

עידן חקלאי- הכלכלה התבססה על פעילות משקי בית, בעיקר בחקלאות

מאה 18- מהפכה תעשייתית- הקמת בתי חרושת. עוד תחום התפתח יחד עם המהפכה התעשייתית- תחום החינוך!

משנת 1992- הפס הרחב עבר לשימוש אזרחי- עידן הידע והצריכה. הרבה יותר נגישות לכל מקום הרבה יותר מגוון, הרבה יותר ידע....

איכות

רמה של משהו, טיב- לבדוק עד כמה משהו טוב, סטנדרט (תקן), רמת התאמה לצרכים של משתמשים

הגדרה שלנו: התאמה לדרישות

דרישות- של הלקוח, בעל הענין

דרישות מספל קפה איכותי:

חם

לא מר/מתוק מדי

חלב בתוקף תקין

חלב שקדים

נוח לאחיזה

לא דליל/לא סמיך

לא מלא עד הסוף (שלא יישפך)- מכסה

ישנן דרישות מוגדרות. ישנן דרישות "משתמעות" – שהן הכרחיות אך לא תמיד ברות הגדרה.

דרישות עבור תוכנה: לקוחות, של הספקים (שוק), רגולציה

אנחנו מדברים על..בדיקות תוכנה.

איכות= התאמה לדרישות- מוגדרות, משתמעות. מקור לדרישות: לקוח, יצרן, רגולציה (בעלי ענין במערכת)

Testing- פעולה אקטיבית של הפעלת תוכנה לשם בדיקה

Qc- בקרת איכות- מוצר. מה עשינו?- האם ספקנו ללקוח את אשר בקש? פעולות כגון- סקר דישות הלקוח, פגישות חוזרות עם הלקוח, קבלת משוב מהלקוח

Qa- הבטחת איכות- תהליך "נכון". איך עשינו. האם תהליך העבודה שלנו היה הכי נכון ויעיל? האם ניתן



QA, QC and Testing in software development process

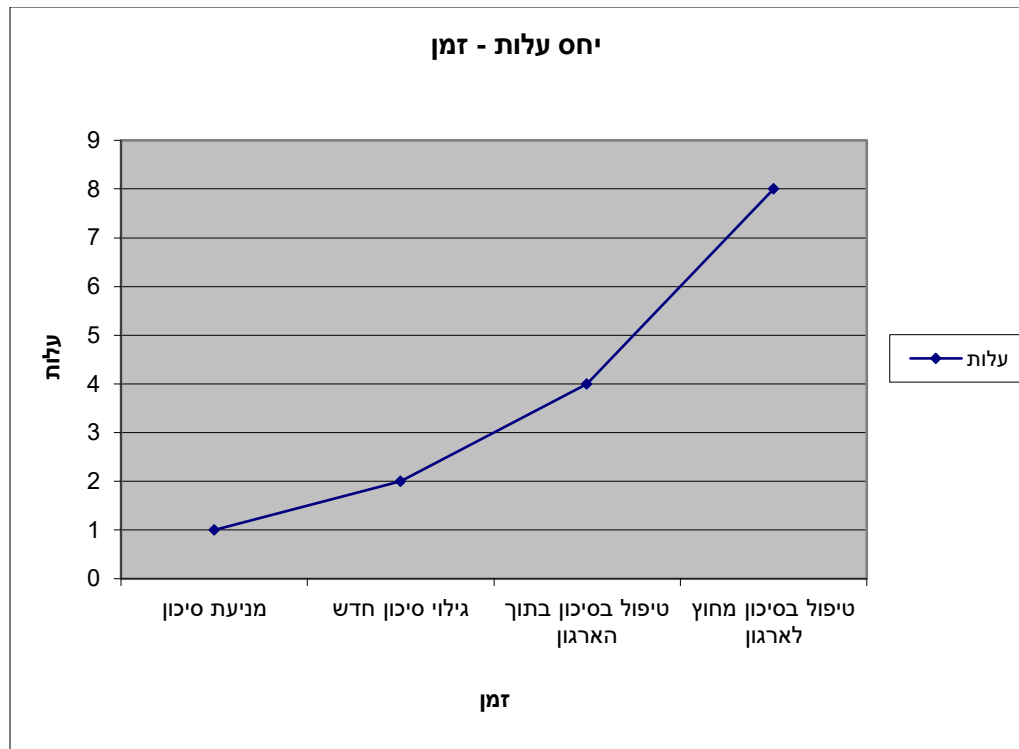
(וכדאי) לבצעו שוב?

"מורשת קרב"- הזכרו במקרים בהם היו בעיות בהבטחת איכות?

- אמש- האטה באינטרנט בישראל. בגלל עדכון של משחק הפורטנייט שהופץ
- השיר גנגם סטייל- מונה הצפיות התאפס- כי השדה לא הצליח להכיל את הערך (!!)
- חברת ריטליקס- בעיה בתוכנת הקופה הרושמת- הפסד כספי
- באג 2000- היה או לא...? 1.11.2011- מספר חברות אשראי גילו שכל כרטיסיהן נהיו פגי תוקף

אז...למה להשקיע בהבטחת איכות???

כדאיות כלכליות.



בדיקות תוכנה- הגדרה:

תהליך בו מתפעלים את התוכנה באופנים שונים, על מנת למצוא טעויות וכדי לוודא, תוך נקיטת פעילויות הולמות, שהתוכנה עומדת בתנאי דרישות בעל העניין בתוכנה וכן, אם יש, דרישות ותקנים תעשייתיים, ממשלתיים ועולמיים

מדדי איכות של תוכנה:

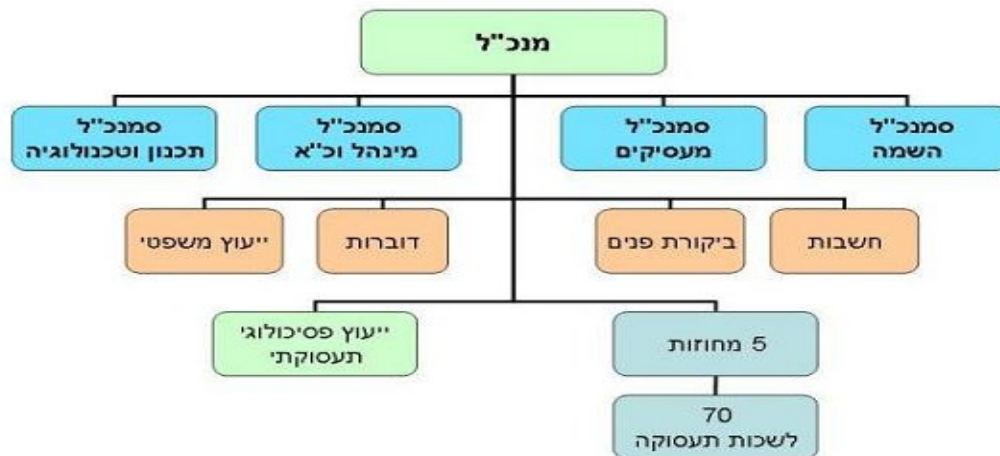
- נכונות= האם הפלט של התוכנה- הוא נכון? איך נוודא: ברור עם הלקוח, אב טיפוס, השוואה עם פלט אחר- שידוע שהוא נכון
- שמישות usability- חווית משתמש. נוחות, יעילות, שביעות הרצון.
- אמינות= עבודה רציפה לאורך זמן- בתנאים קבועים. תבדק טרם אספקה ללקוח
- בר- תחזוקה=עבודה רציפה לאורך זמן- כאשר המוצר אצל הלקוח – תנאים משתנים.
- בר שימוש חוזר= תוכנה שבנויה כך שניתן להשתמש ולעדכן אותה בקלות
- עקיבות traceability – ניתן להוכיח התאמה חד ערכית בין הדרישות לבין המימוש של הדרישות בקוד ובבדיקות.
- כל פעולת שינוי ועדכון- תהיה פשוטה יותר

- "תקשורת עם הלקוח בשפה שלו"
- בצועים performance- איך (וכמה) התכנה מנצלת משאבים- זמן, משאבי חומרה
- Scalability- הפעלת התוכנה בסדרי גודל שונים, סביבות שונות...

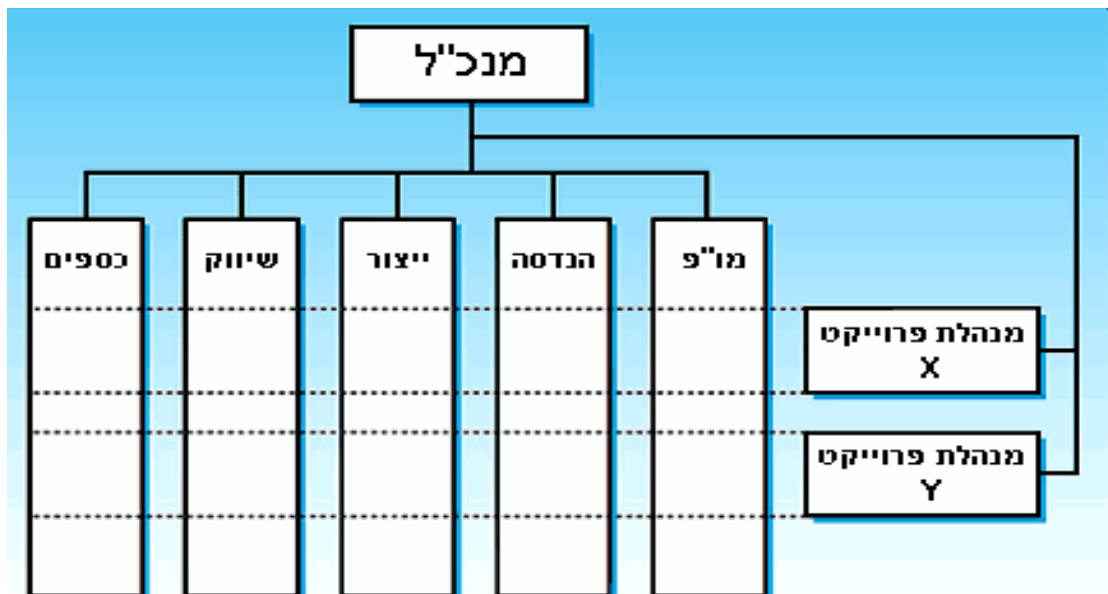
בדיקות בתוך ארגון

ניתן לחלק ארגונים ל2 צורות התנהלות:

פונקציונלית:



מבנה מטריציוני:



ארגון מטריציוני	ארגון פונקציונלי	
דיווח כפול: למנהל הפרויקט, ולמנהל המקצועי	דיווח למנהל הישיר	דיווח
עבודה יותר ממוקדת	עבודה קצת יותר לרוחב	בזהירות- איפה יש מומחיות נקודתית ואיפה "כולם עושים הכל"
כאן יש יותר רגולציה.		בזהירות- איפה יותר נהלים מוסדרים ומוקפדים?

הקשר בין בודקים לשאר הארגון

- תלות מוחלטת בין בודק ומפתח= המפתח בודק את הקוד של עצמו. יתרון: מכיר אין צורך להסביר ולפרט. חסרון: מוטה!
- בדיקת עמיתים- מפתחים בודקים זה את עבודתו של זה=יתרון: פותר את בעיית ההטיה. חסרון: זו לא בדיקה מקצועית
- צרף אנשי בדיקות יעודיים לנושא, בתוך ארגון הפיתוח. יתרון: בדיקה מקצועית. חסרון: חסר ניהול נפרד לנושא הבדיקות.
- בדיקות מנוהלות במקביל לפיתוח ולא בתוכו. תפקיד חדש- מנהל בדיקות
- בדיקות (לעיתים גם קוד)- מחוץ לארגון
 - מיקור חוץ outsource-
 - Off shore - מיקור חוץ, שנעשה עי ארגון מארץ אחרת.
 - On shore - מיקור חוץ, שנעשה עי ארגון מקומי
- Near shore – ארגון חיצוני ברדיוס גיאוגרפי קרוב

ממשקים של צוות בדיקות בארגון

- פיתוח – בשביל התוכנה
- לקוחות- בשביל להבין את הדרישות
- הנהלה- משאבים, לוחות זמנים
- תמיכה טכנית- אם יש בעיה שמגיע מכיוונם
- כתיבה טכנית- כל תוצרי ההדרכה וההטמעה של התוכנה- צריכים להבדק.
- העברה ללקוח- dev ops

סוגי בדיקות

ניתן לחלק את הבדיקות, לסוגים שונים. החלוקה על כל "קוביה" - דיכטומית. אך בדיקה אחת, יכולה להיות כמה סוגים:

סוגי בדיקות				
פרוגרסיה/רגרסיה/שפיות/אימות	קופסא שחורה/אפורה/לבנה	פונקציונלית/לא פונקציונלית	דינמית/סטטית	ידנית/אוטומטית
פרוגרסיה-התקדמות. נבדוק את כל היכולות החדשות של המערכת.	קופסא = אחסון הקוד.	בדיקה פונקציונלית: מה?? התאמה לדרישות שהתקבלו.	בדיקה דינמית: יש צורך בהרצת המערכת הנבדקת לשם בדיקה. בדיקה סטטית: אין צורך בהרצת המערכת הנבדקת, לשם הבדיקה.	בדיקה ידנית: בצוע בדיקה עי' בודק אנושי, בדיקה אוטומטית: בצוע בדיקה באמצעות קוד
רגרסיה- אחורה. נבדוק יכולות ישנות של המערכת- שלא נפגעו עקב העדכונים.	בקופסא שחורה- הקוד אינו נראה. ניתן לבדוק קלט- פלט אך לא ניתן לראות מה קורה בפנים.	בדיקה לא פונקציונלית: איך???	הנבדקת, לשם הבדיקה. דוגמא: קומפילציה. קוד ריווי	
אימות- בדיקה שבאה לוודא האם פתרון שקבלנו לתקלה, אכן פתר את הבעיה.	אם הקוד בקופסא לבנה- אז הקוד חשוף. ניתן לבדוק גם התנהלות בתוך הקוד.			
שפיות- sanity- בדיקה מדגמית כדי לוודא שהגרסא הינה מוכנה לסבב בדיקות מעמיק.	קופסא אפורה= באמצע, לא שחור ולבן- פעילות ועקבות הקוד- חשופים לפנינו- תעבורה, לוגים, בסיסי נתונים			

כל מתודולוגית הבדיקות שנלמד, מתבססת על 7 עקרונות.

עקרון 1: בדיקות מראות אחוז מסוים מהתקלות. => ישנו תקלות שהבדיקות לא יוכלו לתפוס. אין 100% גילוי תקלות.

עקרון 2: גילוי מוקדם עדיף=>תקלה שמתגלה מוקדם יותר- קלה לפתרון וזולה יותר

עקרון 3: bug clustering - "צרות באות בצורות" – תקלות נוהגות להגיע יחד. ואף לגרור חדשות. עלינו ליצר ניהול כזה אשר יהיה גמיש לתקלות ולכלול חזרות ומיצוי בדיקת איזור מסוים

עקרון 4: פאראדוקס "מדביר החרקים"- אם נחפש תקלות אך ורק באיזור אחד- "נפספס" תקלות באזור אחר התוכנה הנבדקת. "לא לחפש רק מתחת לפנס"

מה ניתן לבדוק:

כפתור login

ססמה- כניסה של משתמש רשום עם ססמה נכונה

סוגי אותיות, תווים מותרים ואסורים בקלט

אורך ססמה- כמות תווים והגבלת אורך

בדיקת כניסה עם אי מייל

שפה נכונה

דוגמא: הערך Weinberg העיף את המערכת.

מה עוד אפשר לבדוק פה? (שמות אחרים לא שחזרו את הבעיה)

- שמות באורכים דומים
- אותיות גדולות, שילוב עם קטנות
- שמות דומים
- הבעיה האמיתית: רצף האותיות ein התרגם ל \n

גם בבדיקות אשר חוזרות על עצמן, יש לגוון ככל האפשר (data שונה, התקנה שונה...)

עקרון 5: בדיקות "מתישות" אינן ישימות

דוגמא: נתונה מערכת אשר:

מערכת הפעלה	שפה	חיבור db
Os1	אנגלית	Db1
Os2	עברית	Db2
Os3		Db3

נתון, שמחזור בדיקות אחד ייקח 10 ימים. כמה יקח לבדוק הכל? $3 \times 2 \times 3 = 18$

180 יום כדי לבדוק הכל. אין לנו את הזמן הזה....מה אפשר לעשות?

- להעביר כמה שיותר לאוטומציה
- למצוא מנגנון של תעדוף ובחירת בדיקות מייצגות

עקרון 6: בדיקות הינן תלויות **הקשר**. כשמגדירים בדיקות, יש להתחשב בתחום הדעת של התוכנה. יש להתחשב במשתמשים.

עקרון 7: השקרייות של העדר השגיאות- גם אם המערכת נבדקה ונמצאה יציבה, עדין אין זה אומר שהיא טובה למשתמשים. יש לשמור על אינטרס המשתמש

מבחינת תכנון ובצוע בדיקות: יש לערב כמה שיותר משתמשים

סוגי בדיקות				
ידני/אוטומטי	דינמי/סטטי	פונקציונלי/לא פונקציונלי	קופסא שחורה/אפורה/לבנה	פרורגסיה/רגרסיה/שפיות/אימות

בדיקה פונקציונלית=מה (עבודה לפי דרישות התכנה)

בדיקה לא פונקציונלית= התנהלות. איך. פחות מדידות, מערבות שיקול דעת, יותר אכיתניות מאשר כמותיות.

בדיקה לא פונקציונלית- בדיקת שימושיות

שימושיות usability – חווית משתמש: עד כמה המערכת אינטואיטיבית, יעילה (תפוקה/משאבים), שביעות רצון של המשתמש.

אז איך בודקים שימושיות?

- הערכות מומחים
- פסיכולוג רשת- התמחות בחזוי התנהגות אנושית
- בלשנות= חקר השפה
- הפעלת משתמשים (נסיינים)

עבור 80% מתקלות השימושיות העקריות, כמה נסיינים נצטרך? 5

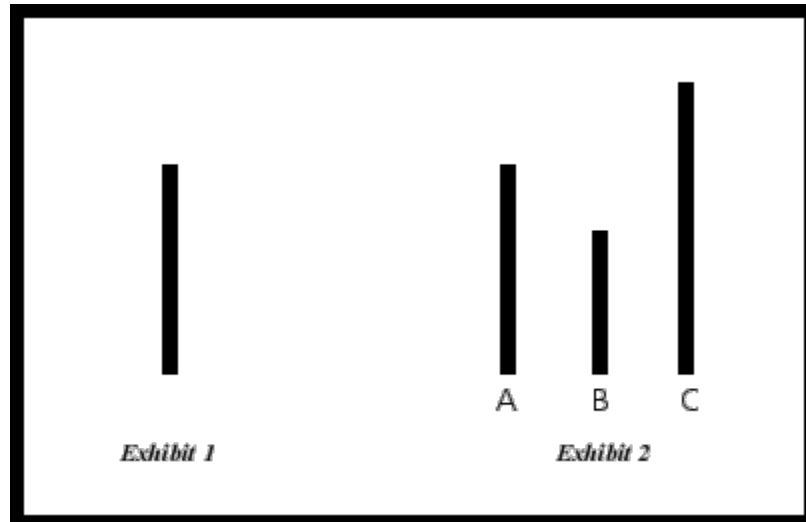
אתגרים בעת הפעלת נסיינים:

- אפקט הותרון- ריצוי מראיין

בראיון ישיר- יכולה להיות הטית התשובות ע"י הנסיין (המראיין)- עקב הנטיה לריצוי מראיין.
מה ניתן לעשות?

