רבעון שלישי 2016 | גיליון מס'06

מגזין עולם הבדיקות

WWW.TESTINGWORLD.CO.IL

דבר העורכת | איריס קוטליצקי

ברוכים הבאים לגיליון השישי של מגזין עולם הבדיקות

הרצון שלנו להרחבת הידע והוספת כלים לבדיקות ידניות ואוטומציה הינו צורך אמיתי יום יומי, אנו רואים זאת בסקר האחרון בו הוזמנתם להצביע "האם קיום מפגשים בין בודקים הינו דבר חשוב?".
74% הצביעו בעד קיום מפגשים בין בודקים, את התוצאות אתם יכולים לראות בעמוד 8 וגם להצביע לשאלה הבאה.

בימים אלו מתחילה ההרשמה לתחרות הפרס למצוינות ישראלית בבדיקות תוכנה 2016,
אם לקחתם חלק בפרויקט חדשני ויצירתי הנכם מוזמנים להירשם, את כל הפרטים תוכלו למצוא באתר של ITCB.

הקיץ כבר בעיצומו זו תקופה מאתגרת של שילוב בין המשך עבודה עם כוח אדם מוגבל, תוך כדי חיפוש הפעלות לילדים וחופשות משפחתיות... - חופשה מהנה.
הגיליון הבא ייצא במהלך החגים - אז נקדים ונאחל לכולכם שנה טובה ומתוקה...

ומה יש לנו בגיליון?
תוצאות ומסקנות ההשוואה בין כלי אוטומציה למובייל (HPE Mobile Center, Appium ו eggPlant Functional) שבוצעו ע"י איל יובל, המשך המאמר אודות בדיקות חקירה (ET) במערכות ללא ממשק גרפי מאת שמואל גרשון וכלים לעבודה עם צוותים במדינות שונות מאת ענבר מאיר.
אנו מתחילים בסדרות חדשות של מאמרים: איך לעבוד עם בדיקות רשת (web) אוטומטיות מאת דייב הפנר ואודות בדיקות עומסים מאת נחום דימר וכמובן הטורים הקבועים אשר מכילים ידע רב.
המאמרים עוברים עריכה ע"י צוות עורכים מדהים השואף לרמת דיוק בתוכן ובשפה עמם אני מאוד נהנית לעבוד ולומדת מהם רבות. אני רוצה להודות לכל העורכים המלווים אותנו מהגיליון הראשון: לעמית ורטהיימר, טל פאר, משה ממיה, אסתר צבר, איסי חזן ומיכאל שטאל ולעורכים שהצטרפו אלינו לא מכבר דורון בר, אפרת ויינברג, רון פרץ ונטלי מהרבן.
בזכות רוח ההתנדבות המדהימה שלכם ושל כותבי המאמרים כל קהילת הבודקים נהנית ממגזין זה.
קריאה נעימה
איריס

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

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 המהדורה הבאה ואנו 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, *RSA, The security division of EMC*
משה מאמיה, *HP Software*
אסתר צבר, *AQA*
אפרת וינברג, מכללת אורט סינגאלובסקי
דורון בר, ***AVG***
נטלי מהרבן, יועצת *Truelogic*
רון פרץ, *Abbott Informatics*

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

השוואת כלי אוטומציה לבדיקות על מכשירי מובייל | איל יובל 4
תוצאות ההצבעה לגיליון #5 9
ראיון עם מנהל בדיקות | שגיא וכמן 10
בחן את עצמך | טל פאר 12
האנציקלופדיה לבדיקות | אייל זילברמן 13
דגשים והמלצות בעבודה עם צוותים גלובליים – ענבר מאיר 15
חדשות מעולם הבדיקות | קובי הלפרין 18
בדיקות מן העולם | אלון לינצקי 19
בדיקות חוקרות (ET) במערכות ללא ממשק גרפי | שמואל גרשון 20
מחפש צרות | מיכאל שטאל 23
איך להתחיל לעבוד עם בדיקות רשת (web) אוטומטיות
דייב הפנר 24
פינת הטיפים | קובי הלפרין 27
מבוא לבדיקות עומסים | נחום דימר 28
בחן את עצמך – תשובה | טל פאר 30

עולם הבדיקות נכתב ע"י בודקים לבודקים

ITCB® מקדמים את קהילת הבודקים בישראל

מגזין עולם הבדיקות

עולם הבדיקות הינו מגזין רבעוני.
כל הזכויות שמורות. זכויות היוצרים על חומר שפורסם על-ידי המפרסם הינו רכושו של המחבר. הדעות המובאות במאמרים והתוכן לא בהכרח מביעות את דעת המפרסם. המחברים הינם האחראים הבלעדיים על תוכן מאמרם. אין לפרסם ו/או לשכפל בכל צורה שהיא את התוכן ללא אישור.
לקבלת אשור לשימוש בתוכן צור קשר במייל
info.testingworld@gmail.com

השוואת כלי אוטומציה לבדיקות על מכשירי מובייל | איל יובל

איל יובל

שמי איל יובל, אני בן 44, נשוי לרינת ואב לאורי, נהוראי ומאיה (הבטחתי להם שהשמות שלהם יופיעו...)
לפני כשלוש שנים עברתי מטעם חברת קוואליטסט לסניף של החברה באנגליה ומאז אני מתגורר בלונדון.
בארץ עבדתי כשש שנים בחברת מרקורי אינטראקטיב (כיום HP) בתפקדי פיתוח ובתמיכה הטכנית. לאחר מכן עברתי לקוואליטסט ישראל וניהלתי פרויקטי אוטומציה רבים בחברות כגון רפאל, אלביט, Objet ו GE Healthcare בעיקר בשימוש עם HP QTP / UFT בסביבות windows.
באנגליה אני מטמיע ומנהל פרויקטי אוטומציה ועומסים בחברות כגון: NetworkRail, giffgaff, Camelot, Bank Santander ו-TUI Group
בשימוש עם סלניום, UFT, eggPlant Performance, eggPlant Functional ו-Neotys NeoLoad.

הקדמה

לפני מספר חודשים קראתי דוח של חברת המחקר פורסטר (Forrester Research) בנושא Automated Testing Improves App Quality. בדוח יש סקירה של לא פחות מ-17 כלי אוטומציה מובילים לבדיקות על מכשירי מובייל שקיימים בשוק.
רשימה חלקית מהדוח:
Appium ,Xamarin Test Cloud ,TestPlant Functional ,Soasta TouchTest ,Perfecto Mobile Continuous Quality Lab ,HP Mobile Center , Experitest ,SeeTest Automation.
למרות שיש כל כך הרבה כלי אוטומציה לבדיקות על מכשירי מובייל, סביר שרובנו מכירים ועובדים עם כלי אוטומציה אחד מתוך הרשימה כי "זה מה שנמצא ובשימוש בארגון".

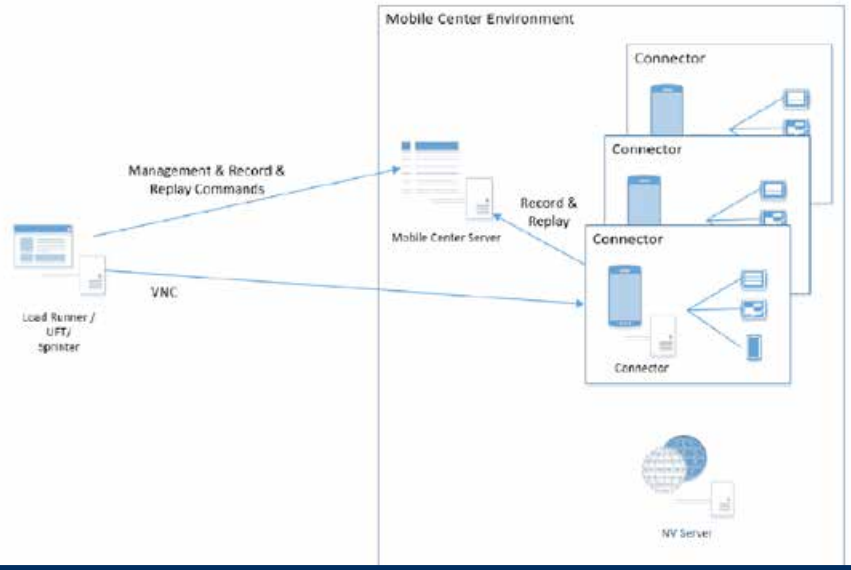
> **המאמר בא לתת רקע על מספר כלי אוטומציה לבדיקות על מכשירי מובייל הקיימים בשוק**

לאחר שסיימתי הטמעת אוטומציה לבדיקת אפליקציה על מספר מכשירי מובייל באחד הבנקים הגדולים באנגליה תוך שימוש ב-eggPlant Functional (בקיצור ePF) של חברת testPlant החלטתי לבחור באחד מהתרחישים שהטמעתי ולפתח את אותו תרחיש בעזרת כלי אוטומציה שונים על מספר מכשירי מובייל וכך לבצע השוואה בין הכלים.
חשוב לציין שההשוואה הינה סובייקטיבית ומתבססת על הניסיון והחוויה האישית שלי בתהליך.
כלי האוטומציה שבחרתי להשוואה הינם HPE Mobile Center, Appium ו eggPlant Functional (ePF). הסיבות שבחרתי בכלים אלו הינן על סמך ניסיון האוטומציה שלי – פיתוח תרחיש ב-Appium דומה לפיתוח בסלניום ו-HPE Mobile Center הינו Add-in מעל UFT שגם בו יש לי ניסיון רב. כמו כן מבחינת רישיונות, Appium הינו חינמי ולגבי כלי HP ו-ePF יש באפשרותי לקבל רישיונות זמניים כשותף.
את התרחישים פיתחתי והרצתי על מכשירי המובייל Nexus5 ו-Samsung S6 (פיתוח עבור iOS דרש מחשב Mac שלא היה ברשותי).
המאמר בא לתת רקע על מספר כלי אוטומציה לבדיקות על מכשירי מובייל הקיימים בשוק ולא בא לדון איזה כלי אוטומציה עדיף. אין כלי רע כפי שאין כלי מושלם. לכל אחד מהכלים יש את היתרונות והחסרונות שלו וכחלק מהתהליך הטמעת בדיקות אוטומטיות בארגון מומלץ וחשוב לבצע הוכחת היתכנות (Proof of Concept) ע"י אנשי הבדיקות מתוך הארגון או בעזרת חברות יעוץ חיצוניות לבחירת הכלי המתאים לארגון.

מידע על כלי האוטומציה שנבדקו
HPE Mobile Center (MC)

פתרון יחסית חדש ומבטיח של HP לתחום בדיקות המובייל. הפתרון הינו לכלל כלי HPE בתחום הבדיקות: LoadRunner, UFT ו-Sprinter.

ארכיטקטורת HPE Mobile Center

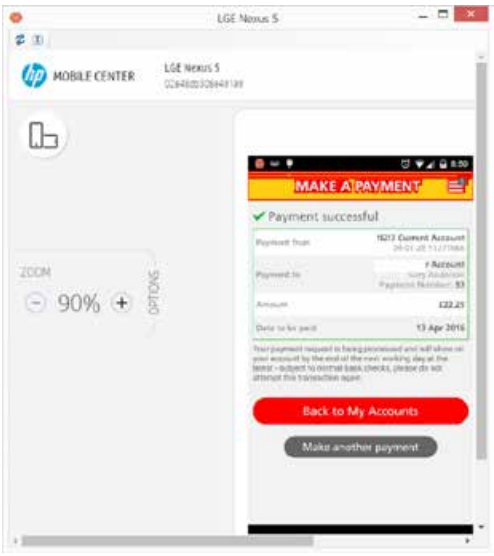


השוואת כלי אוטומציה לבדיקות על מכשירי מובייל | איל יובל

יש צורך להתקין **Mobile Center Server** על מערכת הפעלה **Linux** או **Windows**, אני התקנתי את השרת ללא בעיות על מחשב שבו מערכת הפעלה **Windows 8.1** שבו גם התקנתי את **UFT**.

זיהוי מכשירי המובייל, ניהול האפליקציות לבדיקה וניהול משתמשים נעשה בשרת. החיבור לשרת מתבצע דרך דפדפן כדוגמת **IE**, **Firefox** או **Chrome**.
אופן העבודה עם המובייל ועם האפליקציה הנבדקת דומה לאופן העבודה עם כל **UFT Add-ins** אחר: בפתיחת **UFT** מסמנים את **Mobile Add-in** ומתוך תפריט **UFT** מתחברים לשרת **MC** ובוחרים את מכשיר המובייל והאפליקציה לבדיקה.

משלב זה אפשר להקליט את התרחיש, להוסיף אובייקטים ישירות ל**Object repository**, להשתמש ב-**spy** וכמובן להריץ את התרחיש על מכשיר המובייל האמיתי ולקבל דוח ריצה מפורט.
כל הפעולות על מכשיר המובייל מתבצעות מול השתקפות של המכשיר אצלנו במחשב בתוך חלון של **HPE Mobile Center**



מכיוון שהפתרון יחסית חדש, חסר ב-**Add-in** פונקציונליות הקיימת ב-**Add-ins** האחרים של **UFT** כדוגמת עבודה עם טבלאות (אין אובייקט **MobileTable**) ולכן יש עדכונים רבים ל-**Add-in** כגון **Device.ClickText** לתמיכה ב- **Text Recognition** שנוסף רק לאחרונה לגרסת **UFT 12.52**.
קישור מומלץ על [HPE Mobile Center]

(ePF) testPlant eggPlant Functional

זהו כלי מוכוון תמונה (**image based testing tool**) ומוכוון זיהוי טקסט (**OCR**) ולפיכך בשימוש בכלי אין תלות במערכת ההפעלה ואין תלות בטכנולוגיה. התקנת **ePF** הינה פשוטה ועל המערכת הנבדקת יש צורך רק בהתקנת דמוי **vnc server**. הכלי נועד ומכוון קודם כל לבודקים ולכן הדגש הוא על שימוש בשפת פיתוח פשוטה בשם **SenseTalk** אך זהו גם אחד מחסרונות הכלי (הצורך בלימוד שפה חדשה בשימוש רק עבור פיתוח בכלי).

