רבעון רביעי 2015 | גיליון מס' 03

מגזין עולם הבדיקות

WWW.TESTINGWORLD.CO.IL

בחירת מזהה אובייקטים בסלניום

6 כובעי בדיקות

ייעול סט בדיקות

אוטומציית BDD

איך בודקים DWH

דבר העורכת | איריס קוטליצקי

שלום וברוכים הבאים לגיליון השלישי של "עולם הבדיקות" מגזין המיועד לבודקי התכנה בישראל

בכל נקודת ציון חדשה כגון שנה חדשה או יום הולדת אנו שואלים איך עברה השנה שחלפה, מה אנו רוצים שיהיה הלאה ואיפה אנו יכולים להשתפר בעתיד.
בתחילת **השנה שחלפה** הרגשנו שחסר לקהילת בודקי התוכנה בארץ מקום שיהיה בו קול ובמה ראויים לעולם הבדיקות, מקום לשיתוף ולמידה.
בשנה זו הוצאנו כבר שני גיליונות והנה עכשיו השלישי וזה הזמן להודות לקוראינו שנרשמו, עזרו להפיץ, נתנו לייק ושיתפו מהידע והניסיון שלהם.
לשנה הבאה אני מאחלת שיימשך השיתוף ונגיע לקוראים רבים בקהילת הבודקים בארץ כמגזין המאחד את הקהילה ומרחיב אופקים.

הגיליון הנוכחי עוסק בדרכים שונות להשגת תוצאות יעילות יותר בבדיקות,
דרך חשיבה שונה, מיקוד ושימוש בכלים המתאימים.
כיצד למצוא את הדרך המהירה (אך גם היעילה ביותר) דרך ניסוי וטעיה וחשיבה מחודשת יום יומית.

שנה טובה ופורייה ותודה על השנה שחלפה

מגזין עולם הבדיקות מתפרסם כל רבעון בגרסה אלקטרונית ומופץ חינם.
בכל רבעון תוכלו להתעדכן בטרנדים בעולם הבדיקות וכן במפגשים וכנסים המתקיימים בארץ ובעולם.
אתם מוזמנים להיכנס לאתר המגזין, אליו נעלה את הגיליונות החדשים - וכתבות בודדות בתקווה שיתפתח דיון פורה לגבי תוכנם.
www.testingworld.co.il
אם הנכם מעוניינים לקבל מייל ביציאת כל מגזין והדיונים המתפתחים בעקבותיו - הנכם מוזמנים לשלוח בקשה לכתובת register.testingworld@gmail.com
יש ביכולתכם להשפיע על התכנים שיופיעו במגזין.
לכל הערה, הארה, הצעה ורצון לכתוב אתם מוזמנים לפנות אליי במייל info.testingworld@gmail.com

בברכה,
איריס קוטליצקי

יש לך ראש יצירתי ואת\ה מעוניין\ת ליצור תוכן מעניין ולשתף את קהל הבודקים בישראל?

אנחנו מחפשים אותך!

לא חייבים להיות מומחי בדיקות בכדי להשתתף בפעילות הקהילה

לכל אחד ואחת יש מה לתרום.

בימים אלו צוות המגזין כבר שוקד על המהדורות הבאות אנו מחפשים אנשי בדיקות נוספים שירצו לתרום ולפרסם מאמר מקצועי במגזין "עולם הבדיקות".
גם אם אין לך ניסיון בכתיבה או אם את/ה מתקשה לבטא הרעיון שלך בכתב, אנחנו כאן לעזור!
פנו אלינו לגבי הצטרפות לצוות הכותבים.

כותבים המעוניינים לקחת חלק בכתיבה עבור הגיליונות הבאים מוזמנים ליצור קשר: info.testingworld@gmail.com

מו"ל
Israeli Testing Certification Board
ITCB®

ניהול המגזין
יאן ברון, *iMDsoft*

ניהול התוכן
קובי הלפרין, *Ceragon*

עורכת
אירית קוטליצקי, *Testing World*

עריכה
טל פאר, *Kongsberg Maritime*
איסי חזן, מוביל בדיקות ב-*Intel design center Jerusalem*
עמית ורטהיימר
משה מאמיה, *HP Software*

עיצוב גרפי
בית נלי מדיה
סטניסלב קולנקו
www.beitnelly.com

יצירת קשר
אימייל:
info.testingworld@gmail.com
הרשמה
register.testingworld@gmail.com
פקס:03-6176605
כתובת: ברוך הירש 14 בני ברק 51202

www.testingworld.co.il

www.itcb.org.il

תוכן העניינים

שיקולים והשוואת מזהה אובייקטים בסלניום | אייל כהן

אייל כהן

מהנדס תוכנה בתחום הבדיקות קרוב ל-10 שנים. בשנים האחרונות ממלא תפקידי ארכיטקטורה, פיתוח ויעוץ אוטומציה בכלל וסלניום בפרט ב-Java. כיום עובד בצוות DevOps בחברת Taboola המפתחת מנוע המלצות לצפייה בתכנים באינטרנט (נ.ב. אנחנו מחפשים אנשי בדיקות אוטומציה ופיתוח ניתן לפנות אלי). לכל שאלה ניתן לפנות אליי למייל eyal@seleniumexp.com

או בלינקדאין:

http://il.linkedin.com/pub/eyal-cohen/19/b40/406

במהלך שנות עבודתי כמפתח סלניום נתקלתי רבות בעובדים שמעלים שאלות סלניום כמו איזה מזהה (**locator**) הוא הכי מהיר? איזה דפדפן מוצא במהירות אלמנטים בדום? במה כדאי ואין בכלל להשתמש? התשובות שקבלתי היו שונות ולא חד משמעיות. חלק יגידו ש-**Id** הוא המלך, חלקם יטענו ש-**CSS** הכי נוח, חלקם יגידו ש-**XPath** הוא ממש, אבל ממש איטי. רבים יאתרו אובייקט ע"י הטקסט שבו, ומפתחים אחרים ימנעו מזה. במאמר אנסה לספק הוכחות מספריות ואתן מספר רעיונות לחשוב עליהם.

אתחיל בניסוי שערכתי בכדי לקבל מידע על מהירות המזהים השונים והדפדפנים הנפוצים. על מנת לבצע זאת, בניתי סביבה בדיקות קטנה בה אני פונה לדף **HTML** והמזהים השונים מחפשים את אותו האלמנט. אחרי פתיחה של דפדפן, מאתרים את האלמנט פעם אחת, מיד אחר כך שוב פעם ואז הדפדפן נסגר, נפתח ושוב מתבצעת הבדיקה. כך שוב ושוב וזאת על מנת להשיג ממוצע זמנים נכון ביותר.

נתון האלמנט הבא בתוך דף **HTML**:

```
<div id="auto-div">
  <button id="btn-id" class="btn-class" data-automation="btn-data" name="btn-name"> Click Me </button>
</div>
```

בטבלה למטה נראה את תוצאות הזמנים במילי-שניות (ms) בגרסת Selenium 2.46. ניתן להבחין שממוצע זיהוי אלמנט בפיירפוקס הוא המהיר ביותר (לאחר תיקון מ-2.45!). אחריו, בפי 2 זמן, עומד כרום והרחק מאחור אינטרנט אקספלורר.

כאשר נבחן את התוצאות, נראה שההבדל בין הפונקציות הוא יחסית קטן. מציאת אלמנט ע"י ה-**ID** שלו תהיה המהירה ביותר בכל הדפדפנים, לכן אם קיים Id לאלמנט נעדיף להשתמש בו. אך שימו לב, קיבלנו אותן תוצאות בפיירפוקס ובכרום כאשר השתמשנו ב-CssSelector וגם ב-Class.

Xpath לעומת **CssSelector** – אז כן, CssSelector באמת מהיר יותר מ- XPath לפחות בפיירפוקס וכרום. לכן נעדיף שימוש ב-CSS על פני XPath (בואו לא נשכח שרוב מפתחי ה-UI מכירים Css). אבל ל-XPath יותר אפשרויות במציאת האלמנט עם אפשרות "לטייל" ב-HTML למעלה ולמטה. לדעתי, מוביל האוטומציה בארגון חייב להכיר את שימושי ה-XPath בצורה טובה כי אליו יפנו לעזרה בזיהוי אובייקטים מסובכים באמת. (השוואה נוספת בין Xpath ו-Css כאן)

> **כאשר אתם ניגשים לזהות את האלמנט, מהירות מציאת האלמנט אינה גורם משמעותי בבחירת סוג הפונקציה. ישנם שיקולים אחרים ויותר חשובים בהם כדאי להתרכז.**

Locator	FF 38	Chrome 42	IE 11
By.Id: btn-id	10	23	68
By.ClassName: btn-class	10	23	70
By.CssSelector: [data-automation='btn-data']	10	23	102
By.Name: btn-name	13	25	77
By.XPath: //*[@data-automation='btn-data']	15	25	83
	12.33	24.17	81.33

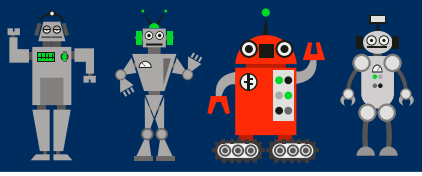
- בטבלה לא מופיעים מזהים אחרים הקיימים בסלניום כמו: **LinkText** אשר השימוש בו פחות נפוץ ו-**TagName** שבדרך כלל נשתמש בו כאשר נגיע לאובייקט מאלמנט עוגן אחר.

שימו לב שההבדל בין מציאת האובייקטים במזהים השונים באחוזים הוא יחסית גדול, אבל כגורם זמן הוא לא משמעותי במיוחד.

לדוגמא: מציאת אלף אלמנטים לפי Id ייקח כ-10 שניות, ומציאת האלמנטים בשימוש XPath ייקח 15 שניות. הבדל קטן של 5 שניות בלבד לאיתור אלף אלמנטים! רבים יגידו שזהו זמן זניח (וכמובן הכל יחסי).

שורה תחתונה: כאשר אתם ניגשים לזהות את האלמנט, מהירות מציאת האלמנט אינה גורם משמעותי בבחירת סוג הפונקציה. ישנם שיקולים אחרים ויותר חשובים בהם כדאי להתרכז.

שיקולים והשוואת מזהה אובייקטים בסלניום | אייל כהן



שיקולים אחרים בבחירת המזהה (Locator)

לרוב נעדיף להשתמש ב-ID או ב-Name כי הם לרוב ייחודיים בדף ובדרך כלל השם מרמז על האלמנט כך קל להבין מה הוא האובייקט לעיתים גם בלי לפתוח את ממשק המשתמש. זיהוי לפי Class גם כן עדיף אך לרוב לא יהיה ייחודי בדף וכנראה שנצטרך להגיע לאלמנט הספציפי מאלמנט עוגן אחר. CSS עדיף מ-XPath כי הוא יותר קריא וקרוב יותר לשפת מפתחי ה-UI. אך באמצעות XPath נוכל לבצע חיפושים מתוחכמים יותר.
אני לא ממליץ להוסיף class או id במיוחד עבור האוטומציה. לכן המזהה, שהוא הקלף המנצח בעיניי, (אם כי הוא לא קיים כחלק מברירת המחדל של סלניום) הוא השימוש ב-**Data Attribute**.

HTML5 הציג לנו את ה-Data Attribute, זהו למעשה כל attribute שנמצא באלמנט ומתחיל ב- 'data-'. הם לא 'נראים' למשתמש והמפתח יכול להשתמש בהם מאחורי הקלעים. את האובייקט המכיל data attribute ניתן לזהות ע"י שימוש ב-CSS או XPath.

אני ממליץ על שימוש בשם data attribute ייחודי (ולא לכתוב שם אחר באלמנטים שונים) שנועד רק עבור האוטומציה לדוגמא: data-automation-id. לשימוש מסוג זה יש מספר יתרונות:

1. איתור האלמנט למפתח האוטומציה נעשה קל וברור.
2. לא נתון לשינויים שנוגעים לעיצוב (לדוגמא: שינוי ב-CSS יגרור אולי שינוי שם ה-class).
3. לא מצריך מהמפתחים להוסיף Id או class במיוחד עבור האוטומציה כך קוד הדף יישאר נקי יותר.
4. בגלל שה-attribute ייחודי, קל למפתח ה-UI לזהות שהאלמנט בשימוש האוטומציה ושינוי שלו כנראה ישבור את הבדיקות.

> **המזהה, שהוא הקלף המנצח בעיניי, (אם כי הוא לא קיים כחלק מברירת המחדל של סלניום) הוא השימוש ב-Data Attribute.**

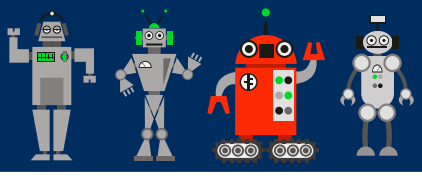
ByChained ולמה עדיף להימנע ממנו

רבים מאתנו מוצאים אובייקט ב-DOM בצורה היררכית ע"י השימוש ב-ByChained. זהו מזהה נפוץ כאשר אנו בונים את קוד הדפים שלנו בצורה הזו . לפונקציה זו ניתן להעביר סידרה של Bys כאשר הראשון הוא האב המכיל את השני שמכיל את השלישי וכן הלאה.
פונקציית ה-**ByChained** היא מאד איטית בגלל שהיא מבצעת מספר לא קטן של לולאות (ראו קוד) ובכל איטרציה אנו משתמשים ב-By בכדי למצוא את האלמנט הפנימי וראינו שכל פעולה כזו עולה לנו לפחות 10MS.
בטבלה הבאה נראה את תוצאות ההרצה בשימוש ב-ByChained בסדרה של 5 Bys. נראה גם כן את אותה היררכיה בשימוש של Css ו-XPath. אפשר להבחין שזמן מציאת האובייקט היחיד גודל פי כמה וכמה.

Locator	FF 38	Chrome 42	IE 11
By.Chained([By.ClassName[Contains]: block-inner, By.CssSelector: div.node-article, By.XPath: //div[contains(@class,'field-name-body')], By.Id: div-id,By.TagName: button])	87	164	3007
By.CssSelector: div.block-inner div.node-article div.field-name-body div#div-id button	10	23	99
By.XPath: //div[@class='block-inner']//div[contains(@class,'node-article')]//div[contains(@class,'field-name-body')]//div[@id='div-id']//button	15	25	83

שימו לב מהו זמן הריצה באינטרנט אקספלורר, יותר מ-3 שניות!! אך נתון מעניין עוד יותר הוא שמציאת אלמנט ע"י שימוש ב-Css או XPath מסובכים וארוכים לא פגעו בכלל במהירות מציאת האלמנט.
אני מציע לא להשתמש ב-**ByChained** ולקודד פונקציה דומה המקבלת סידרה של Bys ומחזירה By בודד ע"י מעבר עליהם ושרשור ל-CSS או XPath.
לדוגמא:
את הסידרה הבאה (By.Id("my-id"), By.ClassName("my-cls") ניתן בקלות לשרשר ל: By.CssSelector("#my-id .my-cls")

שיקולים והשוואת מזהה אובייקטים בסלניום | אייל כהן



מציאת אלמנט ע"י הטקסט

אחת הדרכים לזיהוי אלמנט בדף הוא ע"י הטקסט המופיע בתוך האלמנט. למרות שאני מעדיף לא למצוא אלמנט בצורה הזו, ותמיד משאיר את האפשרות הזו כאופציה האחרונה, לעיתים אין לנו ברירה ואנו חייבים למצוא את האלמנט כך.

טקסט מועד לשינויים לעיתים קרובות יחסית, כלומר כל שינוי תצטרכו לשנות גם את ה-Locator שלכם. בנוסף לכך, תחשבו שאתם צריכים לתמוך בשפה נוספת באתר שלכם וזאת לאחר שיש לכם כבר כמה עשרות אם לא מאות אלמנטים המזוהים בעזרת הטקסט. זה לא משאיר הרבה ברירות, או שתכתבו בדיקות רק לשפה החדשה או שתשכתבו את הקוד שלכם שיתמוך במספר שפות מה שעלול לקחת לא מעט זמן.

כדי להימנע מזה, כבר בתחילת הפרויקט תראו איך אתם מאפשרים את קבלת הטקסט לבדיקה מקובץ המכיל Key:Text. באפליקציות שתומכות במספר שפות, בדרך כלל, תרגום השפות נמצא בקבצים כאלו. ואם הטקסט כבר מקודד באלמנטים (לדוגמא אתם תומכים רק באנגלית), תכינו לכם קובץ כזה ותקראו ממנו את הטקסט לפי המפתח שלו.

<u>לדוגמא</u>:

findElement(By.XPath("//div[text()='" + getTranslation("my.key") + "']))

אז אם כן אתם משתמשים בטקסט לזיהוי אלמנט, הנה מספר דרכים שימושיות לעשות זאת בעזרת XPath:

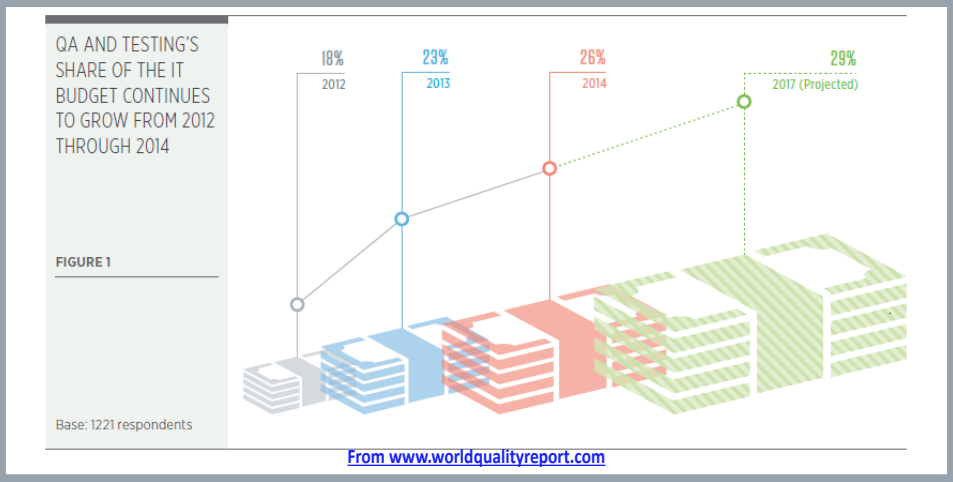
//div[text()='Eggs']	טקסט שווה Eggs
//div[contains(.,'Eggs')]	טקסט מכיל Eggs
//div[not(contains(.,'Eggs'))]	טקסט לא מכיל Eggs
//div[normalize-space(.)='Eggs']	טקסט שווה Eggs לאחר הסרת רווחים, שורה חדשה וכו', (כמו פונקציה trim)
//div[starts-with(.,'Egg')]	טקסט מתחיל עם Egg
//div[@class='my-cls' and descendant-or-self::*[text()='Eggs']]	ולסיום: מציאת אלמנט שה-class שבו שווה ל-my-cls וגם שהטקסט בו או באחד הצאצאים שלו שווה ל Eggs.