- לבקש משוב אנונימי
- נסיינים ללא "אינטרס"
- אם יש, השוואה לנתוני עבר
- לאמר למשתמשים שבדקים דבר אחד ולמעשה לבדוק דבר אחר

○ אפקט סלומון אש- לחץ חברתי. לאיזה קו מימין, מתאים הקו משמאל?



אם נסיינים ישמעו תוצאות של נסיינים אחרים- זה עלול להשפיע גם על התוצאות שלהם.

לכן- רצוי להפריד נסיינים

דרך נוספת להבנת הנסיינים- "מפת חום" מעקב אחרי תנועות עיניים ועכבר של משתמשים כדי להבין, על מה מסתכלים יותר בתוך המערכת.

- בדיקות במעבדה- עבודה לפי היוריסטיקות

קוי מנחה (היוריסטיקה) לבדיקות שימושיות- של נילסן- הוגדרו 10 "קוים מנחים" לשם בדיקת שימושיות

1. חיווי מצב מערכת- המערכת תידע את המשתמש במצבה, בכל עת.

דוגמא בגימיל- רואים מעטפה בצבע אדום, שמתמלאת=>המערכת רומזת למשתמש שהיא עסוקה ויש להמתין

2. התאמה בין המערכת לעולם האמיתי-יש לתקשר עם המשתמשים בשפה ברורה עם דוגמאות מחי היומיום.

מה הבעיה כאן:



3. שליטת משתמש וחופש פעולה- אפשרות לתת גמישות, לחזור לשנות ולבטל פעולות
4. עקביות וסטנדרטים- רצוי לעקוב אחרי סטנדרטים דומים בעיצוב אתרים
5. מניעת טעויות- נגדיר זרימת פעולות, ברירות מחדל- כך שיגנו ככל האפשר, על המשתמש
6. זיהוי ולא זכרון- ננסה לא להתבסס על משאבי המשתמש בכל מקרה!- לא נסמוך על זכרון של המשתמש ולא על ידע קודם שלו. נפרט בדיוק: צעד X מתוך Y, מסכי עזרה הסבר...)
7. גמישות ויעילות שימוש- תמיכה במשתמשים בעלי נסיון שונה במערכת הנבדקת
8. עיצוב מינימלי ואסתטי- נרצה לצמצם עומס חושי, כדי להקל על המשתמש
"פרק הפוקימון"
9. עזרה בזיהוי, ניתוח והתאוששות מטעויות- תמיכה לאחר שנעשתה טעות משתמש
10. עזרה ותיעוד- מסכי עזרה, בוטים, (סיור וירטואלי)

נגישות accessibility

בדיקות נגישות accessibility- תמיכה במשתמשים בעלי מוגבלויות ראייה, שמיעה, מוטוריקה, קוגניציה
נגישות- דרישה לפי חוק- לפיכך ישנו תקן רשמי

W3c

רמות נגישות: AAA AA A

התקן מתייחס ל-4 עקרונות:

- **תוכן נתפס** - חוויי שמיעתי כתחליף לחזותי, ולהפך. דוגמאות: משתמשים בעלי עיוורון צבעים- יהיו צבעים שלא יראו. למשל- כחול ייראה להם כשחור. אם נרצה שמשתמשים רבים יראו את הגרפיקה- נתספק בצבעים בסיסיים שכולם יכולים לראות
- **ניתן להפעלה** - מקלדת, עכבר, טכנולוגיה מסייעת ו"מפת אתר"
- **ניתן להבנה** - הנחיות ברורות
- **יציב** - מערכת יציבה, לא מתרסקת תדיר- חשובה במיוחד לבעלי מוגבלויות

בדיקות לוקליזציה- תמיכה בשפה ותרבות מקומיים

שפות עתירת מלל- הסטרינגים ארוכים יותר, עלול להשפיע על עצוב Ui

בדיקות לא פונקציונליות- צד שרת

שרת- מספק בקושים- לבקשות רבות:

- בצועים (במיוחד- בצועים תחת עומס)
- אבטחה

בצועים= זמני בצוע לפעולות השרת. תחת עומס משתמשים- ביקושים שמגיעים בו זמנית

השרת עמוס...??מה ניתן לעשות?

- חסימת בקשות, תעדוף ניהול תור, לנתק או לתת פג תוקף
- "לשמור בצד" מענה בקשות- proxy – זכרון מטמון של תגובות שנדרשות הרבה
- להוסיף עוד שרת (!)
 - "אשכול"- לחבר מספר שרתים לנתב אחד, אשר יחלק את הבקשות בין השרתים- הלקוח פונה לנתב.
 - מראה- מספר שרתים הפזורים גיאוגרפית במקומות שונים. הלקוח מחליט לאיזה שרת לפנות.

בדיקות לא פונקציונליות	בדיקות (כן) פונקציונליות
איך עושה- תמיכה בפעילות הפונקציונלית למשל- האם כדי לקבל תוצאה, נדרש מאמץ לא סביר? האם זמן הבצוע לא סביר?	מה- מוודאים תוצאות של המערכת הנבדקת, אל מול האפיון המתקבל מהלקוח- מה עושה? בדיקה כמו- האם $4=2+2$?

בצד לקוח- דברנו על בדיקות שימושיות – תמיכה במשתמשים (קוי מנחה ושיטות נוספות) נגישות – תמיכה במשתמשים בעלי מוגבלויות (תקן מחייב) ולוקליזציה

ראינו דוגמא לכלי בדיקות אונליין.

בדיקות לא פונקציונליות- צד שרת

אילו לא פונקציונליות, ניתן לבצע בצד שרת?

בצועים- מודדים זמני התנהלות

- עומס = load – בקושים רבים, שמגיעים בו זמנית
- נפח volume- עבודה עם קבצי ענק- נהוג במערכות שעוסקות תהליכי אצווה
- דחק stress- עומס שיא- בבדיקה הזאת מחפשים את "גבולות"
- התאוששות recovery- תפקוד המערכת לאחר חווית כשל

בדיקות עומסים=פניות רבות לשרת, בו זמנית.

השרת עמוס...??מה ניתן לעשות? להלן 3 שיטות לפיצול העומס

- "אשכול" שרתים = cluster- מספר שרתים, מחוברים יחד לנתב load balancer- הפניות (מהלקוחות) מגיעות לנתב והוא יפנן לשרתים
- "מראה" = mirroring שרתים זהים אשר פזורים במרומות גיאוגרפיים שונים. client בוחר למי לפנות
- proxy = Cache – זכרון מטמון. "כלי עזר" שמכיל בזכרון קצר טווח בקשות פופולריות. מצד שני מול השרת- יכול להסתיר כתובת אמיתית של בקשה

בדיקות עומסים= המטרה הינה: לייצר עומס של פניות לשרת- זאת נעשה על ידי כלי ייעודי שמחולל קריאות url generator. דוגמא: jmeter

אתגרים:

- אם אחולל הרבה קריאות מאותה כתובת/דומיין- השרת יפעיל cache. זה טוב- כי זה תפקיד השרת. זה פחות טוב- כי אז אין העמסה.
 - להגדיר לשרת לעבוד בלי cache בכלל
 - מה נעשה? להטעות את השרת- שימוש במנגנון ip spoofing- נייצר כתובות וירטואליות לכל בקשה, נייצר רנדומציזציה בין הבקשות

- אפקט גשושית- probe effect נבעת בבדיקה פולשנית (כלי המדידה מתערב במערכת הנבדקת)- יש הטיה בבדיקה. אפקט גשושית probe effect.

- הצלבה של תוצאות כמה כלי בדיקות שונים
- ניתן לבדוד עד כמה שאפשר את הנבדק מהכלי הבודק.

מה בודקים?

- Hits per second - כמה קריאות השלים השרת בשניה אחת

- Requests per second - כמה קריאות קבל השרת בשניה אחת
- נחפש גם את היחס בין שני הקודמים
- Duration - משך ממוצע להשלמת קריאה
- Recovery - יעילות ההתאוששות

Fine tune - משנים הגדרות בשרת, כדי למצוא נתונים מטיביים

בדיקות אמינות

אמינות=עבודה רציפה, לאורך זמן- תנאים קבועים, בדיקה שתעשה (במעבדה) טרם שחרור המוצר ללקוח

תחזוקה= עבודה רציפה, לאורך זמן- תנאים משתנים בדיקה שתעשה בסיבת הלקוח, לאחר שהמוצר פעיל production

אמינות- איך מודדים אמינות- אחוזי הצלחה- מד שנקרא mtbf זמן ממוצע בין כשלים. בשביל אמינות טובה- נרצה mtbf כבוה ככל האפשר

- אמינות- הסתברותית
- אמינות- תלוית תוכן
- אמינות- תנאים קבועים

יתירות= redundancy לשריין "עודף" כדי להפעיל בעת הצורך

- יתירות בחומרה hardware – מושג raid - מערך של דיסקים אשר נכנסים לגיבוי דיסק ראשי
- יתירות במידע- meta data – מנגנון כמו checksum מידע נוסף למידע המועבר, רק לשם הפעלת בקרה
- יתירות בתוכנה n version programming - בייחוד בתוכנות שמבצעות אופטימיזציה- מיישמים מספר אלגוריתמים שונים לפתרון אותה בעיה, ומריצים א כולם- ומצליבים את התוצאות

בשביל להפעיל בדיקת אמינות, מעבר למעקב בעת ההעמסה- נייצר כשלים כדי לוודא התאוששות המערכת

- יצירת כשלים בחומרה hardware- ציוד תקול, ציוד מושבת
- יצירת כשל בסביבה בה התוכנה נמצאת – הפרעות רשת, תנאי לחות, לחץ טמפרטורה עוינים
- יצירת כשל בתוכנה- שליחת נתונים לא נכונים, לייצר timeout, לשתול קוד זדוני כדי לראות הפעלה של exceptions

נבדוק גם את איכות ההתאוששות בנוסף לבצועים עצמם של הפעלת היתירות.

בדיקות תחזוקה

תחזוקה- בדיקות שמתבצעות כאשר המוצר במצב תפעולי (אצל לקוח)

אתגרים:

- קשה להפריע לעבודה השוטפת
- סכנה של- תקלות רגרסיה. ולכן טרם כל פעולת תחזוקה נצטרך לבצע הערכה כמה בדיקות רגרסיה
- בעית ידע- לעיתים בעלי התפקידים התחלפו

למה בדיקות ופעולות תחזוקה?

- Corrective – הלקוח מזהה תקלה בתפעוליות המערכת ונדרש תיקון
- Adaptive- משהו בסביבת הלקוח השתנה ונדרש עדכון
- Perfective- הספק מעונין לשדרג מערכת אצל הלקוח

שאלה שתשאל בכל פעילות תחזוקה- האם לתקן, או לכתוב מחדש?- שקול של עלות תועלת

"כללי אצבע" שיכולים להשפיע על ההחלטה לכתוב מחדש:

- Mtbef מתקצר – תכיפות כשלים
- קוד ישן (בזהירות)
- מערכת שעובדת באמולציה – אמולציה= תוכנה שמגשרת בין מערכת ההפעלה לתוכנה שלנו
- סבוכות שהולכת וגדלה

במעבדה- אנחנו מיישמים "תהליך הבדיקות הבסיסי"

- תכנון- בצענו פירוק פונקציונלי לרכיבי המערכת
- ניתוח- לכל פירוק הגדרנו "תנאי בדיקה" אם כאשר אזי
- עיצוב- נעצב מקרי בדיקה- סט הוראות לבודק

תוצאה רצויה	צעד לבצוע
מהדרישות, ומפעילות הניצוח (תנאי הבדיקה)	בשביל לייצר צעדי בצוע רלבנטיים, נשתמש בשיטות עיצוב מקרי בדיקה

חשוב לפרט כי:

- בהירות מול הדרישות מהלקוח
- לא תמיד מי שכותב את הבדיקה- הוא גם זה שיבצע אותה
- בעת מציאת תקלה- חשוב להבין בדיוק, מנין נבעה

שיטות שונות- לעיצוב מקרי בדיקה

1. חלוקת שקילות – בשיטה זאת, **נחלק** כל קלט אפשרי לתוכנה, לקבוצות. בתוך כל קבוצה- כל אבריה משפיעים על המערכת הנבדקת- אותו דבר- **שקולים**. עבור מקרה בדיקה שלם- נקח מייצג אחד, מכל קבוצה

2. בקרת ערכי קצה- יש להוסיף לקבוצות השקילות את הקצוות בכל קבוצה

נתון שדה בן 3 מקומות (כל מקום- נועד לתו אחד), הקולט לתוכו מספרים בלבד _ _ _

מקרה בדיקה:

צעד לבצוע	תוצאה רצויה
הזן 123	יתקבל
הזן 2	לא יתקבל
הזן 12	לא יתקבל
הזן abc	לא יתקבל
הזן a31	לא יתקבל
הזן #@\$	לא יתקבל
הזן ריק	לא יתקבל
הזן 1.8	יתקבל
הזן 18. לצורותיו	??
הזן -12	יתקבל
הזן 12i	יתקבל
הזן e 12	
הזן 5-1	לא יתקבל
הזן 999	
הזן 000	
הזן -99	
הזן -0	
הזן 9.9	
הזן 0.0	

המכונה מקבלת אחד מ-3 כרטיסי אשראי: ויזה, דיינרס, אמריקן אקספרס

מקרה בדיקה:

תוצאה רצויה	צעד לבצוע
יתקבלו	הזן ויזה
יתקבלו	הזן דינרס
יתקבלו	הזן אמריקן אקספרס
לא יתקבל	הזן כרטיס אשראי שאינו נתמך
לא יתקבל	הזן כרטיס שאינו אשראי
	הזן ריק
	הזן אשראי בדיקה
	הזן אשראי שפג תוקפו- היום
	הזן אשראי שפג תוקפו- אתמול

המערכת מציגה עד 20 שורות במסך אחד (אם יש יותר, המערכת מציגה כפתור "הבא" או "חזור" לשם דפדוף במסכים הבאים)

מקרה בדיקה:

תוצאה רצויה	צעד לבצוע
20 שורות ללא כפתורים	הצג מסך הכולל 20 שורות
מסך ראשון עם "הבא", מסך שני עם שורה אחת ו"חזור"	הצג מסך בן 21 שורות
מסך ריק	הצג מסך ריק
מסך ראשון עם "הבא", מסך שני עם שני כפתורים יחד, מסך שלישי עם שורה אחת ו"חזור"	הצג 41 שורות
שורה אחת ללא כפתורים	הצג מסך עם שורה אחת

המערכת מאפשרת הצגת דו"ח של השיחות שבוצעו בטווח 24 שעות אחרונות

מקרה בדיקה:

תוצאה רצויה	צעד לבצוע
מציג שיחות	טווח זמן אקראי
	בדיקה בשעה 23:59
	בדיקה בשעה 00:01
	בדיקה ביממת הזזת שעון (יממה מתארכת/מתקצרת)

נתון המסך הבא: לשם פישוט נניח שעלינו לבדוק הדפסת מסמך בן 3 עמודים

Page Range

☒ All

☐ Selection

☐ Current Page

☐ Pages:

מקרה בדיקה:

תוצאה רצויה	צעד לבצוע
ידפיס הכל	כפתור all
	כפתור —selection לא לסמן כלום
	כפתור —selection לסמן רווחים
	כפתור —selection לסמן את כל המסמך
	כפתור —selection לסמן תמונה
תודפס הפסקה	כפתור —selection לסמן פסקה מסוימת
	כפתור pages- לא להזין דבר
	כפתור pages- להזין 0
	כפתור pages- להזין מספר שלילי

	כפתור pages- להזין תו
	כפתור pages- להזין (מעל למספר העמודים שיש) 4
	כפתור pages- להזין את אותו 2-2
	כפתור Current page - לעמוד כל פעם על עמוד אחר ולראות שמדפיס.

נתון המסך הבא:

מקרה בדיקה:

תוצאה רצויה	צעד לבצוע
	לבחור את אותו יום פעמיים בטווח
	טווח גדול- שנים

	תאריך סיום לפני תאריך התחלה
	טווח רגיל...
	לבחור תאריך שלא קיים למשל 31-11
	לבחר תאריך התחלה ללא סוף
	לבחור תאריך סוף ללא התחלה
	לא לבחור כלום

מעבר בין מצבים

שיטה שטובה למקרים בהם מערכת (data בתוכה) זזים בן מספר סופי של מצבים. לא ניתן להיות ביותר ממצב אחד בו זמנית.

מטרתנו: לזהות מעברים חוקיים, ומעברים לא חוקיים.

"נגן מדיה"

Play

Stop

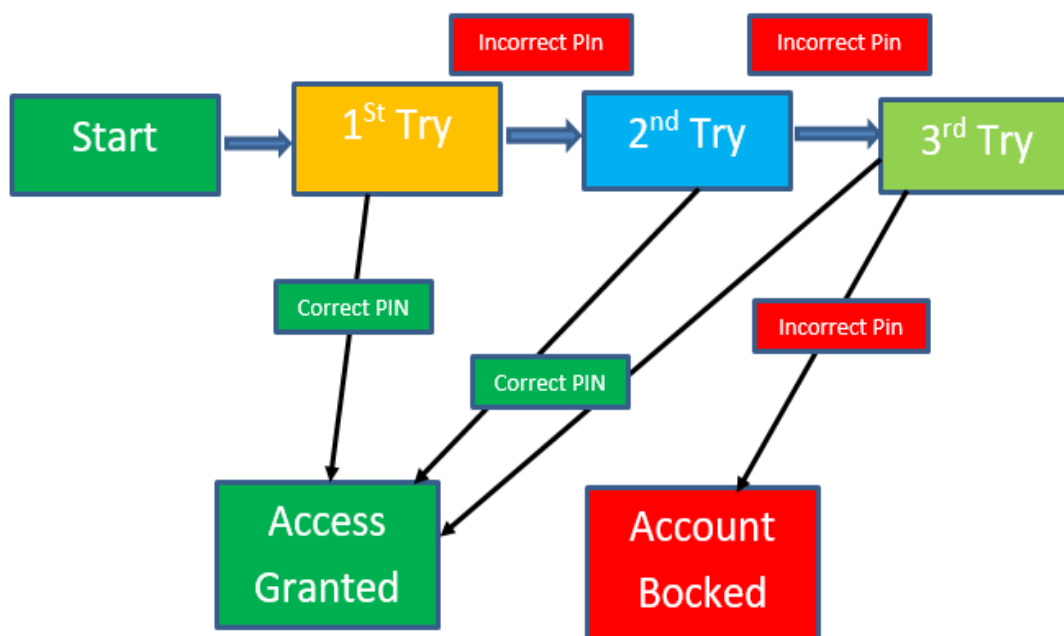
Ff/rew/next

Pause

מקרה הבדיקה:

תוצאה רצויה	צעד לבצוע
יפסיק לנגן-יעצור	התחל play עבור ל stop
מעבר לא תקין- לא יתאפשר	התחל מ stop ועבור ל pause