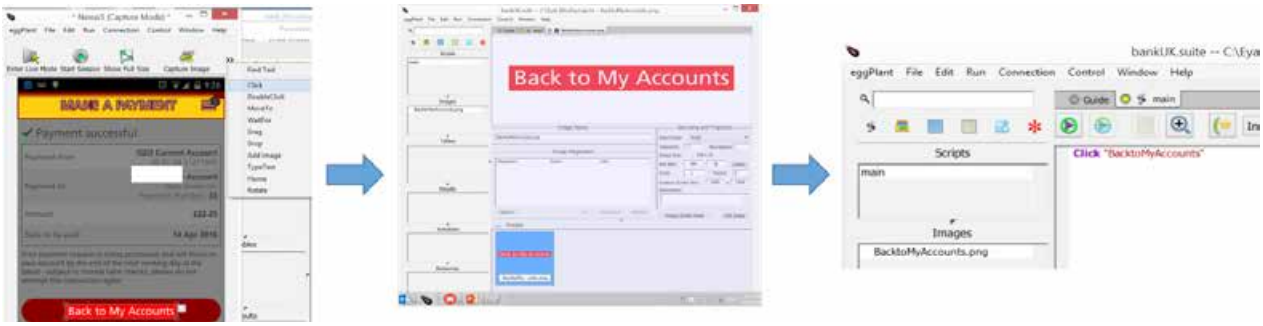
ארכיטקטורת ePF

ePF מותקן על מחשב אחד, במחשב זה מפתחים ומריצים את הבדיקות האוטומטיות. כל סט הבדיקות וסט התמונות נשמרים במחשב זה.
כמו כן **ePF** מנהל את ההתחברות למכשירים השונים ואת הרצת הבדיקות עליה.
במכשירים השונים עליהם מותקנת המערכת הנבדקת יש צורך להתקין **agent** שמבצע את ההתקשרות ל **ePF** (סוג ה-**agent** הספציפי משתנה לפי מערכת ההפעלה של המכשיר הנבדק).

אופן פיתוח בדיקה בכלי הינו תהליך פשוט יחסית – יש לסמן את החלק במכשיר המובייל שעליו רוצים לבצע פעולה מסוימת (בדוגמה למטה סימנתי את כפתור **Back to My Accounts**) ולבחור את הפעולה שיש לבצע (בדוגמה ביצעתי **click**, לחיצה על הכפתור).

השוואת כלי אוטומציה לבדיקות על מכשירי מובייל | איל יובל

ePF מבצע שלוש פעולות: התמונה נשמרת וניתן לראות אותה בחלק של ה-images בכלי, הפעולה שביצענו מתוספת לתרחיש הבדיקה (click על הכפתור) והפעולה שביקשנו מתבצעת (ביצוע לחיצה על הכפתור) על מנת שנוכל להמשיך ולכתוב את צעד הבדיקה הבא:



קישור מומלץ ePF
מאמר אודות ePF מגיליון קודם

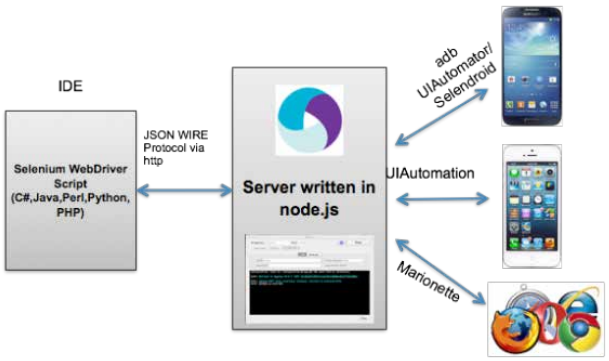
מסקר קצר שערכתי בפורם בדיקות ואיכות תוכנה בפיסבוק רוב המשתתפים ענו APPIUM לשאלה "איזה כלי אוטומציה משתמשים אצלכם בארגון לבדיקות אוטומטיות על מובייל ?".

Appium

מסקר קצר שערכתי בפורם בדיקות ואיכות תוכנה בפייסבוק (דרך אגב פורום מומלץ ביותר לכל בודק) רוב המשתתפים ענו Appium לשאלה "איזה כלי אוטומציה משתמשים אצלכם בארגון לבדיקות אוטומטיות על מובייל?".

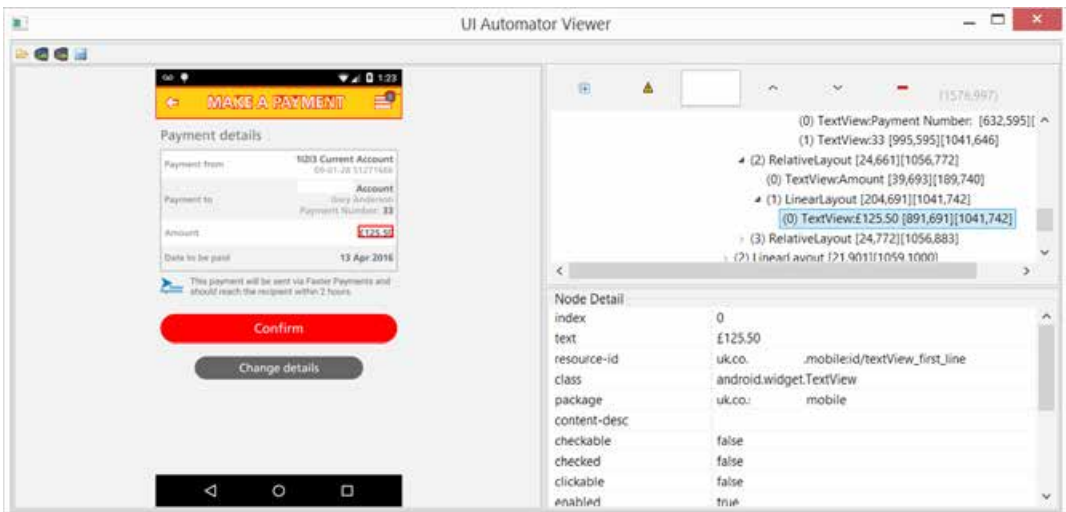
אין ספק ש Appium (המשתמש באותה טכנולוגיה כמו ה Selenium) הינה הכלי הפופולרי בתחום המובייל.
כיום Selenium WebDriver אינו רק כלי אוטומציה לבדיקות web אלא משמש גם כ API שמעליו יש מעטפת של כלי אוטומציה כדוגמת HP LeanFT או SmartBearTestComplete שמוסיפים על השימוש ב-Selenium גם את היכולות שלהם כגון object repository ו-reports משופרים.
תהליך דומה יתכן ויקרה בשנים הקרובות גם עם Appium. כבר היום יש כלים כדוגמת Perfecto Mobile שמשלבים בפתרון שלהם בין יכולות Appium לחוזקות של Perfecto בתחום בדיקות המובייל (למשל שימוש ב Wind Tunnel לדמות חווית משתמש).

ארכיטקטורת Appium



ארכיטקטורת הכלי כוללת את מחשב ה client שבו מתקנים IDE כדוגמת Visual Studio, intelliJ IDEA או eclipse עבור פיתוח הקוד של תרחישי הבדיקה.
במחשב ה client גם מתקנים את שרת ה Appium שפותח ב node.js. שרת ה Appium מבין ומנתח את הבקשות המגיעות מה client ולפי סוג הבקשה פונה למכשיר המובייל המתאים בתקשורת המותאמת למערכת ההפעלה המבוקשת (Android, iOS) ומבצע את הבקשה על מכשיר המובייל.

החוזקות העיקריות של הכלי הינם תמיכה במכשירים אמיתים בסביבת אנדרואיד ו iOS וגם באמולטורים, תמיכה באפליקציות Native, Web ו-Hybrid, הכלי חינמי, יש מידע רב עליו באינטרנט, אפשר לפתח תרחישים בשפות פיתוח רבות כדוגמת java כמו כן יש אפשרות לשימוש ב inspector שהינו חלק מ-Appium או שימוש ב-UIAutomatorviewer שמגיע עם התקנת Android SDK לזיהוי הרכיבים השונים באפליקציה.
דוגמה לאופן שבו UI Automation Viewer מזהה את הטקסט של הסכום לתשלום (121.50) הנמצא בטבלת פרטי התשלום :

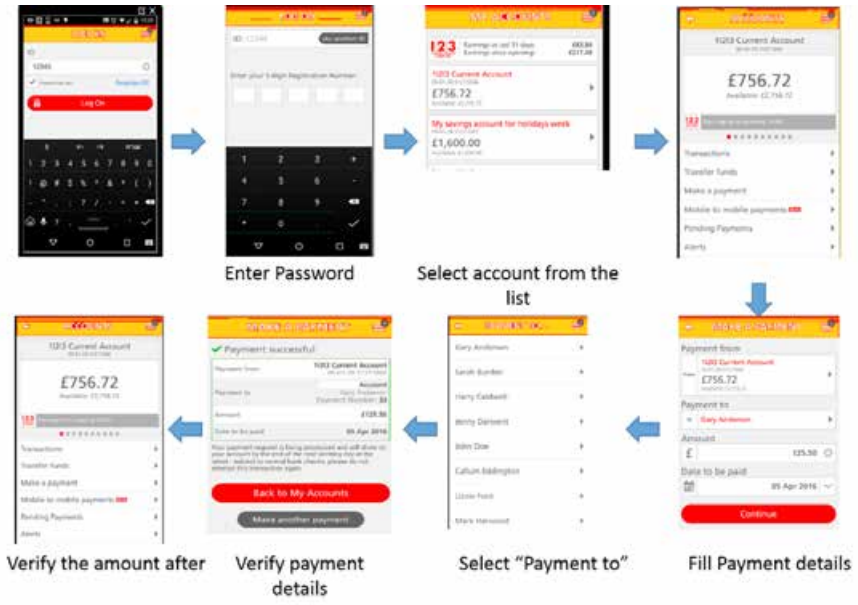


השוואת כלי אוטומציה לבדיקות על מכשירי מובייל | איל יובל

הכלי מיועד לבעלי כישורי תכנות מלאים בשפות כגון java או C# ויש צורך לפתח כל דבר שצריך כגון שימוש בקובץ נתונים חיצוני או דוח ריצה משופר כמו כן ההתקנות והקונפיגורציה של סביבת עבודה אינה פשוטה.
קישור מומלץ על Appium

התרחיש הנבחר לבדיקה

התרחיש הנבחר לבדיקה כולל ביצוע התחברות למערכת הבנקאית, ביצוע תשלום (Make a Payment), בדיקה שכל הנתונים נכונים ושאנו במסך הנכון בכל מעבר מסך ולבסוף בדיקה שהתשלום בוצע (בדיקה שהסכום בחשבון לאחר התשלום הינו הסכום בתחילת הפעולה פחות הסכום לתשלום).
מספר מסכים מהתרחיש באפליקציה הנבדקת:



הפרמטרים החשובים עבורי בהשוואה בין הכלים השונים

הפרמטרים החשובים עבורי בפיתוח התרחיש הנוכחי הינם:

1. יש צורך שהזרמת הנתונים לתרחיש (שם משתמש, סיסמא, מספר חשבון, פרטי תשלום) תתבצע מקובץ נתונים חיצוני.
2. יש צורך בבחירת נתונים דינמית מטבלת נתונים. מספר החשבון לביצוע הפעולה ופרטי מקבל התשלום הינם ערכים דינמיים הנבחרים מקובץ הנתונים החיצוני בזמן ריצה ואז יש צורך לחפש את הערך בטבלה, לדפדף (Scroll up) בטבלה עד שהערך נמצא (או הגענו לסוף הטבלה והערך לא נמצא) ולבחור את הנתון.
3. פעולות על טבלה - בדיקה שהנתונים לתשלום שנמצאים בתוך טבלה נכונים.
4. קריאת נתונים מהמסך - כיצד לבצע את קריאת היתרה בחשבון (בדוגמה המספר **756.72**) לצורך בדיקה שהתשלום בוצע.

פרמטרים נוספים וחשובים בבדיקת יכולות כלי האוטומציה:

5. **Cross devices** - האם התרחיש שפיתחתי ב **nexus5** ירוץ גם על **Samsung s6**? ועל מכשירי מובייל עם מערכת הפעלה **iOS**? מה ההתאמות שצריך לבצע לתמיכה בכך?
6. האם ניתן להריץ על מספר מכשירי מוביילים במקביל?
7. כיצד מזוהים האובייקטים השונים באפליקציה (כפתור, תיבת עריכה, טבלה, תווית)?
8. קלות ההתקנה וקינפוג כלי האוטומציה.
9. קלות לימוד כלי האוטומציה.

השוואת כלי אוטומציה לבדיקות על מכשירי מובייל | איל יובל

טבלת השוואה

	HPE Mobile Center	ePF	Appium
שפת פיתוח תרחישי הבדיקה	VBScript	SenseTalk	Java, C#, Ruby, Python, JavaScript, PHP
זיהוי הרכיבים	Object recognition Image recognition Text recognition	Image recognition Text recognition	Object recognition Text recognition
Cross Devices	התרחיש רץ על שני המוביילים מסוג אנדרואיד ללא צורך בשינוי. עבור iOS יש צורך לבצע התאמות רבות.	רכיבים שזוהו לפי טקסט, זוהו ללא צורך בשינוי עבור כלל המוביילים. רכיבים שזוהו לפי תמונה, נדרש שמירת תמונה עבור כל מובייל או התאמות תמונה למוביילים מאותה משפחה	התרחיש רץ על שני המוביילים מסוג אנדרואיד ללא צורך בשינויים. עבור iOS יש צורך בהתאמות רבות
תמיכה בהרצה על מספר מוביילים במקביל	יש	יש	יש
אופן חיבור המוביילים למחשב הפיתוח	USB Cable	USB Cable Wi-Fi	USB Cable Wi-Fi שימוש במכשירי מובייל שנמצאים בענן ולא בארגון בשימוש אתרים כדוגמת SauceLabs
קלות ההתקנה והקינפוג	נדרשת התקנת שרת אך סה"כ ההתקנה פשוטה	התקנה פשוטה	התקנה מורכבת המצריכה ידע טכני רב
שימוש בקובץ נתונים חיצוני	קבצי אקסל	קבצי csv	ניתן לפתח בכלי ה IDE
בחירת נתונים דינמית	חיפוש רכיב מסוים וביצוע "ScrollOnePage down" לדפדוף וחיפוש חוזר עד שהרכיב נמצא	חיפוש לפי טקסט וביצוע scrollUp לדפדוף וחיפוש חוזר עד שהאובייקט נמצא	חיפוש רכיב לפי טקסט driver.findElementByName (The Text) ושימוש באובייקט TouchAction לדפדוף וחיפוש חוזר עד שהרכיב נמצא
בדיקת נתונים מטבלה	אין אפשרות לזיהוי הטבלה ולכן יש צורך בזיהוי כל רכיב בתוך הטבלה	זיהוי הנתונים בטבלה לפי תמונה שלהם או שימוש בטקסט של נתונים קבועים אחרים הנמצאים בטבלה עבור מציאת נתונים דינמיים	רכיב טבלה מזוהה וניתן לגשת לכל שורה ותא בטבלה
קריאת נתונים מהמסך	**אפשרי בשימוש** GetROProperty ("text") **עבור הרכיב.** **דוגמה:** GetROProperty ("text") MobileLabel ("text")	אפשרי בשימוש readText	אפשרי בשימוש getText עבור WebElement
קלות לימוד כלי האוטומציה	לימוד הכלי הינו פשוט ומהיר	לימוד הכלי הינו פשוט ומהיר	נדרש ידע תיכנותי גבוה באחת השפות הנתמכות בכלי כדוגמת java או C#
ובנימה אישית	הכלי סובל עדיין מתחלואי מוצר חדש (נכון למרץ 2016) אבל מתפתח במהירות, נותן פתרון טוב לפיתוח בדיקות אוטומטיות על מובייל וכל מי שעובד עם UFT יכול במהירות ובקלות לפתח בכלי	הכלי נועד ומכוון קודם כל לבודקים ללא כישורי פיתוח גבוהים ולכן הדגש כאן הינו על שפת פיתוח פשוטה. זיהוי הרכיבים הינו אך ורק לפי תמונה וטקסט כך שאין תלות בטכנולוגיה ובמערכת ההפעלה ופיתוח התרחישים הינו פשוט ומהיר אבל לעיתים גם מקשה על התחזוקה	זה הכלי למפתחים ובודקים בעלי כישורי פיתוח. בדומה ל Selenium גם Appium הופך להיות הכלי הפופולרי בתחום. הכלי חינמי, נותן פתרון טוב לפיתוח בדיקות על מכשירי אנדרואיד ו iOS (אמיתים ואמולטורים) ומכיוון שפיתוח התרחישים מתבצע בשפות פיתוח כדוגמת java יש אפשרות להוספת כל ההרחבות וה-plug-ins הקימים בשפה כדוגמת שימוש ב-mavem, JUnit, Serenity BDD ועוד

מי שמעוניין לשמוע עוד על ההשוואה, להתייעץ או לשתף מניסיונו באוטומציה-בכלל ואוטומציה למובייל-בפרט מוזמן לפנות אלי במייל
eyal.yovel@qualitest.co.uk

מה אתם חושבים?