לסיכום... למדנו שזמן זיהוי האלמנט הוא קטן יחסית והפרש הזמנים בין הפונקציות השונות הוא נמוך ושלא כדאי לקחת את הזמן הזה כשיקול בבחירת הפונקציה. ראינו שפיירפוקס הוא הדפדפן המהיר ביותר בעוד אינטרנט אקספלורר איטי ממנו משמעותית. הבנו שאפשר וכדאי להשתמש ב-data attribute ייחודי עבור האוטומציה. הוכחנו ש-ByChained רוצח לנו את הזמן וכדאי להמיר אותו ל-By יחיד. הוספנו מספר נקודות למחשבה ואיך להשתמש בזיהוי האלמנטים ע"י הטקסט.

חשוב לצפות קדימה, לתכנן מראש ולהכיר שיקולים שונים לפני בניית סביבת בדיקות סלניום.

עד לפעם הבאה... בהצלחה.

הנתח של בדיקות QA1 מתוך תקציב ה-IT - ממשיך לצמוח ביציבות.



From www.worldqualityreport.com

סקרים

בחן את עצמך | טל פאר

בגיליון הקודם הסברתי על סיווג השאלות במבחני ISTQB לפי דרגות שנקראות K-level. במבחני רמת Foundation ניתקל בשאלות ברמות K1 ועד K4.
השאלה שנבחן היום תהיה מרמת K3, כלומר נצטרך לבחור את היישום הנכון של קונספט כלשהו או טכניקה כלשהי וליישם אותה בפתרון השאלה.

השאלה של היום:

את/ה בודק/ת מערכת מסחר אלקטרוני. המערכת מקבלת 4 סוגים שונים של כרטיסי אשראי; לכל כרטיס יש חוקים משלו לגבי חוקיות של מספרים.

לפניך חלק מטבלת ההחלטות של ניהול ההזמנות:

תנאי (condition)	מקרה 1	מקרה 2	מקרה 3
האם מספר הכרטיס חוקי?	לא	כן	כן
האם הקנייה מאושרת?	לא	לא	כן
פעולה (Action)			
הודעת דחייה	כן	כן	לא
קבל תשלום	לא	לא	כן

שילוב התנאים "מספר כרטיס חוקי" (לא) + "האם הקנייה אושרה" (כן) אינו ישים, כלומר לא ניתן לאשר קנייה אם מספר הכרטיס לא חוקי.

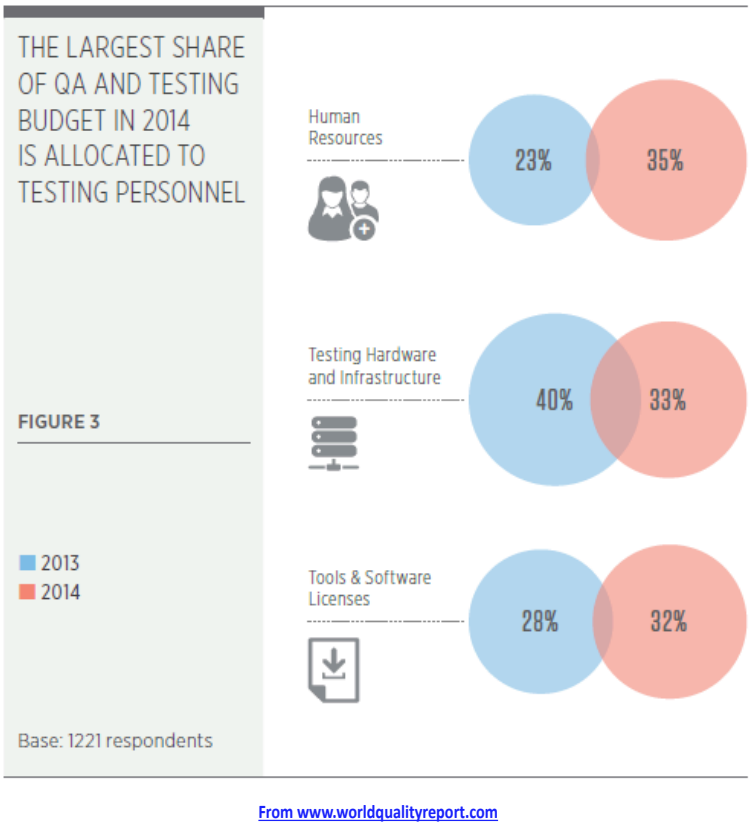
אתה רוצה לבצע בדיקה שמכסה את כל השילובים של מחלקות שקילות (equivalence partitioning) עבור סוגי כרטיסי האשראי השונים, בהתאם לחוקים המופיעים בטבלת ההחלטות למעלה.

כמה מקרי בדיקה (test cases) אתה צריך?

- א – 8
- ב – 16
- ג – 12
- ד – 4

*לצפייה בתשובה המפורטת – דפדפו לעמוד 26

החלק הארי של תקציב הבדיקות QA1 מוקצה לכוח אדם



From www.worldqualityreport.com

סקרים

מחפש צרות | מיכאל שטאל

מסורת

הנה סיפור ששמעתי מיעל (שם בדוי) – עדות ממקור ראשון. ביחידה בה היא שירתה היו בתי מלאכה בהם החזירו לכשירות ציוד שהתקלקל במהלך השימוש. בבתי המלאכה השתמשו בכלי עבודה שונים ובמברשות צבע. על פי הנהלים, בסוף יום העבודה, צריך להחזיר את הכלים למקום, לנקות היטב את מברשות הצבע ולסדר את בית המלאכה. מעשית, כשהגיע סוף היום, כולם היו עייפים או ממהרים או משהו אחר – אז השאירו את בית המלאכה בבלגן, את כלי העבודה על השולחנות ומתחתם, את המברשות עם הצבע עליהן, ויאללה – הביתה.

עד יום המחרת המברשות כמובן התייבשו לגמרי וכל מה שאפשר היה לעשות זה לזרוק אותן לפח.

כיוון שהייתה עבודה חדשה לעשות, לא היה זמן לסדר את בית המלאכה. כלי עבודה נעלמו וזמן יקר בוזבז בלמצוא אותם או ללכת לאפסנאות להביא כלי חדש.

יעל מעידה על עצמה שגם היא התנהגה כך – ולא ראתה בזה בעיה. "ככה עבדו אצלנו" היא אמרה.

כמה חודשים לפני סוף השירות של יעל נוצר מצב מעניין: כל החיילות הוותיקות בבית המלאכה השתחררו, ויעל נשארה היחידה מהצוות הקודם על מנת לקלוט וללמד את הדור החדש של חיילות שהגיעו לבית המלאכה.

יעל החליטה לעשות שינוי דרסטי. יום לפני הגעת הצוות החדש, היא ניקתה וסידרה את בית המלאכה כמו למסדר הרמטכ"ל (לפחות). כשהחיילות החדשות הגיעו, היא הסבירה להם את תהליך ההכשרה אותו יעברו, והדגישה חזור והדגש: "הדבר הכי חשוב אצלנו זה סדר וניקיון. אצלנו אין דבר כזה שהולכים הביתה בסוף היום ומשאירים את בית המלאכה מבולגן. אף אחד לא הולך לפני שכל הציוד חוזר למקום, המברשות נקיות והרצפה מבריקה".

בכל משימת הכשרה שהעבירה לחיילות, היה, בנוסף לרשימת הפעולות שיש לבצע, גם סעיף של "סדר וניקיון". אם מישהי בטעות פספסה את הקטע של הניקיון; אם כלי הושאר על שולחן העבודה – החיילת האחראית נאלצה לשמוע "סדרת חינוך" על חשיבות העל של הסדר והניקיון.

התוצאה? עם סוף תהליך ההכשרה – בערך ארבעה חודשים – גדל דור חדש בבית המלאכה. דור של חיילות ששמר בקנאות על סדר וניקיון ולא העלה בדעתו אפשרות לעבוד אחרת.

מסורת חדשה נוצרה. מסורת של עבודה לפי כללים והקפדה עליהם.

ולמה אני מספר לכם את זה?

> "נהלי עבודה הם ברובם עניין של תרבות ארגונית והרגלים"

כשאנו מנסים להכניס תהליכי איכות לקבוצת פיתוח או בדיקות, יש תמיד התנגדויות: התהליך מעכב את העבודה; התהליך חוסם יצירתיות; כאן זה לא מפעל ייצור; משמעת מוגזמת לא מתאימה לקבוצת פיתוח; וכו' וכו'.

הטענה שלי היא שזה ברובו סתם עניין של מסורת, תרבות ארגונית והרגלי עבודה.

נניח שנעזוב את העבודה שלנו ונעבור לחברה חדשה שבה יש הקפדה על תהליכים מסודרים וכל העובדים ללא יוצא מן הכלל מבצעים אותם במלואם. מיד נתיישר ונקיים את הכללים, וכיוון שכולם מסביב עושים אותם, לא נרגיש פריירים, לא נחשוב שאנחנו מבזבזים זמן או שחוסמים אותנו. נפנים שאלה כללי המשחק ונרגיש טבעי לבצע אותם.

אז האם חייבים לעבור מקום עבודה כדי שזה יקרה?

הכנסת שינויים בדרך העבודה – סט כללים והתנהגויות חדשים – היא בעצם מאמץ לייצר מסורת חדשה. מכיוון שאף אחד (אני מקווה) לא הולך לפטר את כולם ולגייס דור חדש של "חיילים", צריך להצליח לעשות זאת תוך כדי תנועה.

וזה אפשרי – אם יש תמיכת הנהלה ותמיכת כל הקבוצה. מצד ההנהלה – צריך להתחייב שהכללים לא יזרקו הצידה גם אם יש לחץ וקריזה. עם חברי הקבוצה צריך לקבוע מהו המינימום של תהליכים שכולנו מסכימים עליהם – ומתחייבים אחד לשני שלא לוותר על ביצועם בשום אופן, ויהי מה. להסכים שמותר לכל אחד להעיר לשני במקרה של עבירה על הכללים, וזכות לדרוש ביצוע של ההסכמות.

זה קשה. זה לעיתים יראה מיותר לגמרי (הסתדרנו לא רע גם קודם, לא?). אבל אחרי זמן מה, דברים שהקבוצה רואה בהם חשיבות, שבעבר נראו קשים ובלתי ניתנים להשגה, יתחילו לקרות. ואחרי עוד קצת זמן זה אפילו לא ירגיש מאמץ מיוחד לבצע את זה. יותר מזה: לא יידרש מאמץ לשכנע עובדים חדשים שכללים אלה חשובים או מוצדקים, כי זה יהיה חלק ממה שאנחנו קוראים לו "עבודה". כי פשוט: ככה עובדים אצלנו.

ובאותו עניין: שווה להשקיע שעה ולראות את Mars Code – הרצאה מאלפת של ג'רד הולצמן על אימוץ כללי איכות חדשים על ידי קבוצת מתכנתים של NASA. הדרך שלהם ניתנת בהחלט לחיקוי.

האם זה רק Hype או באמת כלי חשוב לבודקים?

לאורך שלוש השנים האחרונות אנו נתקלים שוב ושוב בדיוני פורום, מאמרים במגזינים, פוסטים בבלוגים, ציוצים, מצגות בכנסים ועוד העוסקים בנושא

MindMaps (להל"ן MM), נראה כי "**אם אין לך על מה לכתוב – כתוב על MM**" – אז הנה – גם אני כותב :-)

בכל פעם מחדש אני מנסה להבין למה נושא זה הנו כה "סקסי" לעסוק בו, והאם כלים אלו באמת מצדיקים את כל ה"רעש" בנושא? האם אכן כדאי להשתמש בהם במקום בעצים של כלי ניהול הבדיקות/ALM?

יתרונן של ה- MindMaps טמון במהירות כתיבת הנתונים (אשר אמורה לא להפריע לתהליכי החשיבה), ובהצגה הגרפית המקלה על מיון הנושאים וזכירתם.

כלים אלו משמשים לצרכים רבים מחוץ לעולם הבדיקות, וקיימים ככלים חינמיים, כשירותי וב וככלים מסחריים, ובבדיקות הם משמשים בשלבי תכנון ראשוניים, וגם בהצגת נושאים בקצרה למטרות סקירה (Review).

בזמן שמרבית ה-MM נכתבים ומוצגים בצורה מעגלית סביב השורש (נושא ראשי) כענפים בסדר אקראי ותחתם תתי ענפים/עלים הקשורים אליהם, הרי שיש עוד צורות הצגה רבות (ביניהן עץ אנכי)

> **יתרונן של ה-MindMaps טמון במהירות כתיבת הנתונים (אשר אמורה לא להפריע לתהליכי החשיבה), ובהצגה הגרפית.**

שיטת עבודה זו לתכנון בדיקות בעזרת MM מוצגת כאג'ילית יותר – מכיוון שלרוב לא נוטים לפירוט רב, אלא מסתפקים ברעיונות בדיקה מרכזיים בלי לתאר את דרך מימוש הבדיקה. (עוד סיבה בגללה הכלים הללו תופסים תאוצה במקביל לניסיונות מעבר לפיתוח באג'יל)

לאורך השנים תמיד העדפתי לנהל מידע לצרכים שונים במבני-נתונים – בשל יתרונותיהם בשיתופיות, יכולות עיבוד והצגה בצורות רבות, והקלות בהעברת המידע למדיומים אחרים או מכלי לכלי.

אם אני משווה בין כלי ALM בהם אנו מנהלים בדיקות המציגים לרוב המידע בצורת עץ לכלי ה-MM אני רואה היתרונות והחסרונות הבאים:

1. חלק מכלי ה-ALM דורשים "יותר קליקים" להוספת עלה לעץ.
(לכאורה אין מגבלה טכנית ולכן ניתן לשיפור ובין השאר לעיתים אלו מגבלות שמגדירים מנהלי הכלי בחברה המפריזים בכמות שדות החובה)
2. לעצים המשמשים ב-ALM עדיין ישנם חסרונות בפן הגרפי ביחס ל-MM – ביכולות הוספת צבעים לעלים בודדים ולאיגוד קבוצות, ובחלקן אף אייקונים. (גם כאן לכאורה אין כל סיבה שתכונות אלו אינן נתמכות בעצים – חסרה פשוט היוזמה של אחד היצרנים שתמשוך את האחרים)
3. חסרונן העיקרי של MM הנו במגבלות "Scalability" – מעל גודל מסוים כבר קשה להתמצא ולנהל, והתמונה הגרפית בגדלים אלו כנראה מעל ליכולת התפישה שלנו.
4. כמו כן ל-MM יש פחות יכולות שיתוף, לרוב כלים אלו מיועדים לניהולו של אדם אחד, דבר שמקשה עבודה של צוות למטרה משותפת.
5. חסרון נוסף הוא הקושי היחסי בהעברת הבדיקות ב-MM מייצוג של תכנית לסימון כיסוי ותוצאות.

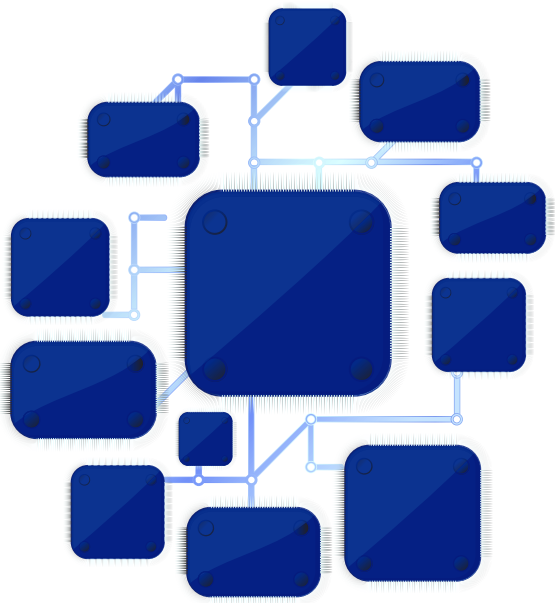
כלומר אנו רואים כפי שצוין לעיל כי כלים אלו משלימים אחד את השני, לעצים במבנה נתונים יש יתרונות בשיתופיות, יכולות עיבוד והצגה בצורות רבות, וקלות בהעברת המידע למדיומים אחרים – ואלו נקודות החולשה העיקריות של MM.

ומאידך במידה ולעצים היו יכולות צבע ואייקונים הרי שהיה ניתן בקלות יחסית לעבור בין תצוגות שונות – תכנון, הצגה ע"פ חשיבות, כיסוי וכד' מתוך מידע נוסף הקיים ב-DB לאותם עלים.

דוגמאות לכמה כלי MindMaps חינמיים נפוצים:

- XMind
- freemind
- www.mindmup.com

> **דווקא שילובם כחלק בלתי נפרד מכלי ניהול הבדיקות/ALM עשוי להדגיש את יתרונותיהם מחד ומאידך להתגבר על חסרונותיהן**



Ministry of Testing מבית מועדון STC הקימו את מאגר ה-MM הראשון לבדיקות (למיטב ידיעתי) – תמצאו שם דוגמאות רבות ומידע רב נוסף, ולאחרונה החברים ב- TestInsane הקימו גם הם אתר המאגד MM בנושאי בדיקות שונים בשיתוף רעיונות מקהילת הבודקים העולמית.