וכן הלאה לפי כל המעברים



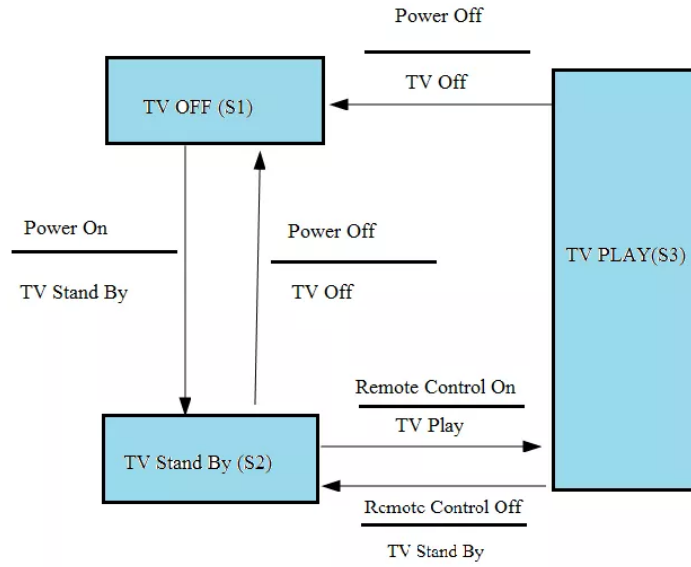
כדי לזהות את כל המעברים ניתן לייצר "טבלת מעברים" - סך הכל כדי למצוא ת כל המעברים (חוקיים ולא חוקיים) נכפול את סך המצבים* סך המעברים (כל סוג מעבר, סופרים פעם אחת). בתוך הטבלה רושמים מהו המצב אליו הגענו

מצב התחלתי	מעבר-קוד נכון	מעבר-קוד לא נכון
התחלה	לא ישים	לא ישים
נסיון ראשון	כניסה	נסיון שני
נסיון שני	כניסה	נסיון שלישי
נסיון שלישי	כניסה	חסימה
כניסה	מעבר לא חוקי (לא יקרה כלום)	מעבר לא חוקי (לא יקרה כלום)
חסימה	מעבר לא חוקי (לא יקרה כלום)	מעבר לא חוקי (לא יקרה כלום)

מקרה הבדיקה יראה כך:

צעד לבצוע	תוצאה רצויה
בנסיון ראשון הקש קוד נכון	נכנס
במצב חסימה הקש קוד נכון	מעבר לא חוקי

וכן הלאה לפי טבלת המעברים



State Transition Diagram

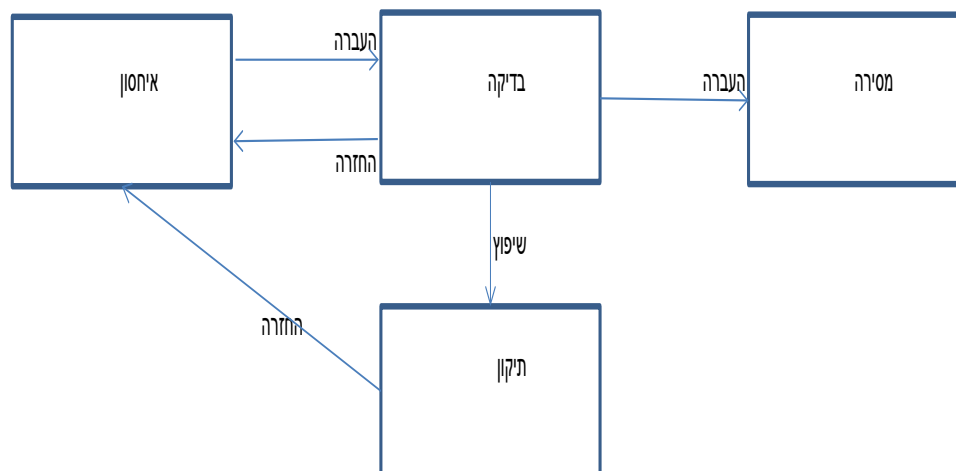
3 מצבים, 4 מעברים (מכשיר או אופ, שלט און אופ)

הדלקת שלט	כבוי שלט	הדלקת מכשיר	כבוי מכשיר	
כלום (מעבר לא חוקי)	כלום (מעבר לא חוקי)	Tv stand by	כלום (מעבר לא חוקי)	Tv off
Tv play	כלום (מעבר לא חוקי)	כלום (מעבר לא חוקי)	Tv off	Tv stand by
כלום (מעבר לא חוקי)	Tv stand by	כלום (מעבר לא חוקי)	Tv off	Tv play

מקרה הבדיקה יראה כך:

תוצאה רצויה	צעד לבצוע
כלום (מעבר לא חוקי)	טלביזה מכובה, הדלק שלט
מנגן	טלביזה stand by הדלק שלט

וכן הלאה לפי טבלת המעברים



4 מצבים, 3 מעברים

	העברה	החזרה	שיפוץ
אחסון	בדיקה	לא חוקי	לא חוקי
בדיקה	מסירה	אחסון	תיקון
תיקון	לא חוקי	אחסון	לא חוקי
מסירה	לא חוקי	לא חוקי	לא חוקי

מקרה הבדיקה יראה כך:

תוצאה רצויה	צעד לבצוע
מגיע לבדיקה	ממצב אחסון בצע העברה
מגיע למסירה	ממצב בדיקה בצע העברה

וכן הלאה לפי טבלת המעברים

שיטת עיצוב מקרי בדיקה- טבלת החלטה

בשיטה הזו נשתמש כאשר כמה קלטים, והפלט הינו תוצאה של תלות ביניהם. שילוב של קלטים- ישפיע על הקלט.

שתי מטרות:

- לפרוש את כל הקומבינציות האפשריות
- נצמצם את האפשרויות- בצורה כזו, שתסכן כמה שפחות (שלא נחמיץ בדיקות חשובות)

נתונה מכונה אוטומטית לממכר גלידה.

- יש לבחור טעם (שוקו/וניל)
- יש לבחור צורת הגשה (כוס/גביע)
- יש לבחור אם רוצים ציפוי (כן/לא)
- כמה גלידות שונות ניתן לקבל מהמכונה? $8=2*2*2$

טעם	גלידה 1	גלידה 2	גלידה 3	גלידה 4	גלידה 5	גלידה 6	גלידה 7	גלידה 8
שוקו	שוקו	שוקו	שוקו	שוקו	וניל	וניל	וניל	וניל
כוס	כוס	כוס	גביע	גביע	כוס	כוס	גביע	גביע
כן	לא	כן	לא	לא	כן	לא	כן	לא

נתונה מערכת הזמנת כרטיסים להופעה.

- יש לבחור את שעת ההופעה הרצויה (20:00, 22:00, 24:00)
- יש לבחור את המיקום הרצוי באולם (אולם, יציע)
- כמה אופציות שונות של כרטיסים ניתן להפיק מהמערכת הזו? $6=2*3$

	1	2	3	4	5	6
שעה	20:00	20:00	22:00	22:00	24:00	24:00
מיקום	אולם	יציע	אולם	יציע	אולם	יציע

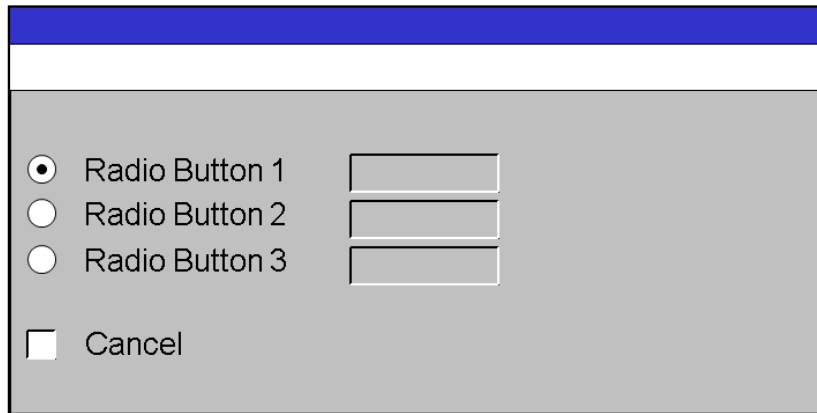
אפשר להזמין:

- לשבת או לקחת
- בפיתה/באגט/צלחת
- מנה קטנה או גדולה

• $12=2*3*2$ אופציות

	1	2	3	4	5	6	7	8	9	10	11	12
לשבת/לקחת	לשבת	לשבת	לשבת	לשבת	לשבת	לשבת	לשבת	לשבת	לשבת	לשבת	לשבת	לשבת
פיתה/באגט/צלחת	פיתה	פיתה	באגט	באגט	צלחת	צלחת	פלטה	פלטה	באגט	באגט	צלחת	צלחת
קטנה/גדולה	קטנה	גדולה	קטנה	קטנה	גדולה	גדולה	קטנה	קטנה	גדולה	גדולה	קטנה	גדולה

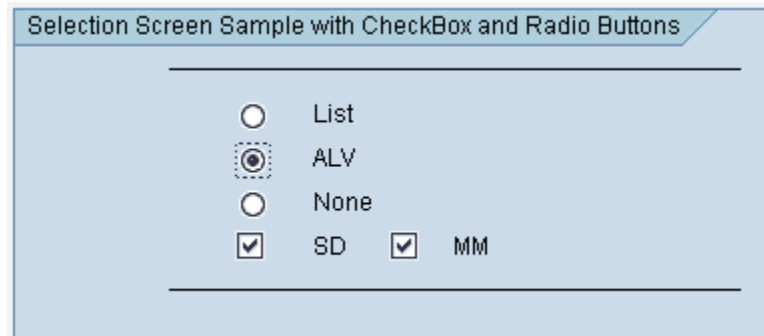
נתון המסך הבא:



כמה מקרי בדיקה אפשריים כאן?

6: radio 3 וכפול 2 עקב אפשרות להדליק או לכבות cancel

וכאן:



$12 = 4 * 3$ אופציות

וכאן:

Checkboxes

- ☒ Option 1
- ☐ Option 2
- ☐ Option 3
- ☒ Option 4

Radio Buttons

- ☐ Option 1
- ☒ Option 2
- ☐ Option 3
- ☐ Option 4

$16 * 4 = 64$ אופציות (16 אפשרויות של צקבוקס * 4 אפשרות radio)

מקרי בדיקה עם טבלת החלטה

דוגמא:

- דירה בת 2 חדרים ומטה תשלם ארנונה מופחתת
 - דירה בת 5 חדרים ומעלה תשלם ארנונה מוגדלת
 - דירות שהן במבנה בית פרטי משלמות מס גינון. דירות בבית משותף לא משלמות מס גינון.
 - דירות שהן בית פרטי חייבות להכיל לפחות 3 חדרים.
 - כמה אופציות ולידיות ניתן לבדוק כאן?
- עלינו לפרוס את כל האופציות בטבלת ההחלטה, לפי האיפיון ניתן לצמצם עמודות שאינן ישימות לבדיקה.

קלטים:

מספר חדרים

בית פרטי/משותף

פלט(תוצאה רצויה): מה משלמים

4	3	2	1	
5 ומעלה	5 ומעלה	2 ומטה	2 ומטה	מספר חדרים בדירה
פרטי	משותף	פרטי	משותף	בית פרטי/משותף
ארנונה מוגדלת+מס גינון	ארנונה מוגדלת	לא ישים	ארנונה מופחתת	תוצאה רצויה: מה לשלם

מקרה הבדיקה יראה כך:

תוצאה רצויה	צעד לבצוע
ארנונה מופחתת	דירת 2 חדרים בבית משותף
ארנונה מוגדלת	דירת 5 חדרים בבית משותף
ארנונה מוגדלת + מס גינון	דירת 5 חדרים בבית פרטי

צמצום טבלאות החלטה

נתונה טבלת ההחלטה הבאה:

8	7	6	5	4	3	2	1	
לא	לא	לא	לא	כן	כן	כן	כן	חרג מתקרת אשראי?
לא	לא	כן	כן	לא	לא	כן	כן	משלם מיידי?
לא	כן	לא	כן	לא	כן	לא	כן	אישור מיוחד?
מאושר	מאושר	מאושר	מאושר	נדחה	מאושר	נדחה	מאושר	תוצאה רצויה- סטטוס ההזמנה

אלגוריתם צמצום טבלאות החלטה:

נחפש 2 מקרים אשר התוצאה הרצויה עבורם זהה, ונבדלים זה מזה- רק במשתנה אחד.

8	7	6	5	4	2	3/1	
לא	לא	לא	לא	כן	כן	כן	חרג מתקרת אשראי?
לא	לא	כן	כן	לא	כן	-	משלם מיידי?
לא	כן	לא	כן	לא	לא	כן	אישור מיוחד?
מאושר	מאושר	מאושר	מאושר	נדחה	נדחה	מאושר	תוצאה רצויה- סטטוס ההזמנה

8	7	6	5	2	3/1	
לא	לא	לא	לא	כן	כן	חרג מתקרת אשראי?
לא	לא	כן	כן	-	-	משלם מיידית?
לא	כן	לא	כן	לא	כן	אישור מיוחד?
מאושר	מאושר	מאושר	מאושר	נדחה	מאושר	תוצאה רצויה- סטטוס ההזמנה

8	6	7.5	4/2	3/1	
לא	לא	לא	כן	כן	חרג מתקרת אשראי?
לא	כן	-	-	-	משלם מיידית?
לא	לא	כן	לא	כן	אישור מיוחד?
מאושר	מאושר	מאושר	נדחה	מאושר	תוצאה רצויה- סטטוס ההזמנה

6/8	5/7	4/2	3/1	
לא	לא	כן	כן	חרג מתקרת אשראי?
-	-	-	-	משלם מיידית?
לא	כן	לא	כן	אישור מיוחד?
מאושר	מאושר	נדחה	מאושר	תוצאה רצויה- סטטוס ההזמנה

6/8	5/7	4/2	3/1	
לא	לא	כן	כן	חרג מתקרת אשראי?
לא	כן	לא	כן	אישור מיוחד?
מאושר	מאושר	נדחה	מאושר	תוצאה רצויה- סטטוס ההזמנה

6/8	4/2	3/1	
לא	כן	-	חרג מתקרת אשראי?
לא	לא	כן	אישור מיוחד?
מאושר	נדחה	מאושר	תוצאה רצויה-

			סטטוס ההזמנה
--	--	--	-------------------------

מקרה הבדיקה יראה כך:

תוצאה רצויה	צעד לבצוע
מאושר	קבל אישור מיוחד
נדחה	חרג מאשראי ללא אישור
מאושר	לא חרג, ללא אישור

נתון האפיון הבא:

אם המוצר נמצא במלאי- ההזמנה מאושרת. אחרת- ההזמנה נמצאת בהמתנה

לקוח שאינו חבר מועדון ושאינו חשבון אשראי- הזמנתו נדחית

פלט: מאושרת, המתנה, נדחית

קלטים: האם במלאי, האם חבר מועדון, האם אשראי

8	7	6	5	4	3	2	1	
לא	לא	לא	לא	כן	כן	כן	כן	אשראי?
לא	לא	כן	כן	לא	לא	כן	כן	מועדון?
לא	כן	לא	כן	לא	כן	לא	כן	מלאי?
נדחית	נדחית	נדחית	נדחית	נדחית	נדחית	המתנה	מאושר	תוצאה רצויה

8	7	5-6	4	3	2	1	
לא	לא	לא	כן	כן	כן	כן	אשראי?
לא	לא	כן	לא	לא	כן	כן	מועדון?
לא	כן	-	לא	כן	לא	כן	מלאי?
נדחית	נדחית	נדחית	נדחית	נדחית	המתנה	מאושר	תוצאה רצויה

7-7	5-6	4	3	2	1	
לא	לא	כן	כן	כן	כן	אשראי?
לא	כן	לא	לא	כן	כן	מועדון?
-	-	לא	כן	לא	כן	מלאי?
נדחית	נדחית	נדחית	נדחית	המתנה	מאושר	תוצאה רצויה

7-8	5-6	3-4	2	1	
-----	-----	-----	---	---	--

אשראי?	כן	כן	כן	לא	לא
מועדון?	כן	כן	לא	כן	לא
מלאי?	כן	לא	-	-	-
תוצאה רצויה	מאושר	המתנה	נדחית	נדחית	נדחית

	1	2	3-4	5-8
אשראי?	כן	כן	כן	לא
מועדון?	כן	כן	לא	-
מלאי?	כן	לא	-	-
תוצאה רצויה	מאושר	המתנה	נדחית	נדחית

מקרה הבדיקה יראה כך:

תוצאה רצויה	צעד לבצוע
מאושר	מוצר במלאי, אשראי+מועדון
המתנה	מוצר לא במלאי, אשראי+מועדון
נדחית	ללא אשראי
נדחית	אשראי לא מועדון

טבלאות החלטה- תרגיל

נתון המסך הבא:

📅

01/02/2017 1:30 PM

📅

01/04/2017 2:00 PM

Apply

Cancel

🕒

1

:

30

PM

🕒

2

:

00

PM

<

Jan 2017

>

Feb 2017

Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa
25	26	27	28	29	30	31	29	30	31	1	2	3	4
1	2	3	4	5	6	7	5	6	7	8	9	10	11
8	9	10	11	12	13	14	12	13	14	15	16	17	18
15	16	17	18	19	20	21	19	20	21	22	23	24	25
22	23	24	25	26	27	28	26	27	28	1	2	3	4
29	30	31	1	2	3	4	5	6	7	8	9	10	11

ממה מתחילים...???

קלטים:

יום מוצא: יום בינואר, יום בפברואר

שעת מוצא: am pm

יום יעד: יום בינואר, יום בפברואר

שעת יעד: am pm

סך האופציות: $16 = 2 \times 2 \times 2 \times 2$ אופציות

	12	11	10	9	8	7	6	5	4	3	2	1	יום מוצא
יום מוצא	פברואר	פברואר	פברואר	פברואר	ינואר	ינואר	ינואר	ינואר	ינואר	ינואר	ינואר	ינואר	יום מוצא
שעת מוצא	am	am	Am	am	Pm	Pm	Pm	Pm	am	am	Am	am	שעת מוצא
יום יעד	פברואר	פברואר	ינואר	ינואר	פברואר	פברואר	ינואר	ינואר	פברואר	פברואר	ינואר	ינואר	יום יעד
שעת יעד	Pm	Am	Pm	Am	Pm	Am	Pm	Am	Pm	Am	Pm	Am	שעת יעד
תוצאה רצויה			לא ישים	לא ישים				לא ישים- אם מדובר באותו היום					תוצאה רצויה

וכן הלאה, ניתן להוסיף ערכי קצה, ולצמצם עוד את הטבלה

תרגיל מסכם- בדיקות למיקרוגל

מכשיר המיקרוגל מסוגל להפשיר או לחמם 4 קבוצות של מזון . יש לבחור את קבוצת המזון הרצויה והאם מעוניינים בהפשרה או בבישול.

יש להזין למכשיר המיקרוגל את קבוצת המזון מתוך ה – 4 לעיל, ואת הפעולה הרצויה (בישול/ הפשרה). משך הזמן לביצוע ייקבע על ידי המערכת, לפי הטבלה שלהלן:

סוג המזון	משך זמן הפשרה	משך זמן בישול
מרק/דייסה/נוזל	1:00	2:50
עוף/בשר	1:30	3:00
ירקות/תפוחי אדמה/אורז/פסטה	0:50	1:30
לחם/פיתות/מאפים שונים	0:40	-

במהלך הביצוע:

ניתן להחליף תוכנית או סוג מזון וללחוץ שוב על "הפעל" –משך הפעולה יתחשב במשך הזמן שהושקע עד כה. לדוגמא: ניתן להפשיר תפוחי אדמה במשך 20 שניות, להשהות, להחליף לבישול ולחדש את הפעילות – במקרה כזה, משך הזמן לביצוע הפעולה יהיה 1:10 בלבד.