הצבעה לגיליון #6

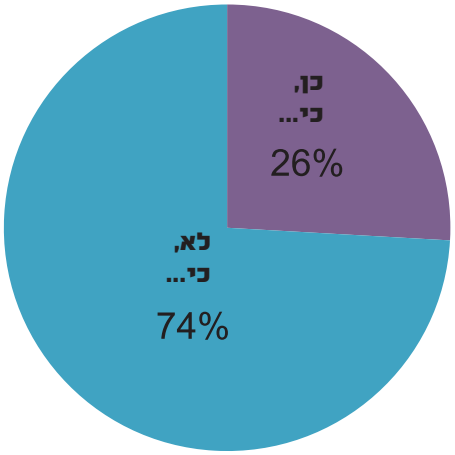
מה מעצבן אותך ביום יום בעבודתך כבודק?

לחצו כאן להצבעה

דעתכם חשובה, אנו מזמינים אותכם להשתתף בכל גיליון בהצבעה אשר את תוצאותיה נשתף עמכם בגיליון הבא.

תוצאות ההצבעה לגיליון #5

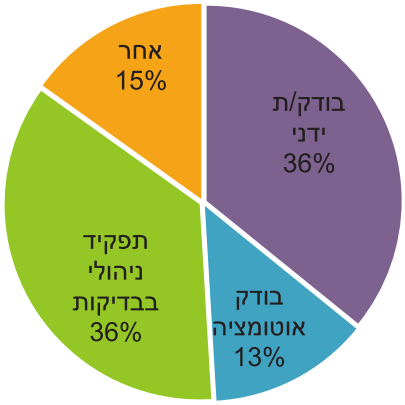
האם מיותר לקיים מפגשי בודקים?



מתוצאות ההצבעה אנו למדים שמפגשי בודקים זהו צורך חשוב מאוד לקהילה, שמחנו מאוד לראות ההיענות לשאלה זו, ואת מגוון התשובות המילוליות שהוספתם אשר ננסה לסכמן כאן. חלק מהעונים לא הבדילו בין כנסים המאורגנים ע"י גורמים שונים (חלקם מסחריים, וחלקם גם בתשלום – ותגובות ה- נגד – היו בעיקרן על נושאים אלו), לבין מפגשי בודקים חינמיים המאורגנים ע"י אנשים מהקהילה לרוב ע"ג פלטפורמת www.meetup.com ומתקיימים במשרדי חברות שונות המאפשרות שימוש בהתנדבות באודיטוריומים ובחדרי הישיבות שלהם, ולעיתים אף מספקות כיבוד קל – בין אלו נזכיר את JeST שקיימה כבר 9 מפגשים בירושלים ובמרכז. אך לשם קיום פעילויות אלו נדרשים מתנדבים אשר ייזמו המפגש, יתכננו התוכן, יקדמו ברשתות וידאגו לחברה מארחת – אנו רואים הרבה יותר מפגשים מתקיימים בתחום האוטומציה, ואלו גם יחסית רבי משתתפים (80-150) לעומת מיעוט מפגשים בנושאים מתודולוגיים וכמות משתתפים דלילה (10-20) דבר הסותר באופן מהותי את תוצאות ההצבעה (כולם רוצים מפגשים חינמיים, אך מעטים יוזמים אותם ומגיעים אליהם).
מהתשובות שהועלו בעד למדנו כי "נחוץ להעלות את הרמה המקצועית", "והפרייה הדדית היא קריטית במקצוע שלנו", "מעניין ללמוד מעמיתים וגם לספר להם על איך אנו עושים דברים", "זו יכולה להיות הזדמנות טובה ליצור קשרים מקצועיים", "אפשר להכיר אנשים חדשים ועבודות חדשות", "ללמוד על גישות שונות לבדיקות, נקודות מבט נוספות, ייעול תהליכים".

לסיכום "מפגשים פנים אל פנים יוצרים קשרים בין בודקים ומייצרים פלטפורמה לשיפור עצמי ולשיפור התחום – החלפת רעיונות, פתרון בעיות וכו'."

> **כולם רוצים מפגשים חינמיים, אך מעטים יוזמים אותם ומגיעים אליהם**



מה מגדיר טוב יותר את התפקיד הנוכחי שלך

בין העונים לסקר כמות הבודקים עלתה אך במעט על כמות המנהלים, וכמו כן היו כמה בעלי תפקידים אחרים כגון מנהלי פרויקטים, יועצים, אנשי אוטומציה ומחפשי עבודה בתחום.

תודה רבה על השתתפותכם!

ראיון עם מנהל בדיקות | שגיא וכמן

מראיין - דורון בר

שמי שגיא וכמן, אני בן 34, נשוי לדיקלה ואב לנועם ומעיין.
אני חי בראשון לציון ועובד בשנה וחצי האחרונות בחברת הסטארט-אפ Glide, חברה אשר הביאה את המושג "Video texting" לעולם אפליקציות התקשורת.
יש לי מעל לעשר שנות ניסיון בתחום ה- QA בחברות כגון Viber, Lifewatch ו-Giant Steps ולמזלי יצא לי ללמוד את רזי המקצוע אצל שני מנטורים נהדרים אשר מלווים אותי גם כיום: אייל כדורי ורמי בן ארי.

איך אתה רואה את תפקידך כמנהל בדיקות?

לדעתי לא משנה כמה התפקיד שלך בכיר במחלקת ה- QA אתה חייב להכיר את המוצר/ים שהחברה שלך מפתחת. כל יום לאחר פגישות הסקראם בבוקר מגיע הזמן לעבודת Hands-On שאינה מסתכמת בלבצע בדיקות למוצר אלא גם בסקירה לבדיקות הנכתבות לכל פלטפורמה ולשבת עם הבודקים ביחד בקריאת מסמכי הדרישות ולעזור ולהדריך אותם כיצד אפשר לשפר את התוצר של כל אחד ואחד מאתנו.
אחד הדברים שאני הכי מאמין בהם כאשר אנחנו יושבים ועושים את הסקירה של הבדיקות הוא שכל אחד צריך להרגיש שהוא יכול להיות הכי פתוח ו"להגן" על העבודה שלו. כבר קרו מקרים שבהם למרות שבתחילה חשבתי שהצורה שבה Test Case כתוב היא שגויה, אחרי שיחה עם הבודק הסכמתי עם ההערכה שלו והשארנו את הCase כפי שנכתב.
במידה ולא יוצא לי לשבת עם הבודק/ת ביחד על ה- Caseים אני דואג לשלוח מסמך אשר מציג את מה שהבודק/ת כתב/ה, בהמשך את איך שאני הייתי כותב את הבדיקה ובסיום השאלה הכי חשובה...
What do you think?
עם כל בודק במחלקה אני דואג לנהל ישיבת 1 על 1 פעם בשלושה שבועות על מנת להתעדכן עם כל אחד באופן אישי מה שלומו/ה ובניית תוכנית עבודה והתקדמות אישית לכל אחד ואחת בהתאם לשאיפות שלהם וצרכי החברה (כאשר לא בכל המקרים השאיפות והצרכים יובילו להישארות של אותו/ה בודק/ת במחלקת ה-QA אלא גם במעבר למחלקות אחרות).

אתגרים שעבר הארגון: המעבר לסקראם

עד לסוף שנת 2015 Glide עבדה בצורה שאני יכול להגדיר כסטנדרטית, לכל פלטפורמה של מובייל (iOS, Android, Windows Phone) היה צוות QA ייעודי אשר עבד בסמיכות עם צוות הפיתוח ומנהל המוצר והיו אחראים על סך כל הבדיקות של הפלטפורמה.
בחודשיים האחרונים של 2015 התקבלה ההחלטה בחברה לאמץ את גישת ה-Feature Teams (סקראם), וצורת העבודה של מחלקת ה-QA השתנתה מהקצה לקצה. בנינו תהליכים חדשים תוך לקיחת דוגמאות מחברות אחרות שעובדות בתצורה זו והתאמתם לאופי העבודה שלנו ב-Glide.
מניהול ישיר של צוותי QA הניהול שלי השתנה לניהול של התהליכים שבהם הקבוצות עובדות, ליווי בתהליכי ישיבות ההשקה של הצוותים והנחיה כיצד נכון יותר בהתאם ללוחות הזמנים החדשים להביא את מסמכי הבדיקות ואיכות הבדיקות למקומות הנדרשים.
נכון להיום מספר Feature Teams שלנו גדל וכעת יש לנו כחמש קבוצות שעובדות בצורה דינמית, מהירה ובחלקן אפילו רוחבית על פני מספר פלטפורמות.
יתרונות ברורים שניתן לראות לאחר השינוי הם בעיקר ברמת עבודת הצוות של כלל האנשים הנמצאים בקבוצה (מפתחים, QA, מנהל המוצר ומוביל הפעילות "Scrum Master"). העובדה שכל הגורמים בקבוצה יכולים להביע את דעתם, להסביר מדוע הם חושבים כפי שהם חושבים והתקשורת השווה העלתה את שביעות הרצון של כולם לגבי חשיבות התפקיד שלהם.
מנגד, חסרון שנמצא עדיין בתהליכי שיפור הוא היכולת של הקבוצה לנקוב בצורה נכונה את המורכבות של העבודה הדרושה ובעצם לספק הערכות זמנים נכונות.
למזלנו שיתוף הפעולה עם ראשי צוותי הפיתוח, מנהל ה- R&D, וה-CPO מאפשר לנו להדריך ולעזור לקבוצות להתקדם עם הבנה של המטרה.

כיצד הנכם פועלים להעשרת הידע של הבודקים?

נושא העשרת הידע של הבודקים בקבוצה הוא אחד מהנושאים העיקריים שאני משתדל לתת לו דגש רב. בפגישות המחלקה אשר מתרחשות אחת לחודש שבהן אני משתדל להביא חומרים חדשים הקשורים לעולם ה- QA, או הבאת "מרצים" ממחלקות אחרות אשר באים להעשיר את העולם הטכני של הבודקים מהעולמות השונים של הפיתוח. כמו כן אני דואג להעביר לכל חברי המחלקה הרצאות שלדעתי יכולות להעשיר את דרך החשיבה שלהם כבודקי תוכנה, כדוגמה הרצאותיו של ג'יימס באך.
ההבנה שככל שהבודק יודע נושאים רבים יותר כך הוא יכול לספק לחברה ערך גבוה יותר היא זאת שדוחפת אותי קדימה. על מנת גם שנוכל לצאת מעבר לעולמות הבדיקה של ה- BlackBox הבודקים עוברים/עברו בזמן האחרון הדרכות על ה- Application Lifecycle של כל אחת מפלטפורמות המובייל שבהן Glide מפתחת.
כדוגמה לערך הנוסף באיתור התקלות וההצדקה לזמן שהוקדש,

מה הסקרים אומרים?... יאן ברון iMDsoft,

סקרים

טור זה מפוזר במספר בלוקים ברחבי המגזין ומציג

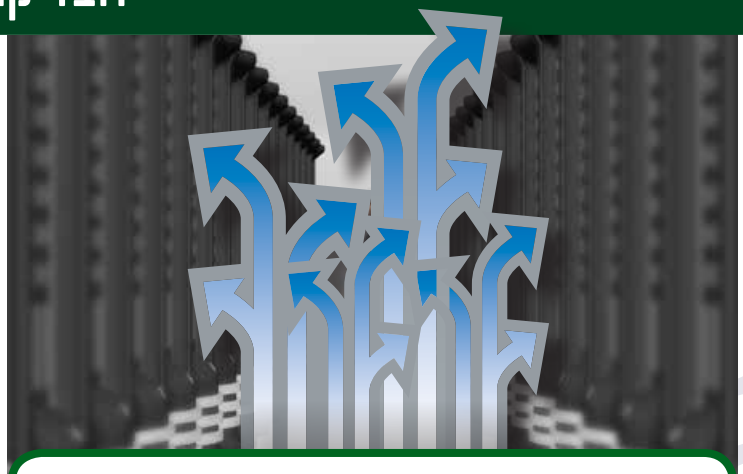
ממצאי סקרים המציגים מגמות מרחבי העולם לעיונכם. היום, אנו מציגים את את הדו"ח שיטות בדיקות תוכנה ברחבי העולם, מנוהל על ידי International Software Testing Certification Board

הסקר מכסה מספר נושאים, החל מהיבטים ארגוניים ותקציביים, לטכניקות / תהליכים / כלים, באמצעות מיומנויות ויכולות.

מחקר גלובלי שנערך באינטרנט עם יותר מ3200 בודקים ומנהלים על פני 89 מדינות.

