונים
  + תשתית (infrastructure)
  + ממשקים (interfaces)
  + תצורת מערכת (system configuration)
  + נתוני תצורה (configuration data).

**דוגמאות לבדיקות אינטגרציה**

(תיתכן יותר מרמה אחת של בדיקות אינטגרציה, בסדרי גודל שונים)

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 (business processes)-עשויים לכלול סדרה של מערכות, תיתכנה בעיות חוצות מערכת (cross platform) משמעותיות.

**שיטות אינטגרציה**

אינטגרציה "במכה אחת" (big bang)

ככל שהיקף האינטגרציה רחב, קשה יותר לבודד פגמים, זהו סיכון מוגבר ונדרש זמן רב יותר לאיתור תקלות (troubleshooting).

אינטגרציה בשלבים (incremental)-הבודקים מתרכזים באינטגרציה עצמה למשל, בביצוע אינטגרציה של מודול א' עם מודול ב', תיבדק התקשורת בין המודולים ולא התפקוד של כל מודול בפני עצמו,

אם בדיקות אלה אכן התבצעו כבר במהלך בדיקות הרכיבים.

בדיקת מאפיינים תפקודיים ולא-תפקודיים ספציפיים (למשל, ביצועים).

**2.2.3 בדיקות מערכת (system testing)**

בדיקות מערכת (system testing)-התהליך של בדיקת מערכת שעברה כבר אינטגרציה כדי לאמת שהוא עונה לדרישות שפורטו. [בעקבות Hetzel]

בדיקות המערכת נוגעות להתנהגות המערכת או המוצר כמכלול.

היקף הבדיקות מוגדר במסמכים: תכנית-האב לבדיקות (master test plan)או תכנית הרמה (level test plan).

**בסיס הבדיקות ומושאי בדיקה לדוגמא**

בסיס הבדיקות (מסמכים)

* + 1. מיפרט הדרישות (requirement specification) של המערכת ושל התוכנה
    2. מיפרט תפקודי (functional specification)
    3. דוחות ניתוח סיכונים (risk analysis reports)
* מושאי בדיקה אופייניים
  + 1. המערכת עצמה, תיעוד הנלווה למערכת: מדריכי המערכת, המשתמש וההפעלה, תצורת המערכת ונתוני תצורה.

**עוד על בדיקות מערכת**

* על סביבת הבדיקה להתאים ככל הניתן לסביבה הסופית של המערכת.
* הבדיקות יכולות להתבסס על:
  + 1. סיכונים, מפרטי דרישות, תהליכים עסקיים, מקרי שימוש , תיאורים מפורטים אחרים (תיאור טקסטואלי או מודל של התנהגות המערכת), אינטראקציות עם מערכת ההפעלה, משאבי המערכת system resources)).

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 {מצב נפוץ מאד!} בתור התחלה, בדיקות מערכת של דרישות תפקודיות עושות שימוש בטכניקות מבוססות-המיפרט (קופסה שחורה, black box) המתאימות ביותר להיבט המערכתי הנבדק.

למשל, יצירת טבלת החלטה (decision table) עבור צירופים של תופעות המתוארות בכללים העסקיים (business rules).

לאחר מכן אפשר להפעיל טכניקות מבוססות מבנה (קופסה לבנה, white box) על מנת להעריך את מידת היסודיות של הבדיקות ביחס לאלמנט מבני, כגון: מבנה תפריט או ניווט בדף אינטרנט (ראו פרק 4)במקרים רבים צוות בדיקות עצמאי הוא המבצע בדיקות מערכת.

**2.2.4 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, וכדי לאפשר למשתמש, ללקוח או לישות מוסמכת אחרת לקבוע אם לקבל את המערכת או לא [בעקבות IEEE 610].

במקרים רבים בדיקות קבלה הן באחריות הלקוח או המשתמשים במערכת, אם כי בעלי עניין אחרים עשויים להיות מעורבים אף הם.

**בסיס הבדיקות**

1. דרישות המשתמש
2. דרישות המערכת
3. מקרי שימוש
4. תהליכים עסקיים
5. דוחות ניתוח סיכונים.

**מושאי בדיקה אופייניים**

1. תהליכים עסקיים על גבי מערכת שעברה אינטגרציה מלאה
2. תהליכים תפעוליים ותחזוקתיים
3. נוהלי משתמש (user procedure)
4. טפסים
5. דוחות
6. נתוני תצורה.

**מטרת בדיקות הקבלה**

לבסס בטחון במערכת, בחלקים ממנה או במאפיינים לא-תפקודיים (non-functional) מסוימים שלה.

מציאת פגמים איננה עומדת במוקד בדיקות הקבלה.

בדיקות קבלה יכולות להיות כלי להערכת מוכנות המערכת לפריסה בשטח ולשימוש, אינן בהכרח מהוות את רמת הבדיקות הסופית.

למשל, בדיקת אינטגרציה רחבת היקף של מערכת עשויה להתבצע לאחר ביצוע בדיקות הקבלה.

**מועד בדיקות הקבלה**

בדיקות קבלה עשויות להתבצע בזמנים שונים במחזור החיים, למשל: בדיקות קבלה של שימושיות רכיב מסוים יתבצעו במהלך בדיקות הרכיב, בדיקות קבלה של שיפור תפקודי חדש יתבצעו לפני בדיקות המערכת.

מוצר-מדף מסחרי (COTS)יעבור בדיקות קבלה בעת התקנתו או שילובו במערכת קיימת.

תוכנת-מדף (off-the-shelf software)

גם: מוצר-מדף מסחרי (COTS, commercial off-the-shelf)

תוכנת-מדף מסחרית (commercial off-the-shelf software).

**בדיקות קבלה על ידי המשתמש**

בדרך כלל מאמתות את כשירות המערכת לשימוש בידי משתמש עסקי.

עשויות להתבצע בידי: בודקי ספק התוכנה, בנוכחות הלקוח, במשותף על ידי בודקי ספק התוכנה ובודקי הלקוח, בודקי הלקוח, בנוכחות בודקי ספק התוכנה, בודקי הלקוח, ללא נוכחות ספק התוכנה.

**בדיקות (קבלה) תפעוליות**

קבלת המערכת על ידי מנהלי המערכת (system administrators), בדיקת גיבוי ושחזור, התאוששות מאסון (disaster recovery), ניהול משתמשים, משימות תחזוקה, עומס נתונים ומשימות הגירה (migration), בדיקות תקופתיות לפרצות אבטחה.

**בדיקות קבלה חוזיות ורגולטוריות**

בדיקות קבלה חוזיות מבוצעות לפי הקריטריונים לקבלה המופיעים בחוזה לייצור תוכנה מותאמת ללקוח.

יש להגדיר את הקריטריונים לקבלה בעת ההסכמה על תנאי החוזה.

בדיקות קבלה רגולטוריות מבוצעות לפי הנהלים והתקנים בהם יש לעמוד למשל נהלים ממשלתיים או משפטיים או תקני בטיחות.

**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 בדיקות אלפא מתבצעות באתר הפיתוח, אך לא על ידי צוות הפיתוח.

בדיקות בטא, או בדיקות שדה, מתבצעות על ידי לקוחות או לקוחות פוטנציאליים במקומות בהם הלקוחות נמצאים.

ארגונים עושים שימוש במונחים נוספים כגון: בדיקות במפעל (factory acceptance testing),

בדיקות באתר (site acceptance testing) עבור מערכות הנבדקות לפני ואחרי ההעברה לאתר הלקוח.

**2.3 סוגי בדיקות**

נזכיר: יש להבדיל ולא להתבלבל בין רמות בדיקה לסוגי בדיקות!

רמות בדיקה הן השלב בבדיקות (יחידה > אינטגרציה > מערכת > קבלה)

סוגי בדיקות מתייחסים אל: יעד הבדיקה (תפקוד, ביצועים...) , טכניקת הבדיקות.

**סוגי בדיקה לפי יעדם**

תפקוד מסוים של התוכנה, מאפיין איכותי, לא-תפקודי כגון: אמינות או שימושיות,

מבנה או ארכיטקטורה של התוכנה או המערכת.

בדיקה הנובעת משינויים, כלומר: אישור שפגמים תוקנו (בדיקות חוזרות) ,חיפוש אחר שינויים בלתי מכוונים (בדיקות נסיגה).

**בדיקות בעזרת מודלים של תוכנה**

בבדיקות מבניות; למשל:

מודל זרימת הבקרה (control flow model) או מודל מבנה התפריט (menu structure model)

בבדיקות תפקודיות; למשל: מודל זרימת תהליך (process flow model), מודל החלף מצבים   
(state transition model), או מיפרט הכתוב בשפה פשוטה (plain language specification)

בבדיקות לא-תפקודיות; למשל: מודל ביצועים, מודל שימושיות, מודל איומי אבטחה (security threat modeling).

**בדיקות מבניות בעזרת מודלים**

מודל זרימת הבקרה (control flow model)

מודל מבנה התפריט   
(menu structure model)

**בדיקות פונקציונליות בעזרת מודלים**

מודל זרימת תהליך (process flow model)

מודל החלף מצבים   
(state transition model)

מיפרט הכתוב בשפה פשוטה  
 (plain language specification)

**בדיקות לא-פונקציונליות בעזרת מודלים**

מודל ביצועים (performance model)

מודל שימושיות   
(usability model)

מודל איומי אבטחה   
(security threat modeling)

**2.3.1. בדיקות תפקודיות (פונקציונליות)**

נחזור על הגדרת בדיקות תפקודיות (functional testing):

בדיקות המבוססות על ניתוח מיפרט הפונקציונליות (תפקודיות) של רכיב או מערכת. ר' גם בדיקות קופסה שחורה.

הפעולות שעל מערכת, תת-מערכת או רכיב לבצע: מתוארות בדרך כלל בתוצרים כגון מפרטי דרישות, מקרי שימוש או מיפרט תפקודי, אם כי לעיתים תיעוד כזה אינו קיים...

במקרה כזה הפעולות הינן "מה שהמערכת עושה".

**השימוש בבדיקות קופסה שחורה**

אפשר להשתמש בשיטות מבוססות מפרטים כדי להפיק תנאי בדיקה ומקרי בדיקה מתפקודיות התוכנה או המערכת.

בדיקות תפקודיות מעריכות את ההתנהגות החיצונית של התוכנה (בדיקות קופסה שחורה).

**הגדרות**

בדיקות קופסה שחורה (black‑box testing)

* + 1. בדיקות, בין שהן פונקציונליות או לא, המתוכננות ומתבצעות...
    2. ללא התייחסות למבנה הפנימי של רכיב או של מערכת.

בדיקות אבטחה (security testing)

* + 1. בדיקות אבטחה, שהן סוג של בדיקות תפקודיות, חוקרות תפקודים (למשל, חומת אש, firewall) הנוגעים לגילוי איומים, כמו וירוסים למשל, שמקורם בגורמים חיצוניים זדוניים.

בדיקות ליכולת פעולה-הדדית (interoperability)

מעריכות את יכולת המוצר לשתף פעולה עם רכיב אחד או יותר או עם מערכת אחת או יותר.

**2.3.2. בדיקות לא-תפקודיות (לא-פונקציונליות)**

בוחנות "כמה טוב" המערכת פועלת

דוגמאות (תפורטנה בשקפים הבאים):

בדיקות ביצועים (performance testing), בדיקות עומס (load testing)

בדיקות מאמץ (stress testing), בדיקות שימושיות (usability testing)

בדיקות תחזוקתיות (maintainability testing), בדיקות אמינות (reliability testing)

בדיקות ניידות (portability testing)

בדיקות ביצועים (performance testing)

* + 1. תהליך בדיקות שנועד לקבוע את הביצועים של מוצר תוכנה.
    2. ר' גם *בדיקות יעילות.*

בדיקות עומס (load testing)

* + 1. סוג של בדיקות ביצועים המבוצעות כדי להעריך את ההתנהגות של רכיב או מערכת עם עומס גובר והולך, למשל מספר המשתמשים הבו-זמניים ו/או מספר התנועות (transactions), כדי לקבוע איזה עומס יכול להיות מטופל בידי הרכיב או המערכת.

ר' גם *בדיקות ביצועים, בדיקות מאמץ*

בדיקות מאמץ (stress testing)

סוג של בדיקות ביצועים המתבצעות כדי להעריך את התנהגות מערכת או רכיב כאשר העומסים:

מגיעים לרמה המירבית הצפויה או שפורטה או מעבר לרמה זו או עם זמינות מופחתת של משאבים כגון גישה לזיכרון או לשרתים.

[בעקבות IEEE 610] ר' גם *בדיקות ביצועים, בדיקות עומס*

בדיקות שימושיות (usability testing)

בדיקות כדי לקבוע את המידה שבה מוצר תוכנה הינו מובן ,קל ללימוד, קל להפעלה ,מושך (אטרקטיבי) עבור המשתמשים בתנאים שפורטו.   
[בעקבות ISO 9126]

בדיקות אמינות (reliability testing)

תהליך הבדיקות שנועד לקבוע את מידת האמינות של מוצר תוכנה

ר' גם *אמינות*

אמינות (reliability)

היכולת של מוצר תוכנה לבצע את התפקידים (functions) הנדרשים בתנאים נתונים ,ולאורך תקופת זמן שפורטה או עבור פעולות (operations) שמספרן הוגדר. [ISO 9126]

בדיקות תחזוקתיות (maintainability testing)

תהליך בדיקות שנועד לקבוע את מידת התחזוקתיות של מוצר תוכנה

ר' גם *תחזוקתיות*

תחזוקתיות (maintainability)

בדיקות שינויים שנעשו במערכת מבצעית (operational) או ההשפעה של שינוי בסביבת הפעולה של מערכת כזו.

בדיקות ניידות (portability testing)

תהליך הבדיקות שנועד לקבוע את מידת הניידות של מוצר תוכנה

ר' גם *ניידות*

ניידות (portability)

מידת הקלות שבה ניתן להעביר מוצר תוכנה מסביבה אחת של חומרה או תוכנה לסביבה אחרת. [ISO 9126]

**2.3.2 בדיקות לא-פונקציונליות**

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".(ISO 9126)

**בדיקות לא-פונקציונליות כמעריכות התנהגות חיצונית**

בדיקות לא תפקודיות מעריכות את ההתנהגות החיצונית של התוכנה.

וברוב המקרים עושות לצורך כך שימוש.

בטכניקות קופסה שחורה לעיצוב הבדיקות.

**2.3.3. בדיקת ארכיטקטורה של התוכנה (בדיקות מבניות)**

בדיקת מבנה/ ארכיטקטורה של התוכנה (בדיקות מבניות)

בדיקות קופסה לבנה (white‑box testing)

בדיקות מבנה (structural testing)

כיסוי קוד (code coverage)

פרק 4 מתייחס בהרחבה לשיטות כיסוי.

**מיקום בסדר הבדיקות**

אפשר לבצע בדיקות מבניות (קופסה לבנה) בכל רמות הבדיקה.

בסילבוס מומלץ ליישם בדיקות מבניות אחרי יישומן של טכניקות מבוססות-מיפרט.

אפשר גם אחרת: תלוי בהקשר!

יישום בסדר כזה מסייע למדידת יסודיות הבדיקות.

על ידי הערכת רמת הכיסוי (coverage) שהושגה עבור מבנה מסוים.