מקרה בדיקה:

צעד לבצוע	תוצאה רצויה
בחר מרק והפשרה	יעבוד 1:00
בחר עוף ובשול	יעבוד 3:00
בחר לחם והפשרה	יעבוד 0:40
בחר לחם ובשול	לא ישים (יש להוסיף את שאר המקרים לפי האפיון- חלוקת שקילות)
בחר מרק ובשול, לאחר 0:59 עבור להפשרה	יעבוד 0:01
בחר מרק ובשול, עבור להפשרה לאחר 1:00	יעצור!
בחר לחם והפשרה, עבור לבשול	יעצור! לא ישים
בחר עוף והפשרה, לאחר 0:59 עבור למרק	יעבוד 0:01
בחר עוף והפשרה, לאחר 1:01 עבור למרק	יעצור!

וכן הלאה, לפי כל המעברים

מכונת משקאות- תרגיל

להלן אפיון.

יש לזהות שיטות עיצוב בדיקות רלבטיות, ולהציג מקרה בדיקה

נתונה מכונה אוטומטית למכירת פחיות משקה.

המכונה מקבלת מטבעות מסוג 25 אג, 10 אג.

נתון שמחיר פחית הינו חצי ש"ח (50 אג)

המכונה תחזיר עודף במטבעות מסוג 25 אג, 10 אג בלבד.

נתון, שברגע שמצטבר במכונה סכום של 50 אג, היא משחררת פחית משקה.

לשם התרגיל נניח שאין לנו מגבלת מלאי במכונה, ואין מגבלת מטבעות כלשהי.

קלט: מטבע - 25 סך הכל 3 אופציות, מטבע 10 סך הכל 6 אופציות. ואז בטבלת ההחלטה יהיו $18=3*6$

פלט: משקה, עודף....

1	1	1	1	1	1	1	1	11	1	9	8	7	6	5	4	3	2	1	
8	7	6	5	4	3	2	3	3	2	2	2	1	1	1	0	0	0	0	סך מטב עות 10
5	5	5	4	4	4	3	3	1	0	2	1	0	2	1	0	2	1	0	סך מטב עות 25
2	1	0	2	1	0	2	2	המכונ ה (תשח רר משק ה)						ממ תין	ממ תין	מש קה	ממ תין	ממ תין	תוצא ה רצויה

מקרה 6:

10-25<-25: משקה+עודף

25-25<-10: משקה+מצב המתנה

25-10<-25: משקה+עודף

מקרה 9:

תלוי בסדר, דומה למקרה 6

מקרה 11:

3 מטבעות של 10+1 מטבע של 25 = המכונה ברווח

--

אנחנו מדברים על..עיצוב מקרי בדיקה. זהו חלק מתהליך הבדיקות הבסיסי. בשלב הזה אנחנו רוצים להגיע למקרה בדיקה:

צעד לבצוע	תוצאה רצויה
למדנו מספר שיטות לעיצוב מקרי בדיקה	מתוך ניתוח הדרישות (שלב קודם- כתבנו תנאי בדיקה לכל דרשה אם כאשר אזי)

שיטות עיצוב מקרי בדיקה:

- חלוקת שקילות- שיטה לבחירת קלט. מחקים את הקלט לקבוצות, בתוך כל קבוצה- כל אברי הקבוצה משפיעים על המערכת הנבדקת אותו דבר (שקולים)
- ערכי קצה
- טבלאות החלטה- כאשר יש לנו מספר קלטים שנותנים פלט אחד. נפרוס את כל הצירופים האפשריים, ואז נצמצם (כאשר ניתן)
- מעבר בין מצבים- כאשר המערכת יכולה להחליף בין מספר סופי של מצבים. נזהה מעברים חוקיים ולא חוקיים

השלב הבא, לאחר עיצוב מקרי בדיקה- בתוך תהליך הבדיקות הבסיסי הינו-

שלב יישום ובצוע

שלב יישום- שלב של הערכות- לאחר כתיבת הבדיקות וטרם הרצתן.

- התקנת סביבות עבודה, רשיונות, קונפיגורציות..
- טיפול בdata:
 - ניתן להשתמש בנתונים קיימים- יש לבצע הליך ערבול srcumbling- הליך ש"מבלבל"
 - בין הרשומות כך שנשתמש ברשומות אמיתיות, אך לא נכוונות
 - דאטה סנטטי- נייצר רשומות בעצמינו

- נקח את מקרי הבדיקה ונארגן אותם למנות בדיקה- חשוב, כדי שבעת ההרצה נצמצם את זמני ההקמה וההערכות לכל בדיקה
- כיול כלים לקראת הבדיקות- הזנת ערכים מתאימים פנוי מקום

בקרת תצורה configuration management: ניהול גרסאות.

שלב בצוע- הרצת הבדיקות

מבחינת מחזור חיים- זה השלב שייקח כ-80% מהזמן

בתחילת כל מחזור בדיקה- נקבל גרסא מבקרת התצורה, ונעביר אותה בדיקת שפיות sanity- בדיקה מקצה לקצה, שטחית יחסית, כדי לוודא מוכנות של המערכת לסבב בדיקות מעמיק

הרצת בדיקות- טיפול בתקלות

תקלה: מצב בו התוצאה הרצויה (כפי שהוגדרה במקרה הבדיקה) != תוצאה בפועל (כפי שנצפתה בעת הרצת הבדיקה)

גלוי תקלה- מחייב דיווח.

עקרונות לדיווח איכותי של תקלה:

דיוק- מה באמת קרה?	יש ארגונים שמצמידים לדיווח התקלה- את מקרה הבדיקה
בידוד	יש לדווח על כל תופעה בנפרד, גם אם נצפו כמה בעיות יחד. בעיות שונות – מטופלו ע"י אנשים שונים
מידע שלם	להוסיף לדיווח כל מידע שעשוי לעזור בהבנת הבעיה: צילומי מסך, קטעי לוג, תעבורת רשת, רשומות db
אובייקטיביות	הבדיקה היא למוצר (ולא למי שפיתח אותו)

סיפורי באגים- כיצד ניתן לשפר את התיאורים- מה חסר, ואיך נשלים זאת? הכוונה אינה לתיקון התקלה, אלא שיפור בדיווח

תיאור !:

"ניסיתי להכנס למערכת עם השם והססמה הרגילים שלי, ולא הצלחתי. נכנסתי בשם וססמה של המשתמש admin, ופתחתי קובץ שהיה לי- והקובץ כולו בתצוגה לא קריאה (גברישי)"

בידוד- מפורטות שתי בעיות:

- כניסה למערכת
 - דיוק: האם קורה אצל משתמשים נוספים? האם בעיה ספציפית של המשתמש או כל ההרשאה הזאת?
- פתיחת קובץ

- דיוק: לברר בלוגים מה נעשה. שקילות על סוגי קבצים – לראות האם קורה בסוג זה או אחר, ערכי קצה על גודל הקובץ

תיאור II:

"בצעתי חיפוש לערך מי-קרוו2 והתוצאות לא נראות טוב, אחרי מספר חיפושים נוספים, הופיעו הודעות שגיאה"

דיוק- מה זה "לא נראה טוב"? מידע שלם- לצרף צילום מסך

מה קרה בחיפושים הנוספים?

חלוקת שקילות על ערכי החיפוש- קורה בתוים אותיות, מספרים, שילוב??? ערכי קצה על אורך הקלט. אולי הבעיה באורך ולא בתוכן?

תיאור III:

"צריך להזין למסך מספר חשבון ותאריך. הזנתי את הערכים 00-00-0000 ואת 111111 בהתאמה, והתוכנה קרסה"

בוצעה כאן בדיקת ערכי קצה. מה עם אברים נוספים בקבוצות השקילות השונות?

תיאור VI:

"אני מקליד הערות למערכת דרך חלון "ביקורת". כשאני יוצא מן המסמך וחוזר אליו, ההערות לא נשמרות ולא מופיעות על גבי המסמך"

יש כאן מעבר בין מצבים. האמנם הנתיב חוקי? האם משתחזר במעברים נוספים?

יש ארגונים שמצמידים לדיווח התקלה- את מקרה הבדיקה	דיוק- מה באמת קרה?
יש לדווח על כל תופעה בנפרד, גם אם נצפו כמה בעיות יחד. בעיות שונות – מטופלו ע"י אנשים שונים	בידוד
להוסיף לדיווח כל מידע שעשוי לעזור בהבנת הבעיה: צילומי מסך, קטעי לוג, תעבורת רשת, רשומות db	מידע שלם
הבדיקה היא למוצר (ולא למי שפיתח אותו)	אובייקטיביות

תיאור V:

"בצעתי מיון על רשימה באחת האופציות למיון שיש שם. אחר כך נכנסתי לאחת הרשומות וחזרתי והמיון השתנה."

גם- מעבר בין מצבים

תיאור VI:

"בחלק מהמסכים הטקסט חתוך"

באילו מסכים? מה חתוך בדיוק? רזולוציה, גודל פונט, אורך טקסט....

טופס דיווח תקלה

subject	כותרת:
דיוק, בדוד מידע שלם אוביקטיביות,	תאור:
הפירוק הפונקציונלי בו נתגלתה התקלה. חשוב, כי זה ייתן כלים להנהלת הבדיקות לקבל החלטות לגבי ההמשך	רכיב בתוכנה:
לפי המספור שנעשה בעת בקרת תצורה	מספר גרסא:
האם רגרסיה- האם התקלה היא על משהו שהיה תקין בעבר, וכעת התקלקל. חשוב לציין, כי זה כלי בתעדוף התקלה לטיפול מעקף- האם למשתמש יש דרך פשוטה להתגבר בכוחות עצמו, על הבעיה	האם רגרסיה כן/לא:
עד כמה יושפע המשתמש מהתקלה. "ציון" לרוב בסולם החל מ minor ועד critical	מעקף workaround:
עד כמה יושפע הארגון המפתח מהתקלה. "ציון" לרוב בסולם החל מ minor ועד critical	חומרה severity:
	עדיפות priority:

חומרה severity- שיקול טכני, נקבע ע"י **הבודק**

עדיפות- שיקול עסקי, נקבע עי הנהלת הפרויקט.

קביעת חומרה של תקלה

חשוב, שהבודקים ישתמשו בסולם אחיד לשם קביעת החומרה.

- "חונכות"- בודק ותיק/ראש צוות עובר על הדיווחים עם הבודקים ומתקנים חומרה לפי צורך
- ישיבות עם בעלי ענין
- עבודה לפי מודל- לרוב יפותח מודל כזה בארגון

דוגמא

על כל תקלה, נשאל שאלות. כל תשובה תקבל ציון, נסכום את הציונים וזה יהווה את חומרת התקלה severity

מה קרה?	טעות קוסמטית ועד להשבתת ציוד, קריסת מערכות...
איפה קרה?	בתהליך המרכזי של המערכת, או באופציה צדדית למתקדמים
מתי קרה?	האם בכל מקרה. האם תלוי Data האם תלוי בקונפיגורציה...
קלות מעקף (ציון הפוך)	ככל שיש מעקף יותר פשוט, הציון נמוך יותר. אם אין מעקף בכלל- הציון גבוה

1. "אין אפשרות להכנס למערכת כמשתמש רגיל אלא רק כ admin"

הערכה ב	הערכה א		
3	5	טעות קוסמטית ועד להשבתת ציוד, קריסת מערכות...	מה קרה?
5	5	בתהליך המרכזי של המערכת, או באופציה צדדית למתקדמים	איפה קרה?
5	5	האם בכל מקרה. האם תלוי בData האם תלוי בקונפיגורציה...	מתי קרה?
5	5	ככל שיש מעקף יותר פשוט, הציון נמוך יותר. אם אין מעקף בכלל- הציון גבוה	קלות מעקף (ציון הפוך)
18	20		ציון סופי של החומרה severity

2. במהלך הבקרה של המערכת, יוצאות "אזהקות שווא" אזהקה אך למעשה אין בה צורך. רגיש מדי. אם נכבה את ה אזהקות- נפסיד גם אזהקות אמת...

הערכה ב	הערכה א		
3	2	טעות קוסמטית ועד להשבתת ציוד, קריסת מערכות...	מה קרה?
5	5	בתהליך המרכזי של המערכת, או באופציה צדדית למתקדמים	איפה קרה?
5	5	האם בכל מקרה. האם תלוי בData האם תלוי בקונפיגורציה...	מתי קרה?
3	5	ככל שיש מעקף יותר פשוט, הציון נמוך יותר. אם אין מעקף בכלל- הציון גבוה	קלות מעקף (ציון הפוך)
16	17		ציון סופי של החומרה severity

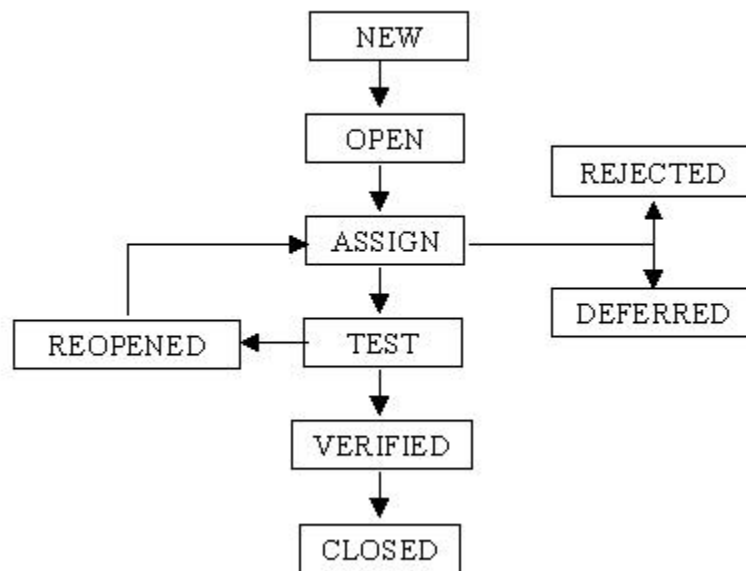
3. פתיחת תפריט "אופציות מתקדמות" קטן מדי ואין שום אופציה להרחיב אותו, כך שלא רואים הכל

הערכה ב	הערכה א		
	3	טעות קוסמטית ועד להשבתת ציוד, קריסת מערכות...	מה קרה?
	2	בתהליך המרכזי של המערכת, או באופציה צדדית למתקדמים	איפה קרה?
	5	האם בכל מקרה. האם תלוי בData האם תלוי בקונפיגורציה...	מתי קרה?
	5	ככל שיש מעקף יותר פשוט, הציון נמוך יותר. אם אין מעקף בכלל- הציון גבוה	קלות מעקף (ציון הפוך)
	15		ציון סופי של החומרה severity

מחזור חיי תקלה- דיאגרמת מצבים של גלגולים של תקלה

תקלה- לאחר שמדווחת, יכול לקרות אחד מ:

- עוברת לטיפול
 - ממצב new (לפי נהלי הארגון) ניתן להעבירה ל open
 - מועברת לטיפול assign
 - אחרי שבוצע תיקון- הקוד החדש ייצא בגרסת תוכנה חדשה
 - צוות הבדיקות יבצע בדיקת אימות- בדיקה שנועדה לוודא שהתקלה באמת תוקנה. לפעמים בדיקת אימות לא מספיקה וצריך לבצע גם בדיקות רגרסיה- ההחלטה על הרצת רגרסיה בעקבות תקונים- החלטה ניהולית שמתקבלת בשלב תכנון הבדיקות
 - אם בדיקת האימות עברה- התקלה נסגרת
 - אם בדיקת האימות לא עברה- התקלה נפתחת מחדש reopen
 - אם בדיקת האימות עברה, אבל נתגלתה תקלה אחרת- סוגרים תקלה נוכחית ופותחים חדשה. חשוב- כדי לשמור על סטטיסטיקות נכונות
- מוקפאת deferred- התקלה לא תטופל בזמן הקרוב
- מבוטלת rejected- התקלה לא תטופל



שאלה למחשבה: יש לנו בודק, שהרבה מדיווחי התקלות שלו, מסתיימים ב reject. מה זה אומר??

- דיווח לא טוב
- הבודק לא מבין את מספיק את התוכנה
- יכול להיות שיש בעית שימושיות- שמטעה את הבודקים

תזכורת:

סוגי בדיקות				
ידי/ אוטומטי	דינמי/ סטטי	פונקציונלי/ לא פונקציונלי	קופסא שחורה/לבנה/אפורה	פרוגרסיה "מה חדש בתוכנה" /רגרסיה- "לא התקלקל"/שפיות sanity/אימות- "לוודא שתוקן" מה כוללת הגרסא כרגע ביחס לגרסאות אחרות
מי מבצע את הבדיקה (בודק אנושי, או קוד)	(לא הרחבנו) מצב התוכנה הנבדקת- רצה או לא	פונקציונלי- מה (בדיקה מול דרישות) לא פונקציונלי- איך, לרוב לא מדיד	נראות הקוד- רואים? לא רואים?	

בדיקות קופסא שחורה	בדיקות קופסא לבנה	בדיקות קופסא אפורה
הקוד ב"קופסא שחורה"- לא רואים אותו. בבדיקה- ניתן לשלוט על הקלטים ולוודא את הפלטים	הקוד חשוף- בבדיקה נוכל לוודא את התנהלות הקוד ולא רק את התוצאה	אנחנו לא רואים קוד. אנחנו לא מסתפקים רק בקלט פלט. אנחנו רואים "עקבות" ל הקוד- לוגים db נתוני תעבורה

מה הכי טוב? תלוי מה המטרה שלנו.

אם נרצה להוכיח (ללוח או בעל ענין) התאמה של המערכת הנבדקת לדרישות- עדיף להשתמש בקופסא שחורה

אם נרצה לוודא שאין בזבז זכרון, שהלוגיקה נכונה, שהסבוכיות לא גבוהה מדי- עדיף בקופסא לבנה.

כמובן שזה לא "חוק" ואפשר ליישם בכל דרך שרוצים

בדיקות קופסא לבנה

נתונה פונקציה, אשר מקבלת מספר חשבון וסכום משיכה. באם יש יתרה בחשבון -אשר מוגדר במערכת, הפונקציה תאפשר משיכה. אם אין, הפונקציה תודיע- אין כסף בחשבון. אם החשבון לא מוגדר, אז לא תתאפשר פעולה.

בקופסא שחורה:

נעצב טבלת החלטות פשוטה לבדיקה זו:

מקרה בדיקה מספר 4	מקרה בדיקה מספר 3	מקרה בדיקה מספר 2	מקרה בדיקה מספר 1	
לא	לא	כן	כן	יש יתרה?
לא	כן	לא	כן	חשבון מוגדר?
ללא פעולה (מצב לא ישים)	יודיע שאין כסף בחשבון	ללא פעולה (אולי חשבון שהושחת או חשבון שנסגר מזמן)	משיכה	תוצאה רצויה

נתון קוד:

```
DrowMoney (int AccNumber,int Sum)
```

```
1  if (enoughMoney(AccNumber,Sum) then
2      PayMoney (sum)
3  else
4      ShowMessage ("Sorry, No money left in your account")
5      if AccNumber==Sum then
6          PayMoney(AccNumber)
7      endif
8  endif
```

שורה 5- פרצה לא הגיונית- אם נזין מספר חשבון הזה לסכום משיכה, נקבל כסף (לא ברור איך)

בבדיקות קופסא שחורה- קצב גילוי התקלות הוא יחסית מהיר- אבל- נעצר

בבדיקות קופסא לבנה- קצב גילוי התקלות איטי, אך ניתן למצוא הרבה יותר תקלות

מושג אחוז כסוי=סך הבדיקות שבוצעו מתוך סך הבדיקות שיש לבצע (100^*)

איך ניתן למדוד אחוז כסוי בבדיקות קופסא שחורה? - ניתן (דרך מנגנון עקיבות) למדוד כמה בדיקות נעשו לכסוי הדרישות ולראות אם כיסינו הכל.