ISTQB® Worldwide Software Testing Practices Report 2015-2016

ראיון עם מנהל בדיקות | שגיא וכמן



ההכרות של הבודקים עם ה- Activity Lifecycle מאפשרת להם לזהות את המיקום ובחלק רב מהמקרים את הסיבה שבגינה נכשלה האפליקציה ולספק את שנדרש (תודות לעבודת לוגים של צוות הפיתוח). כך אנו חוסכים למפתחים זמן חקירה – כאשר בעצם כל המידע הנדרש כבר הופיע מולם בבאג שדווח.

מהן התובנות שלך לגבי תחום הבדיקות?

אחרי עבודה בתחום של יותר מעשר שנים אחת מהתובנות הברורות ביותר שהבנתי היא שאין כזה דבר "פרופיל יחיד ואחיד לבודקים". יכולים להיות בודקים מנוסים, עם מספר שנות ניסיון מאחוריהם שעדיין לא מצליחים לצאת מהקופסה כשצריך ולהתמודד עם שינויים, ומנגד בודקים שרק התחילו לעבוד בתחום בין אם אחרי קורס או פשוט כאפשרות קידום במקום עבודה שלהם שלא יודעים מה ראשי התיבות STD או STP אבל הם בעלי חשיבה חדה וחוש טבעי לכל מה שקשור לתחום.

כמו כן, אחד מהמרכיבים החשובים ביותר של המקצוע שלנו הוא להבין שאנחנו והמפתחים עובדים ביחד, אנחנו יחידה אחת שהמטרה שלה זהה - לספק למשתמשים שלנו מוצר איכותי ולעמוד ביעדים של החברה. לא פעם אני נתקל במרמור מהצד של בודקים על המפתחים, "הם לא מקשיבים לנו", "הם לא מבינים שזה לא אמור לעבוד ככה" ותלונות נוספות... מה שחשוב להבין שבשביל זה הבודקים ומנהלי המוצר נמצאים ועובדים ביחד עם המפתחים, זה התפקיד שלנו לבדוק לא רק שהמוצר עובד אלא גם שהוא נוח ומזמין למשתמש - וכשזה לא המצב זה התפקיד שלנו להרים את הדגל. לרוב פשוט לשבת עם המפתח, לדבר על התהליך, להקשיב לצד שלו ולהציג את הצד שלך בדרך שהיא לא "דיווח באג" יכול לפתור את הבעיה ולייצר דו שיח חיובי וחשוב מאוד בין שני הצדדים.

מה היית ממליץ לבודק תוכנה בתחילת הקריירה שלו?

קודם כל תהנה/י ותלמד/י, מי שגייס אותך יודע שהוא גייס בודק/ת חדש/ה ושצריך להשקיע בך.

יחד עם זאת תמיד תשאף/י ללמוד את המוצר שאת/ה אחראי/ת על בדיקות שלו בצורה הטובה והמלאה ביותר שניתן: לקרוא את כל מסמכי הדרישות, לקרוא ולהריץ את הבדיקות הקיימות ולא להתבייש לשאול שאלות או להציע שיפורים - אף אחד לא מושלם ועוד זוג עיניים תמיד יכול להציע זווית חדשה.

ללמוד להכיר את האנשים (ובכוונה אני אומר אנשים ולא "מפתחים", "מנהלי מוצר", "מנהלי פרויקטים") מאחר ואלה היחסים שיעזרו לך לעבוד נכון ולפתור בעיות וקשיים בצורה המהירה והנכונה ביותר.

ולעצה האחרונה שאני אתן פה, תמיד לשמור על צורת התבטאות מכובדת - כחלק מהעבודה שלנו אנחנו מוצאים בעיות בעבודתם הקשה של אחרים, מעט רגישות can take a long way כמו שאומרים.

פספורט קבוצת Glide QA - ירושלים:

סוג המוצר הנבדק

אפליקציית התקשורת Glide

מוצר cross–platform אשר מאפשר למשתמש לתקשר לא רק בצורות התקשורת הרגילות כיום (טקסט, תמונות וכו'...) אלא לשלוח הודעות וידאו אסינכרוניות אשר ניתנות לצפייה בכל רגע ובכל מקום כל עוד המשתמש מחובר לתקשורת Data במכשיר הטלפון הנייד שברשותו, כולל צפייה ישירה בעת ההקלטה.

גודל הקבוצה

הקבוצה מונה כשניים עשר איש, כאשר מתוכם בודק אחד גם עוסק בכתיבת בדיקות אוטומציה.

וותק הבודקים

הכי צעיר (גויס ישר לאחר סיום הקורס) נמצא איתנו כשלושה חודשים בעוד שהוותיקה ביותר נמצאת מעל לשלוש שנים.

מבנה הקבוצה

הבודקים אשר עוסקים בפלטפורמות המובייל מחולקים ל – Feature Teams (צוותי סקראם).

הבודקים אשר עוסקים בעולמות השרתים וה Web-יושבים עם צוותי הפיתוח ומשמשים כבודקים הייעודים של כלל הסביבה.

סוגי הבדיקות

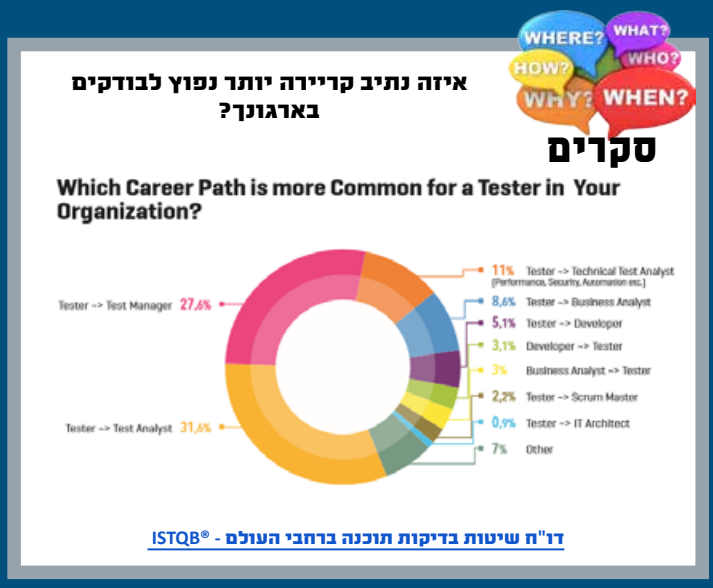
עם מוצר כמו Glide אנחנו בעצם מכסים את כל קשת הבדיקות כולל ביצועים, גלובליזציה, תאימות, תחזוקה וכדומה.

שיטת עבודה

בצוותי המובייל העבודה היא בסקראם, בשאר הצוותים העבודה היא ברוח ה-Agile אך הדגש הוא ה-Feature.

כמות המוצרים הנבדקים וקצב שחרור גרסאות

שלוש אפליקציות מובייל בנוסף לשרתים ו-Web. קצב שחרור משתנה בהתאם לטיב הפלטפורמה והצוות.



במדור היום נמשיך עם שאלה מבחינת הסמכה לרמת Agile.
מלבד רמות קוגניטיביות (K–levels) לכל פרק מוגדרים מספר יעדי לימוד (LO – learning objectives). יעדי הלימוד עוזרים למקד את הלימוד ולהגדיר מה צריך לדעת בודק מוסמך. ביעדי הלימוד מוגדר אם על הבודק לזכור נושא מסוים, להבין אותו, ליישם או לנתח ובהתאם לכך מוגדר ה–K–level של יעד הלימוד.

השאלה של היום:

השאלה של היום תהיה מרמת K3, כדי לקבל הסבר על רמות הבחינה השונות אני ממליץ לקרוא את המאמר של מיכאל שטאל או לחזור לפינה הזו בגיליון מספר 1.
את יעד הלימוד של השאלה נתאר בפתרון שיופיע בהמשך הגיליון.

במהלך פגישת תכנון איטרציה (iteration), משתפים אנשי הצוות את מחשבותיהם בנוגע לסיפור משתמש (user story). מוביל המוצר (product owner) מייעץ שיוצג ללקוח עמוד אחד שבו יוכל להכניס מידע. המפתח מסביר שקיימות מגבלות טכניות לדרישה הזו, בשל כמות המידע שצריך לקלוט מהמסך המדובר. מפתח אחר אומר שיש סיכון לפגיעה בביצועים (performance) מאחר והמידע יאוחסן על בסיס נתונים בענן.

איזה מהמשפטים הבאים מהווה דוגמה טובה ביותר לתרומתו האפשרית של בודק בדיון מעין זה?

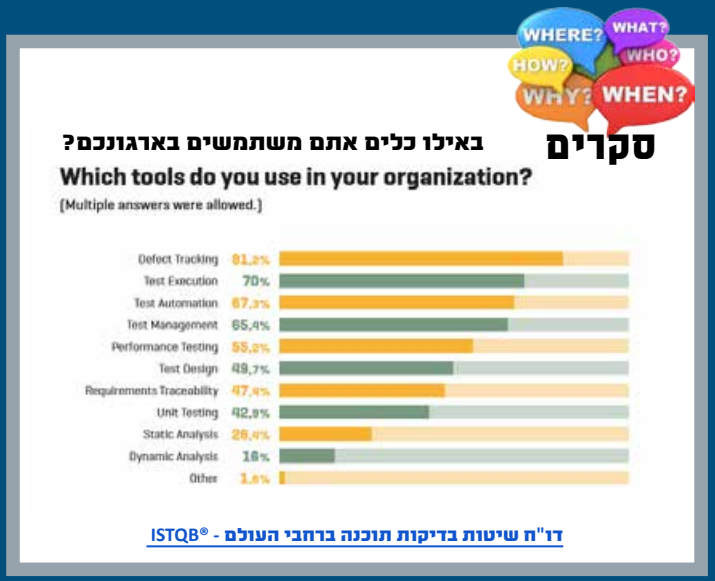
א – הבודק יתמוך בהצעה שהמסך יהיה על עמוד אחד, כי זה יקל על פיתוח אוטומציית הבדיקות.

ב – הבודק יציין שהשימושיות (usability) חשובה יותר מהביצועים (performance).

ג – הבודק יציע שקריטריון הקבלה לביצועים (performance) יהיה: "זמן שמירת המידע יהיה קטן משניה אחת".

ד – הבודק יציין שיש להגדיר לסיפור המשתמש קריטריון קבלה שניתן לבדוק (testable).

*לצפייה בתשובה המפורטת – דפדפו לעמוד 30



מודל שיפור תהליך בדיקות TPI

אייל זילברמן

פעיל בתחום בדיקות התוכנה מזה 20 שנה. החל את דרכו כבודק תוכנה, ראש צוות ומנהל בדיקות. כיום משמש אייל כנשיא חברת קוולישטס, חברת הבדיקות הגדולה בישראל והשנייה בגודלה בעולם. אייל חבר ב-AB של ארגון ה-ITCB.

מה זה TPI?

TPI (Test Process Improvement) הינו מודל המעריך את רמת הבגרות של תהליך הבדיקות בארגונים ומטרתו היא לשפר את תהליך הבדיקות באופן רציף ומתמשך.

למה צריך את זה?

הרבה אנשי בדיקות, מנהלי בדיקות וגורמים בעלי עניין מעוניינים בשיפור תהליך הבדיקות בארגונם. בהרבה מקרים, שיפור תהליך הבדיקות מתרכז במספר נושאים מצומצם כגון "בואו נכניס אוטומציה", "בואו נשפר את מסמכי הבדיקות" או "בואו נשלח הבודקים להדרכות".
בפועל, שיפור תהליך הבדיקות דורש שיפור של מספר רב של אספקטים. קשה להעריך את רמת תהליך הבדיקות כטוב או רע. בכל תהליך בדיקות יש תהליכים שרצים בצורה טובה וכאלה שדורשים שיפור. נדיר עד בלתי אפשרי למצוא צוותי בדיקות מושלמים, דהיינו צוותים שאין מה לשפר בהם.
מודל TPI נועד לסייע לאותם צוותים לשפר את תהליך הבדיקות באמצעות הערכת רמת בגרות התהליך בכל אחד מהאספקטים הקשורים לתהליך הבדיקות ומילוי מטריצת בגרות בדיקות (**Test Maturity Model, TMM**).

היסטוריה ומודלים דומים

TPI הינו מודל בינלאומי מוכר שפותח ע"י חברת **Sogeti** בשנת **1998** ופורסם בספר בשם **TPI**. בשנת **2009** פורסמה גרסה חדשה למודל בשם **TPI Next**. מאמר זה מתייחס בעיקרו לגרסה האחרונה של המודל.
TPI הינו מודל הבגרות לבדיקות המוביל בעולם. מודל נוסף שתופס תאוצה הוא **TMMi** שפותח ע"י ארגון עצמאי העונה לאותו שם בו חברים מספר דמויות בולטות מתחום הבדיקות בעולם ואשר משלים את מודל ה-**CMM** (מודל בגרות של כלל תהליך הפיתוח).

מטריצת בגרות בדיקות (Test Maturity Model, TMM)

נושאי מפתח

מטריצת בגרות בדיקות מחולקת ל-**16** נושאי מפתח (**Key Areas**). הקשורים לתהליך הבדיקות נושאי המפתח מחולקים ל-**3** קבוצות:

1. קשר עם בעלי עניין (**Stakeholder Relations**) – לדוגמה: מחויבות בעלי עניין (**Stakeholder commitment**), אסטרטגית הבדיקות (**Test strategy**), תקשורת (**Communication**) ועוד
2. ניהול הבדיקות (**Test Management**) - לדוגמה: ביצוע הערכות ותכנון (**Estimating and planning**), מטריקות (**Metrics**), ניהול תקלות (**Defect management**) ועוד
3. מקצוענות הבדיקות (**Test Profession**) – לדוגמה: פרקטיקת מתודולוגיה (**Methodology practice**), מקצוענות הבודקים (**Tester professionalism**), כלי בדיקות (**Test tools**), סביבת הבדיקות (**Test environment**) ועוד.

רמת בגרות

עבור כל נושא מפתח נקבעת רמת הבגרות של הנושא בארגון. רמת הבגרות נקבעת באמצעות נקודות בקרה (**CheckPoints**). נקודת בקרה היא נושא אותו הארגון מיישם (או שלא). תהליך הבחינה וההערכה האם הנושא מיושם נבחנת באמצעות בחינת תהליך הבדיקות ותוצרי הבדיקות השונים. נקודות הבקרה מחולקות ל-**13** רמות קושי (**Clusters**) מ-**A** עד **M** בהתאם לרמת הקושי של יישום הנושא בארגון ומחולקות ל-**3** קבוצות: מבוקר (**Controlled**), יעיל (**Efficient**) וממוקסם (**Optimizing**). כל קבוצה כוללת מספר שונה של נקודות בקרה (בין 2 ל-4) ומרמות קושי שונות.