לסיכום – נראה כי המשיכה של רבים ל-MM מוכיחה כי אי אפשר להגדיר כלים אלו כמיותרים, אך לדעתי דווקא שילובם כחלק בלתי נפרד מכלי ניהול הבדיקות/ALM עשוי להדגיש את יתרונותיהם מחד ומאידך להתגבר על חסרונותיהן – אם רק יכולנו להחליף שיטת תצוגה בקלות בין עץ ל-MM בזמן תכנון הבדיקות, ולבחור קבוצת מקרי בדיקה להצגה כ-MM במהלך סקירה, הרי שהיינו יכולים ליהנות משני העולמות.
(ותאורטית נראה כי אין כאן בעיה טכנית ממשית למימוש – דוגמא ראשונית ניתן למצוא בכלי החדש של QAsymphony המאפשר להציג תוצאות ריצה בעזרת MM).

6 כובעים | גיל בלום

גיל בלום

ממייסדי Testuff, מפתחת כלי ניהול הבדיקות Testuff Test Management, הכולל חידושים ופיתוחים ייחודיים בעולם הבדיקות. החברה פועלת מזה למעלה מ 8 שנים (www.testuff.com), ולה לקוחות מרחבי העולם כולו, ומתעשיות שונות ומגוונות.

למה לבודקי תוכנה כובעים רבים

בשנת 1985, הציג אדוארד צ'רלס פרנסיס דה בונו פובליוס (כן - זה שמו האמיתי) מודל חשיבה חדש, בספרו הידוע Six Thinking Hats (בהוצאת Back Bay Books, הוצאה שניה 1999).
על פי מודל חדש זה, ניתן למזג את המרכיבים הטובים ביותר של סגנונות קוגניטיביים שונים, כדי להשיג תוצאות אופטימליות בניהול פרויקט.

ששת ה"כובעים" מקודדי-הצבע כוללים:

- "**כחול ניהולי**" (Managerial Blue). מהי המטרה? מהו הנושא? למה אנחנו כאן?
- "**לבן מידע**" (Informational White). מה הן העובדות? מה הם המשאבים שלנו?
- "**אדום רגשי**" (Emotional Red). מה תחושת הבטן שלי אומרת על הפרויקט הזה?
- "**שחור אבחנתי**" (Discerning Black). איפה הם אזורי הסכנה? מה יכול להשתבש?
- "**צהוב אופטימי**" (Optimistic Yellow). איפה ההזדמנויות? מה הם היתרונות?
- "**ירוק יצירתי**" (Creative Green). האם אנחנו יכולים לעשות את זה טוב יותר? האם אנחנו יכולים לשלב רעיונות חדשים?

> **נחקור כיצד "חבישת הכובעים" האלה יכולה להביא בודקים ליכולת גבוהה יותר, לביצועים טובים יותר וליצירת מוצרים טובים יותר**

על פי המחקר של דה בונו, צוותים שמשלבים את ששת סגנונות החשיבה האלה:

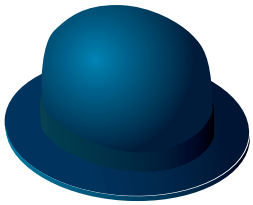
- טועים פחות במהלך ביצוע הפרויקט
- מזהים את ההזדמנויות הגדולות ביותר
- מגיעים לכיסוי רחב יותר של הפרויקט על כל היבטיו

למרות שבמקור נכתב הספר, ופותחה התיאוריה כולה, כעיקרון ניהולי למנהיגים עסקיים, יש לה יישומים רבים בתחומים אחרים - כולל בעולם ניהול בדיקות תוכנה.

בסדרה של שלושה מאמרים בגיליונות הקרובים נצא לחקור כיצד "חבישת הכובעים" האלה יכולה להביא בודקים ליכולת גבוהה יותר, לביצועים טובים יותר וליצירת מוצרים טובים יותר. בכל מאמר נסקור שני כובעים וננסה להגדיר מה עושה כל אחד מהם לתהליך הבדיקות שלנו, מתי נכון לחבוש אותו בתהליך, וכיצד ניתן ליישם את התוצאות לשיפור תהליך פיתוח התוכנה.

> **כשאתה חובש כובע כחול, אתה לא מודאג מצעדים ספציפיים או מטקטיקות. אתה מתמקד באסטרטגיה הכוללת.**

נתחיל עם הכובע הכחול.



המנצח על התזמורת

הגדרת בדיקות תוכנה "כחולות"

חבישת "הכובע הכחול" מאפשרת לבודקים לשאול את השאלות החשובות ביותר. כלומר, לדוגמה:

- מהו הנושא?
- למה אנחנו בודקים היבט זה?
- מהי המטרה שלנו?

הכובע הכחול נמצא ברמה העליונה של התהליך. הוא הנווט שמשרטט את תרשימי הניווט. המנצח אשר מוביל את התזמורת. השופט שקובע את החוקים.

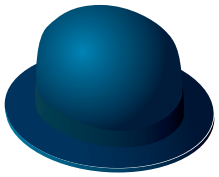
ההוגה שחושב על החשיבה.
בעולם בדיקות תוכנה, הכחול הוא מנהל הבדיקות הרשמי (או הלא רשמי) ש:

- מגדיר את היעדים ומגדיר את המתודולוגיה.
- מקצה את הבודקים וקובע את לוח הזמנים.
- חובש הכובע הכחול מחליט מתי הבדיקות מתחילות ומתי הן מסתיימות - ובאילו תנאים בדיקות אלה מתבצעות (ולמה).

כלומר, הכובע הכחול סב כולו סביב סיכומים ומסקנות - ההחלטות הגדולות והשאלות החשובות ביותר.

הכחול הוא ה"מדוע" של כל הפרויקט.
יישומי הכובע הכחול בבדיקות תוכנה עבור רוב הכובעים בסדרה זו, נציין מתודולוגיות בדיקה ספציפיות, שנראות רלבנטיות. כחול הוא אחד החריגים לכלל זה.
וכאן המקום להזכיר גם את ההבדל בין "חשיבת כובע כחול" לבין STP, Software Test plan. "הכובע הכחול" משמש לאסטרטגיה כוללת, כפי שצוין לעיל. STP מתייחס לעומת זאת לאסטרטגיה ספציפית (או יותר מדויק לטקטיקה) ולתכנון הבדיקות עצמן. ב STP לא מעלים תהיות לגבי המוצר עצמו (למה מפתחים ככה, למי זה מיועד וכו'). ה"כובע הכחול" מאפשר התרחקות וראיה מזווית שונה של בחינת הפרויקט כולו והעלאת נושאים כלליים. אתן דוגמא: אם בדרישות יופיע "תמיכה בשפה הסינית" אז ב STP יוגדר סעיף שלם לתכנון בדיקת המוצר בשפה הזו (בודקים מתאימים, מתרגמים, וכו'). אבל תחת "הכובע הכחול" יתאפשר לשאול את השאלה "למה החלטנו לתמוך בסינית?" ולנסות להגדיר מדדים לבחינת השאלה הזו.

6 כובעים | גיל בלום



כשאתה חובש כובע זה, אתה לא מודאג מצעדים ספציפיים או מטקטיקות. אתה מתמקד באסטרטגיה הכוללת:

- מה אנחנו למעשה בונים/מפתחים?
- מה משתמש הקצה מצפה?
- מהו התפקיד שלנו במימוש הציפייה הזו?

מענה לשאלות אלה יעזור לכם להחליט לגבי המתודולוגיה הנכונה או הכלים הטובים ביותר בשלב מאוחר יותר - בזמן חבישת כובעים אחרים.

למה חשיבה קוגניטיבית "כחולה" חשובה

כל הכובעים במסגרת שבנה דה בונו חשובים וחיוניים. לטענתו צריך שישה מצבי חשיבה כדי לכסות את כל האפשרויות. אבל הכובע הכחול הוא היסוד. צריך לחבוש כובע זה בכל שלבי התכנון הראשוניים של כל פרויקט.

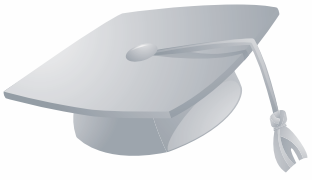
חשיבה מסוג זה מבטיחה כי הקצאת משאבים עתידיים נעשית לפרויקט עם הגיון, ותכנון נכון:

- אם שלב חשיבה זה מתבצע נכון, כל השלבים הבאים יישרו קו באופן חלק יותר. יהיו מכשולים וכישלונות לאורך הדרך. אלה הם בלתי נמנעים. אבל התכנון בעזרת הכובע הכחול מציע מפת דרכים, שתאפשר לך להתגבר על מכשולים אלה. עם יעד ברור, בחירת ההרים לטיפוס והנהרות לחציה קלה יותר. חשיבה "כחולה" מסירה הרבה ניחושים מהמשוואה.
- אם שלב חשיבה זה נכשל או לא מבוצע נכון, הפרויקט שלך יצטרך לאינספור הרעיונות המבריקים האחרים שלא הצליחו למצוא שוק מוכן. כאשר יהיו מחסומים, לא יהיו את המשאבים כדי לעקוף אותם. זמן יקר יבוזבז על טיפוס של הרים שלא צריך לטפס עליהם.

אל תתחיל פרויקט, אלא אם כן מישהו בצוות חובש את "הכובע הכחול". באופן אידיאלי, כובע זה צריך להישאר בהישג יד בכל מחזורי הפיתוח ובדיקות. "הכובע הכחול" מאפשר בחינה מחדש של האסטרטגיה לפני כל ביצוע סבב וכך לבצע התאמות על פי הנדרש.

ועכשיו לשלב הבא במסגרת החשיבה הקוגניטיבית של דה בונו – "הכובע הלבן".

חבישת "כובע לבן": כלי ניהול בדיקות התוכנה החשוב ביותר



בחלקו הראשון של מאמר זה, בדקנו גישה חדשנית לניהול פרויקטים – גישה בה צוותי פיתוח ובדיקות יכולים לשפר את סיכויי הצלחה שלהם על ידי אימוץ ששה סגנונות חשיבה משלימים. "חשיבה לרוחב" לאורך כל תהליך הפיתוח.

המושג הוצג כאמור לראשונה על ידי אדוארד דה בונו. ב-30 השנים שחלפו מאז, השימוש בגישה ייחודית זו חרג מניהול עסקי בלבד והשתלב כאמור היטב בפעילויות נוספות, לרבות בבדיקות תוכנה ופיתוח.

> **כאשר אחד מחברי הצוות שם את הכובע הלבן, הוא הופך למי שבוחן את העובדות בצורה אובייקטיבית ככל האפשר.**

כפי שראינו, הכובע הכחול מייצג את החשיבה ברמה העליונה, והוא נדרש לפני שהפרויקט אפילו מתחיל. שאלות אופייניות לשלב זה יכולות להיות: מהי המטרה? למה אנחנו כאן? מה אנו בודקים? למה אנחנו בודקים את זה?

הגדרת בדיקות תוכנה "לבנות"

כאשר אחד מחברי הצוות שם את הכובע הלבן, הוא הופך למי שבוחן את העובדות בצורה אובייקטיבית ככל האפשר. הוא ה census-taker. באופן ספציפי יותר, חשיבה מסוג זה דורשת להעריך:

- מה אנחנו יודעים כבר?
- מה אנחנו צריכים לברר?
- איך אנו יכולים לקבל את הנתונים החסרים האלה?

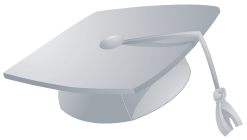
שים לב שיש אפס התייחסויות רגשיות בתהליך זה. צריך לאמץ גישה 'קלינית' ולקבוע מהן פיסות המידע המאומתות, או שניתן לאשש אותן – ומה מבוסס על הנחות. אם וכאשר נמצאים פערים, חשיבה מסוג "הכובע הלבן" מאפשרת לך לזהות דרכים לאחזור המידע שאתה עדיין צריך.

הכוונה היא להסיר הנחות ותאוריות רבות ככל האפשר מהשולחן. שאר סוגי החשיבה, מיוצגים על ידי כובעים אחרים במסגרת של דה בונו, מציעים שפע של הזדמנויות לחקור ולהיות יצירתי. אבל הכובע הלבן נועד לוודא שהמסע מתחיל ממקום של ודאות. מקום של ידע.

התחלת הבדיקות נעשית עם הגדרת מדדים ברורים לביצועם, ומדדים ברורים לניהול התהליך. הכובע הלבן מאפשר הגדרה מדוייקת של תנאים להתחלה וסיום של בדיקות, וכך להוציא מהמשוואה תחושות העלולות להטעות ולגרום לבזבוז זמן.

לדוגמא: תנאי עצירת סבב בדיקות, הינו קריטי למניעת בזבוז משאבי בדיקות. אם לדוגמא מוגדר, שסבב הבדיקות מתחיל רק לאחר שהאוטומציה עברה בהצלחה, ומסתיים אם הסתיימו הבדיקות או שהתגלתה תקלה מרמת חומרה קריטית עשוי להימנע בזבוז משאבים של בודקים וזמן בדיקות – במידה והמערכת לא מוכנה לבדיקה למעשה או כאשר התגלתה תקלה קריטית והבודקים ממשיכים להשקיע זמן מיותר (שהרי ממילא הבדיקות יצטרכו להיעשות שוב לאחר תיקון התקלה הקריטית).

> **כשתסתיים החשיבה עם "הכובע הלבן", אמור להיות תוצר של דו"ח פשוט של מספר עמודים, שמתאר את המצב הנוכחי של הפיתוח.**
> **"זה המקום שבו אנו נמצאים."**



יישומי כובע לבנים בבדיקות תוכנה

בדיקות תוכנה מעוררות לעתים קרובות הרבה רגשות – בעיקר כשמועדי גירסה מתקרבים, כשאנשים עם אופי שונה מתנגשים, או כשתקציבים מקוצצים.

חשיבת "כובע לבן" איננה הזמן ולא המקום להסחת דעת כזו. הדאגה העיקרית בשלב זה היא לקבוע עובדות ניטרליות כמו:

- מהו התקציב (ולא מה הוא צריך להיות)?
- מה המוצר עושה כיום (ולא איך הוא יכול לבצע זאת טוב יותר)?
- איפה המוצר אינו עונה על הנדרש (ולא למה זה נכשל)?

כשתסתיים החשיבה עם "הכובע הלבן", אמור להיות תוצר של דו"ח פשוט של מספר עמודים, שמתאר את המצב הנוכחי של הפיתוח.

עצור וחשוב! האם שיטות אלו הנן היעילות ביותר לצרכינו?

"זה המקום שבו אנו נמצאים."

כדי להגיע לשלב הזה עדיין דרושות כמה בדיקות:

- כדאי לשים דגש על בדיקות רגרסיה (Regression testing) כדי לזהות מה עובד ומה לא עובד. בדיקות מסוג Exploratory יש לשמור למועד מאוחר יותר, כאשר מגיע הזמן הנכון לסוג בדיקות זה.
- יש להשתמש בבדיקות אוטומציה כדי לעבור סבבי בדיקות. אוטומציה לא נתונה למניפולציות, וניתן לחזור עליה בדיוק וללא שינויים.
- תסריטים כתובים ולוחות זמנים שנקבעו מראש, יסייעו לוודא את נכונות המידע הידוע.
- יש לעשות שימוש בבדיקות ידניות כדי להריץ מחדש בדיקות שנכשלו (זו עובדה שהם נכשלו).
- אימות (Validation) של תקלות שטופלו ותוקנו, יוכיח שהם אכן תוקנו ויהפוך אותם לעובדה ידועה (או שלא).
- עבודה עם טבלת ניהול וכיסוי דרישות, עד להסרת כל התקלות שדווחו, תעזור אף היא לגיבוש עובדות חדשות ולווידוא עובדות קיימות.

רוב בודקי התוכנה ירגישו "בבית" עם הכובע הלבן. המקצוע שלנו מונע על ידי נתונים. ובין אם במודע או לא, רובנו אוספים את מה שאנחנו יודעים - לפני שממשיכים הלאה לביצוע העבודה.

עם זאת, חשוב להימנע מהפיתוי להתעסקות ארוכה מדי בשלב הזה. רעיונות ותובנות חדשות יתחילו להתגבש ("אם A זה נכון, אז B יכול להיות הגורם") ממילא.

יהיה מספיק זמן כדי לבחון הנחות אלה בשלבים מאוחרים יותר. לעת עתה, העובדות – ורק העובדות.

למה חשיבה קוגניטיבית "לבנה" חשובה?

הדרך הטובה ביותר להבין מדוע שלב חשיבה זה הינו חשוב הוא לבחון מקרים בהם שלב זה חסר.

הודות לחשיבה בשלב "הכובע הכחול", הצוות מבין את המטרות הבסיסיות של הפרויקט. כולם חולקים מפת דרכים משותפת. אבל בגלל שלא נעשתה "חשיבה לבנה", שלב איסוף העובדות אזי:

- מחצית צוות חושב שהתקציב הוא A - האחרים שזה יהיה B.
- מחצית צוות חושב שרכיב X עובד - החצי השני מבין שזה עדיין מצריך תיקון ובדיקות.
- מחצית מהצוות פועל לשיפור תכונה Y - החצי השני מבין שתכונה זו אינה חלק מגרסת הבטא.

כאשר יש לך הנחות מתחרות, או שונות, באותה הקבוצה, עיכובים ותסכולים הם בלתי נמנעים. ה"כובע הלבן" מסיר ספקולציות מתהליך הבדיקה, ומוודא שכל הצוות נמצא באותו המקום, עם אותו מידע ובאותו כיוון.

זה סביר ובסדר אם סוג חשיבה זה לא בא באופן טבעי. אחרי כמה ניסיונות (ואולי כישלונות), זה יהיה. באופן אינטואיטיבי אנו תופסים את היתרונות של איסוף עובדות, והפצתן. בשלב מסוים רואים את "הכובע הלבן" ככלי בפני עצמו. במובנים רבים, זה הכלי החשוב ביותר שלנו.