**כיסוי קוד (code coverage)**

כיסוי קוד (code coverage)

הכיסוי הוא המידה בה פריט מסוים (כגון: שורות קוד) נבדק בפועל על ידי סדרת בדיקות (test suite).

כיסוי מיוצג על ידי אחוז הפריטים שכוסו מתוך סך פריטים מסוג זה הקיימים בקוד הנבדק,

במידה והכיסוי הוא פחות מ 100%, יתכן צורך לתכנן בדיקות נוספות לבדיקת הפריטים הנותרים כדי להגדיל את הכיסוי.

פרק 4 מתייחס בהרחבה לשיטות כיסוי.

**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 למשל על בסיס מדרג הקריאה (calling hierarchy).

גישות בדיקה מבניות מתאימות ליישום ברמות בדיקת המערכת, האינטגרציה של המערכת או הקבלה.

הערה: בבדיקות קבלה אינן נפוצות.

**מושגים מהמילון**

בדיקות קופסה לבנה (white‑box testing)

* + 1. בדיקות המבוססות על ניתוח המבנה הפנימי של רכיב או מערכת.

שם נוסף: בדיקות קופסה שקופה (clear-box testing)

קרויות גם בדיקות מבניות (structural testing)

כלומר בדיקת 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. (הרחבה בפרק 4)

**2.3.4. בדיקות עקב שינויים: בדיקות חוזרות ובדיקות נסיגה**

בדיקות חוזרות (re-testing) קרויות גם: *בדיקות אישור (confirmation testing)*.

בדיקות המריצות מקרי בדיקה שנכשלו בריצתם האחרונה במטרה לאמת את הצלחתן של פעולות מתקנות שננקטו.

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 או ניפוי באגים (debugging, איתור פגמים ותיקונם) הינו פעילות פיתוח ולא פעילות בדיקה.

**בדיקות נסיגה**

בדיקות נסיגה (regression testing)

בדיקות של תכנית מחשב אשר כבר נבדקה קודם לכן בעקבות שינויים שעברה כדי לוודא שכתוצאה מאותם שינויים פגמים לא הוכנסו או התגלו באזורי התוכנה אשר... **לא השתנו (!)**

מתבצעות לאחר שהתוכנה או סביבתה עברו שינוי.

פגמים שנגרמו או שנחשפו כתוצאה מן השינוי עלולים להימצא בתוכנה הנבדקת, או ברכיב תוכנה אחר שקשור - או אף שאינו קשור ישירות - בתוכנה המפותחת. היקף הבדיקות נקבע על פי הסיכוי להופעת פגמים חדשים עקב השינוי והמשמעות העסקית האפשרית שלהם (הסיכון) ככל שהסיכון הנובע מאפשרות של פגמים חדשים הינו גבוה יותר נשקיע יותר מאמץ בבדיקות נסיגה. על הבדיקות להיות כאלה שניתן לחזור עליהן בדייקנות, אם רוצים לעשות בהן שימוש חוזר לבדיקות אישור ובדיקות נסיגה, אפשר לבצע בדיקות נסיגה בכל רמות הבדיקה והן כוללות בדיקות תפקודיות, לא-תפקודיות ומבניות.

סדרות של בדיקות נסיגה מופעלות מספר רב של פעמים ובדרך כלל עוברות התפתחות הדרגתית

מסיבה זו בדיקות נסיגה הן מועמדות מתאימות מאד לאוטומציה.

**2.4 בדיקות תחזוקה**

במקרים רבים מערכת תוכנה משמשת במשך שנים ואפילו עשורים בזמן זה המערכת, נתוני התצורה שלה, או סביבתה עוברים שינויים, תיקונים או הרחבות.

תכנון מראש של גרסאות הוא חיוני לקיום בדיקות תחזוקה מוצלחות, יש להבחין בין גרסאות מתוכננות לבין תיקוני חרום (hot fixes).

בדיקות תחזוקה מתבצעות על גבי מערכת מתפקדת קיימת, יוזמים אותן בעקבות שינויים, הגירה (migration) או סיום חיי התוכנה או המערכת.

**סוגי שינויים**

שיפורים מתוכננים (למשל, מבוססי גרסה).

תיקונים או עדכוני חירום.

שינויים בסביבה כגון שדרוג מתוכנן של מערכת ההפעלה או של בסיס הנתונים.

שדרוג מתוכנן של מוצר-מדף מסחרי (COTS).

טלאים (patch) לתיקון פרצות ותקלות שהתגלו, תיקון חירום (hot fix) הינו דחוף ומידי,

טלאי (patch) הינו תיקון שבא לפתור בעיות שאינן קריטיות.

**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 (migration, למשל מפלטפורמה אחת לאחרת) יש לכלול בדיקות תפעוליות של הסביבה החדשה כמו גם של התוכנה המעודכנת.

בדיקות הגירה (בדיקות הסבה) נחוצות גם במקרים בהם נתונים מיישום אחר יועברו אל המערכת המתוחזקת.

**סיום חיי מערכת (end-of-life)**

בדיקות תחזוקה לסיום חיי מערכת עשויות לכלול: בדיקות להגירת נתונים או להעברתם לארכיון,

במקרים בהם יש צורך לאחסן נתונים לתקופה ממושכת.

בנוסף לבדיקת השינויים, בדיקות תחזוקה כוללות בדיקות נסיגה של חלקי המערכת שנותרו ללא שינוי.

היקף בדיקות התחזוקה תלוי בסיכון, בגודל המערכת ובהיקף השינוי.

לפי אופי השינויים, אפשר לבצע בדיקות תחזוקה בכל רמות הבדיקה ועבור כל סוגי הבדיקות.

בחינת ההשפעה של שינויים על מערכת קיימת מכונה ניתוח השפעה

מהווה כלי מסייע בקבלת החלטה על היקף בדיקות הנסיגה הדרושות

ניתוח השפעה יכול גם לשמש בבחירת סדרת בדיקות הנסיגה.

**קשיים בבדיקות תחזוקה**

**קושי מרכזי:**

מי שפיתחו ובדקו את המערכת כבר התחלפו...

**שני גורמים המקשים על קיום בדיקות תחזוקה:**

1. העדר בודקים בעלי ידע בתחום
2. ו/או... מפרטים בלתי מעודכנים או חסרים.

**מושגים מהמילון**

**בדיקות תחזוקה (maintenance testing)**

בדיקת שינויים שנעשו במערכת מבצעיתoperational, production) ) או בדיקת ההשפעה של שינוי בסביבת הפעולה של מערכת כזו.

**ניתוח השפעה (impact analysis)**

הערכה של השפעת שינוי על: שכבות של תיעוד הפיתוח, תיעוד ורכיבים בבדיקות, במטרה ליישם את אותו שינוי ביחס לדרישות שפורטו.

**פרק 3: שיטות סטטיות**

3.1 שיטות סטטיות ותהליך הבדיקה (K2)

3.2 תהליך הסקירה (K2)

3.3 ניתוח סטטי באמצעות כלים

**3.1 שיטות סטטיות ותהליך הבדיקה**

בדיקות סטטיות מסתמכות על:

בחינה ידנית (סקירות)

וניתוח אוטומטי (ניתוח סטטי)

של מסמכים ו/או של הקוד

ללא הרצת הקוד.

**בדיקות דינמיות ובדיקות סטטיות**

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.

**חזרה: הגדרת הסקירה**

סקירה (review)

הערכת הסטטוס של מוצר או פרויקט כדי לברר אם קיימים פערים מהתוצאות המתוכננות, להציע שיפורים.

דוגמאות (כולן תפורטנה בהרחבה בהמשך הפרק):

סקירת הנהלה, סקירה לא רשמית, סקירה טכנית, ביקורת (inspection), דיון מודרך (walkthrough).

**מחיר הפגמים**

ניתן לבדוק את תוצרי העבודה (software work products)

כולל הקוד!

באמצעות סקירות אותן אפשר לקיים זמן רב לפני ביצוע הבדיקות הדינמיות.

מחיר הסרתם של פגמים שזוהו במהלך סקירות בשלבים המוקדמים במחזור החיים

(למשל, פגמים שנמצאו ברשימת הדרישות)

נמוך בהרבה מזה של פגמים שיזוהו על ידי הרצת בדיקות על הקוד המתבצע.

**מה בודקים בסקירות ידניות**

הפעילות הידנית העיקרית היא בחינת תוצר העבודה וכתיבת הערות.

ניתן לבדוק, בין היתר: מיפרט הדרישות, מיפרט העיצוב, קוד, תכניות הבדיקה וכן מפרטי 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 למוצאן במהלך בדיקות דינמיות הוא קלוש.

**סיכום המבוא לבדיקות סטטיות בתהליך הבדיקה**

מטרה משותפת לסקירות, ניתוח סטטי ובדיקות דינמיות: זיהוי פגמים.

השיטות משלימות - כל אחת 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, תחזוקתיות (maintainability) בלתי מספקת, מפרטי ממשק שגויים.

**3.2 תהליך הסקירה**

קשת הסקירות מגוונת ונעה בין:

הבלתי רשמיות: מאופיינות בהעדר הוראות כתובות.

שיטתיות ורשמיות יותר, אותן מאפיינים: השתתפות הצוות, תיעוד תוצאות הסקירה , נהלים מתועדים. לביצוע הסקירה.

**שיקולים בבחירת סוג הסקירה**

מידת הרשמיות של תהליך הסקירה תלויה בגורמים כמו:

בשלות תהליך הפיתוח, עמידה בדרישות חוקיות או רגולטוריות , צורך בהצגת נתיב ביקורת (audit trail).

האופן בו מבצעים סקירה תלוי ביעדים המוסכמים של הסקירה, למשל:

מציאת פגמים, הבנת המערכת, למידה על ידי בודקים וחברי צוות חדשים, דיון וקבלת החלטות בקונצנזוס.

**מושגים בסקירה רשמית**

סקירה רשמית (formal review)

סקירה המאופיינת בנהלים ודרישות מתועדים; למשל: ביקורת (ע"ע).

ביקורת (inspection)

סוג של סקירת עמיתים (peer review) הנשען על בחינה חזותית של מסמכים כדי לגלות פגמים,

למשל הפרות של תקני פיתוח ואי-התאמה לרמות תיעוד גבוהות יותר.

טכניקת הסקירה הרשמית ביותר ולפיכך מבוססת תמיד על נוהל כתוב ומתועד.   
[בעקבות IEEE 610, IEEE 1028] ר' גם *סקירת עמיתים*

**אמות מידה לכניסה (entry criteria)**

אמות מידה לכניסה (entry criteria):

סט של תנאים כלליים (generic) ו/או ספציפיים, אשר נדרשים כדי להרשות לתהליך להתקדם עם מטרה מוגדרת, למשל שלב בדיקות.

מטרת אמות המידה לכניסה היא למנוע משימה מלהתחיל באופן שיצריך יותר מאמץ (מבוזבז) בהשוואה למאמץ הנדרש כדי לעמוד באמות המידה.[Gilb and Graham]

ובקיצור: למנוע משימה מלהתחיל כשאיננו מוכנים אליה...

**אמות מידה ליציאה (exit criteria)**

סט של תנאים כלליים (generic) ו/או ספציפיים, המוסכמים עם בעלי העניין (stakeholders),

ואשר נדרשים כדי לאשר רשמית את סיומו של תהליך.

מטרת אמות המידה ליציאה היא למנוע מצב בו משימה תיחשב כאילו הושלמה, כאשר ישנם עדיין חלקים משמעותיים שלה שלא הסתיימו.

אמות מידה ליציאה משמשות לשם דיווח מולן וכדי לתכנן מתי לעצור את הבדיקות. [בעקבות Gilb and Graham]

**3.2.1 פעילויות במסגרת סקירה רשמית**

(חלקן משמשות גם בסקירה בלתי-רשמית; בסקירה רשמית כולן חובה)

1. תכנון
2. התנעה (kick-off)
3. הכנה אישית
4. בחינה/ הערכה/ רישום של תוצאות הישיבה
5. תיקונים
6. מעקב

**סקירה רשמית: תכנון**

1. תכנון

הגדרת אמות המידה של הסקירה (review criteria)

בחירת הצוות

חלוקת תפקידים

הגדרת אמות המידה לכניסה ויציאה ((entry and exit criteria:

עבור סקירות רשמיות יותר (כמו ביקורות)

בחירת המסמכים או חלקי המסמכים שייסקרו, בדיקת אמות המידה לכניסה עבור סקירות רשמיות יותר.

**סקירה רשמית: התנעה והכנה**

2. התנעה (kick-off)

הפצת מסמכים

הסברה למשתתפים בעניין היעדים, התהליך, המסמכים.

3. הכנה אישית

הכנה לישיבת הסקירה על ידי סקירת המסמכים, הכנת רשימה של פגמים פוטנציאליים,שאלות והערות.

**סקירה רשמית: בחינה/ הערכה/ רישום של תוצאות הישיבה**

4. בחינה/ הערכה/ רישום של תוצאות הישיבה

דיון או רישום

כולל (עבור סקירות רשמיות יותר) ,תוצאות מתועדות או פרוטוקול.

רישום של פגמים, הצעת המלצות לגבי הטיפול בפגמים, קבלת החלטות לגבי הפגמים.

בחינה, הערכה ורישום של בעיות העולות במהלך פגישות או מעקב אחר תקשורת אלקטרונית קבוצתית.

**סקירה רשמית: תיקונים ומעקב**

5. תיקונים

תיקון פגמים שנמצאו (מתוקנים לרוב בידי כותב המסמך הנסקר)

בסקירות רשמיות: רישום הסטטוס המעודכן של הפגמים

6. מעקב

ווידוא שהפגמים טופלו

איסוף מדדים = מטריקות (metrics)

בסקירות רשמיות: בדיקת אמות המידה ליציאה.

**מדדים (מטריקות)**

מדד (metric)

סוג המדידה (מה מודדים)

השיטה הננקטת לשם ביצוע המדידות

קנה המידה (scale) של מדידות. [ISO 14598]

דוגמאות למדדים:

רשימת באגים

מידת הביצוע של מקרי בדיקה

מידת ההצלחה של מקרי בדיקה

כלל מקרי הבדיקה לפי תוצאותיהם: טרם בוצע, עבר בהצלחה, נכשל, וכו'.

**3.2.2 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 (follow up) לאחר הישיבה.

אם יש צורך, המתאם מגשר בין נקודות מבט שונות ותכופות הינו האדם שבו תלויה הצלחת הסקירה.

מחבר (author): הכותב או האדם הנושא באחריות העיקרית למסמך(ים) הנסקר(ים).

**3.2.2 תפקידים ותחומי אחריות בסקירה**

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, לעתים נדרש שהתיעוד יהיה בטופס רישום (logging form).

הרשם צריך לוודא שטופס הרישום הינו קריא ומובן.

**רשימות-ביקורת (checklists) ונקודות מבט שונות**

האפקטיביות של סקירות ויעילותן משתפרים מתוך: התבוננות במוצרי תוכנה או תוצרי עבודה נלווים מנקודות מבט שונות.

שימוש ברשימות-ביקורת ((checklists.

לדוגמא:

רשימות-ביקורת המבוססות על נקודות מבט שונות כמו: בודק, משתמש, מתחזק (maintainer), מתפעל.

רשימות-ביקורת של בעיות אופייניות בדרישות.

כולן יכולות לסייע לחשיפת בעיות שלא התגלו עד כה.