איך ניתן למדוד אחוז כסוי בבדיקות קופסא לבנה? איך, בקופסא לבנה, נדע כמה בדיקות צריך לעשות?

נגדיר רמות שונות של כסוי:

הצהרה = פעולה לבצוע. הגדרת משתנה, השמה, חישוב, הדפסה...

for while if החלטה = תשובה לשאלה לוגית, האם להפעיל קטע קוד, או לא.

כסוי הצהרות - יש לייצר מקרי בדיקה (לייצר קלטים לקוד) כך שיופעלו כל ההצהרות בקוד

כסוי החלטות - יש לייצר מקרי בדיקה (קלטים לקוד) כך שיופעלו כל מוצאי החלטות בקוד – גם אם לא קורה בהם כלום

נתון:

```
Read x;
```

```
If (x>3) print "hello";
```

מה יהיה ערכו של X עבור כסוי הצהרות? $X=4$

מה יהיה ערכו של X עבור כסוי החלטות? $X=4, x=2$ פעם אחת להפעיל if ופעם אחת לא.

מספר מקרי הבדיקה בכסוי החלטות הינו לכל הפחות כמו מספר המקרים בכסוי הצהרות, וגם יותר ממנו.

בכל מקרה, מטרתינו- כמה שפחות מקרי בדיקה, וכמה שיותר כסוי!!!!

Read x,y;

If (x>3) then if (y>5) print "hello";

עבור כסוי הצהרות: $x=4, y=6$ שני ה if יתקיימו וההצהרה תתבצע

עבור כסוי החלטות:

$x=4, y=6$ שני ה if יתקיימו וההצהרה תתבצע

$x=2, y=6$ תנאי ראשון לא מתקיים ולכן לא נגיע בכלל לשני

$x=4, y=4$ תנאי ראשון מתקיים ושני לא

3 סטים הספיקו

כסוי תנאים מורכבים - תנאי מורכב: בטוי לוגי, המורכב מתנאים אטומיים, וקשר לוגי ביניהם

((x<6) and (y<8))

נתון קוד:

```
1 read a;
2 c=0;
3 while (a>1){
4     if (a^2>c)
5         c=c+a;
6     a=a-2;
7 }
```

נדרש למצוא ערכים של a לכסוי הצהרות מלא ולכיסוי החלטות מלא

עבור כסוי הצהרות מלא: $a>1$

עבור כסוי החלטות מלא: צריך להתקיים

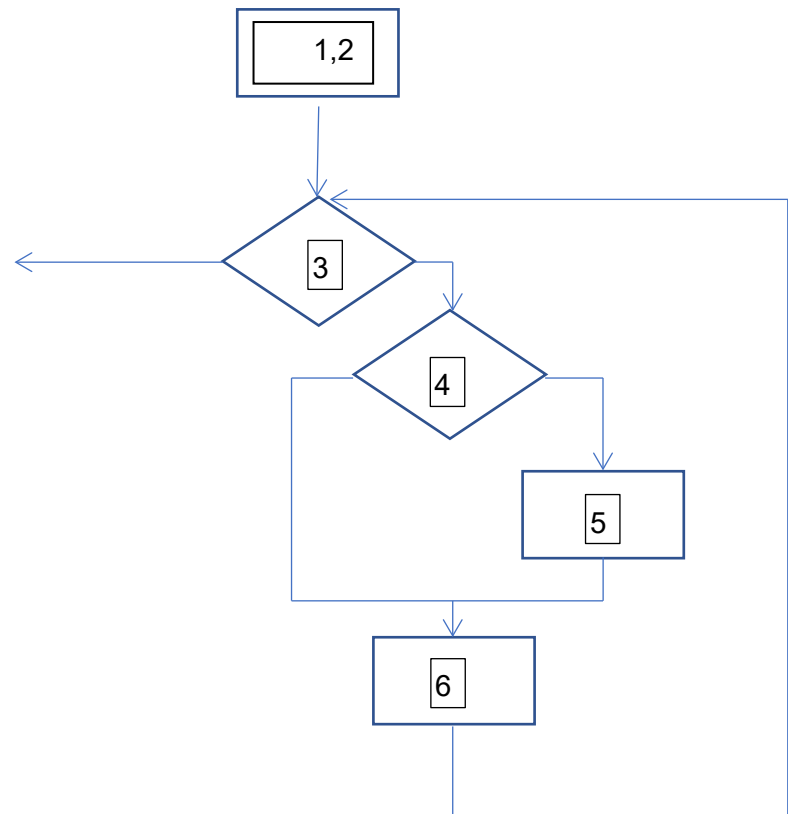
$$(a-2)^2 \leq a$$

הסיבה: בפעם השניה שנכנסים while נרצה שה if לא יתקיים- נרצה לכסות גם אופציה זו.

הפתרון למשוואה: $a=1, a=4$

ולכן, עבור כסוי החלטות מלא, נקח $a=4$

תרשים זרימה:

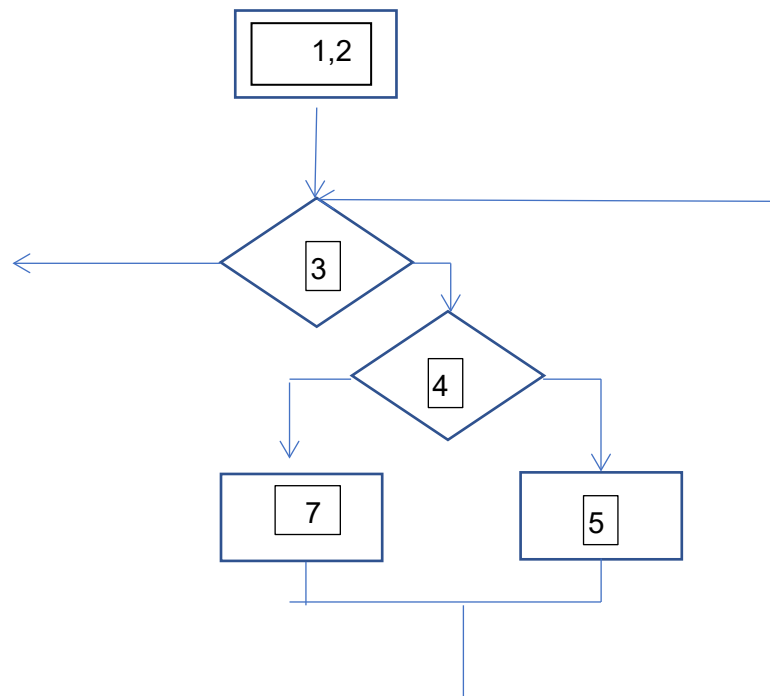


נתון קוד:

```
1 read a;  
2 b=2*a-5;  
3 while (b+a)>10{  
4     if (b+a)>7  
5         b=b-3;  
6     else  
7         b=b+3;  
8 }
```

יש פה פוטנציאל ללולאה אינסופית...נמשיך בשבוע הבא...

תרשים זרימה:

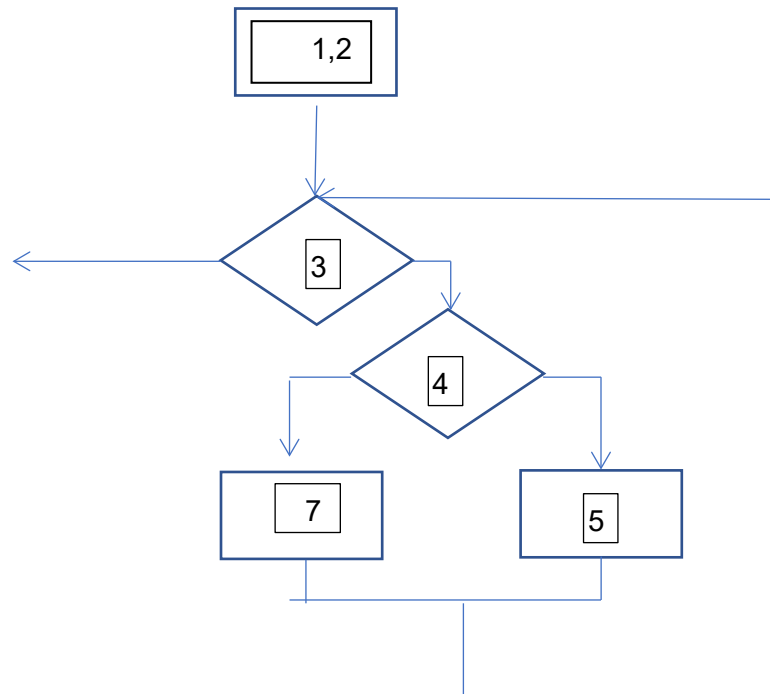


יש למצוא ערכי a עבור כסוי הצהרות מלא, ועבור כסוי החלטות מלא

נתון קוד:

```
1  read a;  
2  b=2*a-5;  
3  while (b+a)>10{  
4      if (b+a)>7  
5          b=b-3;  
6      else  
7          b=b+3;  
8  }
```

תרשים זרימה:



יש כאן פוטנציאל ללולאה אינסופית....

Path Testing

כסוי נתיבים (מסלול, דרך) - נחפש מסלולי מכבר על הקוד- המטרה לעבור על הקוד מקצה לקצה.

המספר המינימלי של המסלולים (השונים) בתוך הקוד, יחושב בעזרת מדד שנקרא cc

$cc = \text{סך ההחלטות הלוגיות בקוד} + 1$

1-Read x, y, z;

2-If (x>1 and y==0) then

3-Z=z/x;

Endif;

4-Write Z;

5-If (x==2 or z>1 then)

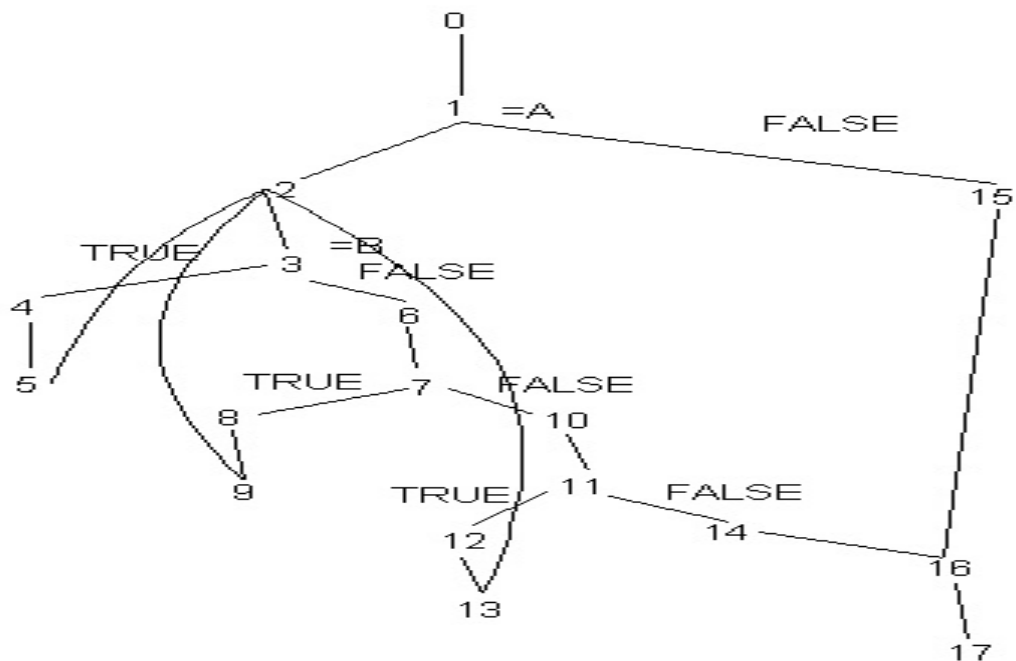
6-Z=z+1;

Endif;

7-Write z;

הcc הינו 3

- 1->2->3->4->5->6->7 •
- 1->2->4->5->6->7 •
- 1->2->4->5->7 •



סך הכל- 4 החלטות לוגיות (1,3,7,11) ולכן ה-cc היה 5

- 0->1->15->16->17
- 0->1->2->3->6->7->10->11->14->16->17
- 0->1->2->3->4->5->2->3->6->7->10->11->14->16->17
- 0->1->2->3->6->7->8->9->2->3->6->7->10->11->14->16->17
- 0->1->2->3->6->6->10->11->12->13->2->3->6->7->10->11->14->16->17

Data flow

בגישה הזאת, אנו עוקבים אחר התנהלות של data בתוך הקוד. בהגדרה data יכול לעבור בין מצבים שונים:

d- defined הגדרה או השמה של משתנה

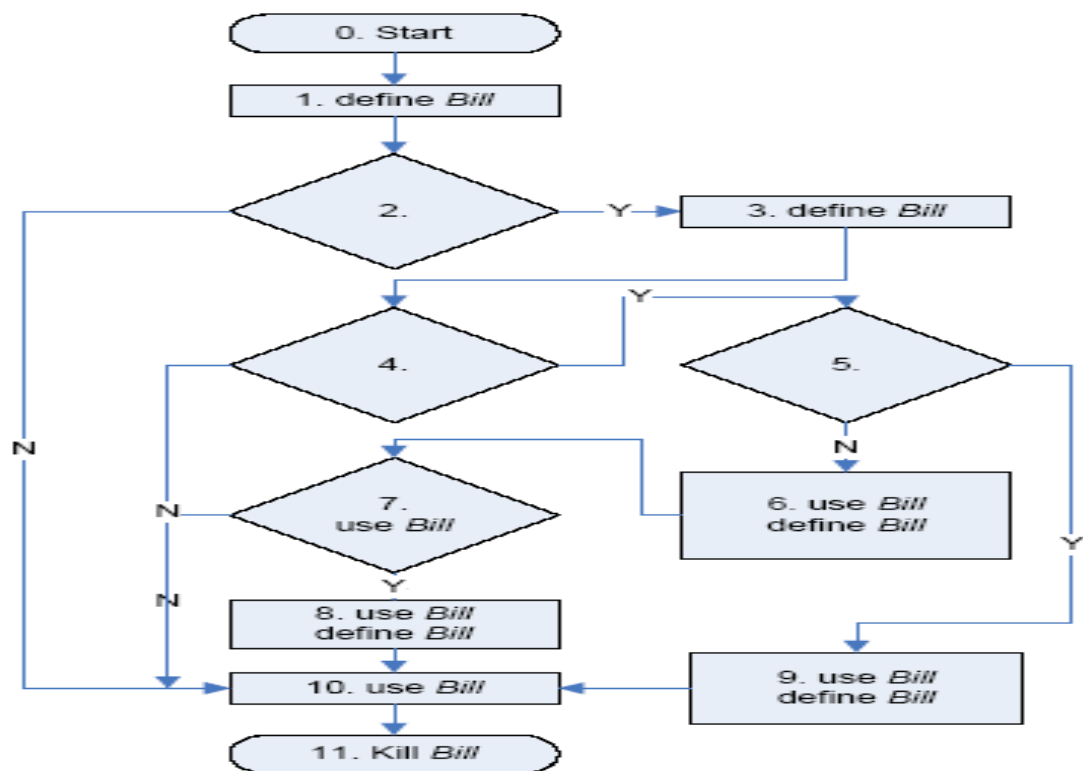
used – u חישוב, עבור הערכה של תנאי $x > 5$, $x++$,

killed – k לשחרר את הזכרון

כלומר אם יש מצבים שונים אנחנו נבדוק...מעבר בין מצבים- והדגש הינו לחפש מעברים חוקיים ולא חוקיים.

מעבר	חוקי?
d->u->k	Ok
d->u->d->k	הגדרה מיותרת, טרם שחרור
d->k->u	שימוש לאחר שחרור- לא חוקי כלל!
d->d->u->k	הגדרה מיותרת..חוקי אבל...מיותר..

```
public static double calculateBill (int Usage)
{
    double Bill = 0;
    if(Usage > 0)
    {
        Bill = 40;
    }
    if(Usage > 100)
    {
        if(Usage <= 200)
        {
            Bill = Bill + (Usage - 100) * 0.5;
        }
        else
        {
            Bill = Bill + 50 + (Usage - 200) * 0.1;
            if(Bill >= 100)
            {
                Bill = Bill * 0.9;
            }
        }
    }
    return Bill;
}
```



בדיקות אינטגרציה

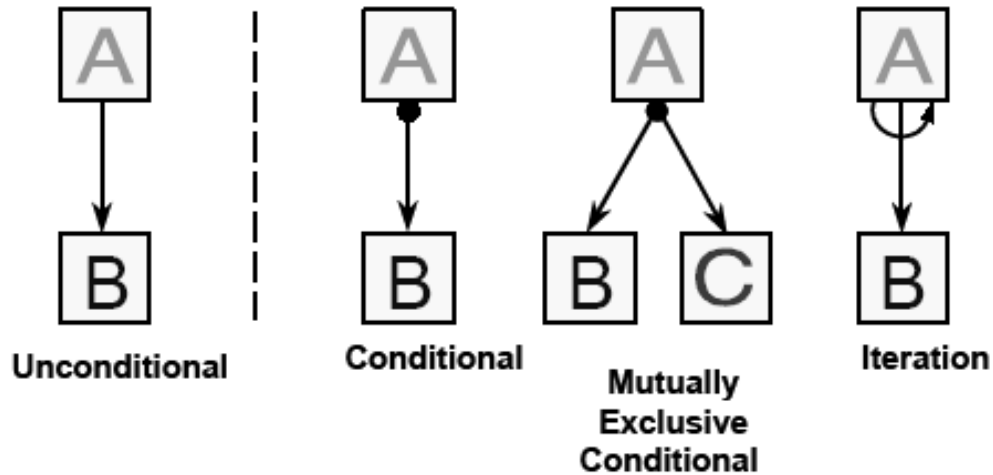
בדיקות אינטגרציה = בדיקות של קריאת קטעי קוד זה לזה. מטרת הבדיקות הללו- לוודא תקינות תקשורת בין רכיבי הקוד. (ולא- התאמה לדרישות)

המטרה שלנו: לזהות ולהפעיל את כל מסלולי קריאות הקוד האפשריים. בשביל למצוא סך מסלולים אפשרי, נשתמש במדד ה cc אך יחושב בצורה שונה.