במאמר השני, בגיליון הבא, נבחן איך מסביר דה-בונו את שלבי החשיבה של:

הכובע האדום (רגשות)

והכובע השחור (מאבחן)

הצטרף לקהילות בודקים

אם לא נחשוף את עצמנו למידע ולרעיונות שונים, לא נוכל להתקדם בתחום המקצועי בצורה יעילה.
להצטרפות לקהילת בודקים יש יתרונות רבים:

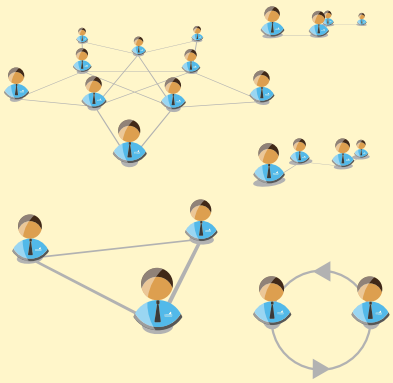
1. חשיפה למידע.
2. חשיפה לרעיונות חדשניים
3. החלפת ופיתוח רעיונות תוך כדי דיון.
4. הזדמנות למפגש עם אנשים בעלי תחומי עניין דומים ויצירת חברויות חדשות.
5. יצירת קשרים בתעשייה.
6. הזדמנויות למיצוב מחדש כאנשים מובילים בתחומנו.

נוכל להתקדם בתחום המקצועי בצורה יעילה רק אם נחשוף את עצמנו למידע ולרעיונות שונים.

גם כאשר אנו עובדים בארגון גדול, בסופו של יום אנו חשופים לכמות קטנה מאוד של בודקים אשר יכולים להפרות אותנו מבחינת שיטות עבודה, ידע טכני וכד' – אפילו אם בקבוצת הבדיקות יש 30 איש, כמה מתוכם אכן מעלים רעיונות חדשים באופן תכוף? לעתים רחוקות יגיע בודק חדש ויביא עמו כמה רעיונות או שיטות עבודה אשר אולי יגרמו לנו לבחון שנית את צורת עבודתנו, אך במרבית הזמן איננו נחשפים למידע חדש רב אלא שקועים בפעילות יום-יומית בשיטות ידועות ומוכרות.
מרביתנו לא עוצרים לחשוב האם שיטות אלו הנן היעילות ביותר לצרכינו?
אילו חידושים יש בשוק ובמקצוע?
רובנו מצטרפים לקהילות הבדיקות כאשר אנו צריכים דבר מה – יכול להיות שאנו מחפשים מקום עבודה, או יכול להיות שבדיוק קיבלנו מינוי לר"צ או מנהל קבוצה ואנו רוצים לבחון את דרכי הניהול, או שעלינו לעשות הסבה לאוטומציה וחסר לנו רקע, או אנו זקוקים לכלי זה או אחר אך איננו יודעים באיזה כלי כדאי לנו לבחור...
אכן הקהילות הנן מקור בלתי נדלה לידע ורצוי לפקוד אותן לצורך קבלת עזרה, אך כאשר מגיעים אליהן אך ורק לשם הגשמת צורך מיידי – רוב הסיכויים הם שנהיה אך אורחים לרגע ולא באמת נשתלב בקהילה.
ואכן רבים מדיי מסתפקים (לטווח ארוך או קצר) בקריאה פסיבית בלבד,
ההשתלבות האמתית בקהילה מתקיימת ברגע בו אנו מפנימים כי גם לנו יש יכולות וידע אשר יכול לעזור לאחרים, ואנו מעוניינים לתרום בחזרה היכן שניתן. ולא – אין צורך שנהיה פורצי דרך בתחומנו בכדי לתרום לאחרים, לעתים קרובות מספיק שנחלוק מניסיוננו, או נדע למי כדאי למחפש המידע לפנות ונוכל לכוונו.
לעתים קרובות אפילו שאלות שנשאל מחוסר ידע, עשויות להביא אחרים להעלות רעיונות חדשניים.
לעתים מספיק שנעזור בפן טכני זה או אחר – כמו איתור פוסטים מעניינים, בניית סקרים, אירוח או ארגון מפגש – כדי שנתחבר יותר לפעילות הקהילה.
הקהילות הווירטואליות מאפשרות לנו להשתתף במרבית הפעילות גם אם איננו ממוקמים באותו אזור כמו מרבית החברים, אמנם יקשה עלינו להצטרף למפגשים באירופה או ארה"ב או הודו לדוגמא, אך עדיין נוכל להשתתף במרבית הדיונים.
וכיום ניתן למצוא קהילות בעלות תחומי עניין מגוונים, החל מנושאים כלליים בתחום הבדיקות – כמו קבוצת הפייסבוק שלנו, דרך קבוצות ומפגשי meetup בנושאים ספציפיים כמו סלניום, אוטומציה, מובייל, אג'יל ועוד.

ההחלטה בידיכם – האם אתם רוצים להתפתח ולתרום?

הצטרפו ל: https://www.facebook.com/groups/IL.Testing.QA



חומר קריאה נוסף:
http://www.mkltesthead.com/2013/08/is-join-software-testing-club-taken-99.html

נשמח לשמוע רעיונות הערות והארות מכם הקוראים
אתם מוזמנים לקרוא טיפים רבים נוספים ב: http://goo.gl/UcW42
ולהוסיף טיפים משלכם לרשימה משותפת זו.

סדרת טיפים זו "כיצד להפוך לבודקים טובים יותר" מתבססת על דיון ב: Software Testing Club
99 Things Testers Can Do To Become Better Testers
ה-eBook החינמי שנוצר בעקבות דיון זה: 99ThingsEbook.pdf
וסדרת פוסטים מאת Michael Larsen בשם: 99 Ways Workshop – בה מיכאל מרחיב על כל אייטם וגם מספק הנחיות כיצד לתרגל את הנושא.

אחוז ניכר מהארגונים ממשיך לשלב ה-QA והבודקים מאוחר מדיי בתהליך הפיתוח

From www.worldqualityreport.com

סקרים

מימוש בדיקות אוטומטיות באמצעות BDD

גלעד ברסלאור

בשנים האחרונות אנו עדים למהפכה של ממש בכל הנוגע לתהליכי בדיקות ופיתוח תכנה. הניסיון המתמיד לשפר את איכות המערכות הביא לשלל מתודולוגיות שונות השואפות לנסות לגשר על פערי התקשורת בין הגורמים השונים המעורבים בתהליך פיתוח התכנה. גם לכלי האוטומציה שהולכים ותופסים נתח משמעותי בעולם פיתוח התכנה, השפעה דרמטית בכל הנוגע לאיכות תכנה.

אחד הקונספטים החדשניים שצצו בשנים האחרונות המשלב כלים טכנולוגיים ושיטות עבודה הוא **BDD**.

> **תסריטי הבדיקה מונעים ע"י הפן העסקי/מוצרי של המערכת**

מה זה BDD?

BDD (Behavior Driven Development) היא מתודולוגיית פיתוח המעודדת שיתוף פעולה בין כל הגורמים המעורבים בפרויקט (טכניים ולא טכניים).

מתודולוגיה זו מאופיינת בתסריטי בדיקה אוטומטיים הכתובים בשפה טבעית וקריאה שגם גורמים מחוץ לצוות הפיתוח (כמו בודקים ומנהלי מוצר) יכולים לקרוא ולכתוב. BDD היא שיטת עבודה מהחוץ-פנימה כלומר תסריטי הבדיקה מונעים ע"י הפן העסקי/מוצרי של המערכת ומתארים פעולות אמיתיות שצפויות להתבצע ע"י משתמשי המערכת. השילוב של תסריטי בדיקה קריאים המאפיינים את התנהגות המערכת מקצה לקצה הופך אותם למעיין תיעוד חי של דרישות המערכת ואף לתחליף למסמכי ה-STD הוותיקים.

קונספט זה הוא מעין אבולוציה של שיטת פיתוח ותיקה בשם **TDD (Test Driven Development)** בה הבדיקות האוטומטיות נכתבות לפני מימוש הפיצ'רים במטרה לכוון את המפתח לכתוב את הקוד בצורה נכונה שתענה על הדרישות וכפועל יוצא לכתוב קוד שניתן לבדוק בקלות. כמו **TDD** גם **BDD** נועד להכריח את המפתחים לממש את קטעי הקוד תוך מודעות לאיכות, כאשר הטסטים משמשים כבקרה ומכוונים אותם למימוש תקין, ההבדל העיקרי הוא ש-BDD ממקד את הפיתוח בהתנהגות הרצויה ונותן פחות מקום לפרשנות, אולם חשוב לציין ש-BDD לא נועד להחליף את TDD אלא לשמש כשיטה משלימה.

BDD בעולם ה-Agile

האבולוציה של שיטות עבודה אג'יליות היוו קרקע פורייה להתפתחות הקונספט.

בעולם הפיתוח האג'ילי נהוג להציג את הדרישות כאוסף של התנהגויות (בצורה של User Stories) וכמו כן יש צורך בפידבק מהיר מאוד. לכן שיטת עבודה כזו שמתמקדת באיכות ומאפשרת לקבל אינדיקציה מהירה תוך כדי תהליך הפיתוח, משתלבת היטב בתהליך האג'ילי.

מנהל קבוצת D&R בחברת Leverate.
עוסק בבדיקות תכנה מזה 10 שנים כבודק וכמנהל צוותים וקבוצות.
בעל ניסיון בהטמעת תהליכי אוטומציה על פלטפורמות וטכנולוגיות שונות כגון:
Selenium לבדיקות ווב, Appium לבדיקות מובייל, SpecFlow לבדיקות BDD ועוד.

הרעיון הוא כזה:

- מנהל המוצר (או כל דמות רלוונטית אחרת) מגדיר דרישות באמצעות User Stories.
- מפתח/בודק/איש אוטומציה כותב תסריטים שמתארים את ההתנהגות הרצויה במצבים שונים
-
- במקביל (או לאחר מכן) המפתח מממש את ה-User Stories
- לאחר המימוש, התסריטים מורצים על מנת לוודא שיש תאימות בין ההתנהגות הרצויה להתנהגות בפועל. רק לאחר שהתסריטים עוברים ובהינתן וכל שאר הפעולות ההכרחיות בוצעו (בדיקות ידניות, תיעוד וכו'), ה-User Story נחשב ל-Done.
- מאותו רגע התסריטים משמשים סוג של דרישות הניתנות להרצה, ומספקים פידבק לגבי התקדמות הצוות לקראת סיום פיתוח של פיצ'ר מסוים. כמו כן בשיטת עבודה זו אנו בעצם בונים מערכת בדיקות רגרסיה אוטומטית כחלק מתהליך הבדיקות השוטף.

> **הפורמט בו נכתבים התסריטים הוא תבנית מוגדרת מראש כאשר ברוב הכלים המקובלים היום משתמשים בשפה הנקראת Gherkin**

איך זה עובד בפועל?

קודם כל יש לבחור את הכלי המתאים לפיתוח BDD בהתאם לשפת הפיתוח בה אנו מעוניינים לפתח. באופן כללי כלי BDD הן בעצם ספריות פיתוח (Frameworks) שמאפשרות כתיבת תסריטים המדמים התנהגות בשפה קריאה. ישנם כלים שונים שמתאימים לטכנולוגיות פיתוח שונות כגון: Cucumber,SpecFlow,JBehave ועוד.

פרויקט BDD מורכב משתי שכבות מרכזיות (לפחות): שכבת התסריטים (Scenarios) ושכבת המשפטים (Steps). שכבת התסריטים מייצגת את ההתנהגות שאותן אנו רוצים לבדוק בצורה אוטומטית, ושכבת המשפטים מייצגת את החיבור בין משפט לבין פונקציה הממשת לוגיקה רלוונטית.

מימוש בדיקות אוטומטיות באמצעות BDD

גלעד ברסלאור

הפורמט בו נכתבים התסריטים הוא תבנית מוגדרת מראש כאשר ברוב הכלים המקובלים היום משתמשים בשפה הנקראת Gherkin בצורה הבאה:

Given: A pre-condition is fulfilled

When: some action is carried out

Then: There is certain outcome

דוגמא:

1) בהינתן ה-User Story הבא:

As a user

I want to be able to reset my password

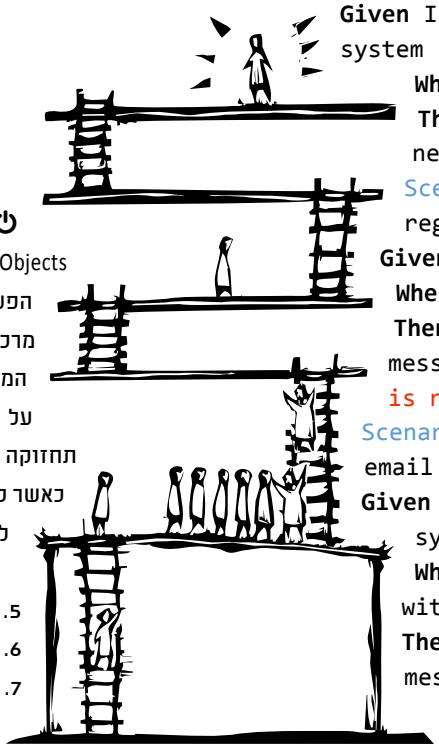
So that I can login to the site in case I forgot it.

Acceptance Criteria:

1. Entering a valid email address and pressing the "Reset Password" button should trigger an email with a new password.
2. Entering a non-registered address and pressing the "Reset Password" should show the following alert to the user: "The Email you have entered is not registered in the system".
3. Entering a non-valid email address and pressing the "Reset Password" button should show the following alert to the user: "The Email you have entered is not valid".

2) נכתוב את התסריטים הבאים:

```
Scenario: Reset password for registered user
Given I am a registered user in the system
When I try to reset my password
Then I receive an email with a new password
Scenario: Reset password for non-registered user
Given I am a non-registered user
When I try to reset my password
Then I see the following error message: The Email you have entered is not registered in the system
Scenario: Reset password with invalid email
Given I am registered user in the system
When I try to reset my password with a non-valid email
Then I see the following error message: The Email you have entered is not valid
```



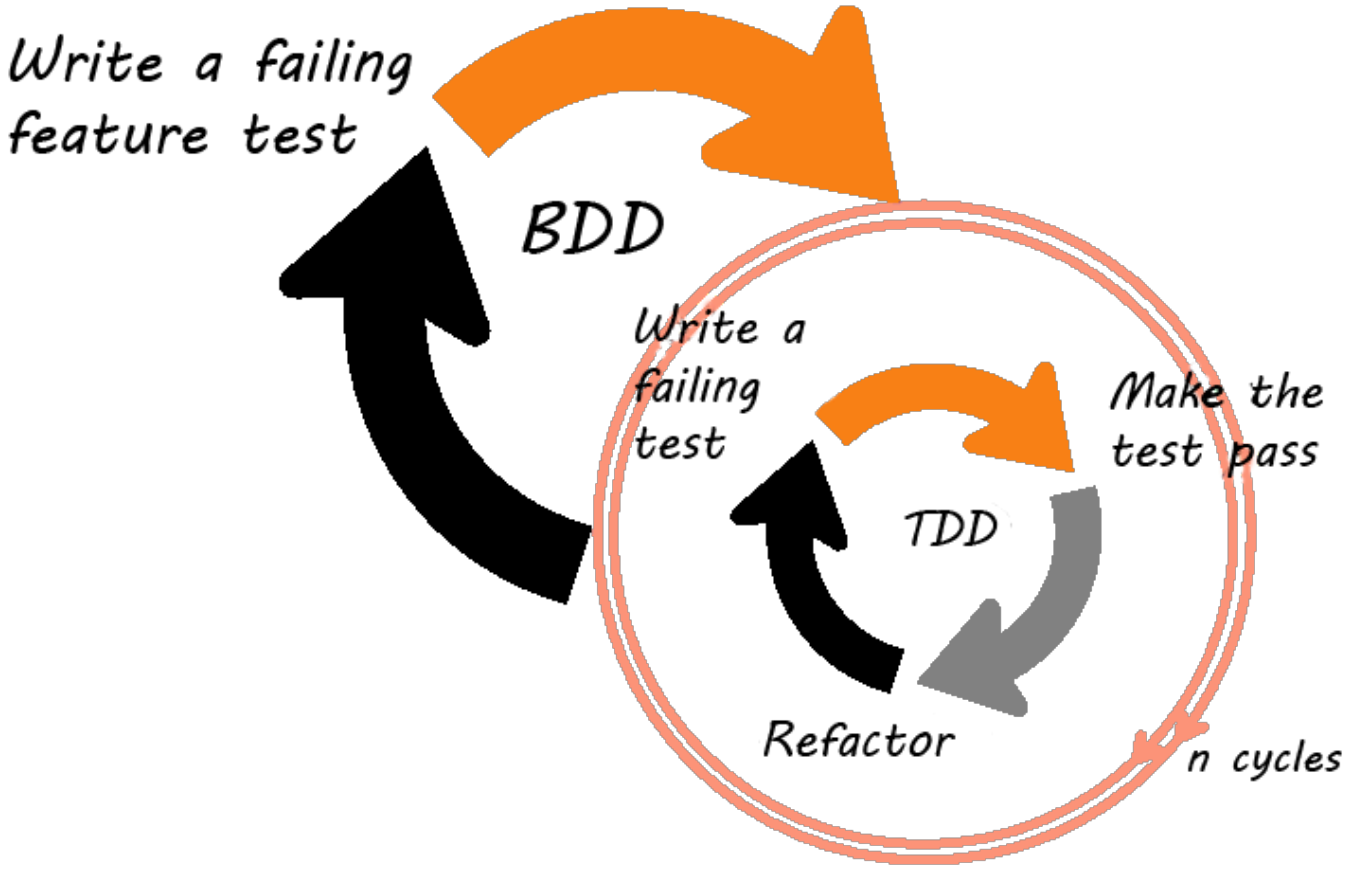
3) עבור כל משפט נממש פונקציה רלוונטית:

```
[Given(@"I am a registered user in the system")]
public void GivenIAmARegisteredUserInTheSystem()
{
    RegisteredUser =  UserAPI.GetRegisteredUser();
}

[When(@"I try to reset my password")]
public void WhenITryToResetMyPassword()
{
   forgotPasswordPage.email = RegisteredUser.email;
   forgotPasswordPage.ResetButton.Click();
}

[Then(@"I see the following error message")]
public void ISeeTheFollowingErrorMessage(string errorMessage)
{
   Assert.AreEqual(errorMessage,     ForgotPasswordPage.ErrorMessage);
}
```

⏻ בדוגמאות שלעיל קיימת שכבה נוספת של לוגיקה (Page Objects) שבה מבוצעות הפעולות בפועל. Page Objects היא בעצם תבנית בה כל חלק מרכזי בממשק (דף, רכיב וכו') ממודל ל-Class החושף את כל המידע והפעולות שאותו ממשק מציע (מילוי שדות, לחיצה על כפתורים וכו'). התבנית הזו מאפשרת שימוש חוזר בקוד, תחזוקה קלה והופכת את הקוד ליותר קריא וברור. הדבר מאוד מקל כאשר כותבים תסריטים באמצעות BDD מאחר ולעיתים נצטרך לקרוא לאותה פונקציה בקוד מתוך משפטים שונים.