**3.2.3 סוגי סקירות**

מוצר תוכנה או תוצר עבודה נילווה (מסמך) עשוי להיות נתון ליותר מסקירה אחת.

אם עושים שימוש ביותר מסוג סקירה אחד, סדר ביצוע הסקירות אינו מוחלט אלא נתון לשינויים. דוגמאות:

סקירה לא רשמית עשויה להתבצע לפני סקירה טכנית, ביקורת על דרישות עשויה להתבצע לפני דיון מודרך עם הלקוח.

בשקפים הבאים רשימת סוגי הסקירות העיקריים:

עבור כל סוג יובאו המאפיינים, האפשרויות, והמטרות עיקריות שלו.

**סקירה לא רשמית**

הגדרת סקירה לא רשמית informal review)):

אין תהליך רשמי.

יכולה להתבצע על ידי מוביל טכני הסוקר את העיצוב והקוד או במסגרת תכנות בזוגות (pair programming).

יתכן תיעוד של הממצאים.

מידת התועלת תלויה בסוקרים.

מטרה עיקרית:

השגת תועלת מסוימת, בעלות נמוכה (של זמן ומשאבים).

**דיון מודרך**

הגדרת דיון מודרך (walkthrough):

פגישה בהנחיית המחבר.

למשל תיאור תרחישים או הרצת הקוד "על יבש" (dry run) תוך השתתפות פעילה של קבוצת עמיתים.

מפגשים ללא סיום מוגדר שאינם מחייבים מסקנות וסיכומים, בהם אפשר (אך לא חובה):

להכין את הסוקרים טרם הישיבה, להכין דוח סקירה ובו רשימת הממצאים, למנות רשם (שאיננו המחבר).

בפועל אופי הדיון המודרך יכול לנוע בין בלתי-רשמי למדי לבין רשמי מאד

מטרות עיקריות: למידה, הבנה, מציאת פגמים.

**סקירה טכנית**

הגדרת סקירה טכנית (technical review):

תהליך גילוי פגמים מוגדר ומתועד הכולל עמיתים ומומחים טכניים בהשתתפות אפשרית של ההנהלה.

עשויה להתבצע כסקירת עמיתים ללא השתתפות ההנהלה.

רצוי שתתבצע בהנחיית מתאם (moderator) שקיבל הכשרה מקצועית לתפקיד ושאינו המחבר.

הכנה מוקדמת של הסוקרים טרם הישיבה.

אפשרות לשימוש ברשימות-ביקורת (צ'ק ליסט).

הכנת דוח סקירה הכולל: רשימת ממצאים, הכרעה בשאלה אם התוכנה עומדת בדרישות, במקרה הצורך המלצות בנוגע לממצאים.

בפועל אופי הסקירה יכול לנוע בין בלתי-רשמי למדי לבין רשמי מאד.

מטרות עיקריות: דיון, הערכת חלופות, מציאת פגמים, פתרון בעיות טכניות, בדיקת התאמה למפרטים, תכניות, נהלים ותקנים, קבלת החלטות.

**ביקורת**

הגדרת ביקורת (inspection):

סוג של סקירת עמיתים (peer review) הנשען על בחינה חזותית של מסמכים כדי לגלות פגמים, למשל הפרות של תקני פיתוח ואי-התאמה לרמות תיעוד גבוהות יותר.

טכניקת הסקירה הרשמית ביותר ולפיכך מבוססת תמיד על נוהל כתוב ומתועד.

בהנחיית מתאם שקיבל הכשרה מקצועית לתפקיד (ושאיננו המחבר), לרוב מתבצעת כבחינת עמיתים (peer examination),תפקידים מוגדרים, כוללת איסוף מדדים (מטריקות).

תהליך רשמי המבוסס על כללים ורשימות-ביקורת.

אמות המידה לכניסה ויציאה מוגדרות לצורך אישור מוצר התוכנה.

הכנה מוקדמת לישיבה.

כתיבת דוח ביקורת, כולל רשימת ממצאים.

תהליך מעקב (follow-up) רשמי (עם אפשרות להוספת רכיבים של שיפור תהליכים).

אפשרות למינוי קורא (reader).

מטרה עיקרית: מציאת פגמים.

**סקירת עמיתים**

הגדרת סקירת עמיתים (peer review):

אפשר לבצע דיון מודרך, סקירות טכניות וביקורות בתוך קבוצת עמיתים כלומר, עמיתים בדרג דומה ברמת הארגון.

סוג זה של סקירה מכונה סקירת עמיתים.

**3.2.4 גורמים התורמים להצלחת סקירות**

בין הגורמים התורמים להצלחת סקירות: קיום יעדים ברורים ומוגדרים מראש, מעורבות של אנשים המתאימים ליעדי הסקירה שהוגדרו.

הבודקים נתפשים כמי שמביאים ערך מוסף להצלחת הסקירה ולומדים על המוצר באופן שמאפשר להם להכין בדיקות בשלב מוקדם יותר.

הפגמים המתגלים מתוארים בצורה עניינית ומתקבלים בברכה.

מתחים בין-אישיים ועניינים פסיכולוגיים מטופלים כראוי (למשל, יצירת חוויה חיובית עבור המחבר).

הסקירה מתבצעת באווירה של אמון: תוצאותיה לא ישמשו בתהליכי הערכת-עובד (evaluation) של המשתתפים.

טכניקות הסקירה מותאמות להשגת היעדים וכן לסוג ולרמה (level) של תוצר העבודה, וליכולת הסוקרים.

נעשה שימוש, לפי הצורך, ברשימות בקרה או בחלוקת תפקידים.

מתבצעת הדרכה בשיטות הסקירה הנתונות במיוחד במקרה של סקירה רשמית יותר כמו ביקורת.

ההנהלה תומכת בתהליך סקירה תקין (למשל על ידי הקצאת זמן מספיק לפעילויות הסקירה בלו"ז הפרויקט).

קיים דגש על למידה ועל שיפור תהליכים.

לסיכום: לשם סקירה מוצלחת יש להכין אותה כראוי ולוודא שתתנהל באופן ענייני ונכון, לפי הסעיפים דלעיל.

**3.3 ניתוח סטטי באמצעות כלים**

מטרת הניתוח הסטטי לאתר פגמים בקוד מקור (source code) ובמודלים של תוכנה.

הניתוח הסטטי מתבצע ללא הרצה בפועל של התוכנה שהכלי בודק, לעומת בדיקות דינמיות שבהן כן מריצים את הקוד.

ניתוח סטטי מסוגל לגלות פגמים שקשה לחשוף בבדיקות דינמיות.

ניתוח סטטי, בדומה לסקירות, חושף פגמים ולא כשלים.

כלי הניתוח הסטטי מנתחים את קוד התוכנית: למשל זרימת הבקרה וזרימת נתונים וכן גם פלט כמו HTML ו XML.

(כלומר: כאן הבדיקה הסטטית באה לאחר הרצה דינמית, ובודקת את תוצריה).

**מושגים בניתוח סטטי**

ניתוח סטטי (static analysis)

ניתוח של תוצרים (כתובים) הקשורים בפיתוח תוכנה, למשל דרישות או קוד הנעשה ללא ביצוע או הרצה שלהם.

ניתוח סטטי נעשה לרוב בעזרת כלי תומך.

זרימת בקרה (control flow)

רצף של אירועים(או נתיבים, paths) בריצה של רכיב או מערכת.

**התועלת בניתוח סטטי**

גילוי מוקדם של פגמים עוד בטרם ביצוע בדיקות.

התראה מוקדמת על היבטים חשודים בקוד או בעיצוב אשר מתקבלת על בסיס חישובי מדדים

למשל: ערך גבוה של סיבוכיות (complexity; הגדרתה בשקפים הבאים).

איתור פגמים שקשה למצוא באמצעות בדיקות דינמיות.

גילוי תלויות ((dependencies וחוסר עקביות במודלים של התוכנה, כמו למשל קישורים (links).

שיפור תחזוקתיות הקוד והעיצוב (design).

מניעת פגמים, אם מחלקת הפיתוח מיישמת לקחים שנלמדו מהסקירה

(במקרה זה: הניתוח הסטטי).

**מושגים**

סיבוכיות (complexity)

הדרגה שבה עיצוב ו/או מבנה פנימי של רכיב או מערכת קשים:

להבנה, לתחזוקה, לאימות.

קומפיילר, מהדר (compiler)

כלי תוכנה אשר מתרגם לשפת מכונה תכניות שנכתבו בשפות מסדר גבוה.

מורכבות ציקלומטית (cyclomatic complexity)

המספר המרבי של נתיבים (paths) לינאריים ועצמאיים הקיימים בתוכנה מסויימת.

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 (כאן הכל מהווה חלק אחד.

(למשל גרף המקבל קריאה (called graph) או שגרת-משנה (סאברוטינה, subroutine)

[בעקבות McCabe] החישוב כאן: 3=2\*9-8+1

**פגמים אופייניים הנחשפים על ידי 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.

שגיאות תחביר (syntax) בקוד ובמודלים.

**שימוש בכלים בידי מפתחים ומעצבים**

כלים לניתוח סטטי בדרך כלל משמשים את המפתחים (הבודקים את התאמת הקוד לכללים מוגדרים מראש או לתקני תכנות), לפני ובמהלך בדיקות רכיבים ואינטגרציה או הכנסת קוד ((check-in אל תוך כלי ניהול תצורה (configuration management).

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 (קומפיילרים, compilers) התומכים באופן חלקי בניתוח סטטי, כולל גם חישוב מדדים.

**פרק 4: טכניקות לעיצוב הבדיקות**

4.1 תהליך פיתוח הבדיקות (K3)

4.2 סוגי טכניקות לעיצוב בדיקות (K2)

4.3 טכניקות מבוססות מיפרט או טכניקות קופסה שחורה (K3)

4.4 טכניקות מבוססות מבנה או טכניקות קופסה לבנה (K4)

4.5 טכניקות מבוססות-ניסיון (K2)

4.6 בחירת טכניקות הבדיקה (K2)

**4.1 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 (קונטקסט) של הבדיקות, כגון: בשלות תהליכי הבדיקות והפיתוח, אילוצי זמן, דרישות אבטחה ורגולציה, האנשים המעורבים.

**תנאי הבדיקות**

במהלך ניתוח הבדיקות, מנתחים את מסמכי בסיס הבדיקות.

על מנת להחליט מה יש לבדוק, כלומר, מזהים את תנאי הבדיקות (test conditions).

תנאי בדיקה מוגדר בתור: פריט או אירוע שניתן לאמת באמצעות מקרה בדיקה אחד או יותר;

למשל פונקציה, תנועה (transaction), תכונה, מאפיין, אלמנט מבני.

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

יצירת נֶעֱקָבוּת מתנאי הבדיקה חזרה אל המפרטים והדרישות.

מאפשרת ניתוח השפעה (impact analysis) אפקטיבי כשהדרישות משתנות, וגם את קביעת רמת כיסוי הדרישות המושגת על ידי סדרה של בדיקות.

בעת ניתוח בדיקות מיישמים את פרטי גישת הבדיקות   
(detailed test approach) על ידי בחירת טכניקות עיצוב בהן נשתמש.

הבחירה מתבססת, בין השאר, על הסיכונים שזוהו (מידע נוסף על ניתוח סיכונים מופיע בפרק 5).

**מקרי הבדיקה (test cases)**

מקרי הבדיקה ונתוני הבדיקה (test data) מיוצרים ומאופיינים במהלך עיצוב הבדיקות. מקרה בדיקה מורכב מ: סדרה של ערכי קלט, תנאים מוקדמים לביצוע (execution precondition), תוצאות צפויות

ותנאי בתר-ביצוע (execution postconditions).

התנאים מוגדרים כך שיתנו כיסוי ליעדי בדיקה או לתנאי בדיקה מסוימים .

תקן IEEE לתיעוד בדיקות תוכנה ( Standard for Software Test Documentation, IEEE Std 829-1998)

מתאר את תוכן מיפרט העיצוב של בדיקות (המכיל תנאי בדיקה) ואת תוכן מיפרט מקרי הבדיקה.

**התוצאות הצפויות (expected results)**

יש להגדיר, כחלק ממפרט מקרה הבדיקה, את התוצאות הצפויות.

על אלה לכלול: פלטים, שינויים בנתונים ובמצבים וכל השלכות אחרות של הבדיקה.

אם התוצאות הצפויות לא הוגדרו, יתכן מצב בו תוצאה סבירה אך שגויה, תתפרש כתוצאה תקינה.

רצוי מאד להגדיר את התוצאות הצפויות לפני ביצוע הבדיקה!

**נוהל בדיקות (test procedure)**

במהלך יישום הבדיקות מקרי הבדיקה עוברים:

פיתוח, יישום, תיעדוף, ואיגוד למפרט נוהל בדיקות ( IEEE Std 829-1998).

נוהל בדיקות מגדיר את רצף הפעולות לביצוע הבדיקה.

כאשר מריצים בדיקות באמצעות כלי לביצוע בדיקות, רצף הפעולות מפורט בתסריט בדיקות שהוא נוהל בדיקות אוטומטי.

**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 ולעיתים אף תסריטי הבדיקות האוטומטיות.

הלו"ז לביצוע לוקח בחשבון גורמים כגון: בדיקות נסיגה, תלויות טכניות ולוגיות.

**4.2 סוגי טכניקות לעיצוב בדיקות**

המטרה של טכניקה לעיצוב בדיקות היא לקבוע את: תנאי הבדיקה, מקרי הבדיקה, נתוני הבדיקה.

**מושגים**

טכניקת קופסה שחורה (black-box test design technique).

טכניקת קופסה לבנה (white‑box test design technique).

טכניקות מבוססות-ניסיון (experience‑based test design technique).

טכניקת עיצוב בדיקות (test design technique).

**קופסה שחורה**

מקובל להבחין בין טכניקות קופסה שחורה וקופסה לבנה.

טכניקות קופסה שחורה (מכונות גם טכניקות מבוססות-מיפרט):

הן דרך להפיק ולבחור תנאי בדיקה, מקרי בדיקה או נתוני בדיקה בעקבות ניתוח של תיעוד בסיס הבדיקה.

בשיטה זו כלולות בדיקות תפקודיות ושאינן תפקודיות (functional and non‑functional testing).

מעצם הגדרתן בדיקות קופסה שחורה אינן משתמשות במידע המתייחס למבנה הפנימי של הרכיב או של המערכת הנבדקת.

**קופסה לבנה**

טכניקות קופסה לבנה (מכונות גם טכניקות מבניות או מבוססות-מבנה): מבוססות על ניתוח מבנה הרכיב או המערכת.

אפשר לשלב בדיקות קופסה לבנה או קופסה שחורה עם טכניקות מבוססות-ניסיון על מנת לנצל את ניסיונם של המפתחים, הבודקים והמשתמשים בקבלת ההחלטה מה יש לבדוק.

**קופסה שחורה מול קופסה לבנה**

ישנן טכניקות המסווגות בברור תחת סיווג אחד, לטכניקות אחרות יש יותר מסיווג אחד.

סילבוס זה מתייחס לטכניקות מבוססות-מיפרט כאל טכניקות קופסה שחורה, ואל טכניקות עיצוב מבוססות-מבנה כאל טכניקות קופסה לבנה.

בנוסף לאלה, תכנית הסילבוס מכסה טכניקות עיצוב מבוססות-ניסיון.