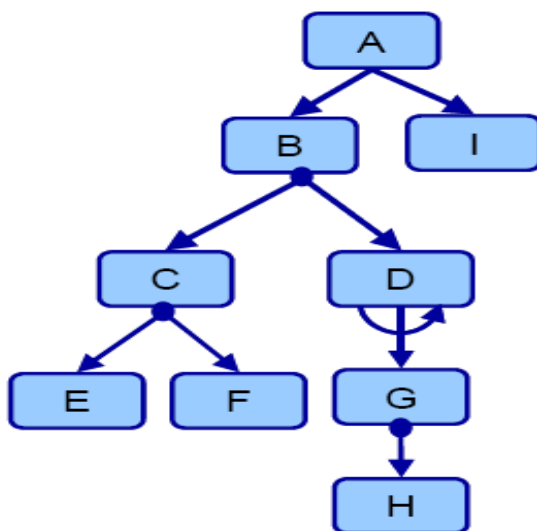
Cc תמיד מתחיל מ-1

Types of Interactions

Types of unit/component interactions:



קריאה לא מותנית (תמיד A קורא לB)	אין השפעה על CC
קריאה מותנית	$CC=cc+1$
Mutually exclusive יחידה אחת קוראת לאחת מהיחידות מתחתיה	$CC=cc+(n-1)$ כאשר נהינו סך ה יחידות להן ניתן לקרוא
קריאה באיטרציה כלומר יחידה אחת תפעיל את היחידה תחתיתה פעם אחת או יותר	$CC=cc+1$

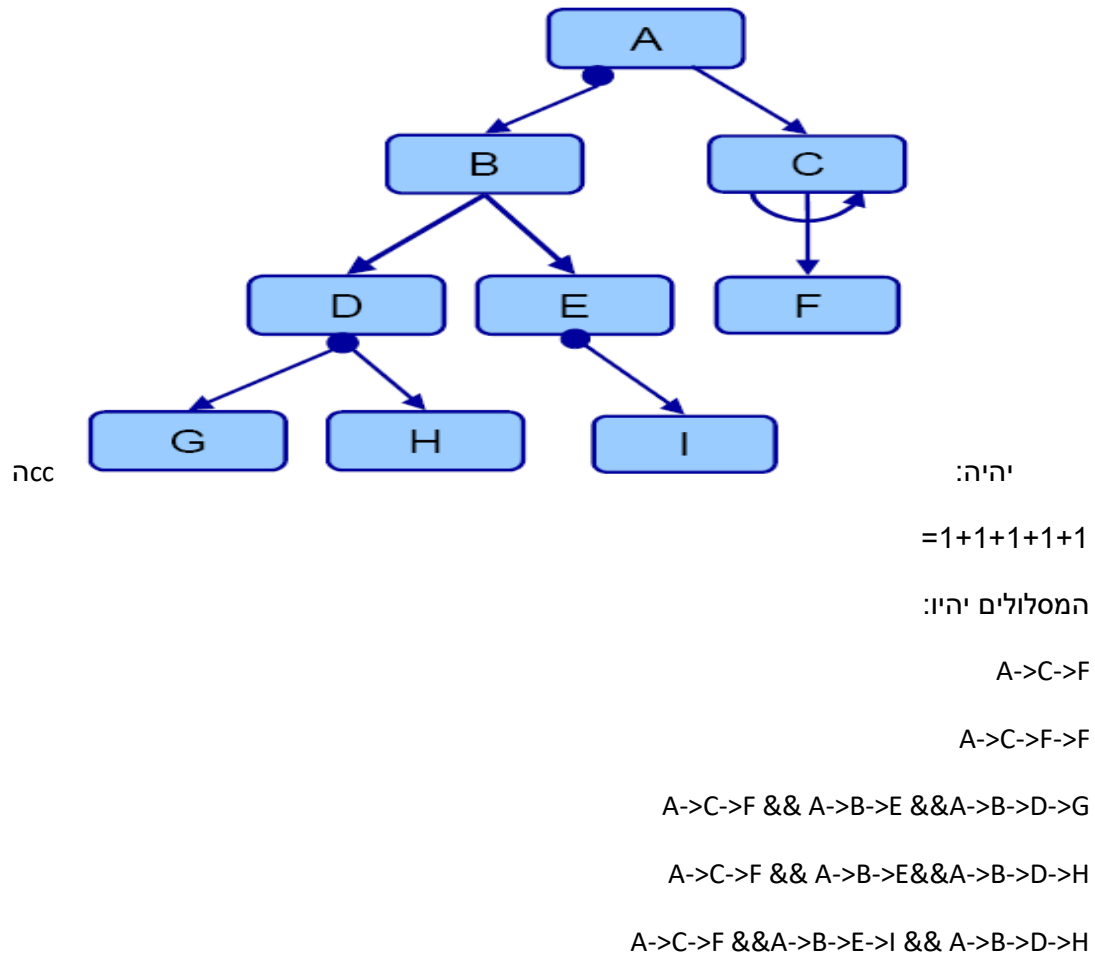


$$CC=1+(2-1)+(2-1)+1+1=5$$

- $A \rightarrow I \ \&\& \ A \rightarrow B \rightarrow C \rightarrow E$
- $A \rightarrow I \ \&\& \ A \rightarrow B \rightarrow C \rightarrow F$
- $A \rightarrow I \ \&\& \ A \rightarrow B \rightarrow D \rightarrow G$
- $A \rightarrow I \ \&\& \ A \rightarrow B \rightarrow D \rightarrow G \rightarrow G$
- $A \rightarrow I \ \&\& \ A \rightarrow B \rightarrow D \rightarrow G \rightarrow G \rightarrow H$

נתון מבנה אינטגרציה:

קריאה לא מותנית (תמיד A קורא לB)	אין השפעה על CC
קריאה מותנית	$CC=cc+1$
Mutually exclusive יחידה אחת קוראת לאחת מהיחידות מתחתיה	$CC=cc+(n-1)$ כאשר N הינו סך ה יחידות להן ניתן לקרוא
קריאה באיטרציה כלומר יחידה אחת תפעיל את היחידה תחתיתה פעם אחת או יותר	$CC=cc+1$



נתונה טבלה: students

<u>id</u>	name	class	grade	College_id

נתונה השאילתא:

Select avg(grade),class from table group by class;

- מצב בו יש רק קיבוץ אחד

בכל השורות- אותו Class

<u>id</u>	name	class	grade	College_id
1		A		
2		A		

- מצב בו מצב בו יש class רבים

<u>id</u>	name	class	grade	College_id
1		A		
2		A		
3		B		
4		B		

- מצב בו אין קיבוץ כלל (ואז יש להסיר שורות עם Class ובמקומן):

<u>id</u>	name	class	grade	College_id
5		Null		
6		Null		
7		Null		
8		null		

- ניתן להוסיף שורות אשר כוללות ערך ב Grade ולא כוללות

בהנתן הטבלאות:

Students

<u>id</u>	name	class	grade	College_id

Colleges

<u>College_id</u>	College_name	city

והשאלתא

Select name,grade ,city from students join colleges on **students.college_id=colleges.colleg_id**

שליפת נתונים ביותר מטבלה אחת.

בשביל שתנאי יתקיים:

<u>id</u>	name	class	grade	College_id
1	avi	A	100	afka

Colleges

<u>College_id</u>	College_name	city
afka	Afeka	Tel aviv

בשביל שהתנאי לא יתקיים:

- להכניס מכללה ללא סטודנטים

<u>id</u>	name	class	grade	College_id
1	avi	A	100	afka

Colleges

<u>College_id</u>	College_name	city
afka	Afeka	Tel aviv
stam	stam	stam

- להכניס סטודנט ללא מכללה

<u>id</u>	name	class	grade	College_id
1	avi	A	100	afka
2	moshe			null

- לא ניתן ליישם מקרה בו יש college_id בטבלת סטודנטים, שאינו מוגדר בטבלת colleges

Students

<u>id</u>	name	class	grade	College_id

נתונה שאילתא:

Select name from students where grade=(select max(grade) from students)

השאילתא הזאת משתמשת בתת שאילתא sub query

- תת שאילתא מחזירה ערך
 - יש ציון מכסימלי
- תת שאילתא לא מחזירה ערך (לא מוצאת שורות)
 - במקרה הזה, לא ישום
- תת שאילתא מחזירה null

- אין ציונים בטבלה

נתונה שאילתא:

Select name, grade, class from students where grade, class in (select max(grade), class from students group by class)

כאן, התת שאילתא מחזירה 2 עמודות class max(grade) ולכן ניישם את כסוי התנאים המורכבים לכל עמודה בנפרד

דברנו על- תהליך הבדיקות הבסיסי

קלט לתהליך:

- התוכנה הנבדקת
- מפרט דרישות מהתוכנה- כפי שמגיע מבעלי ענין

תכנון	פירוק פונקציונלי, בחירת גישה לבדיקות, קביעת מדדים להמשך, קביעת לוחות זמנים...
ניתוח ועיצוב	תנאי בדיקה, עיצוב מקרי בדיקה
יישום ובצוע	שפיות, הרצת בדיקות (גם אוטומציה) וגם דיווח תקלות – כל תקלה בנפרד, כולל הטיפול
הערכה ודיווח (על סיום הבדיקות)	מחליטים על סיום הבדיקות+מדווחים על כלל תהליך הבדיקות
סיום	

הערכה ודיווח

מתי יודעים, (איך יודעים) שנסתיימו הבדיקות?- לפי קריטריון היציאה- שנקבע עוד בשלב התכנון

- הערכה אל מול קריטריון יציאה

מוודאים האם אנחנו משיגים את היעד?- מעקב

- בקרה- פעולה (אקטיבית) שנועדה "להחזיר למסלול". דוגמא: התחלנו בבדיקות פרוגרסיה (=בדיקות של יכולות חדשות בתוכנה) אך קצב מציאת התקלות הינו איטי ביותר. הפתרון: התחלנו לשלב בדיקות רגרסיה (שנוי בל"ז התוכן)

- דיווח- מסמך software test report -str – מסמך אשר מסכם את פעילות הבדיקות שנעשתה.
 - את מצב המערכת הנבדקת, התפלגות וסטטיסטיקות שונות על תקלות, הרצות וכד.
 - מתאר את העבודה שבוצעה, את השינויים שקרו בפועל מתוך התכנון-ולמה, וכן המלצות להלאה. דיווח מסכם של כל התקלות – ההתפלגות שלהן, קצב מציאתן, קצב פתרון וכד.

סיום בדיקות

- סגירת קצוות פתוחים
 - בודקה וסביבות עבודה- לארגן מקרי בדיקה בצורה מסודרת בתוך כלי הבדיקות, לארגן (ולנקות) סביבות עבודה אם יש עדיין תקלות פתוחות שלא נפתרו= אפשר לעשות אחת מ-2: להקפיא את התקלה (לשמור לגרסא הבאה) או לעשות סבב סגירה מהיר.
- Lesson learned- ישיבה מיוחדת אשר בה משתתפים כל בעלי הענין בבדיקות כדי לדבר על הפקת לקחים והמלצות להבא.
- העברת ידע הלאה- ברגע שמסתיים סבב בדיקות- יש להעביר את הידע, לגורם הבא בארגון שיטפל באותה הגרסא.

איך ישתלב תהליך הבדיקות הבסיסי- בתהליך הפיתוח של התוכנה בארגון?

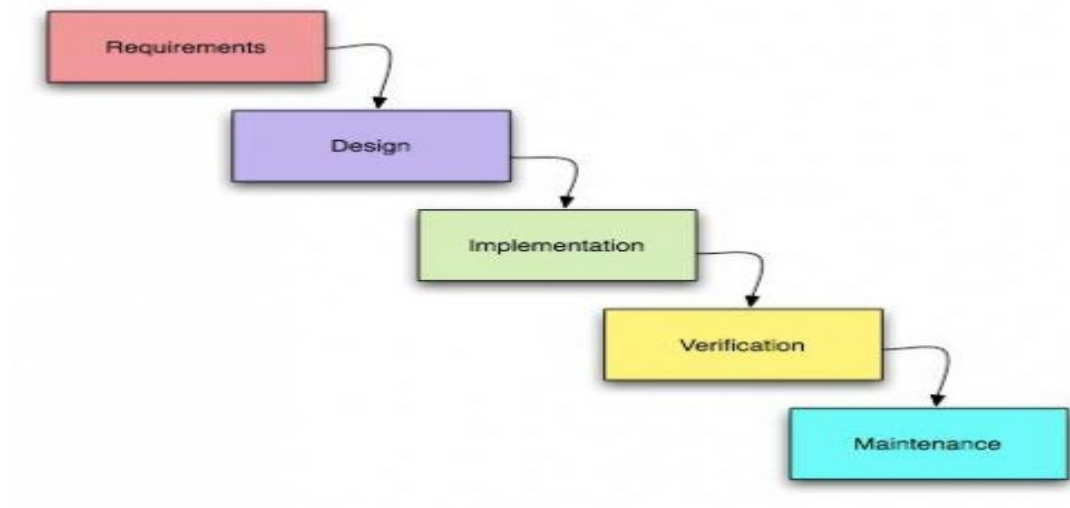
ניהול פרויקטים- פיתוח תוכנה

מודל קוי- הנחת היסוד היא שכל הדרישות מוגדרות מראש. לאחר הפיתוח- התוכנה נמסרת ללקוח (מסירה אחת)	מודל איטרטיבי- הנחת היסוד היא, שהדרישות אינן ידועות במלואן מראש. הדרישות יתפתחו במהלך הפיתוח. התוכנה תמסר ללקוח במספר פעימות
"בנה ותקן"- מסירה אחת ותיקונים כאשר המוצר מותקן אצל הלקוח	מודל מפל- שלבים ברורים ומוגדרים בעת הפיתוח, מתקדמים בפיתוח משלב לשלב
דגש רב יותר, על בדיקות רגרסיה (מסירה חוזרת=> יותר רגרסיה(רגרסיה: לוודא ששום דבר לא התקלקל, עקב עדכון הקוד)	
דגש מובהק- על אוטומציה!!!	
מנגנון עקיבות traceability- מנגנון אשר מקשר בין דרישה לבין מימושה בקוד (ובבדיקות)	ננהל את נושא תיעוד הדרישות ומקרי הבדיקה בצורה שונה user story

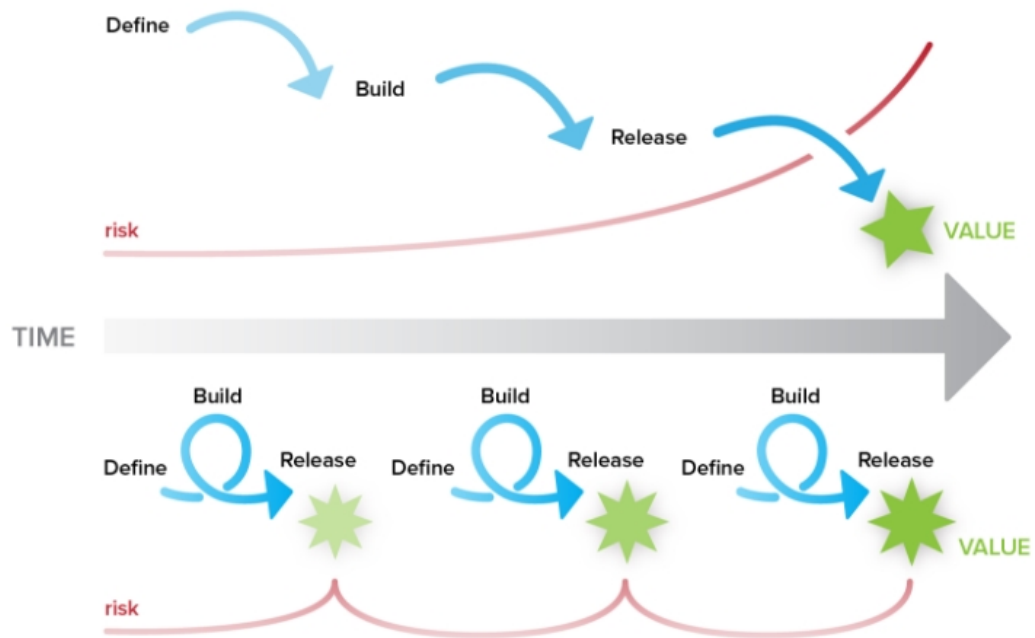
מודל מפל

השלבים הינם:

- אפיון (מפרט דרישות)
- תכן לוגי- בחירת טכנולוגיה הגדרת ארכיטקטורה, הגדרת מודולים עקריים
- תכן פיסי- תאור מפורט של המודולים שנקבעו
- כתיבת הקוד
- תחזוקה- פעילות המערכת אצל הלקוח



WATERFALL



AGILE



מסירה ללקוח	בנה ותקן	מפל	איטרטיבי
רמת רשמיות, נהלים ורגולציה בעת הפיתוח	חד פעמית	חד פעמית	חוזרת
גודל המוצר שייצא ללקוח בסיום הפרויקט	?	מוצר עצום	מוצר גדול
גודל הצוות המפתח	?	צוות ענק- לעיתים מורכב ממספר ארגונים	בהגדרה הצוות קטן- כדי לא להגיע ל"תקורת ניהול" לרוב עד 10 אנשים (מן הסתם יכול להשתנות)

רמות בדיקה- מודל וי

מודל המשלב פעילויות בדיקה שונות, בתהליך פיתוח התוכנה

צד שמאל של המודל- הינו שלבי הפיתוח כפי שראינו קודם

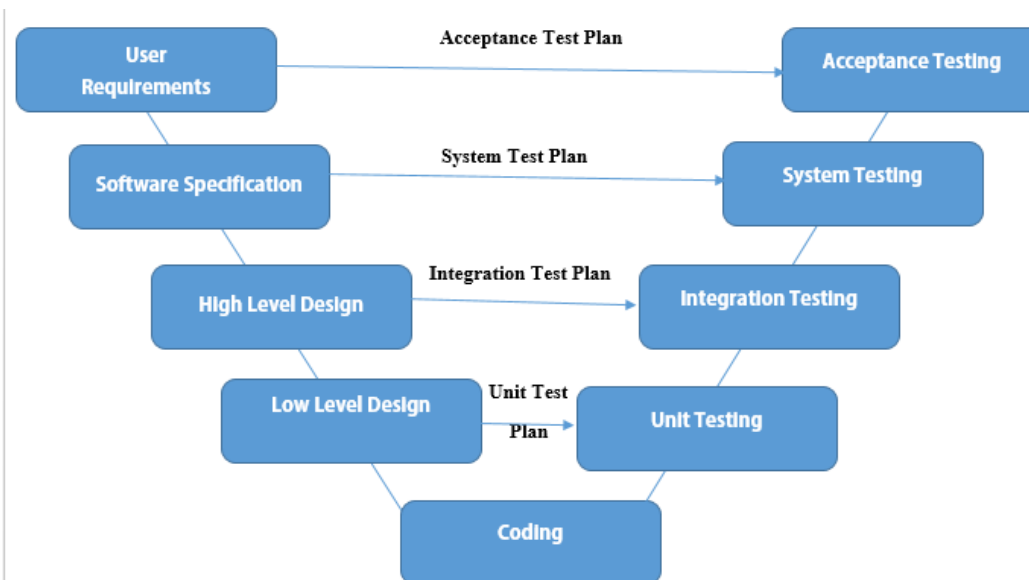
צד ימין- צד הבדיקות. כל שלב נקרא "רמת בדיקה"

רמת בדיקה= פעילות בדיקות, המתייחסת לשלב במקביל בפיתוח, לפי מודל V. בכל רמת בדיקות נפעיל מחדש את תהליך הבדיקות הבסיסי (מפני ש- בכל רמה היעדים של הבדיקות הם שונים)

מהו ההבדל (או הקשר) בין רמת בדיקה לבין סוג בדיקה (סוג בדיקה- ידני/אוטומטי, פונקציונלי/לא פונקציונלי, רגרסיה/פרוגרסיה)? ברמת בדיקות מסוימת, ניתן להפעיל סוגי בדיקות שונים. אין משמעות להפוך

- מבחינה עסקית: האם אנו בונים את המערכת הנכונה (ולידציה)- האם נתנו מענה לדרישות שקבלנו?
- מבחינה טכנית: האם תהליך העבודה שלנו נכון? (ורטיפקציה)- האם עברנו נכון משלב פיתוח אחד לבא אחריו?

בשלבים המוקדמים של הפיתוח- עדיין אין קוד. לא ניתן להריץ בדיקות על קוד... אבל- כן ניתן לבצע בדיקות סטטיות (שלא דורשות הרצת קוד)- סקרים reviews



V-Model

רמות בדיקה

בדיקת יחידה unit component - בדיקת היחידה התכנותית הקטנה ביותר. לרוב מבוצע ע"י מפתח הקוד, על סביבת הפיתוח. לעיתים, עבור השלמת הבדיקה, יש צורך ב"תחליפים" זה נקרא "רמת בדיקות" - רכיבי תוכנה אשר מהווים תחליף לרכיבים שאינם זמינים כרגע, רק לצורך השלמת הבדיקה. לרוב נשתמש כאן בגישת בדיקות - קופסא לבנה.