Topic	#	Key Area	CPs	Controlled				Efficient				Optimizing		
Stakeholder Relations	1	Stakeholder commitment	10	A	A	A	B	E	G	G		J	L	L
	2	Degree of involvement	9	A	B	C	E	H	H	J		L		L
	3	Test strategy	9	A	A	B	E	F	F	H		K		L
	4	Test organization	11	A	D	D	E	I	I	J	J	K	L	L
	5	Communication	9	B	C	C	D	F	F	J		M		M
	6	Reporting	8	A	C	C		E	G	G		K		K
Test Management	7	Test process management	9	A	A	B	B	G	H	J		K		M
	8	Estimating and planning	11	B	B	C	C	J	H	I	I	K	L	L
	9	Metrics	9	C	C	D		G	H	H	I	K		K
	10	Defect management	11	A	A	B	D	F	F	H	J	K	L	L
	11	Testware management	10	B	B	D	E	I	I	J		L	L	L
Test Profession	12	Methodology practice	9	C	D	E		F	H	J	J	M		M
	13	Tester professionalism	11	D	D	E	E	G	G	I	I	K	K	M
	14	Test case design	10	A	A	E		F	I	I	J	K	K	M
	15	Test tools	10	E	E	E		F	G	G	I	L	M	M
	16	Test environment	11	C	D	D	E	G	H	J	J	L	M	M

מודל בגרות הבדיקות בארגון

דוגמה – נקודות בקרה עבור נושא מפתח מס' 6 – דיווח (Reporting)

על מנת להגיע לרמה "מבוקר" (הרמה הנמוכה ביותר) יש לעמוד ב-3 נקודות הבקרה הבאות:

רמה A – הדיווח בארגון כולל היבטים של זמן או מחיר, תוצאות וסיכונים (מאומת בדרך כלל באמצעות מעבר על דו"חות המופקים ע"י צוות הבדיקות ווידוא שנהם כוללים מידע זה).

רמה C – תכיפות ותוכן הדיווח תואם לדרישות הבסיס של בעלי העניין בארגון לצורך קבלת החלטות (מאומת באמצעות ראיון עם בעלי העניין בארגון ווידוא כי הם מקבלים את המידע הנדרש להם מצוות הבדיקות לצורך קבלת החלטות).

רמה C – הדיווח נעשה בכתב (מאומת באמצעות מעבר על דו"חות שוטפים).

על מנת להגיע לרמה "יעיל" (הרמה האמצעית) יש לעמוד ב-3 נקודות הבקרה הבאות:

רמה F – יישום הדיווח בהתאם לדרישות בעלי העניין, הנדרש לצורך תהליך קבלת החלטות יעיל, מאוזן עם המאמץ הנדרש לאספקת הדיווח (יש לוודא כי המשאבים הנדרשים לצורך הפקת הדו"חות עומדים ביחס סביר לערך שיש לדו"חות לארגון ואין חוסר יעילות בכל הקשור להפקת הדו"חות).

רמה G – הדיווח כולל מגמות והמלצות בנוגע להתקדמות תהליך הבדיקות וסיכוני הפרויקט.

רמה G – הדיווח כולל מגמות והמלצות בנוגע ליעדי הבדיקות וסיכוני המוצר.

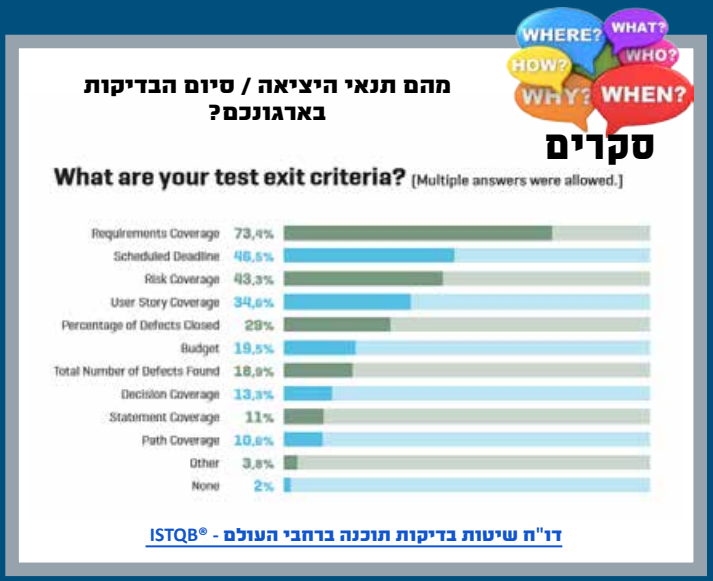
על מנת להגיע לרמה "ממוקסם" יש לעמוד ב-2 נקודות הבקרה הבאות:

רמה K – הדיווח כולל מידע ו/או מדידות שיכול לשמש ושיפור מידי או עתידי של תהליך הבדיקות ותהליך הפיתוח.

רמה K – המידה והמדידות לשיפור תהליך התוכנה מועברים למנהלי הפיתוח בסיום כל פרויקט בדיקות.

סיכום

שימוש במתודולוגיית ה-TPI מסייעת לארגונים להעריך את איכות ובגרות הבדיקות, אולם משמשת בעיקרה לצורך שיפור תהליך הבדיקות תוך ראייה כוללת של כלל מרכיבי הבדיקות בארגון. תוך מתן המלצות נקודתיות וכלליות. גם ללא יישום מלא, המודל כולל הרבה מאוד רעיונות, ולכן מומלץ לקרוא אותו לעומק, בדגש על 158 נקודות הבקרה שבו כאשר כל נקודה מהווה פוטנציאל לשיפור את תהליך הבדיקות בארגון.



דגשים והמלצות בעבודה עם צוותים גלובליים | ענבר מאיר

ענבר מאיר

מנהלת בדיקות בסיסקו נתניה ב-16 שנים האחרונות, בעלת ניסיון של כ-23 שנים בתחום איכות ובדיקות תוכנה.
בעלת תואר ראשון במחשבים וניהול והסמכה של מהנדסת איכות תכנה מארגון האיכות האמריקאי (CSQE).
במהלך השנים הקימה 3 קבוצות בדיקה שונות, השתתפה בבדיקות פנימיות של איזו- 9000 בלוסנט וסיסקו וכן הרצתה בסיגיסט בכנסי איכות ובדיקות שונים.

המציאות הגלובלית בה אנו חיים היום מתבטאת, בין היתר, בכך שצוותים רבים פרוסים על פני מספר מדינות, כאשר העובדים השונים לא חולקים את אותם זמני עבודה (שעות ביממה, חגים וסופי שבוע), אותה שפת אם, אותו מבטא ובהחלט לא חולקים את אותה התרבות.

נשאלת השאלה- האם הפערים האלו, האם השונות הזו ניתנים לגישור? אני חושבת/מאמינה שכן, אבל הדבר מצריך עבודה ומחשבה מראש על דרך הפעולה שלנו. כדי לבצע זאת בצורה מיטבית יש להבין את המדינה והתרבות מולה עובדים (כדי להימנע מסיטואציות לא נעימות ולא מכבדות), ליצור היכרות אישית ואכפתית בין אנשי הצוות, להשתמש בכלים שאולי איננו משתמשים בעבודה לוקלית ומעל הכל להיות בן אדם נימוסי ואדיב הפתוח לשונות תרבותית.

בעבודתי בעבר כמהנדסת בדיקות יצא לי לעבוד עם צוותים בעמק הסיליקון – קליפורניה, בהמשך כמובילת צוות ולאחר מכן כמנהלת יצא לי לעבוד עם עמיתים ועם כפופים בסין, הודו וארה"ב – מזרח, מערב ומרכז.
אשתף אתכם בדברים שלמדתי במסע הזה תוך כדי נתינת דוגמאות שבתקווה יעזרו לכם להתמודד עם הנושא ובנוסף אדגיש מספר נקודות שלדעתי חשוב להתייחס אליהן.

עבודה בזמנים שונים

כיום אני רואה עבודה בזמנים שונים כיתרון צוותי גדול. כאשר הצוות הישראלי בחופש הצוות בארה"ב או בהודו עובדים ולהיפך. יש כאן גיבוי הדדי והספק טוב יותר ומלא יותר של הצוות כולו.

כמה טיפים/נקודות למחשבה

1. **חשוב מאד לכבד את זמני החופשה והחגים** של הצוותים בחו"ל. לפעמים, כאשר מדובר בצוות ישראלי מקומי אנו נוהגים לעבור כאן את הקווים, אך כאשר מדובר בצוות לא מקומי עלינו להקפיד על כך כפליים.

2. **סופי שבוע** כאשר אני נתקלת בצוותים מחו"ל שעובדים בסופי שבוע – אני לא שוכחת להודות להם ובד בבד להזכיר להם שיש להם משפחה ושישנן משימות בעדיפות נמוכה שהם יכולים להשלים בשבוע העבודה הרגיל. מעבר לכך, אני מבקשת מהם ליידע אותי בזמן או מראש בבעיות הספק הזמנים שלהם ע"מ למנוע עבודה בסופי השבוע. לשם כך קיימים בכל חברה כלים שונים מלבד הדוא"ל המוכר לכולנו. מניסיוני ראיתי שהעבודה בסופי שבוע בולטת יותר בצוותי בדיקות מהודו ופחות בצוותי בדיקות מארה"ב. לדעתי עלינו להיות רגישים לנושא הזה ולא לתת הרגשה לצוות כלשהו שהוא רק "סוס עבודה". יש לשקף לצוותים השונים שקיימת ההבנה הרוחבית שבסוף השרשרת קיים אדם שעליו לאזן בין חיי משפחה ועבודה.

> **יש להתחשב בזמני העבודה היומיים של כל צוות**

3. **סנכרון לפני חופשות וסופי שבוע בין הצוותים השונים** – ע"מ לדאוג לעבודה שוטפת ולהשגת מירב האפקטיביות של הצוותים הגלובליים – יש לוודא שכל הצוותים מודעים לחגים ולחופשות ארוכות. יש לשלוח עדכון לכל הצוותים כולל המנהלים הבכירים ולציין אנשי קשר למקרים חשובים (מקרי חירום) במקרה שצריך אני גם מוודאת שלצוותים שממשיכים לעבוד יש גישה למעבדות, למחשבים וכו'.

צוותים בחו"ל יעצרו מלהתקדם במקרה שנתקלו בבעיה הדורשת פתרון מהצוות הלוקלי הנמצא בחופשה, מניסיוני הצוותים בחו"ל לא יתקשרו ישירות לאנשי הקשר למקרי חירום וינסו להשיג מענה דרך מיילים בלבד.

יש לזכור זאת ובמקרה שאתם עומדים לפני אבן דרך חשובה וקריטית- מומלץ מאד במהלך החופשה לוודא שאין בעיות גישה למידע ושלא נוצר צוואר בקבוק כלשהו אצל הצוותים בחו"ל. במקרה ושנוצר מצב שכזה – אני בעצמי מפעילה את האנשים (גם את האנשים הנמצאים בחופשה) ומוודאת שהמכשול יוסר. ההתערבות הספציפית הזו בזמן החופשה משקפת לצוותים שאנו עדין צוות אחד עם מטרה אחת ובמידת הצורך כולם "מתגייסים" לעזרה.

4. **קיום ישיבות בזמנים המתאימים לצוותים השונים** יש להתחשב בזמני העבודה היומיים של כל צוות במיוחד במידה וקיימות הסעות מרוכזות בסוף יום העבודה כפי שקיימות אצלנו בצוות ההודי. כאשר עובדים עם צוות סיני ואמריקאי הדבר קשה במיוחד מכיוון שיש כ--9 שעות הבדל בין הצוותים. הפתרון שהגענו אליו הינו שברוב הישיבות המשותפות עם הסינים הצוות הישראלי מייצג את העובד מארה"ב.

כאשר אנו עובדים עם צוות הודי בלבד- אני מאד מקפידה לבצע את הישיבות מוקדם בבוקר. לפני קביעת הישיבות – אני מוודאת שהצוות ההודי מאשר את זמני הישיבה. אנו מקפידים לא לקבוע ישיבות על זמני ארוחת הצהריים של הצוות ההודי למרות שהדבר מאד נוח מבחינתנו. ההתחשבות הזו חשובה ע"מ ליצור שיתוף פעולה מיטבי והבנה שאנו צוות אחד עם פריווילגיות שוות.

במקרה שיש לשלב את הצוות מארה"ב – אנו מתייעצים וקובעים יחדיו את השעה המאושרת ע"י כולם – צוות אחד יגיע מוקדם וצוות אחר יישאר מאוחר. את זאת אנו מבצעים ברוטציה כך שלא יקרה המקרה שאותו צוות יגיע באופן קבוע מוקדם וצוות אחר יישאר באופן קבוע עד מאוחר.

אנו מקפידים להקליט כל ישיבה ולשלוח לכל הצוותים – כך שבמידה ואיש צוות לא הגיע לישיבה בגלל השוני בזמנים הוא יוכל לצפות בהקלטה.

> **אנו עובדים עם צוותים ששפת האם שלהם אינה אנגלית או שקיים מבטא המקשה עלינו ועליהם להגיע להבנה מלאה.**

דגשים והמלצות בעבודה עם צוותים גלובליים | ענבר מאיר

שפה ומבטא

אנו עובדים עם צוותים ששפת האם שלהם אינה אנגלית או שקיים מבטא המקשה עלינו ועליהם להגיע להבנה מלאה. הקושי מתבטא בעיקר בעבודה עם צוות הודי היודע אנגלית ברמה גבוהה אך הרבה פעמים עם מבטא "כבד ובעבודה עם צוות סיני שבו רוב הצוות לא דובר אנגלית פרט לראש צוות או מנהל הדוברים אנגלית.