מונח נוסף שאינו בשימוש בסילבוס, הוא "קופסה אפורה": כאשר לבודקים ידע מסויים אך מועט וספציפי לגבי מבנה הקוד.

**מאפייני בדיקות מבוססות-מיפרט**

מאפיינים נפוצים של טכניקות עיצוב בדיקות מבוססות-מיפרט: מודלים רשמיים או לא-רשמיים.

משמשים לאפיון הבעיה אותה יש לפתור או לאפיון התוכנה או רכיביה.

ניתן לגזור מקרי בדיקה באופן שיטתי ממודלים אלה.

**מאפייני בדיקות מבוססות-מבנה**

מאפיינים נפוצים של טכניקות עיצוב בדיקות מבוססות-מבנה:

מידע המתייחס לאופן בו בנויה התוכנה משמש לגזירת מקרי בדיקה (למשל, קוד ומידע מפורט אודות העיצוב).

ניתן למדוד את היקף כיסוי התוכנה עבור מקרי בדיקה קיימים, וניתן לגזור מקרי בדיקה נוספים באופן שיטתי על מנת להגדיל את הכיסוי.

**מאפייני בדיקות מבוססות-ניסיון**

מאפיינים נפוצים של טכניקות עיצוב בדיקות מבוססות-ניסיון: הידע והניסיון של אנשים משמשים לגזירת מקרי הבדיקה.

מקור מידע אחד הוא הידע של הבודקים, המפתחים, המשתמשים ובעלי עניין אחרים, לגבי התוכנה, אופן השימוש בה והסביבה בה היא תופעל.

מקור מידע נוסף הוא ידע אודות פגמים אפשריים והיכן הם צפויים להופיע.

**4.3 טכניקות מבוססות מיפרט = קופסה שחורה**

נחזור על ההגדרה: נוהל להפקה או בחירה של מקרי בדיקה, המבוסס על ניתוח המיפרט (specification) של רכיב או מערכת בין שהוא פונקציונלי או לא, ללא התייחסות למבנה הפנימי שלהם.

**4.3.1 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 המיפרט.

חלוקת שקילות, חלוקה למחלקות שקילות (equivalence partitioning)- טכניקת עיצוב בדיקות קופסה שחורה אשר בה מקרי בדיקה מעוצבים לשם ביצוע מדגם מייצג מתוך מחלקות השקילות.

בעקרון, מקרי הבדיקה מעוצבים כך שיכסו לפחות פעם אחת כל מחלקת שקילות.

**4.3.1 חלוקת שקילות: רקע**

במסגרת חלוקת שקילות, הקֶלֶט (input) לתוכנה או למערכת מחולק לקבוצות לפי הדמיון הצפוי בהתנהגות המערכת עבור ערכי קלט מכל קבוצה.

מחלקות שקילות (equivalence classes) קיימות עבור נתונים תקפים (valid), כלומר ערכים קבילים, וכן עבור נתונים לא-תקפים (invalid), כלומר ערכים שיש לדחות.

**מחלקות שקילות**

אפשר להגדיר מחלקות עבור: פלֶט (output), ערכים פנימיים, ערכים תלויי-זמן (למשל, לפני או אחרי אירוע), פרמטרים של ממשק (למשל, רכיבים משולבים הנבדקים במהלך בדיקות אינטגרציה).

אפשר לעצב בדיקות שתכסינה את כל המחלקות התקפות והבלתי-תקפות, חלוקת שקילות מתאימה ליישום בכל רמות הבדיקה, אפשר להשיג יעדי כיסוי של קלט ופלט.

**4.3.2 ניתוח ערכי גבול**

ערך גבול (boundary value)- ערך קלט או פלט הנמצא בקצה של מחלקת שקילות או במרחק המצטבר (אינקרמנטלי) הקטן ביותר מאחד מצידי הקצה, למשל הערך המזערי או המרבי של טווח (range).

בטווח מספרים שלמים בין 0-100, ערכי הגבול הם?

0, 100.

כדאי לבדוק גם: 1-, 0, 1, 99, 100, 101.

ניתוח ערכי גבול (boundary value analysis) טכניקת עיצוב בדיקות קופסה שחורה אשר בה מקרי בדיקה מעוצבים על בסיס ערכי גבול.

סביר יותר לגלות התנהגות לא תקינה בקצה של כל מחלקת שקילות מאשר בתוכה, לכן הקצוות הם אזורים שצפוי שיכילו פגמים.

ערכי המינימום והמקסימום של מחלקה הינם ערכי הגבול:

ערך גבול עבור מחלקה תקפה הינו ערך גבול תקף, ערך הגבול של מחלקה בלתי תקפה הוא ערך גבול לא-תקף.

ניתן לעצב בדיקות כך שיכסו ערכי גבול תקפים ולא תקפים.

כאשר מעצבים מקרי בדיקה, יש לבחור בדיקה עבור כל ערך גבול.

**יישום טכניקת ערכי הגבול**

ניתוח ערכי גבול מתאים ליישום בכל רמות הבדיקה.

טכניקה זו קלה יחסית ליישום ויכולות גילוי הפגמים שלה היא גבוהה.

מפרטים מפורטים מסייעים בבחירת הקצוות המעניינים.

הטכניקה נחשבת לעיתים קרובות כהרחבה של חלוקת שקילות או טכניקות קופסה שחורה אחרות.

ניתן ליישמה גם על מחלקות שקילות עבור קלט מסך מן המשתמש וגם, למשל, על טווחי זמן כגון time out או דרישות למהירות של תנועות (transactions), או טווחי טבלאות.

**4.3.3 בדיקות טבלת החלטה**

טבלת החלטות (decision table)- טבלה המראה צירופים של קלט ו/או מניעים (stimuli; או גורמים causes,) עם הפלט ו/או הפעולות (האפקטים) הקשורים בהם;

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.   
[Veenendaal04]

טבלאות החלטה הן שיטה טובה: לקבל תמונה של דרישות המערכת המכילות תנאים לוגיים ולתעד את עיצובה הפנימי של המערכת. אפשר להשתמש בהן לרשום כללים עסקיים (business rules) מורכבים אשר המערכת נדרשת ליישם.

**צירופים בוליאניים**

כאשר יוצרים טבלאות החלטה, מנתחים את המיפרט ומזהים את תנאי המערכת ופעולותיה.

תנאי הפלט והפעולות בדרך כלל מיוצגים באופן בוליאני (Boolean) כלומר, הם יכולים להיות במצב נכון (true) או לא-נכון (false).

טבלת ההחלטה מכילה את התנאים המחוללים (triggering conditions)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.

לרוב 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.

**4.3.4 בדיקות הֶחְלֵף מצבים**

הֶחְלֵף מצבים, מעבר מצבים (state transition testing) - הֶחְלֵף (מעבר) בין שני מצבים של רכיב או מערכת.

בדיקות הֶחְלֵף מצבים (state transition testing)-טכניקת בדיקת קופסה שחורה שבה מקרי בדיקה מעוצבים כדי להריץ חילופי מצבים תקפים ובלתי-תקפים.

מערכת עשויה להציג תגובה המשתנה בהתאם לתנאים הנוכחיים או להיסטוריה קודמת ("מצב המערכת" - state).

במקרה כזה, היבט זה של המערכת אפשר לייצג בתרשים הֶחְלֵף-מצבים(state transition diagram).

**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 שמעברים מסויימים אינם 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.

**שימושים**

בדיקות החלף-מצבים משמשות רבות את תעשיית התוכנה המשובצת (embedded software) ואת המיכון הטכני באופן כללי.

הטכניקה מתאימה גם למידול עצם עסקי בעל מצבים ספציפיים או לבדיקת זרימת דיאלוג מסך (למשל, עבור יישומי אינטרנט או תרחישים עסקיים).

**4.3.5 בדיקות מקרי שימוש**

מקרה שימוש (use case) מתאר את פעולות הגומלין בין השחקנים (actors: משתמשים או מערכות)

אשר מייצרות תוצאה בעלת ערך למשתמש במערכת או ללקוח.

אפשרי לגזור בדיקות ממקרי שימוש.

אפשר לתאר מקרי שימוש ברמה המופשטת (מקרה שימוש עסקי, ברמת התהליך העסקי וללא התייחסות לצד הטכני של המערכת המבצעת) 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.

כל מקרה שימוש מסתיים בתנאי בתר-ביצוע (post condition) שהם התוצאות הנצפות ומצבה הסופי של המערכת לאחר השלמת מקרה השימוש. בדרך כלל למקרה שימוש יש: תרחיש ראשי (mainstream: זה שיתבצע בדרך כלל) ותרחישים חלופיים.

**שימושים במקרי שימוש...**

מקרי שימוש מתארים את "זרימת התהליך" במערכת בהתבסס על השימוש הצפוי בפועל,

ולכן מקרי הבדיקה הנגזרים מהם יעילים ביותר בחשיפת פגמים בזרימת תהליכים בזמן שימוש אמיתי במערכת.

מקרי שימוש יעילים גם בעת עיצוב בדיקות קבלה (acceptance tests) בהשתתפות הלקוח או המשתמש.

מקרי שימוש מסייעים בחשיפת פגמי אינטגרציה הנובעים מפעולות גומלין או הפרעות בקשר בין רכיבים שונים אשר אינם נראים בבדיקת רכיבים בודדים.

עיצוב בדיקות על בסיס מקרי שימוש ניתן לשילוב עם טכניקות בדיקה מבוססות-מיפרט אחרות.

**4.4 טכניקות מבוססות מבנה = קופסה לבנה**

נחזור על הגדרת בדיקות קופסה לבנה (white-box testing)-בדיקות המבוססות על ניתוח המבנה הפנימי של רכיב או מערכת.

קרויות גם בדיקות מבוססות מבנה (structure‑based testing) -בדיקות מבוססות-מבנה או בדיקות קופסה לבנה מבוססות על כך שהבודקים מכירים מבנים מוגדרים ברמת הקוד או במערכת, כגון:

**מבנים בקוד ובמערכת**

רמת הרכיב (component): המבנים המרכיבים את רכיב התוכנה, כלומר, משפטים (statements), החלטות, הסתעפויות (branches), או אפילו מסלולים.

רמת האינטגרציה: המבנה עשוי להיות עץ קריאה (call tree), כלומר תרשים שבו מודולים קוראים למודולים אחרים.

רמת המערכת: המבנה עשוי להיות מבנה תפריט (menu structure), תהליך עסקי או מבנה דף אינטרנט.

**כיסוי קוד**

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(חלקם יתוארו בהמשך).

**שתי רמות כיסוי קוד בסיסיות**

בסעיף זה נדון בשתי טכניקות מבוססות-מבנה לכיסוי קוד, אשר מבוססות על משפטים, החלטות והסתעפויות.

(בסילבוס כתוב בטעות "שלוש טכניקות")עבור בדיקת כיסוי החלטות, ניתן להשתמש בתרשים זרימת בקרה (control flow diagram) על מנת להציג את החלופות (האלטרנטיבות) לכל החלטה.

**4.4.1 בדיקת משפטים וכיסוי משפטים**

כיסוי משפטים (statement coverage)-אחוז המשפטים (פקודות, statements) שבוצעו בפועל על ידי סדרת בדיקות, מתוך כלל המשפטים (פקודות) בקוד.

במסגרת בדיקות רכיבים, כיסוי משפטים (פקודות) הוא מדד לאחוז הפקודות (ה"משפטים") שהורצו על ידי סדרת בדיקות.

טכניקת בדיקת המשפטים מגדירה מקרי בדיקה שיריצו פקודות מסוימות, לרוב על מנת להגדיל את כיסוי המשפטים.

**כיסוי משפטים**

כיסוי משפטים נקבע על פי: חלוקה של מספר הפקודות שהורצו על ידי מקרי הבדיקה, בסך כל הפקודות הניתנות להרצה בקוד הנבדק.

ניתן לחשב כיסוי משפטים עבור מקרי בדיקה שהורצו בפועל;

ניתן גם לעשות הערכות של הכיסוי שיתקבל כבר בשלב העיצוב (design).

**4.4.2 בדיקת החלטות וכיסוי החלטות**

כיסוי החלטות (decision coverage)- הנקרא גם כיסוי הסתעפויות (branch coverage) הינו הערכת אחוז תוצאות של החלטות אשר נבחנו על ידי סדרת בדיקות.

(למשל, אפשרויות "נכון" ו"לא נכון" – True/False – של פקודת IF).

טכניקת בדיקת ההחלטות מגדירה מקרי בדיקה שיגרמו להחלטות ספציפיות בנקודות הסתעפות בקוד.

בעקבות ההחלטה, מסלול הבקרה משתנה ועובר לחלק אחר בקוד.

**כיסוי החלטות**

כיסוי החלטות נקבע על פי: מספר תוצאות ההחלטות המכוסות על ידי מקרי הבדיקה, חלקי סך כל תוצאות ההחלטות האפשריות בקוד הנבדק.

בדיקת החלטות היא צורה של בדיקת תרשים זרימת בקרה משום שהיא עוקבת אחר מסלול מסוים של בקרה דרך נקודות ההחלטה.

לכיסוי החלטות יש עוצמה רבה יותר מאשר לכיסוי משפטים!

כיסוי החלטות של 100% מבטיח כיסוי משפטים של 100% אך ההיפך אינו נכון.

**4.4.3 טכניקות מבוססות-מבנה אחרות**

קיימות רמות חזקות יותר של כיסוי מבני מאשר כיסוי החלטות, למשל, כיסוי תנאים (condition coverage) וכיסוי תנאים מרובים (multiple condition coverage) לגביהם ראו במילון המונחים.

רעיון הכיסוי מתאים ליישום ברמות בדיקה אחרות, למשל, ברמת האינטגרציה:

אפשר לבטא את אחוז המודולים, הרכיבים או המחלקות שנבחנו על ידי סדרת מקרי בדיקה ככיסוי מודולים, רכיבים או מחלקות.

ישנם כלים התומכים בבדיקות מבניות של קוד, תמיכה זו עשויה להיות שימושית ומועילה מאד.

**4.5 טכניקות מבוססות-ניסיון**

טכניקות מבוססות-ניסיון (experience-based test design technique)

בדיקות חוקרות (exploratory testing)

התקפה מוכוונת ליקויים (fault attack).

**בדיקות מבוססות-ניסיון (experience-based testing)**

בבדיקות מבוססות-ניסיון מגדירים בדיקות המבוססות על: מיומנות הבודקים, האינטואיציה שלהם,

וניסיונם עם יישומים וטכנולוגיות דומים.

כאשר משתמשים בבדיקות אלה כדי להשלים את הבדיקות השיטתיות, הטכניקות מבוססות-הניסיון עשויות להיות יעילות בזיהוי בדיקות מיוחדות שקשה להגיע אליהן באמצעות טכניקות רשמיות.

הדבר נכון במיוחד כאשר משתמשים בשיטות אלה אחרי השימוש בטכניקות רשמיות יותר.

אולם, תיתכנה סטיות משמעותיות בדרגת היעילות של טכניקה זו משום שהיא תלויה בניסיונם של הבודקים.

**ניחוש טעויות (error guessing)**

ניחוש טעויות היא טכניקה מבוססת-ניסיון נפוצה.

בדרך כלל, הבודקים יכולים לחזות פגמים מניסיונם.

קיימת גישה מובנית לטכניקת ניחוש הטעויות לפיה מכינים רשימה של פגמים אפשריים

ומעצבים בדיקות שמכוונות למציאת הפגמים הללו.