בדיקות אינטגרציה - בדיקה שנועדה לוודא תקשורת תקינה בין יחידות תכנותיות שונות. הבדיקה מבוצע על סביבת בדיקות ייעודית, המבצע- תלוי בארגון, ישנם ארגונים שיש בהם מחלקות שמתמחות בכך. גם כאן, ניתן להשתמש ברמת בדיקות (תחליפים ליחידות שכרגע לא זמינות). הרכבת היחידות: top down bottom up לפי פונקציונליות. למדנו שיטות קופסא לשם כך.

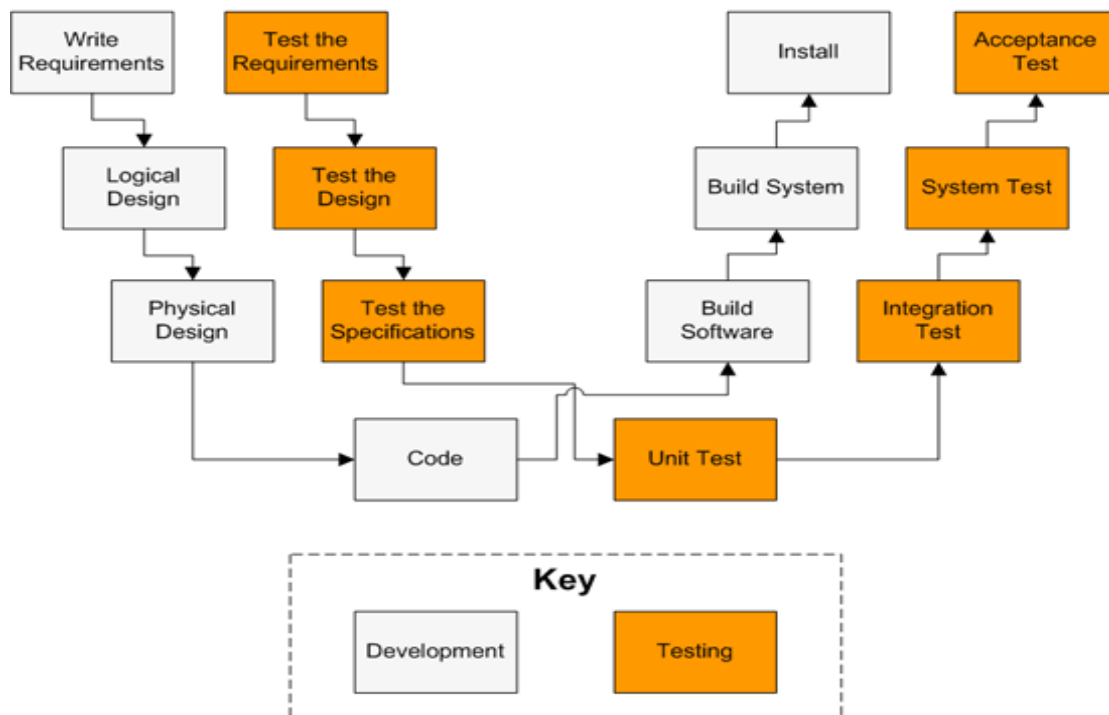
על מה עדיין לא דברנו (בהקשר בדיקות)???... התאמה לדרישות

בדיקות מערכת- בדיקת המערכת כמכלול אחד, פה נכנס אלמנט של התאמה לדרישות הלקוח. הבדיקות יבוצעו על ידי הבודקים!!! על סביבת בדיקות שתהיה תואמת ככל הניתן לסביבת הלקוח. כאן כבר לא נרצה שימוש בתחליפים- ואם אין ברירה, הרי שזהו סיכון שיש לנהל.

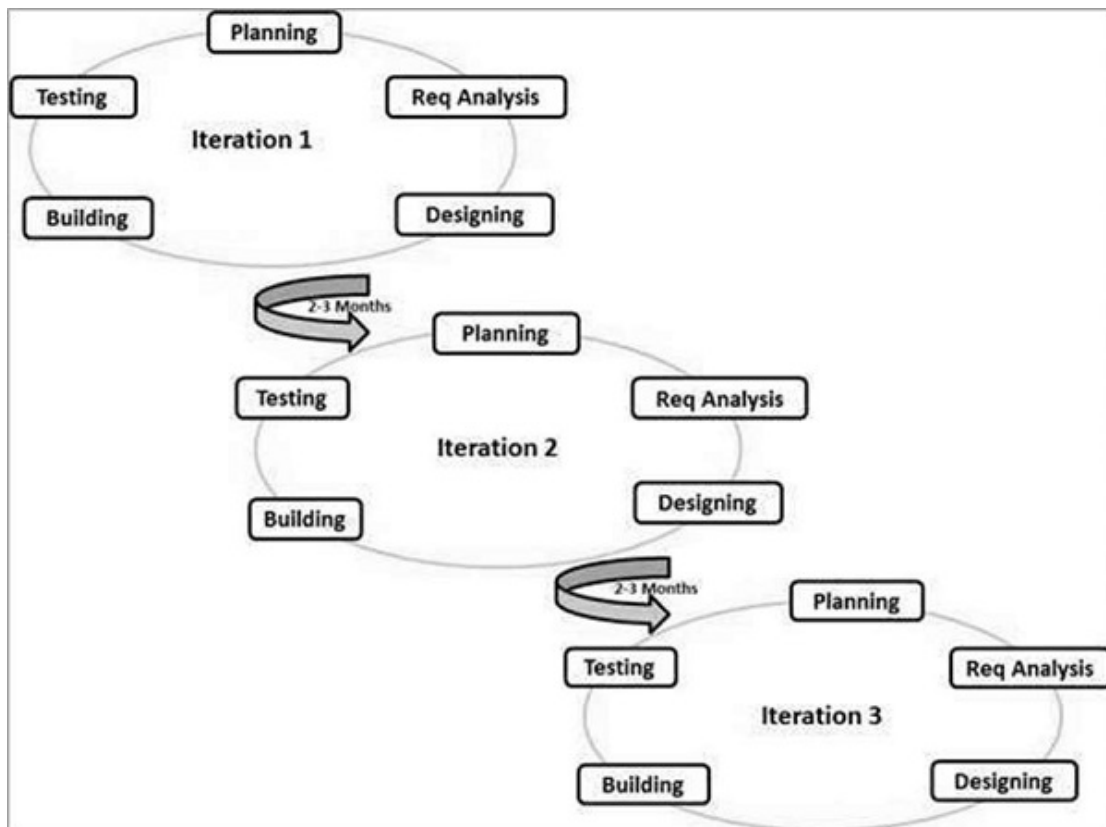
בדיקות קבלה acceptance test - בדיקת המערכת כמכלול אחד, התאמה לדרישות הלקוח כפי שהתקבלו- יבוצעו על ידי נציגי הלקוח!!! (ולא הבודקים)

- בדיקות אלפא- בדיקות שמתבצעות (עי הלקוח)- באתר הספק
- בדיקות ביטא- בדיקות שמתבצעות באתר הלקוח

הרחבה- מודל w



פיתוח אגיל



מנשר האגייל

אנו מאמינים שיש דרכים טובות יותר לפיתוח תוכנה. אנו נעדיף:

אנשים ויחסי גומלין	על פני	תהליכים וכלים
תוכנה עובדת		תיעוד מפורט
שיתוף עולה עם לקוחות		משא ומתן חוזי
תגובה לשינויים		מעקב אחרי תכנית

אנשים ויחסי גומלין על פני תהליכים וכלים

נעדיף אינטראקציה מיידית על פני בירוקרטיה- דוגמא: תהליך (מסורתי) של דיווח תקלה- נשתמש בכלי ייעודי לשם דיווח התקלה- הדיווח כולל תאור נכון, כולל דרוג של חומרת התקלה. הדיווח כולו עובר מסלול. באגייל לא נשקיע כל כך הרבה משאבים בדיווח מסודר- למשל, במקום זאת: עבודת זוגות- פיתוח ותיקון במקביל.

תוכנה עובדת על פני תיעוד מפורט

בכל מקרה- אנו נשחרר ללקוח תוכנה עובדת (!!!!!) הבדיקות מקבלות תעדוף על פני פיתוח, היות וניתן למסור בסבב זה או אחר פחות תוכן- אבל- כולו עובד.

נפנה את כל המשאבים בעיקר לכיוון תוכנה עובדת (פחות תיעוד ורישום)

תוכנה עובדת=עברה בדיקות(!!!!!!!!!!!!!)

נגדיר גישה -tdd test driven design:

נגדיר ונפתח בדיקות הנדרשות לרכיב שמפתחים- טרם הקוד עצמו. הסיבה: כי הפיתוח מותאם לבדיקות שהוגדרו

במודל מסורתי בשביל להגדיר בדיקות: שלבי ניתוח ועיצוב, לא קשורים לתהליכי הפיתוח, שבוצעו במקביל.

באגייל: מושג user story- משימה הכוללת את צרכי הפיתוח ואת הבדיקות שיש לעבור.

שיתוף פעולה עם לקוחות על פני משא ומתן חוזי

באגיל, שיח שקוף ורציף עם הלקוח במהלך כל הפיתוח=>באגיל יהיה בעל תפקיד שנקרא product owner- איש הקשר בן הלקוח לפיתוח

תגובה לשינויים על פני מעקב אחרי תוכנית

באגיל עלינו לממש יכולת לספק ללקוח גרסאות תוכנה בצורה שוטפת- מנגנון ci- סביבה פיתוחית אשר תקבל קוד בכל עת, תעביר אותו בדיקות- ואז תייצר גרסה עדכנית

השוואה קצרה בין גישות מסורתיות ואגיל

מודל מסורתי	מודל אגיל	
תכנון הבדיקות	שלב אחד של תכנון- אשר מתבצע בתחילת הפרוייקט	תכנון ברמת מאקרו- בתחילת הפרוייקט. בכל סבב פיתוח ובדיקות- יהיה גם תכנון ברמת מיקרו ספציפי לסבב הזה
בדיקות ידניות מול אוטומטיות	בדיקות אוטומטיות לאחר בדיקות ידניות- בוחרים מה ייכנס לאוטומציה	בדיקות אוטומטיות מתחילות מרמות הבדיקה המוקדמות ביותר(>=) בדיקות שמוטמעות בתוך הקוד) כאשר השאיפה היא ליותר בדיקות אוטומטיות ברמות בדיקה נמוכות – הקוד ופחות ב gui ועוד פחות בדיקות ידניות
עצמאות בודקים (עד כמה יש הפרדה בין תפקידי המפתח והבודק. עצמאות נמוכה= המפתח בודק לעצמו. עצמאות גבוהה= הבדיקות הינן גוף נפרד מהפיתוח, אפילו ארגון אחר)	שאיפה לעצמאות גבוהה.	יכולה להיות מעט ירידה- מעט חפיפה בין תפקידי בדיקות ופיתוח
עקיבות traceability (=מנגנון אשר מקשר בין דרישות של לקוח לבין מימושן בקוד ובבדיקות)	חשוב מאוד: • בקרת שינויים- כל שינוי נדרש, יהיה קל לזהותו בקוד ובבדיקות ולהגיע אליו בעת הצורך • "לדבר ללקוח בשפה שלו"	הדרישות מתפתחות ומתעדכנות כל הזמן- עקיבות לרוב נהיית קשה לניהול. לכן מנהלים זאת בצורה שונה (userstories)
רמות בדיקה	אבחנה ברורה בין רמות הבדיקה- לא מתחילים רמה מסוימת, טרם הסתיימה רמה קודמת	כיון שכל יחידת קוד יכולה להגיע לci בזמנים שונים- ישנו טשטוש בין התחלה וסוף של רמות בדיקה (כל קוד יכול להיות תאורטית ברמה אחרת)

דוגמא למימוש אגיל- שיטת סקראם scrum

Sprint- ריצה קצרה ומהירה. <=בסקראם הכוונה לפרק זמן קצר 14-4 יום בסופו משחררים ללקוח מוצר
עובד

בעלי תפקידים:

Po - product owner המקשר בין הלקוח לצוות הפיתוח

Scrum master- מדריך הצוות, מאמן הצוות, מקשר בתוך הצוות הדואג לעבודה על פי הסקראם

Team- צוות הפיתוח (והבדיקות)

את מי, לא ציינו פה...??? לא נמצא כאן תפקיד רשמי של ראש צוות

Product backlog- רשימת כל הפיתוחים שירצה הלקוח לגרסא הנוכחית של המוצר, מנוהל על ידי
product ownern

Sprint backlog- רשימת כל הפיתוחים לספרינט הנוכחי, מנוהל על ידי צוות הפיתוח.

מי מחליט מה יכנס ל sprint backlog?- הצוות. כי: חייבים לשחרר מוצר עובד מהצוות נדרשת יכולת
של אמדן מאמץ, מזמנים ותכנון תכולת sprint

Stand up meeting- פגשה יומית שמתבצעת בעמידה לשם תאום ותכנון היום. הרעיון לא לשבת- כדי לא
לבזבז זמן

שיטה לבצוע הערכות ואמדנים של בצוע:

Poker planning-

פוקר מלשון משחק הקלפים. הרעיון הוא- כל אחד יודע מה אצלו, בלי לדעת מה קורה אצל האחרים.

Poker planning- הכוונה שכל חבר צוות ראשית מבצע הערכת זמנים בעצמו, ורק אח"כ חושפים את
ההערכות של כולם, ומקבלים החלטה על אמדן הזמנים.

- מחייב כל חבר צוות לעשות בעצמו
- מונע הטיה של תשובות של חברי הצוות

בעת הצורך לתת הערכות על משימה מסוימת, כל חבר צוות מייצר הערכה רושם לעצמו בפרטיות. לאחר
מכן חושפים את ההערכות של כולם ומקבלים החלטה. הרעיון הינו לצמצם הטיות בהערכה הראשונית

ניהול פרויקט

פרויקט=פעילות, המורכבת ממספר משימות, שנועדה למען מטרה מסוימת

פרויקט מוגדר ע"י 3 מימדים:

- תכולה (מה עושים)
- לוח זמנים (מתי)
- משאבים (מי, ועל מה)

ניהול פרוייקט: מנהל פרויקט צריך לתת מענה לשאלה: מי עושה מה, מתי, (ואיך)

המטרה שלנו: בצוע משימות במינימום זמן (משאבים, כסף)

עבור ניהול מימד התכולה בפרוייקט:

מה עושים בפרוייקט? פעולה שנקראת **פירוק פונקציונלי** – נחלק את כל תכולת הפרוייקט ליחידות עבודה קטנות ורצוי שיהיו אטומיות ככל האפשר.

עבור ניהול מימד לוח הזמנים:

מתי- שיבוץ משימות. נחליט על סדר הפעולות, כאשר ננסה למקבל פעולות – לבצע למקביל- לשם חסכון זמן.

- תזמון: לכל יחידת עבודה נגדיר כמה זמן נדרש להשלמתה
- שיבוץ: נחליט על סדר בצוע המשימות- מה לפני מה, מה אחרי מה, מה במקביל ככל הניתן.

תרשים גאנט

דוגמא:

נתון לנו סט משימות:

זמן (ביחידות זמן)- כמה זמן יקח להשלים משימה זו	תיאור הפעילות	פעילות מקדימה	פעילות
1	הכן אתר	אין	A
6	הכן יסודות ושלד	A	B
3	רכוש עצים וגינון	A	C
2	גג	B	D
3	עבודות פנים	D	E
4	גינון	C	F
1	כניסה לבית החדש	C	G

מהטבלה הזאת, נוכל לבנות תרשים:

a->b->d->e

a->c->f

a->c->g

אורך הפרויקט כולו- נקבע לפי המסלול הארוך ביותר- נתיב קריטי

מטרתנו בניהול פרויקט- לזהות, ולצמצם את הנתיב הקריטי.

- להוסיף משאבים לאותן משימות -וכך לקצר המשך- למשל, אם נותנים משימה אחת ל-2 אנשים, משך הבצוע יקטן
- לבדוק, האם ניתן לפרק משימה מסוימת על הנתיב הקריטי ולמקבל חלק- למשל- אם משימה כוללת חלק טכני וחלק מינהלי, את החלק המינהלי אפשר לעשות בזמן אחר.
- שיפור טכנולוגי- הכנסת אוטומציה, עבור ב-offline- איפה שניתן

תזמון- כמה זמן ייקח לבצע משימה מסוימת? איך יודעים????

- נסיון עבר
- ייעוץ עם בעלי ניסיון אחרים
- שימוש במודל -ארגוני או סטנדרט בתעשיה
 - מודל 3 point estimation מדובר על "ממוצע אינטליגנטי" שמבוסס על מדידות של כ 30 משימות דומות שבוצעו בעבר:
 - A- תוגדר כמשך הזמן המינימלי של 30 משימות דומות שבוצעו בעבר
 - B- תוגדר כמשך הזמן המקסימלי של 30 משימות דומות שבוצעו בעבר
 - M- תוגדר כממוצע 30 משימות דומות שבוצעו בעבר
 - האמדן יהיה $e=(a+4*m+b)/6$

- מודל cocomo- מציג נוסחאות לאמדנים לפי פרמטרים שונים. הפרמטרים מחושבים לפי טבלאות מאותו מודל, מתחשבים בטכנולוגיה, במבנה הארגוני, בכמות שורות הקוד

תהליך הבדיקות הבסיסי:

נושאים שלמדנו בהקשר הזה	שלב בתהליך הבדיקות הבסיסי
תכנון בדיקות- יש לו קשר הדוק לצורת ניהול הפיתוח בארגון- למדנו שתי גישות- מפל (דברנו גם על רמות שונות של בדיקה (מודל וי) ואגיל. המטרה של השלב הזה- להגדיר תכנית עבודה (מי עושה מה, מתי), קביעת סוגי בדיקות נדרשים, פירוק פונקציונלי (פירוק אפיון ליחידות עבודה בדידות)- ובנית גאנט.	תכנון
על כל פירוק מגדירים תנאי בדיקה אם...כאשר..אז..	ניתוח
עיצוב בדיקות קופסא שחורה- למדנו שיטות שונות לעיצוב מקרי בדיקה (=מקרה בדיקה=צעד לבצוע מול תוצאה רצויה). שיטות עיצוב קופסא שחורה- שקילות, ערכי קצה, טבלת החלטה, מעבר בין מצבים. שיטות עיצוב קופסא לבנה- כסוי הצהרות, החלטות, תנאים מורכבים.	עיצוב
מנות בדיקה test sets (ארגון מספר מקרים ליחידה שתרוץ כאסופה אחת). בדיקת שפיות. כתיבת אוטומציה.	יישום
הרצת בדיקות. טיפול בתקלות.	בצוע
ההחלטה על סיום תהליך הבדיקות, סיכום, הפקת לקחים.	הערכה, דיווח, סיום

1. נתונים לכם שני מסמכים: מסמך אפיון ומסמך std מסמך Std הינו מסמך שמכיל את תנאי הבדיקה (אם כאשר אזי) ואת מקרי הבדיקה (צעד לבצוע ותוצאה רצויה)
 - a. כיצד נוכל להוכיח (במה צריך להשתמש) שאכן מסמך ה std מכסה את כל המופיע במסמך האפיון ? מנגנון עקיבות traceability- מקשר בין האפיון לבין מימושו בקוד ובבדיקות. זה חשוב כדי:
 - i. בעת עדכון באפיון- קל לזהות היכן בקוד ובבדיקות יש צורך בעדכון
 - ii. תקשורת עם בעלי העניין של האפיון ב"שפה שלהם"
 - b. מה ההבדל בין הכיסוי מהסעיף הקודם, לבין "כיסוי החלטות"- מתי משתמשים בכל אחד מהכיסויים הנ"ל כסוי החלטות- מושג בקופסא לבנה, כסוי מצואי התחלטות בקוד- סך מוצאי ההחלטות שנבדקו מתוך סך מוצאי ההחלטות בקוד. ההבדל בין כסוי החלטות לבין עקיבות- כל אחד מתיחס למשהו אחר...