5. נריץ את התסריטים ונראה אותם נכשלים
6. נממש את הפיצ'ר הרלוונטי
7. נריץ שוב את התסריטים
8. נתקן באגים ונריץ שוב עד שכל התסריטים יעברו

מימוש בדיקות אוטומטיות באמצעות BDD
גלעד ברסלאור



מקור התמונה: http://elnur.pro/bdd-does-not-replace-testing

לסיכום

ע"י יישום BDD בתהליכי הפיתוח והבדיקות אנחנו יכולים לשפר את איכות הקוד שנכתב, לצמצם את זמן הכתיבה והתחזוקה של בדיקות אוטומטיות ולקבל פידבק מאוד מהיר לגבי איכות המוצר תוך כדי תהליך הפיתוח. אמנם יישום תהליך כזה במחזור חיי המוצר הוא ארוך וכרוך בהרבה חבלי לידה, אולם לאורך זמן עשוי לשפר בצורה ניכרת את תהליכי העבודה ובשורה התחתונה לתת ROI חיובי.

חומר קריאה נוסף:

- הבלוג של דן נורת' הוגה קונספט ה-BDD ומפתח JBehave
  http://dannorth.net/introducing-bdd/
- Specification By Example: ספר חובה לכל מי שמתעניין ב-BDD
  http://specificationbyexample.com

האתגר העיקרי בבדיקת אג'יל הנו חוסר בתהליכים ייחודיים ושיטות אוטומציה

From www.worldqualityreport.com

סקרים

פיתוח מונחה-בדיקות (Test Driven Development)

ההגדרה

פיתוח מונחה-בדיקות : שיטה לפיתוח תוכנה לפיה מפתחים מקרי בדיקה, ותכופות גם ממכנים אותם, לפני פיתוח התוכנה עליה ירוצו מקרי הבדיקה, ושאותה הם יבדקו (מתוך ISTQB Glossary).

מה זה אומר?

פיתוח מונחה-בדיקות הינה גישת פיתוח הנשענת על מחזורי פיתוח קצרים החוזרים על עצמם. המפתח כותב מקרי בדיקה (ברוב המקרים מקרי בדיקה אוטומטיים) לבדיקת השינוי הנדרש, ולאחר מכן כותב קוד מינימלי על מנת שיעבור את הבדיקה שנכתבה.
הבדיקות נכתבות עבור משתנים, אובייקטים, פונקציות, קומפוננטות או מודולים שלמים. כיסוי הבדיקות כולל את הדרישות אך גם את המקרים המיוחדים ומקרי הקצה.
מקרי הבדיקה הנכתבים ע"י המפתחים יכולים להיות מקרי שימוש (Use Cases) או סיפורי משתמש (User Stories).
בניגוד להרבה דעות בשוק, פיתוח מונחה-בדיקות אינה גישת בדיקות אלא גישת פיתוח, למרות שליישום הגישה יש השפעה רבה על תפקיד צוות הבדיקות בארגון כפי שיפורט בהמשך.

היסטוריה, הווה ועתיד

את גישת הפיתוח מונחה-בדיקות יצר קנט בק בשנת 2003. הגישה נוצרה בתחילת המגמה של מעבר לפיתוח אג'יל בדגש על גישת Extreme Programming (שם מתוארת הגישה כ-test-first programming).
כיום הגישה הופכת לרווחת עם יישום שיטות פיתוח מואצות כגון Continuous Integration ו-DevOps. שיטות אלו מחייבות כיסוי מלא של אוטומציה לכלל מרכיבי המערכת והרצה מהירה של הבדיקות לצורך תמיכה במחזורי פיתוח של ימים ולעתים אף של שעות. פיתוח מונחה בדיקות מקל מאוד על יישום שיטות אלו שכן האוטומציה הינה טבועה בקוד וקל מאוד להריץ בדיקות רגרסיה ולאפשר מחזורי פיתוח מהירים, לעתים של שעות ספורות.

ההבדל בין בדיקות יחידה (Unit Testing) ופיתוח מונחה-בדיקות

בניגוד לבדיקות יחידה בהם הבדיקות נכתבות לאחר כתיבת הקוד, בפיתוח מונחה-בדיקות הבדיקות נכתבות לפני כתיבת הקוד.
בפיתוח מונחה-בדיקות הפוקוס של המפתח הוא על הבנת הדרישות ויישום בתהליכים עסקיים בעוד שבבדיקות יחידה הפוקוס הוא על הקוד שנכתב. הבדל שנשמע זניח אולם הוא מהותי ביותר.
היתרון הנוסף והלא פחות משמעותי הוא שפיתוח מונחה-בדיקות מאלץ את המפתחים לקחת אחריות על איכות הקוד אותו הם מפתחים, כפי שתיארה אליזבט הנדריקסון במאמר "בדיקות טובות יותר, איכות גרועה יותר"[1] – ככל שצוות הבדיקות טוב יותר, המפתחים לוקחים פחות אחריות ופשוט "זורקים" קוד ל-QA, דבר שיוצר יותר עבודה לצוות הבדיקות ופוגע באיכות המערכת.

תפקיד הבדיקות בפיתוח מונחה-בדיקות

תפקיד הבדיקות בפיתוח מונחה-בדיקות שונה מארגון לארגון. החל בביטול חלק מצוות הבדיקות והשארת רק צוותי בדיקות אינטגרציה ומערכת וכלה במתן אחריות מלאה על היישום לצוות הבדיקות. ללא קשר למהות השינוי, תפקיד צוות הבדיקות משתנה בארגונים שעוברים לפיתוח מונחה-בדיקות ונוספים תחומי אחריות נוספים שעיקרם:

- סיוע למפתחים בהבנת הדרישות ובקרה על כיסוי מלא הדרישות
- בקרה על הבדיקות המתוכננות (לפני או במקביל לבניית הקוד) וניתוח "נקודות שחורות" – תהליכים או מצבים שלא מכוסים בבדיקות
- ניהול מלאי הבדיקות והרצה של בדיקות רגרסיה, לצד אחריות על בדיקות ממשק UI (ובדיקות קצה לקצה) End to End
- תפקיד הבדיקות בפיתוח מונחה-בדיקות מחייב יותר אינטראקציה ותקשורת מול המפתחים כבר בשלב כתיבת הקוד.

קבלת פיתוח מונחה-בדיקות (ATDD)

קבלת פיתוח מונחה-בדיקות היא גישה קשורה, אך שונה לחלוטין מגישת פיתוח מונחה-בדיקות. גישה זו היא גישת תקשורת בין הלקוח הסופי, המפתח והבודק שנועדה לוודא שהדרישות הוגדרו בצורה תקינה. בגישה זו נכתבות בדיקות קבלה, במקביל או אף לפני כתיבת הדרישות עצמם.

סיכום

גישת פיתוח מונחה-בדיקות הינה גישת פיתוח, אולם לצוות הבדיקות שנושא איכות המערכת עומד לנגד עיניו, אינטרס מהותי ביישום גישה זו ע"י הפיתוח. הגישה תאפשר לצוות הבדיקות לקבל קוד טוב יותר, ולצוות הפיתוח ליישם גישה שתייעל להם את העבודה בצורה משמעותית.

1. http://testobsessed.com/wp-content/uploads/2011/04/btwq.pdf

תהליך אופטימיזציה למציאת סט בדיקות אופטימלי | אלון לינצקי

אלון לינצקי – בשלושים השנים האחרונות התמקצע, הדריך ואימן קבוצות וחברות בפיתוח, בדיקות והבטחת איכות.
תחומי המומחיות העיקריים של אלון הינם: עיצוב בדיקות, תכנון ואופטימיזציה של בדיקות, ניהול יעיל של בדיקות, שיפור ואופטימיזציה של תהליכי בדיקות ופיתוח, בדיקות אוטומטיות בפרוייקטים, ניהול בדיקות מבוסס סיכונים, מדדים ומטריקות לניהול איכות ובדיקות, בניית צוותים יעילים, שיפור תהליכי פיתוח ואיכות, אג'ייל ומעבר לאג'ייל.
אלון מרצה בינ"ל פופולארי מאז 1995, מוביל תוכנית שותפים של ISTQB® Partner Program,
מייסד SIGiST Israel – The Israeli Testing Forum, מייסד שותף של ITCB® – Israel Testing Certification Board. היה אחד מכותבי הסילבוס של ISTQB® Agile Tester – Foundation Level Extension, סגן נשיא ודירקטור שיווק של ITCB®.

הקדמה

הרבה פעמים ראיתי בודקים משתאים ושואלים את עצמם איך ינתחו מסמכי אפיון ועיצוב המערכת וימירו אותם לתרחישי בדיקות משמעותיים. איך יוכלו לבחור את התרחישים המשמעותיים הנכונים עבור כיסוי המערכת, ודרישות הביזנס המוצגות למערכת, כמו גם השכיחות (של השימוש) ורמת הסיכון.
ריצה מהירה – ממסמכים לתרחישי בדיקות – זה משהו שהרבה בודקים מנוסים/בכירים מבצעים, כאשר מבקשים מהם לנתח מסמכי אפיון ולהציג תרחישי בדיקות.
במאמר זה אנתח שיטה ותהליך שאני מבצע כדי לנתח מסמכי אפיון ועיצוב מערכת, כדי להגיע מהר ככל הניתן לתרחישי בדיקות עם כיסוי טוב, ועם סיכון נמוך, בכמות אפשרית לביצוע. את השיטה אפשר ליישם בכל מתודולוגיית פיתוח, מהשיטות המסורתיות ועד שיטות אג'יליות כמו Scrum או Kanban.

אציג:

1) **שלב אנליזה – ניתוח המסמכים**
   - א ניתוח מסמכי האפיון ועיצוב המערכת, ומציאת פרמטרים וערכים חותכים, שמגדירים את מרחב הבדיקות האפשרי, ומסבירים לנו את האתגר שאנו ניצבים בפניו.
   - ב חישוב הכמות המקסימלית (תאורטית) של בדיקות לכיסוי מלא של המערכת הנבדקת (עפ"י המסמך המנותח)

2) **אופטימיזציה**
   - א ניתוח הפרמטרים והערכים המשמעותיים ואישורם
   - ב הקטנת כמות הפרמטרים והערכים למינימום (על בסיס קלט ממנהל המוצר ומחלקת פיתוח)
   - ג חישוב מחודש של כמות התרחישים שיש לבצע

3) **ביצוע הערכה**
   - א סקירה של סט התרחישים שקיבלנו, וקבלת משוב מבעלי עניין לאישורו
   - ב קבלת החלטה לגבי אוטומציה של חלקם/כולם

> **להגיע מהר ככל הניתן לתרחישי בדיקות עם כיסוי טוב, ועם סיכון נמוך, בכמות אפשרית לביצוע**

היתרונות מהגישה המוצגת הינם רבים:

1. כיסוי אספקטים פונקציונאליים קריטיים במערכת
2. הקטנת מספר התרחישים, תוך כדי שמירה על סיכון נמוך בכיסויה
3. הגדלת רמת הידע של הבודקים בעת ניתוח מסמכי האפיון והעיצוב של המערכת
4. מיקוד בתרחישי הבדיקה העיקריים והחשובים לבדיקה
5. בניית מאגר רגרסיה משמעותי וחזק לפעמים הבאות

אם גם אתם מעוניינים לחלוק שיטות ניתוח מסמכי מערכת וכתיבת תרחישים בדרך שאתם משתמשים בה והיא טובה בעינכם, אשמח לשוחח בנושא: אלון לינצקי, alonl@sigist.org.il

שלב האנליזה – ניתוח מסמכים

כאשר אנחנו קוראים ומנתחים מסמך אפיון ו/או עיצוב מערכת, או קטע טקסט קצר המסביר פונקציונליות של יכולות מערכת שאמורה להיות מפותחת, אנו יכולים למצוא פרמטרים, אשר כל שינוי בערכים שלהם, ישנו את "המסלול" בו עוברת התוכנה, ולמעשה המשתמש שלנו, על ידי שימוש בכל שיטה אפשרית (מקלדת, נגיעה במסך, או כל אמצעי הזרקת ערך אחר אפשרי למערכת) יקבל מהלך אחר של זרימת המידע במערכת. שינוי ערך זה, לא רק יביא למסלול אחר עבור המשתמש, אלא גם לתוצאות צפויות אחרות – בין אם הן שינוי ערכים במסך, במסד הנתונים, בנתונים שעוברים בתקשורת וכו'.

ניתן לומר בוודאות, שלא רק הפרמטרים חשובים כאן, והערכים המוזנים בהם, אלא גם צירופי הערכים עבור אותם פרמטרים (למשל עבור אותו מסך, או אותה שאילתה).

פרמטרים וערכים של אותם פרמטרים, הם שני הדברים הראשונים שאנחנו מעוניינים לזהות בעת קריאה וניתוח הטקסט המסביר את הפונקציונליות של המערכת.

תהליך אופטימיזציה למציאת סט בדיקות אופטימלי | אלון לינצקי

דוגמא לפרמטרים וערכים:

למערכת הזמנת טיסות, יש מסך: "הזמן טיסה", בו יש מספר שדות בהם ניתן להזין ערכים שונים. להלן דוגמאות לפרמטרים/ערכים:

סוג טיסה: הלוך-חזור, הלוך, חזור

משדה תעופה: כאן יש שדה בחירה, של שדה תעופה ממנו יוצאת הטיסה. הערכים עלפי המקומות שחברת התעופה יוצאת ממנה עם טיסות.

לשדה תעופה: כאן יש שדה בחירה, של שדה תעופה אליו מגיעה הטיסה. הערכים עלפי המקומות שחברת התעופה מגיעה אליהם עם טיסות.

מתאריך: תאריך ולידי, גדול שווה להיום.

עד תאריך: תאריך ולידי, גדול שווה למחר.

מחלקה: תיירים, תיירים פלוס, עסקים, ראשונה.

כבודה: ללא, יש

מספר נוסעים: 1, 2, 3, 4, 5, יותר

מספר מזוודות (קטן מ 23 ק"ג כל אחת): 1, 2, 3, 4, 5, 6, 7, יותר.

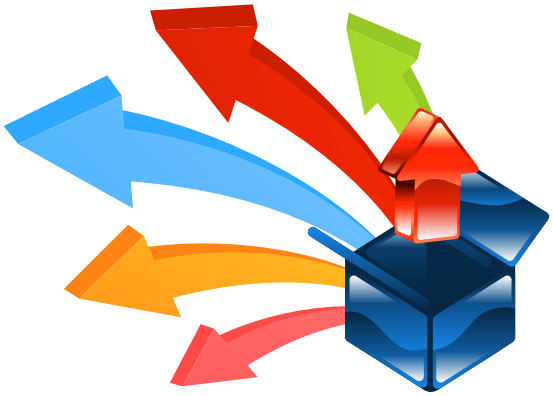
ילדים (עד גיל 18) : כן / לא

ילד 1: תאריך לידה: תאריך ולידי בין 1-1-1997

ילד 2: אותו דבר.

ילד 3: אותו דבר.

ועוד שדות נוספים.



אם ניקח את כלל השדות (ולא רק אלו המוצגים פה כדוגמא), ונחשב את המספר התאורטי של קומבינציות/תרחישים (תוך שימוש רק במספר ערכי תאריך בשדות מתאריך ועד תאריך, ורק במספר שדות תעופה בשדות משדה תעופה ואל שדה תעופה - נניח 2 ערכים לכל שדה במסך), נקבל:
מספר הערכים: סוג טיסה = 2 ערכים, משדה תעופה = 2, אל שדה תעופה = 2, מתאריך = 2, עד תאריך = 2, מחלקה = 4, כבודה = 2, מספר מזוודות (רק אם כבודה = "כן") = 8, מספר נוסעים = 6, ילדים= 2, ילד 1 תאריך לידה = 2 (תאריך ולידי ותאריך לא ולידי), ילד 2 תאריך לידה = 2, ילד 3 תאריך לידה = 2.
סה"כ כ- 196,608 אופציות (כמות מחושבת תאורטית), למסך אחד, עם הרבה הנחות לכמות הערכים והפרמטרים שנבחרו, כלומר המספר גדול יותר באתר עצמו. זהו מספר גבוה לקומבינציות על מסך אחד בלבד בוודאות, למרות היותו מסך מאוד חשוב במערכת ובשירות ללקוח. ברור לכולנו שמספר התרחישים שנרצה לבצע אינו כל כך גבוה, ובוודאות שישנן קומבינציות שהן כל כך לא שכיחות, שמעטים הסיכויים שלקוח יזין אותן למערכת בעתיד (למשל: משפחה בעלת 5 נפשות, עם 3 ילדים, שטסה כולה במחלה ראשונה, ויש להם יותר מ- 7 מזוודות).

כמובן שבמערכות המורכבות שלנו כיום, כמות הקומבינציות שתתקבל פה תהיה משמעותית גדולה יותר (לדוגמא: במערכת blood infusion pump ניתן להגיע למאות אלפים בחישוב כמות תאורטית של בדיקות לכפתור Halt לצרכי חירום כשלוקחים את כל הפרמטרים והערכים בחשבון).

Flight Ticket Website:									
Parameter	**count of Values**	**Values**							
Trip type	2	round trip	multi city						
from location	2	valid location	invalid location						
to location	2	valid location	invalid location						
from date	2	valid date	invalid date						
to date	2	valid date	invalid date						
class	4	economy	economy plus	business	first				
Baggage	2	Yes	No						
# of suite-cases	8	1	2	3	4	5	6	7	more
passengers	6	1	2	3	4	5	more		
children	2	Yes	No						
Child 1 date of birth	2	valid date	invalid date						
Child 2 date of birth	2	valid date	invalid date						
Child 3 date of birth	2	valid date	invalid date						
	196,608								

על פי הדיון מעלה, ניתן כבר לראות כי האופטימיזציה תהיה נסובה סביב קריטריון מרכזי אחד - כאן סביב שכיחות השימוש של הלקוח והסיכון שנובע מכך.
נעבור עכשיו לשלב האופטימיזציה בתהליך שלנו, בו נתחיל להבדיל בין הקומבינציות החשובות יותר והחשובות פחות - בשתי רמות ניתוח.