גישה שיטתית זו מכונה "התקפה מוכוונת ליקויים" (fault attack).

אפשר להרכיב רשימות פגמים וליקויים אלה על בסיס: הניסיון, נתונים קיימים לגבי פגמים וליקויים,

ומתוך הידע הקיים על סיבות לכשלים בתוכנה.

**בדיקות חוקרות (exploratory testing)**

בבדיקות חוקרות מתקיימים בו זמנית עיצוב, ביצוע, רישום ולמידה (שאחרת מתבצעים בתהליכים נפרדים וארוכים)על בסיס אמנת בדיקה (test charter) הכוללת יעדי בדיקה והמבוצעת במסגרת לוחות זמנים מוגדרים.

גישה זו יעילה במיוחד כאשר יש מיפרט מצומצם או בלתי מספיק ואילוצי זמן חריפים, או כאשר יש צורך בהרחבה או השלמה לבדיקות הרשמיות יותר.

גישה זו יכולה לשמש כביקורת על תהליך הבדיקה, ועל מנת להעלות את הסיכוי שמרבית הפגמים החמורים יימצאו.

**4.6 בחירת טכניקות הבדיקה**

הבחירה באיזו טכניקת בדיקה להשתמש תלויה במספר גורמים, בהם:

סוג המערכת, תקנים רגולטוריים, דרישות של הלקוח או דרישות חוזיות, דרגת הסיכון, סוג הסיכון,

מטרות הבדיקה, התיעוד הקיים, הידע של הבודקים, זמן ותקציב, מחזור חיי הפיתוח, מודלים של מקרי שימוש, וניסיון קודם עם סוגי הפגמים שנמצאו.

**תיאור**

טכניקות מסוימות מתאימות יותר למצבים מסוימים ולרמות בדיקה מסוימות, טכניקות אחרות מתאימות לכל רמות הבדיקה.

כשמגדירים 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), על מנת להבטיח כיסוי נאות של העצם הנבדק.

**פרק 5: ניהול הבדיקות**

5.1 ארגון הבדיקות

5.2 תכנון בדיקות והערכת היקף העבודה

5.3 מעקב ובקרה על התקדמות הבדיקות

5.4 ניהול תצורה

5.5 סיכון ובדיקות

5.6 ניהול אירועים

**5.1 ארגון הבדיקות**

**בודק (tester)**

איש מקצוע מיומן המעורב בבדיקות של רכיב או מערכת.

**מנהל בדיקות (test manager)**

אדם האחראי לניהול פרויקט של פעולות בדיקה ומשאביהן,

ולהערכה של מושא (אובייקט) הבדיקות.

האדם אשר מכוון, מבקר, מנהל, מתכנן ומסדיר את הערכת מושא בדיקות.

**מוביל בדיקות (test leader)**

לפי מילון ISTQB זהה למנהל אך אין זה בהכרח כך.

עשוי להיות המוביל הטכני של הבדיקות, מבלי לבצע את פונקציות הניהול.

**5.1.1 ארגון הבדיקות ועצמאותן**

ניתן לייעל את איתור הפגמים על ידי בדיקות וסקרים אם ארגון הבדיקות נהנה מעצמאות.

להלן טווח האפשרויות החל בתלות מוחלטת ועד לעצמאות מרבית:

אין בודקים עצמאיים. המפתחים עצמם בודקים את הקוד שכתבו (**תלות מוחלטת**)

בודקים עצמאיים בתוך צוותי הפיתוח.

צוות או קבוצת בדיקות עצמאיים בתוך ארגון הפיתוח המדווחים למנהל הפרויקט או להנהלה הבכירה.

בודקים עצמאיים מהיחידה העסקית או מתוך קהילת המשתמשים.

מומחי בדיקות עצמאיים לסוגי בדיקות מסוימות, למשל:

בודקי שימושיוּת

בודקי אבטחה

בודקי הסמכה

(שתפקידם לקבוע אם תוכנה עומדת בדרישות תקנים או דרישות רגולטוריות).

בודקים עצמאיים חיצוניים לארגון או העובדים באמצעות מיקור חוץ.

**עצמאות הבדיקות בפרויקטים קריטיים**

במקרה של פרויקט מורכב או קריטי מבחינה בטיחותית, עדיף בדרך כלל להסתמך על מספר רמות בדיקה, כאשר חלקן או כולן מבוצעות על ידי בודקים עצמאיים.

צוות הפיתוח יכול להשתתף בבדיקות, בפרט ברמות הנמוכות יותר, אך העדר האובייקטיביות שלהם מגביל בדרך כלל 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 במפורש לעשות זאת על ידי ההנהלה.

**יתרונות וחסרונות הבדיקה העצמאית**

יתרונות הבדיקה העצמאית כוללים:

בודקים עצמאיים אינם מוטים, ולכן יכולים לזהות פגמים אחרים ומסוגים שונים מאלו שבודקים לא-עצמאיים מזהים.

בודק עצמאי יכול לאמת הנחות שהונחו במהלך אפיון ויישום המערכת.

חסרונות הבדיקה העצמאית כוללים:

הבודקים מבודדים מצוות הפיתוח (כאשר מקפידים על אי תלות מוחלטת).

מפתחים עלולים לאבד את תחושת האחריות והמחויבות לאיכות.

בודקים עצמאיים עלולים להצטייר כצוואר בקבוק או להיות מואשמים בעיכובים בשחרור הגרסה.

**בודקים אפשריים**

משימות בדיקה יכולות להתבצע על ידי מי שהגדרת תפקידו היא "בודק"

או על ידי בעל תפקיד אחר, למשל:

מנהל פרויקט, מפתח, מומחה עסקי או מומחה בתחום,

או אנשי רשת ותשתית של מערכי מחשוב (Infrastructure and IT).

**5.1.2. משימותיהם של מנהל/מוביל בדיקות ושל בודק**

במסגרת תכנית לימודים זו נתייחס לשני תפקידים:

מנהל/מוביל בדיקות ,בודק.

הפעילות והמטלות המבוצעות בידי שני נושאי תפקידים אלה תלויות ב:

הקשר הפרויקט והמוצר, האנשים הממלאים את התפקידים, הארגון.

**מוביל/מנהל בדיקות**

**מוביל/מנהל בדיקות (test leader)**

מכונה לעיתים מנהל בדיקות (test manager) או מתאם בדיקות.

את תפקיד מוביל/מנהל הבדיקות יכול למלא:

מנהל פרויקט, מנהל פיתוח, מנהל הבטחת איכות או מנהל קבוצת בדיקות.

עבור פרויקטים גדולים ייתכנו שני תפקידים נפרדים (וכך רצוי!):

מוביל בדיקות ומנהל בדיקות.

בדרך כלל מוביל/מנהל הבדיקות:

מתכנן, מפקח ומבקר את פעילויות הבדיקה ואת המשימות כפי שהוגדרו לעיל בסעיף 1.4 (תהליכי הבדיקה היסודיים).

**משימות אופייניות של מוביל/מנהל בדיקות**

בין משימותיו האופייניות של מוביל/מנהל בדיקות אפשר למנות:

תיאום אסטרטגיית הבדיקות ותוכנית הבדיקות עם מנהלי פרויקטים ועם אחרים.

כתיבה או סקירה של אסטרטגיית בדיקות עבור הפרויקט, ושל מדיניות בדיקות עבור הארגון.

הוספת היבט הבדיקות לפעילויות אחרות כמו למשל תכנון האינטגרציה.

תכנון הבדיקות תוך התחשבות בהקשר והבנת יעדי הבדיקות והסיכונים –

כולל בחירת גישות בדיקה, הערכת זמן, מאמץ ועלות הבדיקות, הקצאת משאבים, הגדרת רמות בדיקה ומחזורי בדיקה ותכנון ניהול אירועים.

הובלת אפיון, הכנה, יישום וביצוע הבדיקות, וגם: מעקב אחר תוצאות הבדיקות

ובדיקת אמות המידה ליציאה (exit criteria).

התאמת התכנון לתוצאות הבדיקות ולהתקדמותן (לעיתים תוך תיעוד בדוחות מצב)

ונקיטת הצעדים הדרושים בכדי לפתור בעיות שמתעוררות.

יישום ניהול תצורה נאות של מכלול מרכיבי הבדיקות (בּוֹדְקָה, testware) לצורך נֶעֱקָבוּת (traceability).

הגדרת מדדים (metrics) מתאימים למדידת התקדמות הבדיקות ולהערכת איכות הבדיקות והמוצר.

החלטה אילו בדיקות תעבורנה אוטומציה, עד איזו רמה, וכיצד.

בחירת כלים תומכי בדיקות וארגון הדרכות עבור הבודקים על השימוש בכלים.

קבלת החלטה איך ליישם את סביבת הבדיקות.

כתיבת דוחות סיכום בדיקות על בסיס המידע שנאסף במהלך הבדיקות.

**משימות אופייניות של בודק**

בין משימותיו האופייניות של בודק אפשר למנות:

סקירת תכניות הבדיקה והשתתפות בכתיבתן.

ניתוח, סקירה והערכה של דרישות המשתמש, המפרטים והמודלים, מבחינת הבדיקוּת שלהן (testability).

יצירת מפרטי בדיקות (test specifications).

הגדרת ויישום סביבת הבדיקות (לעיתים קרובות בתיאום עם מנהל ה IT והרשת).

השגה ו/או הכנת נתוני הבדיקות.

יישום בדיקות בכל רמות הבדיקה, ביצוע בדיקות, תיעודן ואיסוף תוצאותיהן 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 של בדיקות).

מדידת ביצועים (performance) של רכיבים ושל מערכות.

סקירת בדיקות שפותחו על ידי אחרים.

**עוד על בודקים אפשריים**

האנשים העובדים על ניתוח בדיקות, עיצוב בדיקות, בדיקות ספציפיות או אוטומציית בדיקות...

עשויים להיות מומחים לתחומים אלה, הבאים מחוץ לצוות הקבוע.

בהתחשב ברמת הבדיקות (test level) והסיכונים הקשורים במוצר ובפרויקט, הבודק ישמור על מידה כלשהי של עצמאות.

בדרך כלל בודקים ברמת הרכיב או האינטגרציה יהיו מפתחים, בודקים ברמת בדיקות הקבלה יהיו מומחים עסקיים ומשתמשים, ובודקים ברמת בדיקות קבלה תפעוליות (operational acceptance testing - OAT) יהיו מפעילים (operators).

**5.2 תכנון הבדיקות והערכת היקף העבודה**

אסטרטגיית בדיקות (test strategy)

תיאור כללי מגבוה (high-level) של רמות הבדיקה שיש לבצען והבדיקות הכלולות ברמות אלה

עבור ארגון או תכנית (פרויקט program; ) אחד או יותר.

**גישה לבדיקות**

גישה לבדיקות (test approach)

היישום של אסטרטגיית בדיקות לפרויקט ספציפי.

כוללת בדרך כלל: את ההחלטות שנתקבלו בעקבות מטרת הפרויקט (בדיקות הפרויקט)

ואת הערכת הסיכונים שהתבצעה;

נקודות התחלה ביחס לתהליך הבדיקות;

טכניקות עיצוב הבדיקות שיש ליישם;

אמות-מידה ליציאה;

סוגי בדיקות שיש לבצע.

**5.2.1 תכנון הבדיקות**

בסעיף זה נסקור את מטרת תכנון הבדיקות במסגרת פרויקטים של פיתוח ויישום ובפעילויות תחזוקה.

את התכנון ניתן לתעד בתוכנית אב לבדיקות (master test plan) ובתוכניות בדיקה נפרדות לכל רמת בדיקה כגון: בדיקות מערכת ((system testing ,בדיקות קבלה (acceptance testing).

המתאר של מסמך תכנון בדיקות מכוסה בתקן לתיעוד בדיקות תוכנה ( IEEE Std 829‑1998).

**גורמים משפיעים על תכנון הבדיקות**

התכנון מושפע מ:

מדיניות הבדיקות של הארגון, היקף הבדיקות, היעדים, הסיכונים, האילוצים, מידת הקריטיות (criticality), מידת הבדיקוּת (testability) וזמינות משאבים.

עם התקדמות הפרויקט ותכנון הבדיקות, מתקבל מידע רב יותר וניתן להוסיף פרטים לתכנית 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.

**5.2.2 פעילויות תכנון בדיקות**

פעילויות תכנון הבדיקות של מערכת או חלק ממנה כוללות, בין היתר:

בחינת ההיקף והסיכונים וזיהוי יעדי הבדיקות.

הגדרת גישת הבדיקות הכללית, כולל הגדרת רמות הבדיקה ואמות המידה לכניסה ויציאה .

אינטגרציה ותיאום פעילויות הבדיקה עם פעילויות מחזור חיי התוכנה (הגדרת דרישות, רכישה ואספקה אם רלוונטי, פיתוח, תפעול ותחזוקה).

קבלת החלטות שיקבעו :

מה לבדוק, אילו נושאי תפקידים יבצעו את פעילויות הבדיקה, כיצד יש לבצע את פעילויות הבדיקה וכיצד יש להעריך את תוצאות הבדיקות.

פעילויות ניתוח הבדיקות ועיצובן, קביעת לו"ז וסדר ביצוע.

יישום, ביצוע והערכת הבדיקות, קביעת לו"ז וסדר ביצוע.

מסמכי הבדיקות

הגדרת כמות, דרגת פירוט, מבנה ותבניות (templates).

בחירת מדדים למעקב ובקרה של:

הכנת הבדיקות וביצוען, פתרון פגמים, סיכונים.

הגדרת רמת הפירוט הנדרשת בתיאור נוהל הבדיקות, כך שההכנה והביצוע של הבדיקות יהיו ניתנים לשחזור.

**5.2.3 אמות מידה לכניסה (entry criteria)**

אמות המידה לכניסה מגדירות את הנקודה בה מתחילים לבדוק, למשל תחילת רמת הבדיקה או הנקודה בה סדרה של בדיקות מוכנה לביצוע.

בדרך כלל אמות המידה לכניסה מכסות את הנקודות הבאות:

זמינות סביבת הבדיקות ומוכנותה.

מוכנות כלי הבדיקות בסביבת הבדיקה.

זמינותו של קוד מוכן לבדיקות.

זמינות נתוני בדיקות.

**5.2.4 אמות מידה ליציאה (exit criteria)**

אמות המידה ליציאה מגדירות את הנקודה בה מפסיקים לבדוק, למשל בסוף רמת בדיקה או כאשר סדרת בדיקות מסוימת השיגה מטרה מוגדרת.

בדרך כלל אמות המידה ליציאה מכסות את הנקודות הבאות:

מדדים ליסודיות הבדיקות כמו: כיסוי קוד (code coverage), כיסוי תפקודיות (functionality) ,כיסוי סיכונים.

הערכות של: צפיפות הפגמים (defect density) ,מדדי אמינות (reliability).

עלות.

סיכונים שנותרו, כגון: פגמים שלא תוקנו, חסר בכיסוי בדיקות באזורים מסוימים.

לוחות זמנים כמו: אלה המבוססים על זמן ההגעה לשוק.

**5.2.5 הערכת היקף עבודת הבדיקות**

שתים מן הגישות להערכת ההשקעה הדרושה לביצוע הבדיקות 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 ממומחים.

לאחר הערכת ההשקעה הנדרשת, אפשר לזהות משאבים נדרשים ולהקצותם, ולנסח לוח זמנים וסדר ביצוע.