2. נתונה מערכת המיועדת להרכבת סידורי פרחים:

- ניתן לבחור אחד או יותר מהצבעים הרצויים לסידור הפרחים: אדום, לבן, צהוב
 - ניתן לבחור האם רוצים את הסידור בכלי או כזר.
 - ניתן לבחור האם להוסיף ירק, או לא.
 - ניתן לרכוש כל צירוף אפשרי, אך לא ניתן לרכוש זר ללא פרחים כלל וללא ירק כלל.
- a. באיזו שיטת עיצוב בדיקות, כדאי להשתמש כדי לבדוק את המערכת?

טבלת החלטה- מדובר כאן על צרופי קלט שונים
b. כמה מקרי בדיקה אפשריים למערכת כזו? הסבירו את החישוב.

אדום, צהוב, לבן, כלי/זר, ירק/ללא ירק $2 \times 2 \times 2 \times 2 = 32$ מתוכם נוריד תנאי אחד- נשארו עם 31 צרופים

c. עקב מחסור זמני בפרחים מצבע אדום, כרגע הם אינם מוצעים ללקוחות. כמה מקרי בדיקה אפשריים במצב זה? חצי מהמקרים הקודמים -1=נשארו עם 15 צרופים

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
צהוב	כן	כן	כן	כן	כן	כן	כן	כן	לא	לא	לא	לא	לא	לא	לא	לא
לבן	כן	כן	כן	כן	לא	לא	לא	לא	כן	כן	כן	כן	לא	לא	לא	לא
כלי/זר	כלי	כלי	זר	זר	כלי	כלי	זר	זר	כלי	כלי	זר	זר	כלי	כלי	זר	זר
ירק/ללא	כן	לא	כן	לא	כן	לא	כן	לא	כן	לא	כן	לא	כן	לא	כן	לא
לא ישים לי האפיון																לא

צרוף 1 הינו- פרח לבן+פרח צהוב בתוך כלי, עם ירק

צרוף 9: פרח לבן+כלי+ירק

3. נתונה מערכת לגביית חשבונות מלקוחות. מזינים למערכת את הסך לתשלום (עד 10000 ש"ח) ואת סך התשלומים הרצוי (עד 24 תשלומים). בודק מצא שאם מנסים לשלם חשבון בסך 890 ש"ח ב-6 תשלומים, המערכת מחייבת את כל הסכום בתשלום אחד (לא מחלקת לתשלומים)
a. אילו בדיקות נוספות כדאי לבצע, כדי להגיע לתיאור מדויק של הבעיה?

דברנו על נושא דיווח תקלות- 4 אלמנטים בדיווח תקלה- בדוד, דיוק, מידע שלם, אובייקטיביות. במקרה הזה: בדוד- נבדוק בנפרד נושא גובה תשלום (חלוקת שקילות על הסכום), חלוקה לתשלומים. בשני האלמנטים נחפש שקילות וערכי קצה נוספים כדי לבדוד את הבעיה.

b. אילו שדות יש לציין בטופס דיווח התקלה, אילו ערכים תתנו באותם שדות?

דברנו על שדות כמו- severity priority (severity- דרוג חומרת הבעיה לפי החלטת הבודק. דברנו על שיטות שונות לקבוע זאת. Priority- דרוג עדיפות הטיפול בבעיה לפי החלטת הנהלת הפרויקט- שקול עסקי) מספר גרסא, פירוק, מעקף....

4. בהמשך לשאלה 3. מחלקת הבדיקות קבלה תיקון לתקלה, ומחלקת הפיתוח ציינה שכל מנגנון הגבייה של יותר מ-3 תשלומים, נכתב מחדש.
a. אילו בדיקות כדאי לבצע במקרה זה, ולמה?-

בעת קבלת תיקון תקלה- נבצע בדיקת אימות- לוודא שאכן התיקון פתר את הבעיה. כיון שיש כאן כתיבה מחודשת של קוד- צריך לבצע גם בדיקת רגרסיה (=לוודא שהקוד החדש, לא פגע ביכולות ישנות ובקוד ישן אחר)

b. לאחר קבלת התיקון הסתבר, שהמערכת מחלקת נכון ל-6 תשלומים אך במקרה שמזינים 21 תשלומים, המערכת גובה רק 10 תשלומים. מה הדבר הנכון לעשותו, לגבי דיווח התקלה בנושא?

נושא מחזור החיים של תקלה. במקרה והתיקון פתר את הבעיה, אך התגלתה בעיה חדשה- יש לסגור תקלה נוכחית ולפתוח חדשה. באם המקרה היה שונה- התיקון לא היה פותר את הבעיה, היינו מעבירים את התקלה למצב reopen. חשוב לדייק בכך, כי- הסטטיסטיקה של התקלות הינה כלי עזר לשלבי הערכת התקדמות הבדיקות

5. נתון האפיון הבא: " אם סכום הקניה הינו עד 200 ש"ח, נפתחת אופציה לתשלום אחד בלבד. אם סכום הקניה מעל 200 ופחות מ-1000, נפתחת אופציה לשני תשלומים. מעל 1000 ש"ח נפתחת אופציה ל-3 תשלומים"

a. יש לשרטט תרשים זרימה מתאים

b. כמה מקרי בדיקה נדרשים לכיסוי הצהרות, וכמה לכיסוי החלטות? נא לנמק.

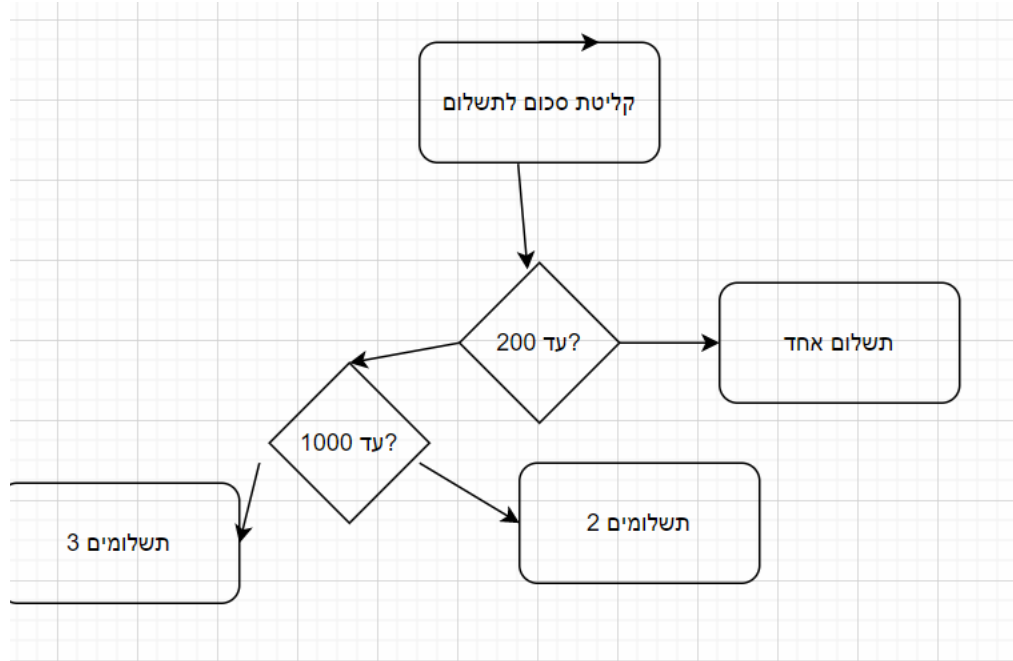
כסוי הצהרות- נייצר קלט או קלטים כך, שכמה שיותר הצהרות בקוד יופעלו, עם כמה שפחות איטרציות.

כסוי החלטות- נייצר קלט או קלטים כך, שכמה שיותר מוצאי החלטות בקוד יופעלו, עם כמה שפחות איטרציות.

הצהרות= פעולה לבצוע

החלטות= תנאי לוגי (מתקיים/לא מתקיים)

בשאלה זאת- סקורים ל-3 מקרים גם עבור כסוי הצהרות וגם עבור כסוי החלטות.



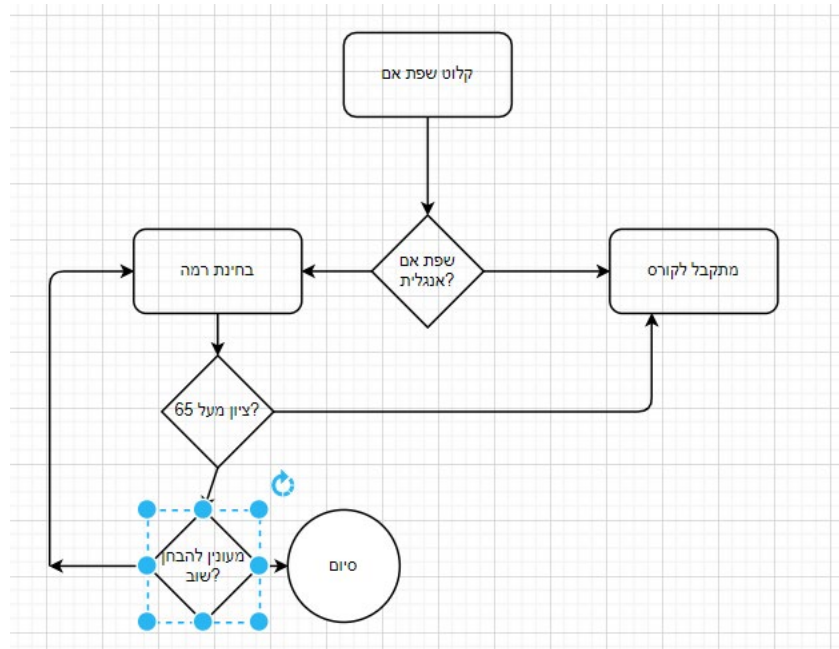
6. נתון האפיון הבא: " אם שפת האם של המועמד הינה אנגלית, מתקבל לקורס. אחרת יש לגשת לבחינת רמה. אם הציון בבחינת רמה גבוה מ-65, מתקבל לקורס. אם הציון בבחינת הרמה נמוך מ-65, ניתן להבחן שוב- אם הציון יהיה גבוה מ-65, מתקבל לקורס. המועמד יכול לוותר ולא להרשם כלל."

a. כמה מקרי בדיקה נדרשים לכיסוי הצהרות, וכמה לכיסוי החלטות? נא לנמק

עבור כסוי הצהרות מלא: מועמד שנבחן ועובר ומתקבל מיד לקורס

עבור כסוי מוצאי החלטות מלא: מועמד שנבחן ועובר ומתקבל מיד לקורס, מועמד שדובר שפת אם, מועמד שנכשל ונבחן שוב והתקבל, מועמד שנכשל וויתר על הקורס.

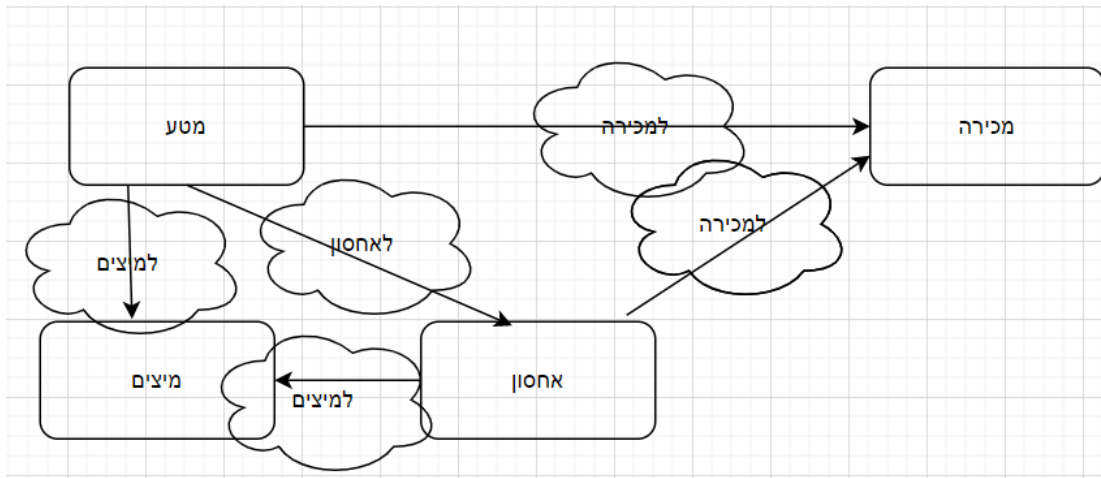
b. יש לציין את כל מקרי הבדיקה לאפיון זה



7. נתון האפיון הבא: " אנו בודקים את הפירות שהגיעו מהמטע. פירות בשלים עוברים מיד למכירה. פירות שאינם בשלים עדיין, עוברים לאחסון. פירות מתאימים ששהו שבועיים באחסון, עוברים למכירה. פירות שאינם מתאימים למכירה לפני או במהלך או אחרי האחסון, עוברים למפעל ייצור מיצים."

a. איזו שיטת עיצוב מקרי בדיקה מתאימה לאפיון זה (האפשרויות הינן: שקילות, ערכי קצה, טבלאות החלטה, מעבר בין מצבים)- לכאן מתאים מעבר בין מצבים

צריך- לשרטט מצבים, לזהות נתיבים חוקיים ונתיבים לא חוקיים



טבלת מעבר בין מצבים:

4 מצבים- מטע, מכירה, אחסון, מיצים

3 מעברים שונים (העברה למכירה, העברה לאחסון, העברה למיצים)

5 מעברים חוקיים+7 מעברים לא חוקיים

	העברה למכירה	העברה לאחסון	העברה למיצים
מטע	מכירה	אחסון	מיצים
מכירה	לא חוקי	לא חוקי	לא חוקי
אחסון	מכירה	לא חוקי	מיצים
מיצים	לא חוקי	לא חוקי	לא חוקי

b. יש לשרטט ולציין את כל מקרי הבדיקה המתאימים לאפיון זה- חוקיים ולא חוקיים.

1. צוות הבדיקות מצא שיש בעית המרה בין המטבעות ש"ח ורופי ודיווח תקלה בהתאם. התיקון שהתקבל כלל כתיבה מחדש של כל מודול ההמרה בין מטבעות. מה יקרה עכשיו?

a. יבוצעו בדיקות אימות ומערכת

b. יבוצעו בדיקות יחידה ורגרסיה

c. תבוצע בדיקת רגרסיה באם בדיקת אימות תעבור אין טעם להתחיל רגרסיה, אם בדיקת אימות לא עברה

d. תבוצע בדיקת אימות באם בדיקת הרגרסיה תעבור

2. צוות הבדיקות מצא שיש בעיה חמורה של המרה בין המטבעות ש"ח ורופי ודיווח תקלה בהתאם. הוחלט שכל מודול ההמרה בין מטבעות יחזור ל product backlog (לא יצא בספרינט הנוכחי) מה יקרה לדיווח התקלה במקרה זה?

מושגים מתוך מתודולוגית סקראם (מימוש של אגיל)

Product backlog- תכולת כל היכולות שהיו בתוכנה שיקבל הלקוח. רשימה שמתוחזקת ע"י לקוח ונציגו

Sprint backlog - תכולת כל היכולות שישוחררו ללקוח בספרינט הקרוב (ספרינט= פרק זמן קצוב שבסופו הלקוח יקבל חלק מהמוצר הסופי). רשימה מתוחזקת על ידי צוות הפיתוח!!

- a. יעבור ל deferred- כיון שהוחלט לא לטפל בקוד הזה כעת, התקלה מוקפאת
- b. יעבור ל reject
- c. החומרה תרד
- d. יעבור ל closed

3. צוות הבדיקות בצע את הבדיקה לפי התאור הבא:

"ניתן להפעיל את המכשיר ברציפות עד 100 דקות בעצמה בינונית או עד 75 דקות בעצמה גבוהה. הפעלתי את המכשיר בעצמה גבוהה ולאחר הדקה 74 עברתי לעצמה בינונית"

. איזו בדיקה בוצעה כאן?

- a. חלוקת שקילות וערכי קצה
- b. מעבר בין מצבים וערכי קצה
- c. ערכי קצה וטבלת החלטה
- d. טבלת החלטה, שקילות, ומעבר בין מצבים
- e.

4. בישיבת צוות הועלתה הצעה, להשתמש בכל בדיקות הרגרסיה שהורצו ועברו בגרסא הישנה- גם לגרסא הנוכחית **כלשון**

למדנו 7 עקרונות בדיקה. אחד העקרונות הינו עקרון "מדביר החרקים"- אם נבדוק כל הזמן באותה צורה, ללא גיוון- נחמיץ בעיות אחרות. אם נריץ הכל אותו דבר- ללא גיוון- עלולים להחמיץ בעיות אחרות

- a. ההצעה טובה לפי עקרון "צרות באות בצרורות"
- b. ההצעה לא טובה לפי עקרון "צרות באות בצרורות"
- c. ההצעה טובה לפי עקרון "פאראדוקס מדביר החרקים"
- d. ההצעה לא טובה לפי עקרון "פאראדוקס מדביר החרקים"

5. נתונות מספר פעולות. יש לציין ליד כל פעולה, לאיזה חלק מתהליך הבדיקות הבסיסי היא שייכת:

תכנון
ניתוח
עיצוב
יישום
בצוע
הערכה, דווח (של מצב התוכנה כולה, לא תקלה ספציפית), סיום

א. בחירת ערכי קצה- עיצוב

- ב. בדיקת פרוגרסיה- בצוע
- ג. הגדרת קריטריון יציאה- תכנון
- ד. ערבול נתונים (data scrambling)- יישום. בשביל להריץ בדיקות על סביבת בדיקות צריך- תוכנה, סביבה, וגם- נתונים! מנין נשיג נתונים? לייצר לבד, אפשר לקחת נתוני אמת ולשם שמירה על סודיות "לערבל" אותם. כך השתמשנו בנתנים אמיתיים, אך לא נכונים.
- ה. פירוק פונקציונלי - תכנון
- ו. כתיבת תנאי בדיקה – אם כאשר אזי- שלב ניתוח
- ז. הגדרת תוצאה בפועל- בצוע. תוצאה בפועל נצפית רק במהלך הרצת הבדיקות, להבדיל מתוצאה רצויה, אשר מוגדרת בעת עיצוב הבדיקות.
- ח. קביעת עדיפות –priority מדובר על אחד משדות דיווח תקלה- בצוע
- ט. בחירת רמת כסוי- רמת כסוי= כמה מקרי בדיקה הופעלו מתוך כמה מקרי בדיקה אפשריים. הבחירה לאיזו רמה צריך להגיע- הינה חלק משלב התכנון- זוהי החלטה אסטרטגית, שנקבעת בתחילת תהליך הבדיקות.
- י. מנות בדיקה (test set)- מנת בדיקה= אסופת מקרי בדיקה שנריץ יחד. מטופל בשלב היישום- אחרי שסיממנו לעצב את הבדיקות, ולפני שהתחלנו להריץ אותן.