תהליך אופטימיזציה למציאת סט בדיקות אופטימלי | אלון לינצקי

שלב האופטימיזציה – פרמטרים וערכים

בשלב ראשון, נבחן את הפרמטרים והערכים שזיהינו עם מנהל המוצר, מחקר ופיתוח, והביזנס. ניפגש איתם ונקבל משוב על הבחירות שלנו, הסיכונים המשתמעים, ואילו פרמטרים וערכים הלקוחות משתמשים בהם באופן שכיח, או הם הקריטיים ביותר בעיניהם.
ניתוח הבעיה לדוגמא שלנו מעלה:
פרמטרים שכיחים: שדה משדה תעופה ואל שדה תעופה, מתאריך ועד תאריך, מחלקה, כבודה, מספר מזוודות, מספר נוסעים, ילדים, תאריך לידה ילד 1.
ערכים שכיחים: מקום ולידי בשדה משדה תעופה ואל שדה תעופה, תאריך ולידי בשדה מתאריך ובשדה עד תאריך, מחלקה = תיירים/עסקים, כבודה הערכים כן/לא, מספר מזוודות = 1 או 2, מספר נוסעים = 1, 2, 3, ילדים = כן ולא, תאריך לידה ולידי של ילד 1.
הערה בנושא סיכון: פרמטרים וערכים אלו מכסים משפחות קטנות ואנשי עסקים – שהינם מרבית המזמינים באתר האינטרנט כרטיסי טיסה (94%).

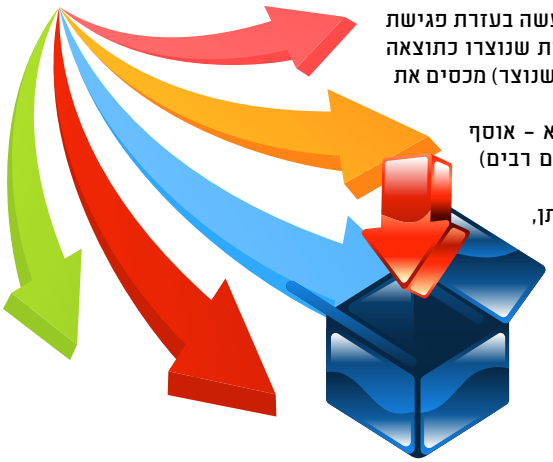
בחירה זו תיתן לנו: 1*1*1*1*1*2*2*2*3*2*1 = 48 קומבינציות (שוב, לא נפחית את כמות הקומבינציות במקומות שיש ערך "כן" בשדה מקדים ותלוי).

Flight Ticket Website:									
Parameter	**count of Values**	**Values**							
Trip type	1	round trip	~~multi city~~						
from location	1	valid location	~~invalid location~~						
to location	1	valid location	~~invalid location~~						
from date	1	valid date	~~invalid date~~						
to date	1	valid date	~~invalid date~~						
class	2	economy	~~economy plus~~	business	~~first~~				
Baggage	2	Yes	No						
# of suite-cases	2	1	2	~~3~~	4	~~5~~	~~6~~	~~7~~	~~more~~
passengers	3	1	2	3	4	~~5~~	~~more~~		
children	2	Yes	No						
Child 1 date of birth	1	valid date	~~invalid date~~						
~~Child 2 date of birth~~	1	~~valid date~~	~~invalid date~~						
~~Child 3 date of birth~~	1	~~valid date~~	~~invalid date~~						
	48								

קומבינציית תרחישי בדיקה זו, תוגדר אצלנו כקבוצה בעדיפות 1. אין זה אומר שנריץ רק את תרחישים אלו, ולא את אותם 196,560 התרחישים האחרים (ואפילו שימוש בערכים לא ולידיים הם קומבינציות נכונות לבדיקה).
כלומר, כשישנה מגבלה קשה של זמן ותקציב/בודקים, אנו בוודאות נשתמש באלו. בד"כ נגדיר את הסט המינימלי הזה כ- Sanity test set, ואזור המערכת שניתחנו, ובמקרים רבים נוסיף אליהם מספר תרחישים שליליים (Negative tests) והשלמת סט ה Sanity.
האופטימיזציה נעשתה קודם כל על ידי בחירה (selection) של פחות פרמטרים, ואח"כ על ידי בחירה של פחות ערכים. יחד קיבלנו סט זריז, ועם סיכון נמוך (לאחר אישורו על ידי בעלי העניין כפי שצויין קודם לכן).
לעיתים לא נפחית הרבה בעת בחירת הפרמטרים, ואז אנו מצפים להפחית הרבה יותר קומבינציות בבחירת הערכים.
ניתן לשלב ידע בשימושיות של המערכת על ידי הלקוחות, לצורכי הבחירה. זאת בהנחה שאנו אספנו מידע בנושא הן מניתוח הפעילות המנוטרת של המערכת, הן על ידי עזרה מאנשי התמיכה בלקוחות, מאנשי DevOps, והן לידי שימוש במידע שנאסף מהלקוחות עצמם.

> **אילו פרמטרים וערכים הלקוחות משתמשים בהם באופן שכיח, או הם הקריטיים ביותר בעיניהם.**

שלב ההערכה



שלב זה נועד לבחון שה- Sanity מתאים לדרישות הביזנס, המוצר והמחקר והפיתוח – דבר שנעשה בעזרת פגישת Review איתם. יש לעבור על הפרמטרים והערכים שאושרו קודם לכן, ולראות את הקומבינציות שנוצרו כתוצאה מביצוע האופטימיזציה (הורדת פרמטרים ו/או ערכים) תוך בחינה – האם התוצר (סט הבדיקות שנוצר) מכסים את מרבית הסיכון עבורנו.
את הצגת האופטימיזציה בפגישת ה- Review, ניתן לערוך באופנים רבים. ההעדפה שלי היא – אוסף תרחישים/קומבינציות ב EXCEL, או דרך טבלאית אחרת, ומעבר אחד אחד או (אם ישנם עדיים רבים) קבוצה קבוצה – עלפי חיתוך של ערכי פרמטרים מרכזיים.
לאחר ביצוע ה- Review ואישור, נריץ את התרחישים, נזהה תקלות – אם קיימות – נתעד אותן, ונשתף במידע הזה עם בעלי העניין.

תהליך אופטימיזציה למציאת סט בדיקות אופטימלי | אלון לינצקי

את כלל תהליך האופטימיזציה, ניתן לבצע כמובן בעזרת כלי Pairwise, וניתן להוריד כלים חינמיים, Open source בנושא מאתרים שונים (לדוגמא: PICT של מיקרוסופט). אך יש לזכור כי כלים אלו לא "יגידו" לנו מהי הקריטיות של הערכים מבחינת המערכת עבור הלקוחות שלנו, או עבור הביזנס. במקרה שלנו, pairwise

```
Trip-type:round trip,multi city
from-location:valid location,invalid location
to-location:valid location,invalid location
from-date:valid date,invalid date
to-date:valid date,invalid date
class:economy,economy plus,business,first
Baggage:Yes,No
#No-of-suite-cases:1,2,3,4,5,6,7,more
passengers:1,2,3,4,5,more
children:Yes,No
Child-1-date-of-birth:valid date,invalid date, NA
Child-2-date-of-birth:valid date,invalid date, NA
Child-3-date-of-birth:valid date,invalid date, NA

IF [children] = "No" THEN [Child-1-date-of-birth]="NA" AND [Child-2-date-of-birth]="NA" AND [Child-3-date-of-birth]="NA";
IF [children] = "Yes" THEN [Child-1-date-of-birth] <>"NA" AND [Child-2-date-of-birth] <>"NA" AND [Child-3-date-of-birth] <>"NA";
```

צמצם את 196,608 מקרים להיות 25 מקרים:

Trip-type	from-location	to-location	from-date	to-date	class	Baggage	passengers	children	Child-1-date-of-birth	Child-2-date-of-birth	Child-3-date-of-birth
multi city	invalid location	valid location	invalid date	invalid date	economy plus	Yes	4	Yes	valid date	valid date	valid date
round trip	valid location	invalid location	valid date	valid date	business	No	more	Yes	invalid date	invalid date	invalid date
multi city	valid location	valid location	invalid date	valid date	business	Yes	3	No	NA	NA	NA
multi city	invalid location	invalid location	valid date	invalid date	economy	Yes	5	Yes	invalid date	invalid date	valid date
round trip	valid location	invalid location	invalid date	invalid date	first	No	2	Yes	valid date	valid date	invalid date
round trip	invalid location	invalid location	valid date	invalid date	business	No	4	No	NA	NA	NA
round trip	valid location	valid location	valid date	valid date	business	No	5	Yes	valid date	valid date	valid date
multi city	invalid location	valid location	invalid date	valid date	economy plus	No	1	Yes	invalid date	invalid date	invalid date
round trip	valid location	valid location	invalid date	invalid date	economy	Yes	more	No	NA	NA	NA
multi city	valid location	invalid location	invalid date	valid date	economy	No	4	Yes	valid date	invalid date	invalid date
round trip	invalid location	invalid location	valid date	valid date	economy plus	Yes	more	No	NA	NA	NA
multi city	invalid location	valid location	valid date	valid date	first	Yes	more	Yes	invalid date	valid date	valid date
multi city	valid location	invalid location	invalid date	valid date	first	Yes	5	No	NA	NA	NA
multi city	invalid location	valid location	valid date	valid date	economy	Yes	2	No	NA	NA	NA
round trip	invalid location	invalid location	valid date	invalid date	economy plus	No	3	Yes	valid date	valid date	invalid date
round trip	valid location	invalid location	valid date	invalid date	first	Yes	1	Yes	valid date	valid date	valid date
multi city	valid location	valid location	invalid date	invalid date	first	Yes	4	Yes	invalid date	invalid date	invalid date
round trip	valid location	invalid location	invalid date	valid date	first	No	3	Yes	invalid date	invalid date	valid date
multi city	invalid location	invalid location	valid date	valid date	business	No	1	No	NA	NA	NA
round trip	invalid location	invalid location	valid date	invalid date	economy	Yes	3	Yes	valid date	valid date	valid date
round trip	valid location	invalid location	invalid date	invalid date	economy plus	Yes	5	Yes	invalid date	invalid date	invalid date
round trip	valid location	invalid location	invalid date	valid date	economy plus	Yes	2	Yes	invalid date	invalid date	valid date
round trip	valid location	invalid location	invalid date	valid date	business	No	2	Yes	invalid date	valid date	valid date
multi city	valid location	valid location	valid date	invalid date	economy	No	1	Yes	valid date	valid date	invalid date
round trip	valid location	valid location	invalid date	invalid date	economy plus	Yes	more	Yes	valid date	invalid date	valid date

לאחר עדכון סט התרחישים שלנו (האופטימלי), והרצת כלי אופטימיזציה לתרחישי הבדיקות, סיימנו את שלב ההערכה והאופטימיזציה באופן מיטבי.
סיום שלב זה, עוזר לנו מאוד בהקמת סט בדיקות הרגרסיה העתידיים שלנו (עדיין יש לזכור שפרדוקס ה- Pesticide עדיין עומד מעל ראשנו... - המערכת מתחסנת בפני בדיקות הרגרסיה שלנו).

לסיכום

ניתוח מסמכי אפיון ועיצוב מערכת לצרכי כתיבת סט תרחישי הבדיקות הנכון עבור שחרור הגרסה, אינו תהליך קל ו/או פשוט. מצד אחד עלינו לבחור סט מצומצם ככל הניתן, ומצד שני עלינו לבחור סט כזה שייתן מענה במישור הסיכון, שכיחות השימוש, אופי פעילות הלקוחות, ורצונות הביזנס ועדיין יהיה בר ביצוע במגבלות הזמן/משאבים/תקציב שלנו.
זיהוי סט קומבינציות מינימלי אופטימלי (min-max) העונה לדרישות הללו היא פעילות רצויה ומחויבת המציאות עבורנו הבודקים, ושימוש בתהליך מסודר ובכלים (דוגמת Pairwise) שמסייעים בתהליך היא פעולה שלדעתי כל בודק בכיר צריך לדעת לעשות ולהדריך אחרים בנושא בארגון.
בכל מקרה, אם אפשר לייצר אוטומציה לבדיקה של כלל הקומבינציות - למה לא?
בהצלחה!

מהו DWH ואיך בודקים אותו | רם קדם

מהו Data Warehouse ואיך בודקים אותו?

מחסני נתונים, (DWH, Data Warehouse), הם חלק אינטגרלי מכל ארגון אשר שואף לקבל תובנות עסקיות מהמידע שקיים ברשותו. ארגונים רבים משקיעים זמן, כסף ומאמץ רב בבניית מחסני נתונים ובעבודה מול כלי תחקור שונים. ואכן, פיתוח מסוג זה הוא פעולה מורכבת שדורשת תכנון קפדני, הבנה עסקית ומעורבות של גורמים רבים בארגון.

מאמר זה נועד לתאר את סוגי מאגרי המידע השונים, את התהליך שמעביר מידע ביניהם, את כלי ה – BI השונים, ואת האופן שבו ניתן לבדוק את כל הרכיבים הללו.

מהו Data Warehouse ומה ההבדל בינו לבין OLTP ?

את מאגרי המידע המנוהלים בצורה טבלאית (RDBMS) ניתן לחלק לשתי קטגוריות מרכזיות: מערכות תפעוליות (OLTP) ומחסני נתונים (DWH). ההבדלים המהותיים בין שתיהן מתבטאים בצורה בה הטבלאות מתוכננות ובמטרה לשמה הן קיימות.

(Online Transaction Processing) OLTP

מאגר נתונים תפעולי (OLTP), הוא מאגר שנועד לנהל את הנתונים המתעדכנים ברמה היומיומית, לדוגמה, בחברת תקשורת סלולרית יוקם מאגר נתונים לטובת לקוחות החברה. כל שינוי שלקוח יבצע יעודכן במאגר נתונים זה: שינוי חבילה, הוספת מנוי, עדכון כתובת וכד'. מאגר נתונים הזה ישמש את אנשי החברה, כגון נציגי השירות ואנשי התמיכה, כדי לנהל את הפעילות היומיומית שלה.

מאגר נתונים זה לא נועד לשמור מידע היסטורי, לדוגמה, נציג שירות לא יוכל להשתמש במאגר נתונים זה כדי לדעת באיזו חבילת תקשורת היה לקוח מסוים לפני שלוש שנים וכמה שילם עליה. עבור מידע מסוג זה קיים מחסן הנתונים.

(Data Warehouse) DWH

מחסן נתונים (DWH) הוא מאגר שאוגר מידע שמגיע ממסד הנתונים התפעולי, ותפקידו לנהל את הנתונים הארכיוניים כדי לספק תשובות לשאלות עסקיות שונות.

בדוגמה בה הסתכלנו קודם, תעביר חברת התקשורת כל פרק זמן מסוים את נתוני המערכת התפעולית של לקוחות החברה למחסן הנתונים, כך שהן נתוני הלקוח העכשוויים והן אלו ההיסטוריים ישוקפו במאגר. לדוגמה, בעוד שהמערכת התפעולית תראה שכרגע לקוח מסוים משלם 29 ₪ לחודש, מחסן הנתונים יראה שלפני מספר חודשים הוא שילם 69 ₪ לחודש, לפני שנה 99 ₪, ולפני דצמבר 2011, מסיבה עלומה, שילם בממוצע 500 ₪.

באמצעות הנתונים שנאגרים במחסן הנתונים, גופים שונים בחברה יכולים להפיק מגוון תובנות. לדוגמה, אנשי המכירות יוכלו להפיק תובנות לגבי הצלחת קמפיינים פרסומיים. הם יוכלו לשאול: כמה לקוחות חדשים, באזור גוש דן, בגילאי 25–20, רווקים, הצטרפו אל החברה בעקבות קמפיין הפרסום האחרון ביחס לכמות המצטרפים בחיתוך זה בכל רבעון בעשור האחרון. כמו כן, מחלקת שימור הלקוחות תוכל לבדוק את שיעור הנטישה

רם קדם

dba בעל שנות ניסיון רבות בתחום מסדי הנתונים על צורותיו השונות. מרצה, מעביר קורסי הכשרה בנושא, ומנהל אתר לימודי העוסק בעולם תוכן זה.

> **פיתוח מחסן נתונים הינו פעולה מורכבת שדורשת תכנון קפדני, הבנה עסקית ומעורבות של גורמים רבים בארגון.**

הרבעוני מול אסטרטגיות שימור שונות בהן נקטה כדי להחליט על הגישה היעילה ביותר, ועוד.

נסכם ונאמר כי Data Warehouse הוא לרוב מאגר נתונים טבלאי שמכיל נתונים היסטוריים ומתוכנן בעיקר עבור ניתוחים עסקיים (Subject-oriented) ופחות עבור טרנסאקציות. וכאמור, המידע של מחסן הנתונים מגיע אחת לפרק זמן מסוים מן המערכות התפעוליות. לתהליך העברת נתונים זה אנו קוראים ETL.

ETL – Extract Transform Load

תהליך העברת הנתונים מן המערכת התפעולית אל מחסן הנתונים מתחלק לשלושה שלבים:

EXTRACT – בשלב הראשון הנתונים מחולצים מן המערכת התפעולית לקראת העברתם אל מחסן הנתונים. בשלב זה אנו קובעים אילו טבלאות ועמודות יהיו מעורבות בתהליך.

TRANSFORM – בשלב השני, לקראת העברת הנתונים, אנו יכולים לבחור לבצע שינויים שונים על נתוני המערכת התפעולית אשר חילצנו. לדוגמה, משיקולי תכנון שונים, העמודות במחסן הנתונים לא תמיד יהיו תמונת מראה של העמודות במערכת התפעולית. כך למשל, בעוד ששם הלקוח במערכת התפעולית יכול להיות מורכב משתי עמודות: Last Name, First Name, במחסן הנתונים, שמו יורכב מעמודה אחת: Full Name. במהלך טרנספורמציה מסוג זה, נחבר באמצעות שרשור (Last Name + ' ' + First Name) את שתי העמודות המחולצות ונייצר "עמודה מחושבת" חדשה, שבה נשתמש לשלב השלישי והאחרון, שלב ה Load.