מאמץ הבדיקות עשוי להיות תלוי במספר גורמים, ביניהם:

מאפייני המוצר: איכות המפרטים ומידע נוסף המשמש להגדרת מודלים לבדיקות   
(כלומר: בסיס הבדיקות, test basis),

גודל המוצר, מורכבות התחום בו עוסקת התוכנה, דרישות האמינות והאבטחה ודרישות התיעוד.

מאפייני תהליך הפיתוח: יציבות הארגון, הכלים בשימוש, תהליך הבדיקות, מיומנויות האנשים המעורבים, ואילוצי זמן.

תוצאות הבדיקות: מספר הפגמים וכמות העבודה הקשורה לתיקונים.

**5.2.6 אסטרטגיית הבדיקות, גישה לבדיקות**

הגישה לבדיקות הינה יישום של אסטרטגיית הבדיקות עבור פרויקט מסוים.

את גישת הבדיקות מגדירים ומלטשים במסגרת תכניות הבדיקות ועיצוב הבדיקות.

בדרך כלל גישת הבדיקות כוללת החלטות שהתקבלו על בסיס מטרת הפרויקט והבדיקות, והערכת הסיכונים.

גישת הבדיקות הינה נקודת ההתחלה לתכנון תהליך הבדיקות: בחירת טכניקות עיצוב הבדיקות וסוגי הבדיקות שייושמו.

הגדרת אמות המידה לכניסה וליציאה.

**הגישה הנבחרת**

הגישה הנבחרת היא תלוית הקשר ועשויה להביא בחשבון:

סיכונים, סכנות ובטיחות, משאבים ומיומנויות זמינים, טכנולוגיה, אופי המערכת (כגון מותאמת ללקוח או מוצר-מדף מסחרי), יעדי הבדיקות, תקנות הרגולציה.

**גישות אופייניות**

גישות אופייניות הינן:

גישות אנליטיות, כגון בדיקות מבוססות סיכונים בהן הבדיקות מכוונות כלפי האזורים בסיכון הגבוה ביותר.

גישות מבוססות מודל, כגון בדיקות אקראיות (stochastic testing) המשתמשות במידע סטטיסטי

אודות שיעורי הכשל (למשל מודלים להתפתחות האמינות והאיכות - reliability growth models)

או אודות השימוש (למשל פרופילים תפעוליים).

גישות שיטתיות, כגון:

גישה מבוססת כשל, כולל: ניחוש שגיאות (error guessing) והתקפה מוכוונת ליקויים (fault attacks)

גישות מבוססות ניסיון, גישות מבוססות רשימת ביקורת (צ'ק-ליסט) ,גישות מבוססות מאפייני איכות.

גישות תואמות תהליך או תקן, כגון: אלה המוכתבות על ידי תקנים תעשייתיים או על ידי שיטות זריזות (agile) למיניהן.

גישות דינמיות והאוריסטיות (heuristic) כגון בדיקות חוקרות (exploratory testing) בהן הבדיקות משתנות בהתאם לתגובות המערכת, ואינן מתוכננות בדייקנות מראש.

ביצוע הבדיקות והחלטה מה לבדוק בשלב הבא הן משימות השלובות זו בזו.

גישות ייעוציות כגון אלה בהן כיסוי הבדיקות מונע בעיקר על ידי ייעוץ והדרכה של מומחים חיצוניים מהתחום העסקי או הטכנולוגי, שאינם שייכים לצוות הבדיקות.

גישות המתמקדות בהפחתת הסיכון לתופעות נסיגה (נוגדות-נסיגה regression-averse approaches), כגון אלה הכוללות שימוש חוזר ב:מקרי בדיקה קיימים, אוטומציה נרחבת של בדיקות-נסיגה תפקודיות, סדרות בדיקה סטנדרטיות.

ניתן לשלב גישות שונות, למשל, גישה דינמית שהיא גם מבוססת-סיכונים.

**5.3 מעקב ובקרה של התקדמות הבדיקות**

מושגים (יפורטו בשקפים הבאים)

צפיפות פגמים (defect density)

שיעור כשלים (failure rate)

בקרת בדיקות (test control)

ניטור בדיקות (test monitoring)

דו"ח סיכום בדיקות (test summary report)

**מושגים**

צפיפות פגמים (defect density)

מספר הפגמים המזוהים ברכיב או מערכת מחולק בגודל הרכיב או המערכת;

הגודל מבוטא במונחי מדידה תקניים, כמו מספר שורות קוד, מספר מחלקות (classes) או מספר נקודות פונקציה (function points).

שיעור כשלים (failure rate)

שיעור מספר הכשלים מקטגוריה נתונה ביחס ליחידת מדידה נתונה,

למשל כשלים ליחידת זמן, כשלים למספר תנועות (transactions), כשלים למספר הרצות מחשב. [IEEE 610].

ניטור בדיקות (test monitoring)

משימת ניהול בדיקות העוסקת בפעילויות הקשורות בבדיקות תקופתיות של מצב (סטטוס) פרויקט בדיקות.

במסגרתה מכינים דוחות המשווים את המצב הנוכחי בפועל לעומת מה שתוכנן.

בקרת בדיקות (test control)

מטרה של ניהול בדיקות, העוסקת בפיתוח ויישם סט של פעולות מתקנות כדי להחזיר פרויקט בדיקות למסלולו, כאשר הניטור (monitoring) 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 exit criteria))   
[בעקבות IEEE 829].

**5.3.1 מעקב אחר התקדמות הבדיקות**

מטרת המעקב אחר בדיקות היא לספק משוב ולתת תמונת מצב ברורה של פעילויות הבדיקה.

אפשר לאסוף את המידע אחריו עוקבים באופן ידני או אוטומטי, ואפשר להשתמש בו כדי למדוד אמות מידה ליציאה כמו למשל רמת כיסוי.

ניתן לנצל מדדים כדי להעריך את ההתקדמות ביחס ללו"ז ולתקציב המתוכננים.

**מדדים להתקדמות הבדיקות**

בין המדדים השכיחים ניתן למנות את:

אחוז עבודת ההכנה של מקרי בדיקה, שבוצע בפועל (או אחוז מקרי הבדיקה המתוכננים שהוכנו).

אחוז עבודת ההכנה של סביבת העבודה, שבוצע בפועל.

ביצוע מקרי בדיקה. למשל: מספר מקרי הבדיקה שהורצו/ שלא הורצו, מספר מקרי הבדיקה שעברו/ שנכשלו.

מידע אודות פגמים (למשל, צפיפות פגמים, פגמים שהתגלו ותוקנו, שיעורי כשלים ותוצאות בדיקות חוזרות).

שיעורי הכיסוי על ידי הבדיקות בתחומים הבאים: דרישות, סיכונים, קוד.

תחושת הביטחון הסובייקטיבית של הבודקים באיכות המוצר.

מדדים הקשורים במועדיהן של אבני דרך (milestones) בבדיקות.

עלויות בדיקה: כולל השוואת עלות ביחס לתועלת שבמציאת הפגם הבא או בהרצת הבדיקה הבאה.

**5.3.2 דיווח על בדיקות**

דיווח על בדיקות מתייחס לסיכום המידע אודות מאמץ הבדיקות, כולל:

מה אירע במהלך תקופת הבדיקות המדווחת למשל תאריכים בהם הושגו אמות המידה ליציאה.

ניתוח של מידע ומדדים אשר יתמוך בהמלצות ובהחלטות לגבי פעולות בהמשך, כגון:

הערכת כמות הפגמים הנותרים, התועלת הכלכלית של המשך הבדיקות, סיכונים שעדיין קיימים,

דרגת האמון בתוכנה הנבדקת.

**תקן התיעוד, והמדדים**

המבנה והסעיפים של דו"ח סיכום בדיקות מופיעים בתקן לתיעוד בדיקות תוכנה (IEEE Std 829‑1998).

את המדדים יש לאסוף במהלך רמת הבדיקה ובסופה, על מנת להעריך את:

התאמתם של יעדי הבדיקות לרמת הבדיקה הנדונה, התאמת גישות הבדיקה שננקטו, יעילות הבדיקות ביחס ליעדיהן.

**5.3.3 בקרת בדיקות**

בקרת בדיקות מתארת פעולות הנחיה או תיקון שננקטות כתוצאה ממידע או מדדים שנאספו ושדווחו.

פעולות אלה עשויות לכלול כל פעילות בדיקה שהיא, ועשויות להשפיע על כל פעילות או משימה אחרת במחזור חיי התוכנה.

**דוגמאות לפעולות בקרת בדיקות**

קבלת החלטות המבוססות על מידע שהתקבל מניטור הבדיקות.

תיעדוף מחדש של בדיקות בעקבות התממשות סיכון צפוי (למשל איחור באספקת התוכנה).

שינוי בלו"ז או בסדר ביצוע הבדיקות בעקבות שינוי בזמינות סביבת העבודה.

הגדרת אמת מידה לכניסה הדורשת שמפתח יבצע: בדיקה חוזרת של תיקונים(בדיקות אישור: confirmation) לפני קבלתם אל תוך הגרסה.

**5.4 ניהול תצורה**

ניהול תצורה (configuration management)

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. [IEEE 610]

נקראת גם: *בקרת גרסאות (version control).*

**ניהול תצורה מצד הבדיקות**

כללית, ניהול תצורה מיועד לבסס ולשמר את השלמות (integrity) של מוצרי התוכנה או המערכת (רכיבים, נתונים ותיעוד) לאורך מחזור חיי הפרויקט והמוצר.

מצד הבדיקות, ניהול תצורה קשור להבטחת הנקודות הבאות:

יש זיהוי חד-משמעי לכל פריט במכלול מרכיבי הבדיקות.

עבור כל פריט יש בקרת גרסה, ומעקב אחר שינויים.

כל הפריטים מקושרים זה לזה או לפריטי פיתוח (מושאי בדיקה, test objects) באופן שמאפשר נֶעֱקָבוּת 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 (test harness) והבדיקה עצמה.

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

**5.5 סיכון ובדיקות**

סיכון מוגדר כסיכוי להתרחשות אירוע, סכנה, איום או מצב, שיובילו להשלכות בלתי רצויות או לבעיה אפשרית.

רמת הסיכון נקבעת בהתאם לסבירות שאירוע לא רצוי יתרחש, ולהשפעה (הנזק) שתגרם כתוצאה מהאירוע.

סיכוני פרויקט (project risk).

סיכוני מוצר (product risk).

בדיקות מבוססות סיכון (risk-based testing).

**סיכון ובדיקות: מושגים**

סיכון (risk)

גורם שתוצאותיו בעתיד עשויות להיות שליליות;

לרוב מבוטא כהשפעה (impact) או כסבירות (likelihood).

סיכון פרויקט (project risk)

סיכון הקשור ישירות לניהול ובקרת פרויקט (הבדיקות), למשל: מחסור באנשי צוות, קווי-סיום (deadlines) קשוחים, דרישות משתנות, וכדומה.

סיכוני מוצר (product risk)

סיכון הקשור ישירות למושא הבדיקות (test object).

בדיקות מבוססות סיכון (risk-based testing)

גישה לבדיקות המיועדת להפחית את רמת סיכוני המוצר וליידע את בעלי העניין על מצב הסיכונים,

וזאת כבר משלביו הראשונים של פרויקט.

כרוכים בכך זיהוי של הסיכונים המוצריים, ושימוש ברמות הסיכון כדי להנחות את תהליך הבדיקות.

**5.5.1 סיכוני פרויקט**

סיכוני פרויקט נוגעים ליכולת הפרויקט לעמוד ביעדיו, למשל:

גורמים ארגוניים: חוסרים במיומנות, בהכשרה ובאנשי צוות, בעיות בתחום משאבי אנוש, בעיות של פוליטיקה ארגונית, כמו: קושי של הבודקים לתקשר אפקטיבית את צרכיהם ואת תוצאות הבדיקות.

הצוות אינו פועל בעקבות המידע העולה מן הבדיקות והסקירות (למשל, אין משפרים את נוהלי הפיתוח והבדיקות).

יחס לא הולם כלפי הבדיקות או ציפיות לא הולמות מהן (למשל, העדר הכרה בערך מציאת פגמים במהלך בדיקות).

**סיכונים טכניים**

בעיות טכניות:

קושי בהגדרת הדרישות הנכונות, היקף אי-העמידה בדרישות כתוצאה מאילוצים קיימים, סביבת הבדיקות אינה מוכנה בזמן, איחור בהמרת נתונים, בתכנון ופיתוח הגירה (migration) ובבדיקת המרת הנתונים/ כלי ההמרה (conversion).

איכות נמוכה של: העיצוב, הקוד, נתוני התצורה, נתוני הבדיקות והבדיקות.

**סיכוני צד שלישי, וסיכום**

בעיות מצד ספקים: כשל צד שלישי (כגון: אי עמידה בהתחייבויות), בעיות חוזיות.

כאשר מנהל הבדיקות מנתח, מנהל וממתן את הסיכונים הללו...

הוא פועל על פי העקרונות המקובלים לניהול פרויקטים.

תקן תיעוד בדיקות תוכנה ( IEEE Std 829‑1998) דורש שסיכונים ותגובות אפשריות יופיעו בצורה מפורשת במסמכי תכנון הבדיקות.

**5.5.2 סיכוני מוצר**

אזורים מועדים לכשל (אירועים בלתי רצויים בעתיד או סכנות) בתוכנה או במערכת ידועים כסיכוני מוצר, משום שהם מסכנים את איכות המוצר.

ההבדל בין סיכוני פרויקט ומוצר: סיכוני פרויקט נוגעים ליכולת לעמוד ביעדים כנדרש יעדי עמידה בזמנים, אך גם יעדי איכות שהינם בעצם סיכוני מוצר.

סיכוני מוצר נוגעים למוצר "נטו" (בלי קשר לשאלות לוח זמנים).

**דוגמאות לסיכוני מוצר**

אספקת תוכנה שנוטה להיכשל.

האפשרות שתוכנה או חומרה עלולים לגרום נזק לאדם או לגוף עסקי.

רמה נמוכה של תכונות התוכנה, למשל: תפקודיות, אמינות, שימושיוּת וביצועים.

רמה נמוכה של איכות ושלמות הנתונים, למשל: בעיות בהגירת נתונים, בעיות בהמרת נתונים, בעיות בהעברת נתונים, הפרות של תקנים הנוגעים לנתונים.

תוכנה שאינה מבצעת את הפעולות שלשמן נועדה.

**סיכונים ובדיקות**

שימוש בסיכונים עוזר להחליט היכן להתחיל לבדוק והיכן לבדוק יותר;

הבדיקות משמשות להפחתת הסיכון להתרחשות תופעה שלילית, או להפחתת היקף השפעתה של התופעה השלילית.

סיכוני מוצר הם סוג מיוחד של סיכון להצלחתו של הפרויקט.

הבדיקות, כפעילות של בקרת סיכונים, מספקות משוב לגבי הסיכון שנותר, וזאת באמצעות מדידת האפקטיביות של הסרת פגמים קריטיים ושל מימוש תכניות חירום (contingency plans).

**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, ביצוען.

**שימוש בסיכונים שזוהו**

בגישה מבוססת סיכונים אפשר להשתמש בסיכונים שזוהו כדי:

לבחור את טכניקות הבדיקה, לקבוע את היקף הבדיקות אותן יש לבצע, לתעדף את הבדיקות.