ע"מ להתמודד עם האתגר הזה יש לוודא מספר דברים:

1. **קיום ישיבות** – יש לקיים את הישיבות לא בשעות מאוחרות בהן הריכוז שלנו כבר ירוד ומקשה עלינו להבין מבטאים שונים.
2. **שילוב אדם חדש** – לאדם חדש בישיבה המשלבת אנשים מצוותים שונים בעלי מבטא שונה יהיה קשה לעקוב אחר מהלך הישיבה ולהבין את מלוא הנאמר בה, לוקח זמן להתרגל למבטאים שונים ו"לסנן" מהם את התוכן הנאמר. לכן, מומלץ לשתף בישיבה אנשי צוות שכבר השתתפו בישיבות האלו והם ישמשו כמתווכים ו"מסננים" למשתתף החדש עד שהוא יתרגל למבטאים השונים.
3. **לחזור על הדברים תוך סיכומם** וקבלת אישור שהנאמר הובן במלואו. קיימת אפשרות לבצע זאת בע"פ (" אני רוצה לוודא שהבנתי – ....") ובסיום להעלות הסיכום בכתב או לסכם הנאמר ישירות במהלך הישיבה. חשוב לוודא שהסיכום הכתוב מוצג אונליין לצוותים ע"מ לוודא שהסיכום מייצג נאמנה את הנאמר. בעבודה עם צוותים סינים הוכח שסיכום אונליין חשוב לאין שיעור והצוות התערב ותיקן דברים רבים, הצוות הסיני באופיו מקפיד על כל קוצו של יוד (מומלץ לקחת זאת בחשבון מכיוון שהישיבה מאד מתארכת).

כאשר עובדים עם צוות הודי ההתייחסות בקשר לסיכום האונליין משתנה בהתאם להכרות עם הצוות, מעמדו בארגון ו"גבולות" הישיבה (אישית או רבת משתתפים). במקרה שהצוות ותיק ומעמדו יציב– הוא יגיב ואף יבקש תיקונים והבהרות על הסיכום גם בישיבה אישית וגם בישיבה רבת משתתפים לעומת זאת, במקרה והצוות חדש או מיוצג ע"י מהנדס חדש לא יעלו הרבה בקשות לתיקונים.

4. **ביטויים משפת האם** – האמריקאים משתמשים לעיתים בביטויים משפתם ותרבותם שלא תמיד מובנים על ידי האחרים. ע"מ לוודא שהכל הובהר יש לבקש בצורה עדינה ומנומסת הבהרה לביטוי. הדבר חשוב בעיקר במקרה שמשתתפים בפגישה צוותים סיניים שהשפה האנגלית לא שגורה בפיהם כמו השפה האנגלית אצל הצוות ההודי.
5. **שימוש בקיצורים** שימוש בראשי תיבות הינו דבר היכול להיות ברור לנו מאוד אך לא ברור לאנשי צוות אחרים ולהיפך. הצוותים ההודים מאד אוהבים להשתמש בקיצורי מילים ויש מספר רב של קיצורי מילים המאוד מאפיין את הצוות האמריקאי. חשוב לוודא שכל קיצור מוסבר הן במסמך והן בשיחה במהלך הישיבה.

<u>לדוגמא</u> – השימוש בקיצור "HL" (High-Level), שהיה בכותרת מסמך בדיקות שעבר סקירה (review) ע"י צוות ישראלי וצוות סיני גרם לנו לעיכוב של כ-40 דק' בישיבה. אחרי קרוב ל40 דקות הבנו את מקור חוסר ההסכמה של הצוות הסיני לשימוש בקיצור הנ"ל. הצוות הסיני תירגם את ראשי התיבות האלו לשם של מדינת "ישראל" ולכן לא הסכים שהמסמך המשותף לשתי הקבוצות יישא רק את שם המדינה של אחד מהצוותים. הפתרון שהגענו אליו היה לא להשתמש בקיצור הנ"ל בכותרת וכן לוודא שקיים מקרא לכל הקיצורים המופיעים במסמך.

הכרת המעמדות/התרבות

1. **הפניית הבקשה – קדימות בפנייה:** בעבודה עם צוותים שונים, במיוחד צוות הודי וצוות סיני, יש חשיבות גדולה למעמד העובד. במקרה שבמפגש/ישיבה משתתפים מנהלים, ראשי צוותים ועובדים יש לפנות תחילה לעובדים הבכירים יותר. במקרה שרוצים להציג צוות חדש לצוות מקומי (ישראלי) ניתן לבקש מכל עובד בצוות המרוחק להציג עצמו אך מקובל יותר ומכובד יותר לתת לעובד הבכיר להציג את שאר העובדים.

כאשר רוצים להשיג הסכמה מהצוות המרוחק, בעיקר כאשר משנים תהליך עבודה או מוסיפים עבודה, יש לפנות לעובד הבכיר ולהפנות אליו את הבקשה. שוב – הדבר מקבל בעיקר חשיבות בעבודה עם צוותים הודים וצוותים סינים. בצוות הסיני שעבדנו אתו בעבר העובדים לא פנו אלינו כלל. הכל נותב ונוהל ע"י המנהלת הסינית שייצגה את הצוות, ייתכן והדבר נבע גם מידיעת השפה האנגלית אך בהחלט יש למעמדות תפקיד.

מעבר לכך יש להדגיש גם את צד התרבות שלנו, חשוב להבהיר שאנו תרבות המעודדת פנייה ישירה וכנה לכל עובד לא משנה מעמדו. אני מדגישה זאת במצבים שונים לצוות ההודי ומעודדת אותו לשאול אותי שאלות, ליצור קשר במידה ויש בעיה ולא להמתין לישיבה מסודרת או לפנייה רשמית ממני.

2. **השגת המידע** – בעבודה עם צוותים הודים נתקלנו רבות בבקשות של הצוותים להשיג להם מידע שהם יכולים להשיג בעצמם ממקורות שונים כמו מסמכים ו/או צוותים אחרים, עד שהמידע לא הועבר לצוות ההודי לא הושגה מבחינתו שום התקדמות בעבודה. לדעתי הדבר נובע מאורך החיים של המעמדות הגבוהים בהודו. ע"מ להפחית תרעומת של הצוות המקומי (המתבקש כל הזמן להשיג מידע לצוות המרוחק) חשוב להסביר את שורשי ההתנהגות ולפעול יחדיו לשנותה. במקביל חשוב להבהיר בעדינות לצוות ההודי שמידע כזה או אחר הקשור ישירות להתקדמותו בעבודה הינו באחריותו ועליו להשיגו בעצמו.

ע"מ לוודא שהדבר אכן יקרה – מומלץ לציין זאת כמשימה קטנה ומדידה הרשומה בסיכום הישיבה או להעביר זאת בדוא"ל ספציפי.

> **בעבודה עם צוותים שונים (במיוחד צוותים הודים וסינים) יש חשיבות גדולה למעמד העובד**

3. **החגים בתרבויות השונות כדרך לקירוב** אנשי הצוות זה לתרבותו של זה– החגים השונים בתרבויות השונות יכולים להיראות לנו משונים עד מצחיקים. ע"מ לקרב בין הצוותים, ליצור הבנה לתרבות האחר ולהראות שאנו בעצם לא כל כך שונים, אני נוהגת להסביר על החג שלנו לצוותים האחרים ומבקשת מהם הסבר על החג שלהם.

לדוגמא – בעבר, כאשר הסברתי לצוות סיני מהו חג "טו– בשבט" עשיתי זאת בצורה שתיצור קשר לתרבותם וחגיהם הקשורים לטבע. הסברתי שאנו חוגגים חג לעצים – שרים שירים לעצים ולטבע וכן נוטעים עצים חדשים. ההסבר הזה קירב ויצר דמיון בין החגים הסינים לחגים היהודים.

בצוות הגלובלי שלנו הגענו למצב שאנו שולחים תמונות של המשפחות מחופשות בפורים מישראל, מקבלים תמונות של בית מסודר ונקי לחג הודי ומשפחה אמריקאית עומדת ליד עץ אשוח מקושט. אנו שולחים ברכות "חג שמח" בין חברי הצוות המקומי וכן גם לצוותים הרחוקים בחגים מסוימים ועיקריים. יש לציין שהצוות ההודי התגלה כצוות מאד סקרן לגבי החגים היהודיים וההיסטוריה העומדת אחריהם ונוצר שיח מעניין בנושא. השיתוף הבין-תרבותי הזה יצר אצלנו רובד נוסף של שיתוף וחיבור.

כלי עזר

1. כדי ליצור קשרים מעבר לקשרי עבודה בסיסיים וליצור הזדהות ועזרה הדדית נעימה בתוך הצוות, **יש לקיים ישיבות כלליות וישיבות מצומצמות** המועברות בווידאו. אחת לתקופה אני מקפידה לקיים ישיבה בטלפרזנס בו התחושה שהצוותים השונים נמצאים יחדיו באותו החדר.
2. **סיכומי ישיבה** והקלטות ישיבה ברמת הציוותית וברמה של הישיבות האישיות (1:1) עם אנשי צוות רחוקים – על הנושא הזה וחשיבותו דיברתי לעיל.

> **לא משנה השפה, המבטא, המיקום הגאוגרפי, החגים השונים, הלבוש והמאכלים השונים בסופו של דבר אנו כולנו אנשים**

דגשים והמלצות בעבודה עם צוותים גלובליים | ענבר מאיר

הפן האישי

עלינו לזכור שלא משנה השפה, המבטא, המיקום הגאוגרפי, החגים השונים, הלבוש והמאכלים השונים- בסופו של דבר אנו כולנו אנשים עם שאיפות לבצע עבודה מאתגרת ומעניינת בשעות העבודה, ולהתמקד במשפחה, בתחביבים ובנושאים אחרים מחוץ לשעות העבודה.

לכן לא משנה מיהו הצוות שמולו אני עובדת – אני מכניסה פן אישי עם אנקדוטות על משפחתי ועל דברים שקרו לי בעבודה. שמתי לב שהדבר חשוב במיוחד לצוות ההודי ולנשים בתוכו המעוניינות לשמוע על השילוב בין עבודה טכנולוגית ומשפחה עם ריבוי ילדים (לרוב העובדים ההודים בצוות שלנו יש רק ילד 1).

לצוות הסיני היה קשה מאד לשלב דברים אישיים מצדו – זו תרבות מופנמת יותר בהשוואה לתרבות המערבית בכלל ולתרבות הישראלית בפרט. לאחר כ-8 חודשי עבודה ומספר רב של ישיבות ארוכות "נשבר המחסום" הזה מול המנהלת סינית שעבדתי מולה וחלקנו גם פן נוסף בעבודתנו – את הפן האישי.

ע"מ להכניס פן אישי אבל לא אישי מידי אנו נוהגים לציין הישגים אישיים של אנשי הצוות שלא קשורים לעבודה בישיבות צוות. חשוב לבקש את האישור של העובד "לפרסם" זאת בישיבת הצוות.

יחד עם זאת יש להיות פה מאד רגישים (בייחוד עם צוותים אמריקאים), שימוש באינטליגנציה רגשית בהחלט עוזרת כאן.

לסיכום קיימת שונות בין אנשים, קיימת שונות בין תרבויות וקיימת שונות בין זמני העבודה של הצוותים. בבואנו לעבוד עם צוותים הפרושים גלובלית עלינו להכיר בכך ולהתייחס לזאת בדרך העבודה היומיומית. כאשר מטרתנו הינה לגבש צוות אחד עם מטרה משותפת ולהגיע אליה ביעילות עלינו להיות רגישים, אנושיים ולהתייחס לנקודות שהעליתי לעיל:

- הכרת התרבות
- התייחסות לזמני חופשה, זמני סופ"ש וזמני עבודה יומיים
- שפה ומבטא
- התייחסות לפן האישי של אנשי הצוות
- שימוש בכלי עזר (ווידאו, סיכומים)

למען הקהילה | ITCB® Israeli Testing Certification Board

הסכם שת"פ מחקרי בתחום הבדיקות ושיתוף אנשי אקדמיה נחתם בין גרמניה לישראל בקידום משרד הכלכלה, מרצי אוניברסיטאות ועמותת ITCB®

הבודקים כברווז גומי

כשאנו באים לפתור בעיה לעיתים קרובות כל שאנו נדרשים הוא לומר אותה בקול - ובאורח פלא התשובה לבעיה צצה ועולה במוחנו, למרות שייתכן וחשבנו על הבעיה כבר זמן רב ולא מצאנו לה פתרון עד כה.

> אנו נאלצים במקרים אלו לתאר מילולית את הנושא לרוב תוך כדי שאנו מקבלים אפיק משוב נוסף מקריאת הקוד עצמו

בדומה להדרכה של אדם אחר המאלצת אותנו להעמיק בהבנת החומר בכדי שנוכל לתאר אותו לאחר, כך ואף יותר אנו נאלצים במקרים אלו לתאר מילולית את הנושא לרוב תוך כדי שאנו מקבלים אפיק משוב נוסף מקריאת הקוד עצמו.

לרוב זה יקרה כאשר נבקש להתייעץ עם חבר ותוך כדי ההסבר נזהה בעצמנו את נקודות החולשה - אני מאמין כי מי מכם שניסה לתחקר בעיה או לכתוב אוטומציה נתקל כבר בתופעה זו.

אך מה נעשה שלא תמיד יש לידנו חבר פנוי ולא נרצה כל הזמן להיאלץ לקום ולחפש מישהו לדבר עמו, ומאידך איננו רוצים לדבר בקול לעצמנו - כי רובנו מעדיפים לשמור על פאסון ולא נרצה כי תדמיתנו תהיה כ- של משוגעים המדברים לעצמם, ולכן בשנים האחרונות בעקבות הספר The Pragmatic Programmer אשר תיאר בין השאר תופעה זו, המושג ברווז גומי (Rubber Duck) הפך לנפוץ - ועל יותר ויותר שולחנות של תכנתים ניתן למצוא בובת ברווז או אחרת המשמשת לצורך זה.

עיקרון זה בא לידי ביטוי בפורמט שונה במקצת במקומות בהם נהוג לבצע סקירת קוד (Review) מסוג Walk Through בו התכנת מציג את הקוד שלו באופן יותר או פחות מפורט לר"צ שלו או לתכנת מקביל - מעבר למשוב המתקבל מאופן חשיבה שונה ותשומת לב לפרטים שונים בין התכנתים הללו, לעיתים קרובות זהו התכנת המציג את הקוד שלו שמזהה פתאום חולשות אשר קודם נעלמו מעיניו - והרי הוא המתאים ביותר לזיהוי חולשות נסתרות מכיוון שהוא מכיר הכי טוב את הקוד והקשרים הפנימיים בין חלקיו.

> לעיתים קרובות – בעיקר בעת ניתוח תופעות משותף בינינו לבין התכנתים – אנו הבודקים משמשים מעין ברווז גומי

מניסיוני לעיתים קרובות - בעיקר בעת ניתוח תופעות משותף בינינו לבין התכנתים - אנו הבודקים משמשים מעין ברווז גומי - במיוחד אם אנו מראים התעניינות בפעולות אשר הם מבצעים במהלך החיפוש אחר מקור הבעיה, ושואלים שאלות לגבי מבנה הקוד (לכך יש יתרונות נוספים מבחינתנו המתבטאים בעיקר בהרחבת והעמקת הידע שלנו במוצר ומאפשרים לנו להעלות רעיונות בדיקה חדשים).