LOAD – בשלב השלישי והאחרון של העברת הנתונים, נטען את הנתונים שחולצו מהמערכת התפעולית (ושאולי עברו טרנספורמציה כלשהי) אל מחסן הנתונים, ונקבע את העמודות וטבלאות היעד.

לאחר שתהליך ההעברה הסתיים, ניתן להתחיל ולנתח את הנתונים במחסן הנתונים באמצעות כלי BI שונים.

מהו DWH ואיך בודקים אותו | רם קדם

Business Intelligence

ההגדרה הכללית של כלי Business Intelligence (BI) היא קבוצת יישומים שנועדו להקל על המשתמש לפענח את המידע הממוקם במחסן הנתונים ולהפיק ממנו תובנות. כיום קיימים כלי בינה עסקית שונים בעלי שימושים רבים: כריית מידע (Data Mining), עיבוד אנליטי מקוון (OLAP), חיזוי אנליטי (Predictive Analytics), דוחות משתמשי קצה (Reporting) ועוד.

בדיקות DWH

אז איך בודקים את ה DWH? ראשית, חשוב להבין כי בדיקת מחסן הנתונים היא פעולה שמערבת גורמים שונים בחברה, החל מהאנליסטים אשר מגדירים את הדרישות העסקיות, ה- DBA שיוצרים את התשתית ובודקים את הביצועים ואנשי ה- QA שמבצעים את הבדיקות. יש לציין שעל פי רוב, תידרש מאנשי ה QA רמת שליטה גבוהה ב SQL על מנת לבצע בדיקות אלו.

את בדיקת ה DWH נהוג לחלק לארבעה חלקים: מקור הנתונים, שכבת ה- ETL, היעד (ה- DWH) וכלי ה- BI אשר פועל על ה- DWH. כל אחד מחלקים אלו דורש בדיקות נקודתיות הרלוונטיות אליו.

בדיקת מקור הנתונים

לעיתים, יגיעו אל מחסן הנתונים נתונים ממקורות שונים ולא רק ממאגרי מידע טבלאיים, לדוגמה: קבצי אקסל, XML, CSV וכד'. בשלב בדיקות זה יש לשים דגש על אינטגרציה תקינה בין מקורות המידע השונים (כגון תאימות מבני נתונים ו- Data types), טיוב נתונים בסיסי, ווידוא שכל המידע הנדרש נגיש, שאין נתונים חסרים, וכי המידע תואם את הלוגיקה העסקית.

> **Business Intelligence – BI – היא קבוצת יישומים שנועדו להקל על המשתמש לפענח את המידע הממוקם במחסן הנתונים ולהפיק ממנו תובנות.**

בדיקת שכבת ה ETL והיעד (DWH)

יש להבחין בין סוגים שונים של תהליכי ETL:

- תהליך ETL שעובד על DWH חדש, טעינה ראשונית של נתונים.
- תהליך ETL שפועל על DWH קיים ומחליף תהליך נוכחי. לדוגמה: כדי לשפר ביצועים או לחסוך בעלויות, לקוח יכול להחליט שתהליך ETL שרץ באמצעות כלי מסויים, ירוץ מעתה באמצעות כלי אחר.

> **את בדיקת ה DWH נהוג לחלק לארבעה חלקים: מקור הנתונים, שכבת ה-ETL, היעד (ה-DWH) וכלי ה-BI אשר פועל על ה-DWH.**

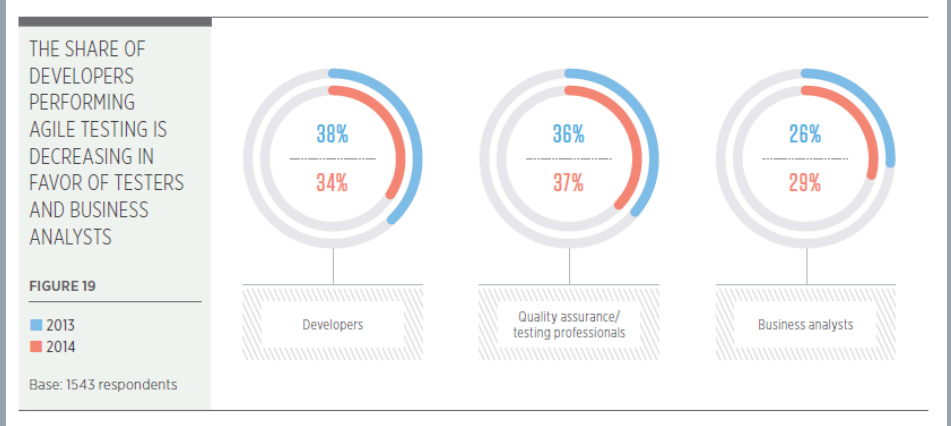
- תהליך ETL שעובד על DWH קיים ומוסיף פרוצדורות חדשות שלא היו קיימות קודם לכן לכל אחד מסוגים אלו יש להתאים אסטרטגיית בדיקות מתאימה ולוודא כי:
  - ✔ מתבצעת טרנספורמציה תקינה לנתונים.
  - ✔ כל הנתונים הועברו בהצלחה, ללא כפילויות, איבוד מידע או דריסת מידע קיים.
  - ✔ תהליך ה ETL יודע להתמודד בהצלחה עם מידע לא תקין (לדוגמא דוחה אותו או משתמש בערכי Default במקום) ומדווח על הבעיה.
  - ✔ מבנה ה DWH אכן תואם את העיצוב שהוגדר לו (סוגי עמודות תקינים, אילוצים [constraints] מתאימים, קשרים נכונים בין טבלאות וכד').
  - ✔ מסגרת הזמן בו מתבצע תהליך ה ETL היא המסגרת אשר הוגדרה עבורו.בדיקת שכבת ה ETL מערבת הבנה עסקית, הערכת זמנים מוגדרת עבור הבדיקות, תכנון Test cases והפקת דוחות המייצגים את תוצאות הבדיקה.בדיקת פעילות כלי ה **BI**.

מאחר וקיימים כלי BI שונים, אופי הבדיקה מאוד תלוי בסוג הטכנולוגיה. לדוגמה, בכלים אשר מציגים דוחות קצה למשתמש כמו ה- Reporting Services של מיקרוסופט או Business Object, אנו נתמקד בתצוגה ונוודא שכל החישובים עבור האובייקטים הגרפיים הוגדרו כראוי (פרמטרים, משתנים וכד'), שהשאילתות בהן השתמשו כדי להציג את המידע תקינות, הפורטל באמצעותו ניתן לראות את הדוחות נגיש, ההרשאות המתאימות ניתנו למשתמשים השונים, ועוד.

לסיכום

בדיקות DWH הן נושא המורכב מחלקים רבים הדורשים הבנה טכנית, עסקית ושיתוף פעולה בין גורמים שונים בתוך הארגון. קצרה ירעית המאמר מלהקיף את ארכיטקטורת ה- DWH לעומק ומלתאר את כל צורות הבדיקה הקיימות. מאמר זה נועד לספק את ההבנה הבסיסית אשר תוכל לסייע לכם לבצע בדיקות יעילות יותר בסביבת ה- DWH ולהבין מעט יותר את העולם המרתק אשר עומד מאחורי מאגרי הנתונים.

אחוז המתכנתים המבצעים בדיקות באג'ייל נמצא בירידה ביחס לבודקים ואנליסטים עסקיים

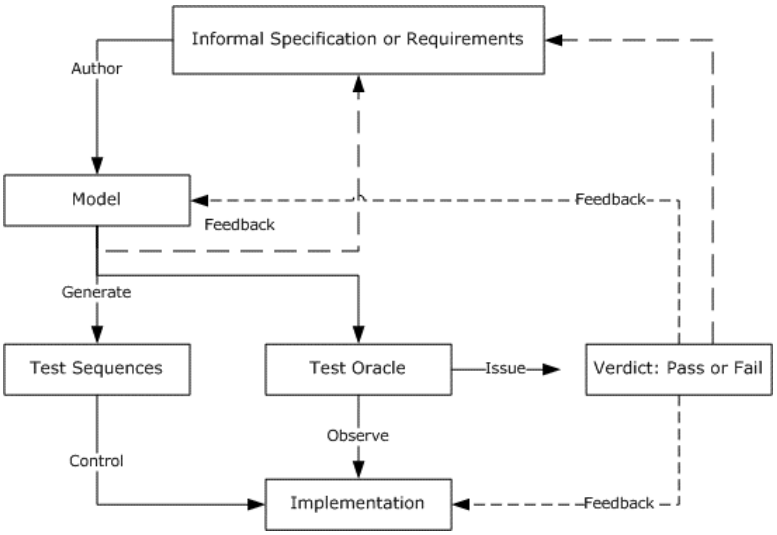


סקרים

בדיקות מבוססות מודל
MBT – Model Based Testing

אלון לינצקי, מנכ"ל Best – Testing, מייסד פורום סיגיסט ישראל, ומייסד-שותף של ITCB®

MBT הינו יישום של **Model Based Design** עבור עיצוב בדיקות פונקציונליות/מערכת והאפשרות להרצתן באופן אוטומטי. ניתן להשתמש במודלים לייצוג מערכת נבחרת לבדיקה (SUT – System Under Test) והתנהגותה, או לייצג אסטרטגיות בדיקה וסביבות בדיקה.



רצ"ב מודל סכמתי כללי של MBT (כפי שמציגים אותו במיקרוסופט – ראו לינק):

תחום ה– MBT, הינו השלב האחרון היום בהתפתחות האוטומציה לבדיקות, המגיע אחרי השלבים: בדיקות ידניות, הקלטה והשמעה (record & replay), בדיקות מונחות Scripts, בדיקות מונחות Keywords, וכעת – Model Based Testing.

למעשה משתמשים ב MBT לייצר פרוצדורות לבדיקות מערכת תוך כדי שימוש במודלים ויצירת מודל פורמלי של דרישות המערכת וההתנהגות שלה. למרות שצורת עבודה זו דורשת עבודה מקדימה רבה הרבה יותר בבניית המודל, היא מציעה יתרונות משמעותיים גבוהים יותר אל מול השיטות המסורתיות של בדיקה מסוג זה, ועל כך נדבר בהמשך.

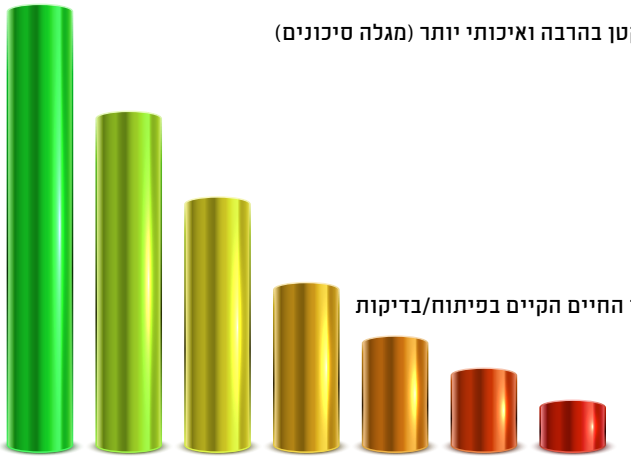
המודל בד"כ נכתב ידנית (על ידי מהנדסי בדיקות) מתוך נתונים לגבי דרישות המערכת, ו/או אפיוני המערכת. שלב יצירת המודל עצמו, עשוי לספק לנו משוב רב וטוב ביותר על המערכת, וה– ambiguity של הדרישות שתוארו במסמכי הדרישות/אפיונים, מכיוון שהתהליך של יצירת המודל מחייב את הממדלים (האדם/אנשים שמייצר/ים את המודל) לשאול שאלות שמובילות למצוא מידע חסר, מידע סותר וכו' תוך שהם מבקשים בהירות בהגדרת הדרישות.

לאחר יצירת המודל, פתירת כל הסתירות והשלמת כל המידע החסר, כלי המידול וההרצה (ישנם כלים שונים בשוק), מייצר אוטומטית test suites אשר מכילים את ה– test sequences ואת ה– test oracle (כל כלי מיישם את האלמנטים הללו אחרת ולעיתים מכיל אלמנטים אחרים).

תפקיד ה– test sequences לבקר את ה SUT, ולהוביל אותה למצבים השונים לבדיקה שזוהו תוך ניתוח המודל, על מנת לבדוק שהמערכת עומדת בהגדרות המודל (להתנהגות המערכת). תפקידו של ה– test oracle, לבחון ולנטר את התהליך של הבדיקה ולייצר חיווי של Pass או של Fail.

היתרונות המרכזיים של MBT :

- תחזוקה קלה ביותר של מערך תרחישי הבדיקה
- אוטומציה של תהליך ה– test design (חוסך בזמן, משאבים וכסף)
- איכות תרחישי בדיקה טובה יותר (כלומר: אין טעויות אנוש, מציאת מגוון קומבינציות רחב יותר למערכות מורכבות יותר)
- יכולת ביצוע אופטימיזציה לכמות תרחישי בדיקה גדולה, והמלצה על סט תרחישים קטן בהרבה ואיכותי יותר (מגלה סיכונים)



האתגרים המרכזיים של MBT:

- הצורך ברכישת כלי מידול יקר (יחסית), שעשוי להגיע לעשרות אלפי ש"ח
- הצורך בהדרכת מהנדסי בדיקות ביכולות כלי מורכב
- הצורך בלימוד שיטת עבודה שונה בהרבה מהקיים
- האתגר בשיתוף פעולה הדוק יותר עם ארכיטקטים בפיתוח/ עם צוות המוצר
- הצורך בשילוב עבודה של כלי ה– MBT עם עבודת כלי בדיקות אוטומטיות ועם מחזור החיים הקיים בפיתוח/בדיקות

כלים שונים הקיימים בשוק (ראה טבלת השוואה):

- Leirios test generator
- MaTeLo
- Qtronic
- Conformiq
- RBT
- Reactis
- Spec Explorer

למידע נוסף:

https://msdn.microsoft.com/en-us/library/ee620469.aspx

https://en.wikipedia.org/wiki/Model-based_testing

http://queue.acm.org/detail.cfm?id=2723708

http://www.methodsandtools.com/archive/archive.php?id=102

מה עשית היום בגן ילד קטן שלי?!

מאיה בוכניק

מאיה בוכניק –
יועצת חיפוש עבודה וקריירה.

בעלת הבלוג Career Tips 4 Geeks שבו תמצאו מאות טיפים לתכנון קריירה ולחיפוש עבודה, שימוש נכון בלינקדאין וברשתות חברתיות, כתיבת קורות חיים, הצלחה בראיונות והכל מזווית מרעננת ומשעשעת (עד כמה שאפשר).

השאלה **"איפה את/ה רואה את עצמך בעוד 5 שנים?"**– צצה מידי פעם בראיונות עבודה ולא בכדי, כי כל מעסיק מאוד מעונין לדעת מהן התוכניות שלכם לעתיד ואיך התפקיד אצלו משתלב בהן.
בראיון עבודה זו שאלה שאתם חייבים לענות לה – פשוט בגלל ששאלו אתכם, אבל מה שהכי חשוב מבחינתי הוא שתדעו לענות לה קודם כל עבור עצמכם. יתרה מזאת... זו שאלה שאתם חייבים לשאול את עצמיכם בכל פעם מחדש, אחת לתקופה: "ועכשיו מה?", "מה הצעד הבא שלי?", "לאן אני מתקדם מכאן?!".

אל תבינו אותי לא נכון, הכוונה איננה להכניס אתכם למשטר של לחץ, אבל כן לעשות בחירות מתוך חשיבה אמיתית ועמוקה ותכנון ולא מתוך זרימה.
ועוד דבר– התשובה שאתם נותנים לעצמכם יכולה להשתנות עם הזמן. היא לא חייבת להיות אותה התשובה מאז סיום הלימודים, או מלפני 10 שנים. החיים דינאמיים והמציאות משתנה, כך שמותר שגם התשובה תשתנה בהתאם. השורה התחתונה היחידה כאן היא שחשוב שפשוט תהיה לכם אחת. הכוונה לתשובה, כמובן.

ברגע שתהיה לכם תשובה כזאת, יהיה לכם גם הרבה יותר קל לחפש עבודה– הרי כבר ידוע מזמן שכשהיעד ברור, הדרך מתקצרת. כשאתם מחפשים עבודה מסיבות של מיצוי, שחיקה או חיפוש אתגר חדש (ולא מחוסר ברירה כי לפני חודשיים פוטרתם ואתם חייבים לחזור לשוק העבודה) – חשוב לבחור בהצעה הטובה ביותר לקריירה שלכם. זכרו: בתכנון נכון של קריירה, אומרים "I DO" רק לתפקידים שמקדמים אתכם צעד אחד נוסף לעבר היעד.

> **אנו חייבים לשאול את עצמנו אחת לתקופה:**
> **"ועכשיו מה?", "מה הצעד הבא שלי?",**
> **"לאן אני מתקדם מכאן?!".**

אבל כל זה הוא רק הקדמה לטיפ האמיתי שלי בפוסט הזה, כי לחפש עבודה אנחנו עושים אמנם מספר פעמים בחיים (מי יותר ומי פחות...) אבל לעבודה אנחנו הולכים כל יום. וקריירה ניתן ורצוי לקדם בכל יום שגרתי של עבודה, בתוך מקום העבודה.

ובאמת, רוב העובדים עושים את הקפיצות הגדולות בקריירות שלהם לא דרך דילוגים ממקום עבודה אחד לשני, אלא דרך מעברים בין תפקידים בתוך אותו ארגון. קל יותר למעסיקים לקדם עובד פנימי לתפקיד בכיר יותר (גם אם אין לו בדיוק את הניסיון הקודם הנדרש) מאשר לגייס עובד מבחוץ לאותו התפקיד וללא הניסיון הקודם הנדרש. אמנם בשני המקרים מדובר בעובד שנכנס לתפקיד חדש עם פוטנציאל אבל ללא הניסיון הספציפי, אבל את ההזדמנות לקידום הזה מקבלים בדרך כלל כשכבר מוכרים לארגון והפוטנציאל ידוע ובטוח ולא כשמנסים לעשות מעבר ממקום עבודה אחד לשני.