במטרה לגלות פגמים קריטיים מוקדם ככל האפשר.

לבחון פעילויות שאינן בדיקות שעשויות להפחית סיכונים(למשל, ארגון הכשרה למעצבים חסרי ניסיון).

**סיכום**

בדיקות מבוססות-סיכון מסתמכות על הידע והתובנות של בעלי העניין כדי לזהות סיכונים ולקבוע את רמות הבדיקה הנדרשות להתמודדות איתם.

כדי להבטיח את הפחתת הסיכוי לכשל המוצר, פעולות ניהול סיכונים מספקות גישה מסודרת שבאמצעותה אפשר: להעריך (ולהעריך מחדש באופן קבוע) מה עשוי להשתבש (סיכונים),

להחליט באילו סיכונים חשוב לטפל, וליישם פעולות לטיפול בסיכונים אלה.

בדיקות עשויות לעזור ב: זיהוי סיכונים חדשים ובבחירת הסיכונים אותם יש להקטין, הפחתת חוסר הוודאות לגבי סיכונים.

**5.6 ניהול אירועים**

ניהול אירועים (incident management).

רישום אירועים (incident logging).

דו"ח אירועים (incident report).

**מושגים**

ניהול אירועים (incident management):

התהליך של: הכרה, חקירה, נקיטת פעולה וסילוק של אירועים.

כרוכים בו: רישום אירועים, סיווגם, וזיהוי השפעתם.

רישום אירועים (incident logging)

תיעוד הפרטים של כל אירוע שהתרחש, למשל תוך כדי בדיקות.

דו"ח אירועים (incident report)

מסמך המדווח על כל אירוע שהתרחש, למשל בזמן בדיקות, ואשר דורש חקירה.

**רקע**

יש לרשום כאירוע כל אי-התאמה בין תוצאות צפויות לתוצאות בפועל מאחר ואחד היעדים של בדיקות הוא למצוא פגמים.

כל אירוע חייב להיחקר, שכן הוא עלול להתגלות ככשל שמקורו בפגם;

שימו לב להבדל בין אירוע לבין כשל ופגם!

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 כל האירועים עד להשלמתם.

**מתי ועל מה לדווח**

תמיד, על הכל!...

אפשר לדווח על אירועים במהלך כל שלב: הפיתוח, הסקירה, הבדיקות או השימוש במוצר תוכנה.

אפשר לדווח על: בעיות בקוד או במערכת שכבר פועלת, או בכל סוג של תיעוד, כולל:

הדרישות, מסמכי הפיתוח, מסמכי הבדיקות.

מידע למשתמש כגון "עזרה" (Help) או הוראות ההתקנה.

**יעדי דוחות האירועים**

אלה היעדים של דוחות אירועים:

לספק משוב למפתחים ולקבוצות אחרות לגבי הבעיה, שיאפשר להם: זיהוי, בידוד ותיקון, לפי הצורך.

לספק למובילי הבדיקות אמצעי מעקב אחרי: איכות המערכת הנבדקת (system under test) התקדמות הבדיקות.

להעלות רעיונות לשיפור תהליך הבדיקות.

**פרטי דו"ח אירוע**

מבנה דו"ח אירועים מצוי בתקן תיעוד בדיקות תוכנה ( IEEE Std 829‑1998).

הפרטים המופיעים בדוח אירוע עשויים לכלול:

תאריך כתיבת הדוח, שם מחבר הדוח והארגון אליו הוא שייך.

תוצאות צפויות ותוצאות בפועל.

פרטים מזהים של: פריט הבדיקות (test item) / פריט התצורה (configuration item) ,סביבת הבדיקות.

השלב - במחזור חיי התוכנה או המערכת - בו נצפה האירוע.

תיאור של האירוע, אשר יאפשר את שחזורו ופתרונו, כולל תיאור מפורט של הצעדים שהובילו לאירוע, ופרטים נוספים כגון: לוגים (logs), תוכן בסיסי נתונים (קבצי dump), צילומי מסך (screenshots).

היקף או דרגת ההשפעה על אינטרסים של בעלי עניין.

חומרת ההשפעה (impact) על המערכת.

דחיפות או סדר עדיפות לתיקון.

מצב האירוע (status). למשל: פתוח, דחוי (deferred), כפול (duplicate), ממתין לתיקון, מתוקן וממתין לבדיקה חוזרת, סגור.

מסקנות, המלצות, אישורים.

בעיות כלליות, כמו למשל:

אזורים אחרים שיושפעו משינוי קוד הנובע מתיקון הבעיה שדווחה באירוע.

היסטורית שינויים, למשל רצף הפעולות שננקטו על ידי צוות הפרויקט לצורך בידוד האירוע, תיקונו ואישורו כמתוקן.

הפניות, כולל זיהוי מדוייק של מקרה הבדיקה אשר חשף את הבעיה.

**פרק 6: כלים תומכי בדיקות**

6.1 סוגים של כלי בדיקה

6.2 שימוש אפקטיבי בכלים: תועלת וסיכונים אפשריים

6.3 הכנסת כלי לארגון (K1)

**6.1.1 כלים תומכי בדיקות**

הערה: הגדרות מפורטות לכלים תבואנה בסעיפים המתאימים)

כלי בדיקה יכולים לשמש לפעילות תומכת בדיקות אחת או יותר.

בין הכלים הללו נמנים: כלים המשמשים באופן ישיר לבדיקות, כמו למשל: כלי ביצוע בדיקות, כלים ליצירת נתוני בדיקה , כלים להשוואת תוצאות.

כלים המסייעים בניהול התהליך, כמו אלה המשמשים לניהול של: הבדיקות, התוצאות, הנתונים, הדרישות, האירועים, הפגמים וכו'.

ולדיווח ולמעקב אחר ביצוע הבדיקות.

כלים המשמשים לחקירה (exploration). למשל: כלי העוקב אחרי פעולות שיישום תחת בדיקה מבצע על קבצים (פתיחה, סגירה וכו').

כל כלי המסייע לבדיקות (במובן זה, גם גיליון אלקטרוני נחשב כלי בדיקה).

**מטרות הכלים תומכי הבדיקות**

לשימוש בכלים תומכי בדיקות עשויות להיות אחת או יותר מהמטרות הבאות, בהתאם להקשר:

שיפור יעילות פעילויות הבדיקה על ידי: אוטומציה של פעולות החוזרות על עצמן, או תמיכה בפעילויות בדיקה ידניות כמו: תכנון בדיקות, עיצוב בדיקות, דיווח על בדיקות וניטור.

אוטומציה של פעילויות הדורשות משאבים רבים כאשר מבצעים אותן ידנית למשל, בדיקות סטטיות.

אוטומציה של פעילויות שאינן ניתנות לביצוע ידני למשל, בדיקות ביצועים נרחבות של יישומי שרת-לקוח.

שיפור אמינות הבדיקות, למשל על ידי: אוטומציה של השוואת כמות גדולה של נתונים.

הדמיה של התנהגות המערכת הנבדקת.

**מסגרות בדיקה (test frameworks)**

הערה: השימוש במונח מסגרות בדיקה (test frameworks) נפוץ בתחום בדיקות התוכנה בשלוש משמעויות שונות לפחות:

ספריות בדיקה הניתנות לשימוש חוזר או להרחבה ואשר אפשר להשתמש בהן לבניית כלי בדיקה מכונות גם רתמת בדיקות (test harness).

סוג העיצוב של אוטומציית הבדיקות; למשל: מונחית נתונים (data driven), מונחית מילות-מפתח (keyword driven).

שם כולל לתהליך ביצוע הבדיקות.

במסגרת תכנית לימודים זו, המונח ישמש בשני המובנים הראשונים.

**6.1.2 סיווג כלי בדיקה**

קיימים כלי בדיקה רבים, התומכים בהיבטים שונים של בדיקות.

ניתן לסווג כלים לפי קריטריונים כגון: מטרה;

מסחרי/ חינמי/ קוד פתוח/ תוכנה שיתופית;

טכנולוגיה;

ועוד.

בתכנית לימודים זו הכלים מסווגים לפי פעילויות הבדיקה בהן הם תומכים.

ישנם כלים אשר תומכים בברור בפעילות אחת, אחרים תומכים ביותר מאחת אך מסווגים תחת הפעילות העיקרית לה הם מיועדים.

כלים מסָפָּק יחיד, בעיקר כאלה שתוכננו לעבוד יחד, משווקים לעיתים כמערכת משולבת אחת.

ישנם כלים שאופי התמיכה שלהם מתאים יותר למפתחים. למשל: כלים המופעלים במהלך בדיקות רכיבים או בדיקות אינטגרציה של רכיבים, כלים אלה מסומנים באות "**מ**" ברשימה שלהלן.

**כלים פולשניים ואפקט הגשושית**

כלים מסוימים הינם פולשניים

במילים אחרות: הם עשויים להשפיע על תוצאת הבדיקה.

לדוגמא: תזמון המערכת בפועל (actual timing) משתנה כתוצאה מהפקודות הנוספות המבוצעות על ידי הכלי, או שמתקבל ערך שונה כאשר מודדים את כיסוי הקוד.

ההשלכות של כלים פולשניים ידועות בשם אפקט הגשושית (probe effect).

**אפקט הגשושית (probe effect)**

אפקט הגשושית

האפקט של כלי בדיקה על רכיב או מערכת כאשר הללו נמדדים על ידו, למשל על ידי כלי בדיקות ביצועים,

או מנטר (monitor).

לדוגמא, הביצועים עשויים להיות מעט גרועים יותר כאשר מופעלים כלי בדיקת ביצועים {כי הכלים עצמם דורשים משאבים, הנגרעים מהרכיב או המערכת הנבדקים}.

הערה: האפקט עלול להשפיע על תוצאת הבדיקה, ובמקרים מסויימים אפילו לעוות אותה משמעותית.

**6.1.3 כלים לתמיכה בניהול תהליך הבדיקה והבדיקות**

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

סקירה של כלי ניהול הבדיקות הקיימים בשוק: HP , מיקרוסופט, IBMואחרים.

**כלי ניהול בדיקות (test management tools)**

ברבים מכלים אלה קיים ממשק לכמה יכולות: ביצוע בדיקות, מעקב אחר פגמים, ניהול דרישות;

זאת, יחד עם תמיכה בניתוח כמותי...

ודיווח על מושאי הבדיקות (test objects).

חלק מהכלים תומכים גם בקישור מושאי בדיקה למפרטי דרישות: לעיתים יש בהם יכולות בקרת גרסה עצמאיות או ממשק לבקרת גרסה חיצונית.

**כלי ניהול דרישות (requirements management)**

בכלים אלה מפורטות הדרישות מהתוכנה, וכן:

התכונות(attributes) של הדרישות, כולל סדר העדיפות של כל דרישה.

הכלי מספק מזהה ייחודי (unique identifier) לכל דרישה, וצריך לאפשר קישור ((tracing בין דרישות לבדיקות.

כלים אלה יכולים לסייע בזיהוי דרישות לא עקביות או חסרות.

כאמור, יכולות אלה הן לרוב חלק מכלי ניהול בדיקות כללי.

**כלי ניהול אירועים (incident management)**

לרוב קרויים "כלי ניהול באגים" (למעשה: כלים למעקב אחר פגמים).

בכלים אלה שומרים ומנהלים דוחות על אירועים כגון:

פגמים, כשלים, בקשות לשינויים בעיות ואנומליות.

כלים אלה אף מסייעים בניהול "מחזור החיים" של אירועים;

בחלקם קיימת תמיכה בניתוח סטטיסטי.

גם כאן, יכולות אלה הן לרוב חלק מכלי ניהול בדיקות כללי.

**כלי ניהול תצורה (configuration management)**

אלה אינם כלי בדיקות בהגדרה, אבל...

הם הכרחיים לשם שמירה וניהול גרסאות של הבּוֹדְקָה (מכלול מרכיבי הבדיקות, testware) והתוכנות הנלוות, בעיקר כאשר בודקים על יותר מסביבת תוכנה או חומרה אחת במונחים של גרסאות מערכת הפעלה, קומפיילרים (מהדרים, (compilers, דפדפנים וכו'.

באופן כללי אלו כלים החיוניים לשמירת איכות התוכנה: חשוב להכיר גם את הכלים המשמשים את המפתחים לניהול תצורה לרוב זה יהיה אותו כלי.

**6.1.4 כלים לתמיכה בבדיקות סטטיות**

כלי בדיקות סטטיות מספקים שיטה יעילה מבחינה כלכלית   
(cost effective) לגילוי יותר פגמים בשלבים יותר מוקדמים של תהליך הפיתוח.

וזאת על ידי ניתוחים האפשריים עוד לפני הרצת הקוד: ניתוח מסמכים, ניתוח קוד בטרם הרצתו, אימות מודלים של תוכנה.

**כלי סקירה (review)**

כלים אלה תומכים ב: תהליכי הסקירה, רשימות ביקורת (checklists), הנחיות סקירה.

הם משמשים לשמירה ולהעברת הערות סקירה בין אנשי הצוות ולדיווח על פגמים ועל התקדמות העבודה.

אפשר להיעזר בהם גם לתמיכה בסקירות מקוונות (online) במקרה של צוותים גדולים מאד או מבוזרים גיאוגרפית.

**כלי ניתוח סטטי (static analysis) (מ)**

כלים אלה מסייעים למפתחים (וגם לבודקים) למצוא פגמים לפני ביצוע בדיקות דינמיות (כלומר: לפני הרצת הקוד).

זאת, על ידי תמיכה באכיפת תקנים לכתיבת קוד כולל כתיבת קוד מאובטח (secure coding) וניתוח מבנים ותלויות (dependencies).

הם יכולים לתרום לתכנון הבדיקות או לניתוח סיכונים, בעזרת המדדים שהם מספקים ביחס לקוד

למשל סיבוכיות (complexity).

**כלי מידול (modeling) (מ)**

גם הם בעיקר בשימוש המפתחים.

משמשים לאימות מודלים של תוכנה למשל, מודל נתונים פיזי (Physical Data Model - PDM).

לבדיקת הקישורים שקיימים בין טבלאות של מסד-נתונים טבלאי (relational database).

וזאת על ידי גילוי פגמים וחוסר עקביות (inconsistency).

לעיתים קרובות כלים אלה מסייעים ביצירת מקרי בדיקה המבוססים על המודל.

**6.1.5 כלים לתמיכה במפרטי בדיקות**

**כלי עיצוב בדיקות (test design)**

כלים אלה משמשים לייצור: קלט בדיקות, קוד בדיקה שניתן להרצה (executable) ו/ או אורקלים לבדיקות ((test oracles.

זאת, מתוך: הדרישות, הממשקים הגרפיים, מודלי העיצוב (מצבים, נתונים או עצמים), או מן הקוד.

**כלים להכנת נתוני בדיקה (test‑data preparation)**

כלים להכנת נתוני בדיקה מבצעים שינויים ועדכונים של: מסדי נתונים, קבצים, תשדורות נתונים (data transmissions).

על מנת להגדיר נתוני בדיקות שישמשו במהלך ביצוע הבדיקות.

מאחר והנתונים המיוצרים בעזרת הכלי הם אנונימיים, מובטח שלא נוצרות בעיות של אבטחת מידע

(security through data anonymity).