בעבר כאשר התכנתים והבודקים שכנו תחת אותה קורת גג ואף באותה קומה - התכנתים נהגו להגיע לשולחננו בין אם בחדר ובין אם במעבדה לצורך חקירת הנושא והפרייה הדדית זו התבצעה באופן יותר טבעי. לצערנו עם המעבר לחברות גלובליות אבדה במעט האפשרות למפגשים בלתי אמצעיים אלו - ובשנים האחרונות חוצץ בינינו חיבור מחשב מרוחק וטלפון - מה שמגביל את היקף השיחות - קשה ליצור הווי משותף כאשר איננו חווים חלק מהתגובות הגופניות של בן שיחנו - והוא גם לא נחשף למבטי התמיהה שאנו מעלים לחלק מאמירותיו וכך השיחה מצומצמת יותר.

אך המעבר לצוותי עבודה מצומצמים בחלק מן השיטות האג'יליות חוזר ומשפר האינטראקציות במידת מה.

אז כן - אולי חלקכם נעלב תחילה מקריאת כותרת הפוסט הזה - "מה? זה כל מה שאנו בשביל התכנתים?" - אך אם תפנימו כי אין בכך כל פסול - תוכלו לא רק לשמש חלופה יעילה יותר לברווז הגומי - אלא אף להפיק ממפגשים אלו ידע חשוב נוסף.

חומר קריאה נוסף בנושא:

https://en.wikipedia.org/wiki/Rubber_duck_debugging

http://www.jrothman.com/mpd/thinking/2016/06/tell-your-problems-to-the-duck

יישום מעבר ארגוני גדול לאג'ייל – LARGE SCALE AGILE

פרק 1

יישום אג'ילי רחב היקף בארגונים, אינו ממוקד טכנולוגיה. זוהי צורת חשיבה שונה לחלוטין.
ארגונים יכולים ליישם מעבר לאג'ייל בשלוש רמות: רמת הפרויקט – הקלה יותר לביצוע, הפורטפוליו – שהינה יותר קשה להשגה ומחייבת ראייה רחבה יותר, ורמת הארגון – שמחייבת חשיבה שונה ואחרת של כלל התהליכים והמודל התפעולי של החברה.
אבל כששינוי ומעבר כזה קורה, היתרונות רבים ביותר, והיעילות גבוהה בהרבה.

> היישום של ארגון לאג'ייל הינו אבולוציה ולא רבולוציה

האתגרים והקשיים

ארגונים חווים אתגרים רבים ובעיות בהתמודדות בשוק של היום, דבר שגורם להם לחשוב על שינוי ומעבר ארגוני לשיטות אג'יליות

- זמן ארוך מידי מזמן החשיבה על feature ועד יציאתו לשטח (Production)
- דרישות משתנות, עם זמן ארוך מדי למחזור הפיתוח שלהן
- קושי מהותי בשינוי עדיפויות בזמן ומהלך ביצוע מחזור חיים של גרסה
- תקורה בתהליך, בניהול ויעילות נמוכה בעקבות זאת
- קושי בתכנון - תכנון זמן ארוך מדי, ללא יכולת גמישות לשינויים, התחייבות ל- 100% מראש
- מדידה - קושי במדידת גוף ה- R&D ותוצריו/ביצועיו
- WIP גבוה (Work in Progress)
- מעגלי משוב ארוכים מדי (לא יעילים לצורכי שינוי ושיפור מיידי)
- חוסר שקיפות בהתקדמות הפרויקט
- בעיות איכות שנובעות מתהליכים לא מסודרים והגעה מאוחרת של קוד לבדיקה

כתוצאה מכך: פספוס הזדמנויות שונות בשוק

> ניהול סיכונים וחשיבה על פתרונות, הינה משימה חשובה בתוך כלל התהליך הזה, וצריכה להיות מנוהלת תדירות.

הארגון הטיפוסי

הארגון הטיפוסי שמתחיל לבחון יישום ומעבר לאג'ייל ארגוני, כולל מספר מאפיינים עיקריים/משותפים, די דומים

- מאות עובדים ומהנדסים בפרויקטים
- תפעול מורכב, עם תשתית מורכבת מאוד
- מספר צוותים גדול מאוד (עשרות ויותר), מספר מחלקות רב - עם דרישת תאום מדוקדק וגבוה ביניהם
- מוצרים מורכבים
- משרתים שוק שעד עתה יכול היה לקבל גרסה אחת בשנה, אולי 2, וכעת חייב להציג ללקוחותיו חידושים מספר פעמים בשנה
- אי קיום מתודולוגיה סדורה ושיטתית בתחומי הפיתוח, הבדיקות, המוצר, התפעול, הניהול הפרויקטלי, ניהול Program ואבטחת האיכות

שלבים אפשריים למעבר ארגוני בטוח יותר לאג'ייל

מניסיוני, ארגונים זקוקים למתווה ברור, דרך סדורה ושיטתית למעבר.
שלבים ברורים למעבר יכולים לכלול:

הגדרת הצורך העסקי, הסיבות והיתרונות למעבר לאג'ייל
הבנת נקודת הפתיחה -שהינה שונה בין ארגון לארגון - הערכה ראשונית של: התרבות הארגונית ותמיכתה במעבר לאג'ייל, הבנת התנהגות הגופים השונים בתפעול המערך הכולל, הסביבה הטכנית טכנולוגית והתשתית לקראת המעבר, רשימת בעיות ו"צרות חולות" בארגון כבר כיום עם ההשפעה שלהם כיום (מתוך מטרה למזער אותם או להעלימם לאחר המעבר).

אלון לינצקי, מנכ"ל "QualityWize™", מייסד פורום סיגיסט ישראל, ומייסד-שותף של ITCB®

בניית Business Case למעבר
בניית צורך עסקי ברור, מנותח ומוגדר, עירוב מנהלים מכל הדרגים בבנייה ובדיון כולל מנהלים בכירים מאוד, בניית צוות חשיבה מתאים להובלת הדיון והמטווה העסקי (שיוביל את ה-Gap Analysis), פרסום ותקשור האתגרים והבעיות והצגת המתווה העסקי שבונים לרוחב הארגון (מעיין Road-show לצורכי דיון, התלבטות, חקירה, ניתוח וקבלת החלטות).

בניית מודל עסקי תפעולי חדש לארגון שמתאים לצרכיו
שימוש בפורום חשיבה שנוצר להובלת השינוי ובניית מודל עסקי חדש, תוך חשיבה על הסיכונים בדרך ודרכי טיפול בהם. עירוב של מנהלים רבים בדיון, ומספר מצומצם של מנהלים בהחלטה, בתהליך מוסכם מראש ולוח זמנים ברור. רצוי ליווי משמעותי של התהליך על ידי מומחים מהתחום, שייעצו לצוות החשיבה לאורך כל הדרך (תהליך, שלבים, תוצרי ביניים, וכו').

קביעת תוכנית עבודה High Level שמתפתחת עם הזמן
בניית תוכנית רב שלבית, במבט כללי, שמתפתחת עם הזמן ומתעדכנת כתוצר של צוות ההובלה והחשיבה על המעבר. הבנת התוצרים של כל שלב- מי מעורב בו, מי מקבל את ההחלטות, מהם הקלטים שמצופים בשלב זה בתהליך, ומהם הפלטים ואיך הם מתקשרים ומתחברים לשלב הבא.
במהלך ביצוע כל שלב, ישנה חשיבה מסודרת על התכנון של השלב הבא, תוך לקיחת התוצאות בשטח משלב נוכחי, הראייה ארוכת הטווח והמתווה העסקי שהוגדר קדם לכן.

יישום
יישום התוכנית וכל שלב בפרויקט הפיילוט או בכלל הפרויקטים שנבחרו למעבר.

כאן ישנה אפשרות לגישה שונה:

1. **גישת הפיילוט** – ביצוע פיילוט במספר פרויקטים, ומעבר אג'ילי שלהם תוך בחינת התוצאות וליווי מתמיד וסיוע לפרויקטים הללו לעבור אותו, ואז יישום כלל ארגוני.
2. **גישת הארגון** – יישום קפדני של כלל המעבר האג'ילי בחטיבה/ארגון. כאן יש לחשוב על תמיכה מאסיבית יותר וסיוע רב היקף יותר לצוותים/מחלקות/חטיבות בארגון לעבור.

ביצוע איסוף משובים במעגלי משוב קצרים וארוכים, לרוחב ולאורך הארגון/הפרויקטים, ניתוח המידע, ושימוש בו לצורכי התקדמות הלאה.
היישום של ארגון לאג'ייל הינו אבולוציה ולא רבולוציה, ולכן שינויים רבים וקטנים, יובילו את מרבית זמן היישום. יש לשים לב אליהם, לנהל אותם, ולהחצין אותם, לשנות – ולהתקדם הלאה. הנקודה הבולטת ביותר בשינויים מהסוג הזה הינם ההתנגדויות העולות וצצות בדרך. אנשים רגילים לדרך מסוימת ולתרבות והתנהגות מסוימת, והמעבר מחייב שינויים נרחבים (לעיתים) בתרבות והתנהגויות אלו, שמן הסתם מעלים התנגדויות אצל העובדים והמנהלים, אותם יש לנהל.
בגיליון הבא, תוכלו לקרוא על פעולות קריטיות להצלחת היישום של המעבר האג'ילי בארגון, אותם מבצעים במקביל לשלבי היישום והשיפור.

בדיקות חוקרות (ET) במערכות ללא ממשק גרפי | שמואל גרשון

תקציר

קל להבין את הרעיון של חקירה וגילוי (exploration) על ידי בדיקות, כשמדובר בתוכנה שיש לה ממשק משתמש גרפי. במקרים אלו החקירה והגילוי הן תוצאות ברורות של מעורבות החושים, והתוכנות שאנו פוגשים יום יום מציגות לבודקים הנחייה ויזואלית וקלטים המזוהים בקלות.

אבל כיצד חוקרים מערכות משובצות (embedded) המשולבות בחומרה, מרוחקות (remote), או מערכות שירות כגון web services שיושבות בתוך מערכות אחרות, הרחק מעיניהם ומידיהם של הבודקים? כיצד פותרים את הבעיה על פיה בדיקת ממשקים מסוג אלו דורשת פיתוח של קוד בדיקה, דבר שקשה לעשותו באותו הקצב המהיר של בדוק-למד-שנה-בדוק שבו נעשת בדיקה חוקרת של מערכות שיש להן GUI?

אנו יכולים לפתור בעיה זו באמצעות חדשנות בשיטת הבדיקה ובמכשירי הבדיקה. ישנם מספר דפוסי יסוד של חקירה הניתנים ליישום על מערכות שאינן-GUI: זיהוי הדרכים להזנת קלט ולגרימת שינויים במערכת; שליטה על הבדיקות ועל פירושן; ויצירת מודלים עבור מכשירי בדיקת התוכנה באופן שמאפשר אינטראקציה מהירה יותר.

בגיליון הקודם סקרנו את שלושת העקרונות הראשוניים מתוך השישה:

1. **זהה את המַבְדֵקים**
2. **טפח פעלים**
3. **הישאר בשליטה**

וכעת נסקור את השלושה הנוספים:

4. **הוו מתונים בדין**
5. **סלק את המתווך**
6. Tradittore, Traduttore.

שמואל גרשון

הוא בודק עם ניסיון בתוכנה וקושחה.
ביום יום הוא בודק תוכנה ומתאם כלים ואסטרטגיות עם צוות של גיבורי-על, מלמד בדיקות ועוזר לחברים, וחוזר חלילה במוצר הבא.
שמואל עבד בחברות גדולות, קטנות וכעצמאי, מדרום אמריקה ועד ישראל. למרות שהתחיל את הקריירה בתור מתכנת, גילה שבדיקות תוכנה זה כיף כפליים. עם הזמן השתכנע שהגורם החשוב ביותר במרדף שלנו אחרי איכות הוא האנשים ויחסם, ולא כלים או טכנולוגיות.
אפשר למצוא אותו בכנסים (SIGiST, EuroSTAR, AgilePractitioners, ועוד...), לקרוא מרעיונותיו ב-e ו-, ולהוריד תוכנה חינמית לרישום רשומות בבדיקות ב-e

3.4 הוו מתונים בדין[1]

> **מהות הבדיקות היא בהפתעות שהם חושפות בעת ביצועי הבדיקות**

להיות מתון בדין – בשיפוט – פירושו לקחת בחשבון את הצורך שבן אדם אנושי יחליט מהו באג ומהי תכונה.

לרוב אנו רואים מערכות בדיקה אוטונומיות שבהן בדיקות (לכאורה) מחליטות מתי הבדיקה "עברה" (לא נתגלה שום באג) ומתי היא נכשלה (זוהה באג שהוגדר מראש). אם כל מאמץ הבדיקה כולו מוכל בתוך מערכת בדיקות אוטומטית כזו, נפספס בעיות רבות שלא ניתן להגדיר מראש אך בני-אדם היו מזהים (הטבע האנושי מוגל לזהות בעיות, חוסר-עקביות וסיכונים; בודקים מיומנים משפרים עם הזמן יכולת זו).

נוסף על כך, מהות הבדיקה היא בהפתעות שמוצאים במהלך ביצוע הבדיקות. הרי בעולם היפותטי שבו אפשר לנבא את כל הבעיות והפגמים כדי להפוך את הבדיקות עבורם לאוטומטיות, מוטב היה לתקן אותם ישירות בקוד המוצר במקום לבדוק אם הם קיימים בשלב מאוחר.

לפיכך, יש ערך רב שתהיה לנו מערכת בדיקות שמאפשרת לבודק לשפוט ולפרש את התוצאות (בשימוש בידיעותיו מַבְדֵקים ומפַעלים.

הסברנו הקודם על פעולת המשוטט של NASA מבהיר שהמהנדס הוא זה שנמצא בתפקיד ניתוח תוצאות והסקת מסקנות. משימה זו לא הואצלה למכונה.

א נ י לפעמים מעיר בהומור שאם NASA היו הופכים את פירוש התוצאות לאוטומטי (כמו שאנו לפעמים מנסים לעשות עם בדיקות ובאגים), הם היו מתכתים את המשוטט לחפש את הצורה הידועה של חייזר: ישויות כחושות בירוק-אפור, עם ראש גדול ועיניים גדולות. המשוטט היה משוטט על פני השטח של המאדים ומוצא גוף שהוא גדול, שחור לבן, בעל 4 רגליים ועושה 'מוּ', והיה אוטומטית דוחה אותה כ"לא חייזר, זו רק פרה".