אז לכל אלה שביניכם שיש להם יעד ברור ושחשוב להם לקדם את הקריירה, הנה 4 דברים שאתם חייבים לעשות תוך כדי עבודה ומאוד מאוד רצוי שבכל יום. כמו בכותרת שכאן למעלה, שאלו את עצמכם "מה עשינו היום בגן"? = מה עשינו היום בכדי לדחוף את עצמנו קדימה?

> **שאלו את עצמכם "מה עשינו היום בגן"? = מה**
> **עשינו היום בכדי לדחוף את עצמנו קדימה?**

1. טיפוח של ידע מקצועי – זוכרים את האמירה "מה שלא הולך קדימה, הולך אחורה"? זה בדיוק זה. בכל תחום שדורש ידע או מומחיות מקצועיים, אתם כבעלי מקצוע חייבים להישאר כל הזמן עם היד על הדופק. אתם חייבים כל הזמן להמשיך ללמוד, לקרוא, לאסוף מידע, לדעת מה קורה בשוק, מה חם ולאיזה כיוון פונות המגמות החדשות. אם אתם מתחילים להרגיש שהידע שלכם הולך ומתיישן– זה הזמן לרענן אותו. זיכרו! לישן ולעתיק יש ערך רב, אבל רק במקום אחד– בשוק הפשפשים... ללמוד ולהתעדכן בחומר מקצועי אפשר, אגב, תוך כדי עבודה. אז למה באמת שלא תהיו אתם שגרירי החדשנות אצלכם בארגון?? :)

2. מופע מעולה ללא קהל לא שווה הרבה... – ובתרגום חופשי: הצלחה מדהימה שאף אחד לא יודע עליה, כמוה ככישלון. אני קצת מקצינה כמובן בכדי להעביר את המסר... אל דאגה– אני לא מנסה להפוך אתכם לחנפנים ולגאוותנים הכי גדולים בארגון, אבל הנקודה היא פשוטה: אם אתם עושים משהו מדהים תוך כדי עבודה, דאגו לכך שמי שצריך לדעת על זה– ידע. צניעות וענווה הן תכונות יפות בהחלט– תראו כמה רחובות קרואים על שמו של רבי עקיבא...– מצד שני, אם אף אחד לא היה מספר לנו עד כמה צנוע ועניו הוא היה– לא היינו שומעים על זה מעולם וגם לא היינו לומדים עליו בתושב"ע...

3. יצירה ועיבוי של קשרים מקצועיים – תוך כדי עבודה מתחככים ונתקלים בהרבה מאוד אנשים. לימדו לשמור על הקשרים האלה (על כל אחד ואחד מהם), עזרו לקולגות כשהם מבקשים טובה וכשאתם יכולים לסייע. מי יודע, אולי הם יוכלו בתורם לסייע לכם בפעם הבאה שתחפשו עבודה, עובד חדש לצוות, קשר עסקי וכו'...

4. דעו מה ערך השוק שלכם בכל רגע נתון – זה באמת לא יפה להעריך בני אדם כמו מוצרים בעלי ערך שוק, אבל, מה לעשות החיים קשים ות'כלס זה ככה וזה מה יש! לכל אחד מאיתנו יש שווי שנגזר ממצב הביקוש וההיצע בענף שלו בשוק. להיות בת יענה שטומנת את ראשה בחול זה אולי קל יותר, אבל החכמים מבינינו ידאגו כל הזמן לרחרח ולגשש, לקרוא את מפת הביקוש וההיצע, לזהות ירידות ערך ולעשות צעדים ולפעמים גם שינויי כיוון בהתאם לשוק.

בכל יום שגרתי בעבודה ניתן לבצע לפחות אחת מהנקודות שכאן למעלה. יום שבו ישאלו אתכם "מה עשית היום בגן, ילד קטן שלי?" ושתענו בו "כלום"– הוא יום אבוד לקריירה שלכם. זיכרו לעשות בכל יום דבר אחד קטן שיתרום לכם בעתיד בגדול!

> ***נצלו את הזמן הפנוי בין לבין לחשיבה – איפה**
> **אתם רוצים לראות את עצמכם בעוד 5 שנים,**
> **או עיזבו – פשוט יותר – איפה אתם רוצים לראות את**
> **עצמכם בקרוב??**

פתרון השאלה

שאלה זו מתייחסת לפרק 4 בסילבוס שמדבר על טכניקות לעיצוב בדיקות. טכניקות אלה מסייעות לבודק להגדיר את האסטרטגיה שתשמש לבחירת הקלט במקרי הבדיקה (test cases) השונים ולניתוח תוצאות הבדיקה. הסילבוס לרמה הבסיסית מתאר ארבע טכניקות בדיקה, בעוד שהסילבוס לרמת המתקדמת מתאר שתי טכניקות נוספות. כמובן, שישנן עוד טכניקות בהן ניתן להשתמש בהתאם לצרכי הבדיקה כאשר חשוב לזכור שהמטרה היא לבחור את מקרי הבדיקה המעניינים, כאלה שיהיו עם הסיכוי הטוב ביותר למצוא תקלות.

השאלה מציגה שתי טכניקות עליהן נעבור בקצרה:

הראשונה היא טכניקה הנקראת מחלקות שקילות (equivalence partitioning). בטכניקה זו מחלקים את הקלט האפשרי לקבוצות לפי הדמיון הצפוי בהתנהגות המערכת לערכי הקלט מאותה קבוצה. מחלקות שקילות קיימות עבור נתונים תקפים (valid) ונתונים לא תקפים (invalid). מחלקות שקילות יכולות להיות רצף של מספרים (למשל, כל תחום המספרים שמערכת יכולה לקבל) או בעלת ערכים בדידים (למשל, זכר/נקבה בטופס המבקש לקבל את מין המשתמש), כאשר כל תנאי מגדיר גבול בין מחלקות שקילות.

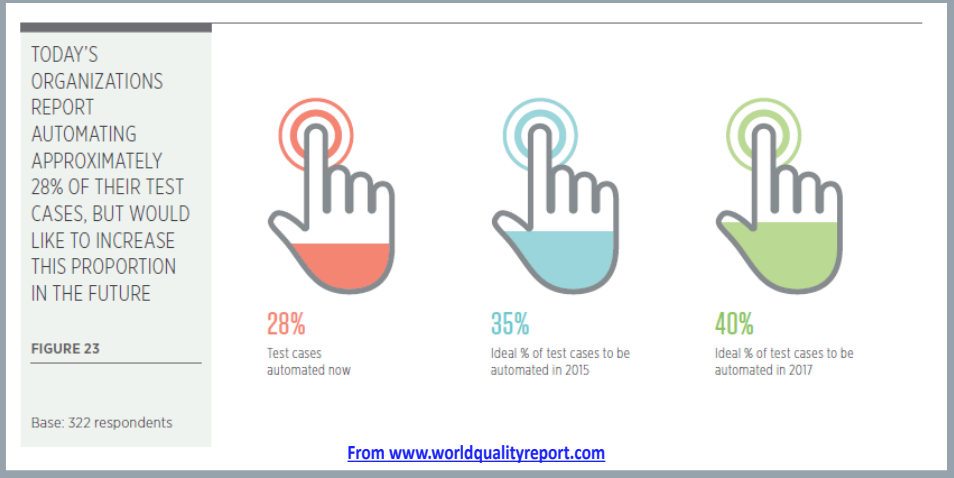
הטכניקה **השנייה** היא בדיקות טבלת החלטה (decision table). טבלת החלטה מתארת כללים עסקיים אשר המערכת נדרשת ליישם. תנאי הפלט והקלט מתוארים בדרך כלל באופן בוליאני של נכון (true) או לא נכון (false). לטבלה יש שני חלקים: החלק הראשון מציג את התנאים (conditions) המחוללים תוצאה מסוימת והחלק השני מציג את הפעולות (actions) הצפויות להתרחש מצירוף תנאים. כל טור בטבלה מציג כלל עסקי שהוא צירוף של תנאים.

בשאלה הזו הטכניקה של מחלקות שקילות מיוצגת בצורה פשוטה – לכל תנאי קלט יש שתי מחלקות, אחת תקפה (כן) ואחת לא תקפה (לא). טבלת ההחלטות מתארת 3 מצבים אפשריים ובמקביל נאמר שהמצב הרביעי לא יכול להתקיים (2 תנאים שלכל אחד מהם 2 אפשרויות יתנו 4 מצבים).

נאמר שהמערכת מקבלת 4 סוגים של כרטיסי אשראי ולכל אחד מהכרטיסים הבודק צריך לבצע את כל השילובים שבטבלה.

מאחר שצריך לבדוק שלושה מקרי בדיקה לכל סוג כרטיס, נזדקק בסך הכל לבצע 12 בדיקות שונות. ולכן התשובה הנכונה היא תשובה ג'.

הארגונים מדווחים על אוטומציה של כ 28% ממקרי הבדיקה ומעוניינים להגדיל נתח זה בעתיד.



סקרים

התוכנית לקידום המחקר האקדמי בתחום איכות ובדיקות תוכנה

ITCB® הינה עמותה מקצועית, שקמה בשנת 2004 כחלק מהפעילות העולמית של ארגון ISTQB®, כדי לקדם את כל פעילות בדיקות התוכנה בישראל. החברים בארגון הינם אנשי מקצוע בתחום הבדיקות, שפועלים במסגרת הארגון בהתנדבות ושלא למטרות רווח.
חברי הארגון הישראלי לוקחים חלק פעיל ומשפיע בוועדות ההיגוי והעשייה השונות בארגון הבינלאומי.
מתוך כוונה להשפיע ולעודד שיתוף פעולה בין האקדמיה לתעשייה בתחום חשוב זה, הארגון יוצא בקול קורא לעידוד מחקרים אקדמיים בתחום בדיקות תוכנה ואיכותה.
לאחרונה אנו חשים התעוררות בקרב הקהילה האקדמית והגברת המודעות לענף איכות ובדיקות התוכנה. ארגון ITCB® מברך על כיוון חיובי זה ומעודד מעורבות נוספת ואינטראקציה בין התעשייה ובין האקדמיה בכל הקשור לאיכות ובדיקות תוכנה.
על מנת לעודד ולהחיש את המחקר האקדמי בנושאים הקשורים לכך החליט ITCB® לתמוך בתהליכי המחקר באמצעות הענקת מלגת לימוד שנתית למחקרים אקדמיים בתחום.
המלגה תבטא את הערכת הקהילה המקצועית לתרומתו הסגולית ולרמתו של המחקר שיתבצע. המלגות יתנו בשלוש רמות של מחקרים:

- מלגה בסה"כ 3000₪ לעבודת מחקר במסגרת תואר ראשון (פרויקט גמר)
- מלגה בסה"כ 5000₪ לעבודת מחקר ברמת תואר שני (Master)
- מלגה בסה"כ 10000₪ לעבודת מחקר ברמת תואר שלישי (PHD)

ייבחנו עבודות מחקר שייעשו במהלך שנות 2015 - 2016 ואשר יגיעו לידי סיום לפני סוף שנת 2016

תחומי מחקר (רשימה חלקית כל תוספת תיבחן)

- מדידה ובקרה של תהליכי איכות בתוכנה
- מיכון תהליכי בדיקה
- שילוב תהליכי איכות בפרויקט תוכנה
- מחקרים אמפיריים בתעשייה בתחום בדיקות התוכנה
- שיטות לניתוח קוד ואיכותו
- בדיקות התוכנה במציאות הטכנולוגית החדשה
- ALM ומחזור האיכות של התוכנה
- בדיקות תוכנה וניהול סיכונים בארגון המפתח
- בדיקות בעולם המוביל
- בדיקות משולבות תוכנה/חומרה
- בדיקות שירותי תוכנה בענן
- SOA ובדיקות תוכנה
- כלים ותשתיות לתהליכי איכות ובדיקות תוכנה
- תהליכי שיפור לבדיקות התוכנה
- שיטות לימוד והקניית ידע בנושא איכות ובדיקות

תנאי ההגשה

- עבודת המחקר צריכה להיות עבודה מקורית המתבצעת במסגרת מחקר אקדמי בליווי ופיקוח של איש סגל אקדמי מטעם המחלקה בה למד הסטודנט
- תינתן עדיפות למחקרים המתבצעים בשיתוף עם התעשייה
- המחקר מתקיים ומסתיים באותן שנים בהם מוענקת המלגה
- תקציר להצעת מחקר הכולל את תיאור נושא המחקר, תהליך העבודה עיקרי לביצועו ושמות המנחים האקדמיים יש להגיש לוועדה המקצועית אשר תוקם מטעם הארגון, לפני חודש אוקטובר 2015 (לשנה הקרובה).
- אין הארגון מתחייב להעניק מלגה כלשהי למחקר באם לא יוגשו הצעות ברמה אקדמית ומקצועית מספקת.

תאריך אחרון להגשת בקשות למלגה: 18.11.2015
תהליך בחינת הזכאות למלגה יתבצע באמצעות צוות המורכב מפרופ' מכובדים מהמוסדות האקדמיים בארץ וממומחים מהתעשיה.
הענקת המלגה תתבצע תחת פיקוח ובקרת הגופים המשפטיים המלווים את ITCB®.

ליצירת קשר והגשת העבודות – דני אלמוג 050-6316423 dani.almog@itcb.org.il
ובאתר ITCB® – www.itcb.org.il

ITCB® Israeli Testing Certification Board
ISTQB® International Software Testing Qualifications Board

צוות הכותבים

יאן ברון – M.Sc מדעי המחשב
הסמכת ISTQB® FL מארגוני הסמכה אמריקאי וישראלי
35 שנות ניסיון בהנדסת תוכנה וניהול בדיקות.
תפקיד נוכחי: מנהל בדיקות, **iMDsoft** ישראל
תחומי עניין – מתודולוגיה וכלי פיתוח לאזורים שונים בבדיקות. בדיקות אוטומציה, ניהול מחזור חיי מוצר: מתודולוגיה וכלים.
התנדבות – **ISTQB®**, ראש ועדת סקרים. **ITCB®**, ועדה המנהלים. **SIGIST** ישראל, יו"ר תכנית הכנסים.

קובי הלפרין – עוסק בבדיקות מזה מעל ל-20 שנה בעיקר בתחום התקשורת טלקום/דאטהקום (ECI, Alvarion, Axerra, Ceragon), בעל מעורבות גבוהה בקהילות בעולם ובארץ ומוביל מספר פורומים אינטרנטיים, מחפש להרחיב את הדיון בתחום הבדיקות, שיפור כלים והרחבת הידע של הבודקים.
פעיל בוועד המייעץ של ITCB® – בעיקר בשיפור האתר www.itcb.org.il והקשרים ברשתות החברתיות.

אייל זילברמן – מומחה בדיקות בכיר ונשיא קבוצת קווליטסט, חברת הבדיקות המובילה בישראל והשנייה בגודלה בעולם. פעיל בתחום בדיקות התוכנה משנת 1995, עבד כבודק תוכנה, מנהל ומוביל בדיקות במספר ארגונים כמו בנק ירושלים, Teleknowledge, T-Mobile ועוד. פרסם עשרות מאמרים מקצועיים ונואם פעיל בכנסים ישראליים ובינלאומיים. אייל הוא בין המקימים של "חושבים בדיקות" מגזין בדיקות התוכנה הישראלי הראשון וחבר בצוות ה-AB של ITCB® – ארגון ההסמכה לבדיקות הישראלי.
לינק: www.qualitestgroup.com

אלון לינצקי – בשלושים השנים האחרונות התמקצע, הדריך ואימן קבוצות וחברות בפיתוח, בדיקות והבטחת איכות.
תחומי המומחיות העיקריים של אלון הינם: עיצוב בדיקות, תכנון ואופטימיזציה של בדיקות, ניהול יעיל של בדיקות, שיפור ואופטימיזציה של תהליכי בדיקות ופיתוח, בדיקות אוטומטיות בפרוייקטים, ניהול בדיקות מבוסס סיכונים, מדדים ומטריקות לניהול איכות ובדיקות, בניית צוותים יעילים, שיפור תהליכי פיתוח ואיכות, אג'ייל ומעבר לאג'ייל.
אלון מרצה בינ"ל פופולארי מאז 1995, מוביל תוכנית שותפים של ISTQB® Partner Program,
מייסד The Israeli Testing Forum, SIGiST Israel – מייסד שותף של Israel Testing Certification Board – ®ITCB. היה אחד מכותבי הסילבוס של ISTQB® Agile Tester – Foundation Level Extension, סגן נשיא ודירקטור שיווק של ®ITCB.

טל פאר – בעל ניסיון של כ-20 שנים כבודק ומנהל בדיקות במגוון חברות וטכנולוגיות במודלי פיתוח שונים, וכמו כן משמש גם כיועץ ומדריך בדיקות. כיום טל מנהל בדיקות בחברת Kongsberg Maritime.
טל חבר ב-® ITCB ומוביל את קבוצת התהליכים ב-®ISTQB.

מיכאל שטאל – הוא ארכיטקט בדיקות תוכנה באינטל, ישראל. במשך 15 השנים האחרונות מיכאל בדק בתחום מודם כבלים, Wi-Fi, טלוויזיות חכמות, כרטיסים גרפיים, ולאחרונה הוא עובד בקבוצה שבודקת את התוכנה שמאחורי טכנולוגיית RealSense (מצלמות תלת-מימד) של אינטל.
מיכאל חבר ב – ITCB ומוביל את פעילות ה – Advisory Board.
במסגרת תפקידו, מיכאל מגדיר שיטות בדיקה ומתודולוגיות עבודה, עוסק הרבה בהדרכה ולפעמים אפילו מרשים לו לבדוק משהו (שזה הכי כייף).
מיכאל מציג תכופות בכנסים בארץ ובחו"ל והציג בין היתר ב – SIGiST Israel, STAR conferences, QA&Test ובכנסים אחרים. מיכאל מלמד בדיקות תוכנה בפקולטה למדעי המחשב באוניברסיטה העברית. ניתן לראות חלק מהמצגות והמאמרים שלו באתר www.testprincipia.com.