**6.1.6 כלים לתמיכה בהרצת בדיקות וברישום**

**כלי הרצת בדיקות (test execution)**

כלים אלה מאפשרים הרצה אוטומטית או חצי-אוטומטית של בדיקות באמצעות כתיבת תסריט בשפת תסריטים (scripting language) ותוך שימוש בקלט ובתוצאות צפויות המוגדרים בתסריט.

בדרך כלל כלים אלה מפיקים יומן בדיקות (לוג) עבור כל הרצת בדיקה.

ניתן להשתמש בחלק מכלים אלה גם להקלטה של מהלך בדיקה ובעתיד להריץ שוב את אותה הבדיקה.

הדבר מאפשר אוטומטיזציה של בדיקות גם ללא כישורי פיתוח קוד.

לרוב הם תומכים ב: שינוי פרמטרים (parameterization) של ערכי נתונים בבדיקות, והתאמות (customization) אחרות של הבדיקות, בעזרת: שפת תסריטים (scripting language).

מחייב ידע מסויים בכתיבת סקריפטים כאלו או ממשק משתמש גרפי (GUI-based configuration).

מאפשר התאמה ללא צורך בכתיבת סקריפטים.

הסבר: חשיבות היכולת לביצוע התאמות כמתואר לעיל.

הסבר: ההבדל בין סקריפט (תסריט) לבין קוד.

**מערכת/ רתמה (harness) לבדיקות יחידה (מ)**

(בשימוש המפתחים, האחראים על בדיקות יחידה לקוד שלהם)

מערכת לבדיקות יחידה (unit‑test framework tool) ו/או רתמת בדיקות (test harness).

מסייעת בבדיקת רכיבי מערכת או חלקים ממנה על ידי הדמיה (סימולציה) של סביבת ההרצה של מושא הבדיקה.

ההדמיה מושגת באמצעות אובייקטים מדומים (mock objects) כגון תותבים (stubs) או דרייברים (drivers).

חזרה על הסבר ההדמיה, שלמדנו בשיעורים קודמים.

**כלי השוואה (test comparator)**

כלי ההשוואה משווים: קבצים, מסדי נתונים או תוצאות בדיקות.

כלי הרצת בדיקות בדרך כלל כוללים כלי השוואה דינמיים, אך השוואה לאחר ההרצה ניתנת לביצוע על ידי כלי השוואה נפרד.

כלי להשוואת בדיקות עשוי להשתמש באורקל בדיקות, במיוחד במקרה שהוא ממוכן (אוטומטי).

**כלים למדידת כיסוי (coverage) (מ)**

בשימוש המפתחים, כי הכלים קשורים להרצת הקוד ולעתים כרוכים בתוספות לקוד)

כלים אלה מודדים את אחוז הקוד אשר נבדק על ידי סידרת בדיקות באמצעים פולשניים (intrusive) או בלתי-פולשניים.

האחוז מחושב ביחס לסוגי הכיסוי השונים, למשל: כיסוי משפטים (statement coverage).

כיסוי הסתעפויות או החלטות (decision coverage, branch coverage).

כיסוי תנאים (condition coverage) וכיסוי תנאים/החלטות (condition/decision).

כיסוי תנאים מרובים (multiple condition coverage).

וכן קריאות מודול או פונקציה.

**כלים לבדיקות אבטחה (security)**

כלים אלה משמשים להערכת מאפייני האבטחה של תוכנה, כולל הערכת יכולת התוכנה להגן על:

סודיות מידע, שלמות מידע (data integrity), אימות זהות (authentication), הרשאה (authorization), ואי-איבוד הרשאות (non-repudiation).

זמינות (availability).

כלי אבטחה מתמקדים לרוב בהיבטים מסויימים של התוכנה: טכנולוגיה, פלטפורמה, ו/או מטרות ספציפיות.

**6.1.7 כלים לתמיכה בביצועים ובמעקב**

**כלי ניתוח דינמיים (dynamic analysis) (מ)**

כלי ניתוח דינמיים מגלים פגמים שנחשפים רק כשהתוכנה רצה, למשל: תלויות זמן (time dependencies) דליפות זיכרון (memory leaks).

הכלים הללו בדרך כלל משמשים בבדיקות רכיבים ואינטגרציה של רכיבים, וכן כאשר בודקים תוכנת ביניים (middleware).

**כלי דיבוג (ניפוי באגים, debugging ) (מ)**

משמשים את המפתחים למציאת הבאג (הפגם) שגרם לתקלה.

**כלי בדיקות ביצועים/ עומסים/ מאמץ**

כלי לבדיקת ביצועים (performance testing tool).

כלי בדיקות עומסים (load testing tool).

כלי בדיקת מאמץ (stress testing tool.

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 (ramp‑up pattern),וכן התדירות והאחוז היחסי של התנועות (transactions).

ההדמיה של העומס מתקבלת באמצעות יצירת משתמשים מדומים (וירטואליים) המבצעים סידרה נבחרת של תנועות על ידי מספר מחשבי בדיקה.

מחשבי בדיקה אלה נקראים "מחוללי עומסים" (load generators).

**כלי לבדיקת ביצועים (performance testing tool)**

כלי לבדיקת ביצועים:

כלי שנועד לתמוך בבדיקות ביצועים, שלו בדרך כלל שתי יכולות עיקריות: יצירת עומס, ומדידת תנועה (transaction).

יצירת העומס יכולה לדמות ריבוי משתמשים ו/או נפחים גדולים של נתוני קלט.

במהלך הביצוע מדידות זמן תגובה נלקחות מתנועות נבחרות, ונרשמות.

כלי בדיקות ביצועים מספקים לרוב דוחות המבוססים על: רישומי הבדיקה (test logs), גרפים של העומס כנגד זמן התגובה.

**כלי בדיקות עומסים ומאמץ**

כלי בדיקות עומסים (load testing tool):

כלי המיועד לתמוך בבדיקות עומסים, אשר יכול לדמות עומס גובר והולך, למשל מספר המשתמשים הבו-זמניים ו/או מספר התנועות (transactions)במהלך פרק-זמן שפורט.

כלי בדיקת מאמץ (stress testing tool):

כלי התומך בבדיקות מאמץ... תזכורת להגדרת בדיקות מאמץ: בדיקות המתבצעות כדי להעריך את התנהגות מערכת או רכיב כשהעומסים מגיעים לרמה המירבית 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, מנתחים באופן רצוף את נתוני השימוש ומייצרים דוחות על נתונים אלה.

הכלים גם מתריעים על תקלות אפשריות בשירותי המערכת.

**6.1.8 כלים לתמיכה בצרכי בדיקה ספציפיים**

**הערכת איכות הנתונים**

נתונים הם המוקד של פרויקטים מסוימים כמו למשל פרויקטים להמרת נתונים או הגירתם, או מחסני נתונים (data warehouses).

מאפייני נתונים אלו שונים בנפחם וברמת החשיבות (הקריטיות) שלהם.

בהקשרים כאלה יש צורך בכלים להערכת איכות הנתונים, כלים שיכולים לסקור ולאמת את המרת הנתונים בהתאם לכללי ההגירה וכך להבטיח שמידע שעבר עיבוד הוא נכון, מלא, ותואם את התקן שהוגדר בהקשר-ספציפי זה.

**כלים אחרים קיימים לשם בדיקות שימושיוּת (usability).**

**6.2.1 תועלת וסיכונים אפשריים בכל הכלים לתמיכה בבדיקות**

קניה של כלי לא מבטיחה הצלחה בשימוש בו.

עבור כל סוג כלי נדרשת לרוב השקעת מאמץ נוסף לשם השגת תועלת אמיתית לטווח ארוך.

השימוש בכלים לבדיקות טומן בחובו תועלת ואפשרויות רבות, אך קיימים גם סיכונים.

**תועלת אפשרית משימוש בכלים**

צמצום המטלות המונוטוניות (החוזרות על עצמן); למשל: הרצת בדיקות נסיגה, הקלדה חוזרת של אותם נתוני בדיקה, השוואת הקוד לתקני כתיבת קוד.

עקביות וחזרתיות (repeatability) טובות יותר; למשל: בדיקות המבוצעות עם כלי באותו סדר ובאותה תדירות, בדיקות המופקות מן הדרישות.

הערכה נטולת פניות; למשל: מדידות סטטיות, כיסוי.

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: תפקודיות (פונקציונליות) ונוחות השימוש.

אומדן-חסר (underestimating) של המשאבים, הנדרשים להכנסתו הראשונית של הכלי:

הזמן, העלות , המאמץ (כולל הכשרה, והזדקקות למומחים 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 שנדרש לתחזק את מה שהכלי מייצר.

הסתמכות יתר על הכלי: שימוש בו כתחליף לעיצוב בדיקות או שימוש בבדיקות אוטומטיות כאשר עדיף להשתמש בבדיקות ידניות.

הזנחת בקרת הגרסאות של נכסי הבדיקה (test assets) הקשורים לכלי.

הזנחת קישוריות ובעיות ביכולת חיבור ובפעולה הדדית (interoperability) בין כלים חיוניים; למשל בין: כלי ניהול דרישות, כלי בקרת גרסאות, כלי ניהול אירועים, כלי מעקב אחר פגמים, כלים מספקי תוכנה שונים.

הסיכון שספק של כלי מסוים ייעלם מהשוק, או: יפסיק את שיווק הכלי, יעביר אותו לספק אחר.

היענות נמוכה של הספק לפניות: בנושאי תמיכה, שדרוג ותיקון פגמים.

בכלים חינמיים או כלים שמקורם הוא קוד-פתוח (open source): סיכון של השעיה (הפסקת הפרויקט suspension).

סיכונים בלתי צפויים, כגון: חוסר יכולת לתמוך בפלטפורמה חדשה.

**6.2.2 שיקולים מיוחדים עבור סוגי כלים מסוימים**

**כלי ניהול בדיקות**

כלי ניהול בדיקות צריכים להתממשק עם כלים אחרים וכן עם גיליונות אלקטרוניים (!)

כדי שניתן יהיה להפיק מהם מידע שימושי בפורמט המתאים לצרכי הארגון.

במיוחד חשובה היכולת להוציא ולקבל נתונים בפורמט אחיד ומקובל, כמו: אקסל

קבצי ערכים מופרדי-פסיק (Comma-Separated Values - CSV) רצוי גם וורד, כולל עיצובים.

**כלי ניתוח סטטי(static analysis)**

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.

אמנם 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, על ידי סינון התחלתי של חלק מן ההתראות.

**כלי ביצוע בדיקות**

כלי ביצוע בדיקות מריצים את מושאי הבדיקות באמצעות תסריטי בדיקות ממוכנים (automated test scripts).

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 ידני נראית מושכת, אבל...

גישה זו אינה מתאימה בפרויקטים שדורשים כמות גדולה של תסריטים אוטומטיים.

תסריט מוקלט חוזר על מקרה בדיקה יחיד ו"לינארי" מסלול אחד בקוד, עם נתונים ופעולות מסוימים מאד.

סוג זה של תסריט עלול להיות בלתי יציב אם בזמן הרצתו מתרחשים במערכת אירועים לא צפויים

(שלא קרו בזמן הקלטת התסריט).

כמו כן הוא קשה לתחזוקה ומחייב שינויים תכופים בעיקר עם כל שינוי בממשק המשתמש (GUI).

**גישת בדיקות מונחית-נתונים (data driven testing)**

גישת זו מפרידה את קלט הבדיקות (הנתונים) שנרשמים לרוב בגיליון אלקטרוני, ומשתמשת בתסריט בדיקות כללי (גנרי) יותר המסוגל לקרוא את נתוני הקלט ולבצע את אותו תסריט בדיקות עבור ערכי קלט שונים.

בודקים אשר אינם מכירים את שפת התסריטים, יכולים ליצור נתוני בדיקה עבור תסריטים מוגדרים מראש (גנריים) אלה.

**עוד טכניקה מונחות-נתונים**

ייצור נתונים בשעת ההרצה באמצעות אלגוריתם ,המבוסס על פרמטרים הניתנים להגדרה ואשר מסופקים ליישום, במקום להציב אוסף נתונים קבועים (hard-coded) בתוך הגיליון האלקטרוני.

למשל, כלי עשוי להיעזר באלגוריתם המייצר מספר משתמש אקראי.

כדי שהרצות חוזרות של הבדיקה ישתמשו באותם מספרי משתמש (לצורך חזרתיות):

האקראיות נקבעת על ידי שימוש בגרעין אקראיות (seed, random seed).

**גישה מונחית מילות-מפתח (keyword‑driven testing)**

בגישה זו הגיליון האלקטרוני מכיל מילות-מפתח המתארות פעולות שיש לנקוט בהן (מכונות גם מילות-פעולה, action words) ונתוני בדיקה.

הבודקים יכולים להגדיר בדיקות באמצעות מילות המפתח אותן ניתן להתאים ליישום הנבדק וזאת גם אם אינם מכירים שפת תסריטים.

**הערות**

מומחיות טכנית בשפת כתיבת תסריטים דרושה בכל הגישות או מצד הבודקים או מצד המומחים לאוטומציה של בדיקות.

בלי קשר לטכניקת התסריטים בה משתמשים...

יש צורך לשמור את התוצאות הצפויות עבור כל בדיקה, לצורך השוואה בזמן ההרצה.

**6.3 הכנסת כלי לארגון**

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

הערכת בשלות הארגון, נקודות החוזק והחולשה שלו, וזיהוי הזדמנויות לשיפור תהליך הבדיקות באמצעות כלים.

הערכת הכלי מול דרישות ברורות וקריטריונים אובייקטיביים.

**שיקולים בבחירת כלי חדש**

הערכת האפקטיביות של הכלי: עם התוכנה הנבדקת ובמסגרת התשתית הקיימת,

וכן זיהוי שינויים הדרושים לשם שימוש אפקטיבי בכלי.

הערכה כזאת דורשת שימוש מעשי בכלי, על ידי ביצוע מהלך של הוכחת היתכנות והתאמה (POC).

הערכה של הספק: כולל יכולתו לספק הכשרה ותמיכה, והיבטים מסחריים או של ספקי שירות תמיכה במקרה של כלים לא מסחריים.

זיהוי דרישות פנים-ארגוניות להכשרה בשימוש בכלי: אימון (coaching), חונכות (mentoring).

הערכת צרכי ההכשרה, בהתחשב במיומנויות האוטומציה של צוות הבדיקות.

הערכת יחס עלות-תועלת מבוססת על נתוני פרויקט אמיתי (concrete business case).

**תכנית ניסוי (פיילוט) להכנסת כלי חדש**

הכנסת הכלי לארגון מתחילה בתכנית ניסוי (פיילוט) עם היעדים הבאים:

למידת פרטים רבים יותר אודות הכלי.

הערכת מידת התאמתו של הכלי לנהוּג בחברה ולתהליכים הקיימים, והחלטה מה צריך להשתנות.

קבלת החלטה לגבי שיטות מוסכמות: לשימוש, ניהול, אחסון ותחזוקה של הכלי ושל נכסי הבדיקה (test assets): למשל שיטה מוסכמת למתן שמות לקבצים ולבדיקות, יצירת ספריות והגדרת המודולאריות של סדרות בדיקות.

ניסיון להעריך אם התועלת הצפויה תושג בעלות סבירה.

**הגורמים להטמעה מוצלחת של כלי חדש**

בין הגורמים התורמים להטמעה מוצלחת של הכלי בארגון ניתן למנות:

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 המשתמש בכלי.

הפקת לקחים מפעילות כל הצוותים.