1 שם העיקרון מקורו בפרקי אבות, בו מופיע כעצה הראשונה הניתנת על-ידי אנשי הכנסת הגדולה לדורות הבאים של שופטים, ומדגיש שיחד עם הכללים לשפיטה יש מקום לשיקול-דעת אנושי.

עבור מערכות בדיקה עם בדיקות אוטומטיות, ההרגשה שלך לגבי כמות המאמץ שמושקע בניתוח התראות שווא או בהוספת בדיקות חדשות עבור באגים שנמצאו בשלב מתקדם, יכולה לתת אינדיקציה על רמת השיפוט עליה אתה מוותר ועל רמת השיפוט שמערכת האוטומציה משאירה בידך.

שיטה אחת ליצור מכשיר בדיקה שמשאיר את השיפוט בידיים אנושיות היא לבנות מכשיר שאינו ממש "בדיקות", אלא רק את הפונקציות שמסייעות לבדיקה (למשל, קריאה לכל פונקציה ב-API והצגת הפלט באופן נוח לקריאה) ואז המכשיר הוא ספציפי לטכנולוגיה שעמה הוא נמצא באינטראקציה, ובו-בזמן הוא גנרי מבחינת הבדיקות שאפשר לעשות בעזרתו.

רבים ממכשירי הבדיקה בהם אנו משתמשים הינם כבר בלתי-שיפוטיים, וכדאי להעריך עד כמה המכשירים שהנך מייצר מתקדמים על סקלת השיפוט. לדוגמה, בתחום דפי האינטרנט, מכשיר ה-BrowserStack הפופולרי מאפשר לצפות בדף אינטרנט כפי שיוצג בדפדפנים שונים, מערכות הפעלה שונות

וקונפיגורציות שונות, אבל לבסוף הוא מציג את תוצאות הדף לשם שיפוטו של המשתמש. מכשירים אחרים, כמו W3C Validator הינם שיפוטיים ומספקים תוצאה של עובר/נכשל (יחד עם רשימת בעיות צמודה). מכשיר הבדיקה שלך יכול לנווט במסלול ביניים שבו הוא לא נוקט בשיפוט סופי, אלא ממכן את איסוף המידע ומספק התרעות או אזהרות על התנהגויות חשודות כדי לעזור לבודק בקבלת החלטות.

3.5 סלק את המתווך

סילוק המתווך פירושו להסיר את כל מה שנמצא בין הפעולה שמופעלת על המוצר לבין התגובה שמוחזרת ממנו, ואפילו יותר מזה – להסיר את כל מה שנמצא בין העלאת רעיון בדיקה לבין היכולת ליישם אותו מייד בלי לאבד את חוט המחשבה.

בפעילות כמו בדיקות, שבה הבודק עסוק בעיון בתוצאות, בהשוואת באגים ובקישור התנהגות המערכת למודל המנטלי של המוצר, כמות המשתנים שהבודק מחזיק בראש ברגע נתון גדולה, והפרעות קוגניטיביות יכולות לשבור בדיקה שאלמלא כן הייתה בדיקה מעולה[2].

> **הפרעות קוגניטיביות יכולות לפגוע בבדיקה שהיא נהדרת מכל בחינה אחרת**

Zork שומר על חיבור ישיר בין השחקן לבין המשחק, וזאת לא מפני שזהו משחק פשוט: רוב המשחקים מכל דור שהוא עושים מאמץ שיהיה להם מעגל פקודות ישיר ומהיר, מפני שבלי זה השחקן מאבד את מעורבותו הקוגניטיבית במשחק (בדומה מאד לבודק בפעילות בדיקה)[3].

2 Testing on the Age of Distraction". Zeger Van Hese, EuroStar 2013" http://bit.ly/AgeDstrctn

3 יש הרבה מקבילות בין הסביבות האופטימליות למשחקים ולעבודה. מחקר אחד מגדיר את המושג gameplay כ"אינטראקציה עם תכנון משחקי תוך ביצוע של משימות קוגניטיביות, עם מגוון רגשות שעולים מאלמנטים שונים של מוטיבציה, ביצוע משימה והשלמת משימה או קישורה אליהם"; בדומה לאופן בו נוכל להגדיר

בדיקות חוקרות (ET) במערכות ללא ממשק גרפי | שמואל גרשון

בדוגמה של NASA יש מרחק רב יותר (תרתי משמע) בין פעולה לתגובה, מפני שהתשובה לכל בדיקה או פקודה חדשה לוקחת מספר דקות או שעות, מעֵבר לאינספור המנגנונים המורכבים והמסובכים שסביר שיהיו מעורבים בבדיקה.

כשאין שום איש-ביניים (משמעותי) – ההרגשה היא כאילו אתה יכול להפעיל את המוצר בזמן קצר ו/או במספר צעדים. כאשר אתה רוצה לעשות בדיקה טיפה אחרת מזו שתוכנה מראש, המאמץ הטכני של ביצוע השינוי מסיט את הבודק מפעילות הבדיקה ומנתק את חוט המחשבה שלו. דוגמה לכך היא בדיקה אוטומטית שפותחת 10 חיבורי רשת וצריכה להיכתב מחדש ולעבור קומפילציה מחדש על מנת לפתוח 15 חיבורי רשת.

הזכרנו קומפילציה. גם Compiler (מהדר הקורא את כל התוכנית ורק אז מפעיל אותה כולה) היא דוגמה למתווך בהשוואה לשפות שעוברות interpreter הקורא ומפעיל שורת-קוד אחר שורת-קוד. תכנות בשפה שעוברת קומפילציה דורש צעדים רבים מהרגע שמשנים את הקוד ועד לרגע בו הוא מופעל (cleaning, linking, compiling, waiting, building...), לעומת שפות עם interpreter שהן מוכנות להפעלה מיד כאשר נשמר השינוי בקוד.

בעת בניית מערכת מכשירי בדיקה, כדאי לנסות לסלק את כל הצעדים הבלתי נחוצים. לפעמים ניתן להשיג זאת על-ידי פישוט השיטה, אך לרוב זה יושג על-ידי החבאת צעדים מאחורי הקלעים ועשייתם לאוטומטיים (רמת האוטונומיה הבריאה שהוזכרה קודם לכן). יוריסטיקה טובה אחת כדי לנסות ולשמור על מצב עבודה ללא-מתווך מביאה בחשבון את העקרונות הראשונים: אם אתה מסוגל לאסוף את כל (או רוב) הפעלים והמַבְדֵּקים לכלי או פאנל אחד, סביר שהתוצאה הסופית תדרוש רק מעט תיווך – במיוחד מפני שאילו הפקודה הייתה דורשת צעדי ביניים, היינו צריכים להחביא אותה בעת הוספת הפועל לפאנל המקבץ.

כדאי להשיג מיידיות-מדומה על-ידי החבאת גורמי ביניים; הם נמצאים, אבל הבודק לא צריך לחשוב עליהם.

3.6 Traduttore, Traditore [1]

> **המטרה היא יצירת מערכת אוטומטית עבור בדיקות ידניות**

תרגום חופשי של פתגם איטלקי זה הוא "כל מתרגם הוא בוגד". כאשר יוצרים מימוש חדש למשהו שקיים כבר, הצורה החדשה אינה זהה למקור, לא משנה עד כמה היא נראית נאמנה לו.

משפט זה בא להזהיר, שמערכת בדיקה עם פעלים ומַבְדֵּקים שנראית מיידית ובשליטה, היא תרגום של המערכת המקורית שנבדקת; מודל שנבנה לשם הנוחות שלנו.

המעורבות שנוצרה על-ידי תגובה מהירה ושליטה מלאה היא כזו, שבמחשבתנו מוחלף המוצר-האמיתי-הבלתי-נראה בייצוג שלו, מודל שאנו עצמנו יצרנו[2]. בבדיקות זה יכול להיות מסוכן, כי לבסוף ייתכן שנעסוק במגבלות המודל וביכולותיו, במקום בתוכנה המקורית, בלי שאפילו נשים לב לכך.

אפקט התרגום היה ברור בדוגמת משחקי Zork, שחקן Zork יוצר מפה במחשבתו ולרוב גם על נייר, מפה שאיננה העולם של Zork (שהוא הפשטה דיגיטלית), אלא ייצוג פיזיקלי (של הפשטה זו). העובדה ששני שחקנים שונים המשרטטים מפות זהות בדיוקן יפיקו תוצאות שונות מחזקת נקודה זו.

ניתן להבחין באפקט זה בהרבה מהתוכנות בהן אנו משתמשים. נראה שההתייחסות ל- Mind Maps , לדוגמה, היא כאל ביטוי להבנה של אדם. אבל מפות חשיבה מוגבלות, לפחות על-ידי יכולתו של האדם לבחון בקפדנות את מחשבתו, על-ידי מיומנותו בהבעת מחשבותיו, על-ידי התווך המוצג (פורמט flat Mind Map), ועל-ידי אפליקציית Mind Map בה משתמשים. גם אחרי ההבנה הזו Mind Maps יהיו שימושיות כמקודם, אבל תובנה זו עושה אותן בטוחות יותר לשימוש.

כאשר אתה מארגן את שיטות הבדיקה שלך, אין שום אפשרות להימנע מצורות תרגום כלשהן3 - תוכנה היא הפשטה לא-מוחשית, וכל אינטראקציה עמה דורשת תרגום.

כל פעילות בדיקה מבוססת על מודלים[4], ובניית מודלים לתוכנה הנבדקת באמצעות תוכנה אחרת היא דרך טובה לחשוף את המודל המנטלי שלך לבחינה קפדנית ולהוכחה אמפירית, עם היתרון הנוסף שהידע בנקודה זו בר-הפעלה וניתן לעדכון5 בקלות.

3.7 סיכום ביניים

תיארנו שישה עקרונות שישפרו את יכולותיו של צוות לחקור ולגלות אפילו מוצרים שקשה להגיע אליהם ולראות אותם.

השתמשתי באותן שש נקודות במסמך אסטרטגיית בדיקות שכתבתי במסגרת עבודתי, והדרך הטובה ביותר שמצאנו כדי להעביר אותן לצוות במהלך מאמצי תכנות כלי הבדיקה שלנו הייתה המוטו: "יצירת מערכת אוטומטית עבור בדיקות ידניות", שחזרנו עליו כמנטרה, בניגוד לנטייה הטבעית ליצור מערכת בדיקות אוטומטיות עצמאית.

4 יישום באופן פרקטי

ביצירת מערכת אוטומטית עבור בדיקות ידניות יש חלקים קלים וחלקים קשים. חלק מהקשיים הם טכניים, אך חלקם נוגעים לדינמיקה האנושית בצוות.

4.1 יישום במקום העבודה שלך

במידה מסוימת, מימוש ארגז כלים כזה פירושו לעבוד עם מה שיש לך כרגע: סיווג פעלים קיימים, מיפוי מַבְדֵּקים קיימים, ארגון המַבְדֵּקים בצורה שניתנת לשליטה, וזכירת המגבלות שבארגז הכלים. שלב זה הוא יחסית פשוט, מפני שפחות מורגשת השפעת השינוי על התוכניות אותן אתה מתכנן, על לוחות הזמנים ועל המוצרים שתפיק.

אבל, אחרי שלב הסיווג הזה, כאשר זה מגיע לשיפור היכולות בכל אחד מהעקרונות, ייתכן שיהיה צורך בשינויים במוצר (כדי לשפר מַבְדֵּקים ופעלים שאר והיבטי הנבדקות של התוכנה) ובתהליך (כדי לתאר עבודה עם מכשירים לא-שיפוטיים).

מלבד תגובה שלילית לשינויים טכניים, הצוות והמנהלים אולי נזהרים מלהשקיע מאמצים במכשירי תכנות ואוטומציה שבראייה ראשונה אינם מממשים בדיקות בעצמם.

אבל סביבת חקירה–וגילוי מכניסה את צוות הבדיקות לכושר ליצירת פרוצדורות אוטומטיות עבור בדיקות ספציפיות: הצוות נמצא בעמדה ייחודית ללימוד על הסביבה, המוצר והבאגים, לבחירת הצעדים הכדאיים ביותר והאופן הטוב ביותר לכתיבת אוטומציה. כפי שנראה, מערכת שלומדת מעקרונות אלה היא כבר שני שלישים מהדרך אל מערכת אוטומציה חסינה.

4.2 השכבות באוטומציית הבדיקה

התחלת המערכת עם פעלים גנריים ויכולות מבדקיות מספקת שכבת אינטראקציה בה ניתן להשתמש הן עבור בדיקות ידניות והן עבור אוטומציה של צעדים.

מפרט השכבות הזה הוא עוצמתי: דפוס לא יעיל של בניית מערכות אוטומטיות הוא התחלה ברשימת בדיקות, שמובילה למאמץ לכתוב אוטומציה כל תסריטי הבדיקה (test flows) בזה אחר זה, שלבסוף גורם למבנה ללא שכבות, שבו צעדי בדיקה ופונקציות טכנולוגיות משולבים–ביניהם ויש ביניהם תלות–הדדית. כאשר רשימת הבדיקות מתרחבת, זה נעשה מונוליטי (מקשה אחת) ולא גמיש. דרך יעילה יותר

ציפיות במקום העבודה.

1 הפתגם באיטלקית הובא לתשומת ליבי על-ידי זיכרונותיו של Gregory Rabassa, שבילה את חייו בתרגום ספרים לטינו-אמריקאים כגון Gabriel Garcia Marques. Jorge Amado, ו-Julio Cortazar. בהתייחס לפתגם ולמלאכת התרגום, הוא הכתיר את זיכרונותיו בכותרת "If This Be Treason". באופן מפתיע, על אף מסירותו לתרגום מדויק, Rabassa מסביר שאפילו תרגומים מדוייקים מפספסים את המשמעויות והכוונות המקוריות: "שפות הן תוצרי תרבות", "מה ש'כלב' מציין עבורי, הוא כנראה שונה ממה ש'cao' מרמז עבור Anotonio Lobo Antunes".

2 Wikipedia on the relationship between an object and its representation http://bit.ly/Rprsnttn

3 הרעיון של מודלים כבגידה מופיע גם בציור המפורסם של Magritte "The Treachery of Images". ציור המקטרת אינו מקטרת (שאם לא כן היינו יכולים לפטפט אותה בטבק ולשאוף ממנה). מושא הבגידה מתאים, מפני שמשתמע ממנו שיש אמון עיוור בייצוג עד שאנו שמים לב למגבלות שלו. זה לא הופך את המודל ללא-שימושי; אם אנו מודעים למגבלה שלו, תיאור המקטרת יכול להיות שימושי כדי לדון בענייני מקטרות, כדי ללמוד אודות חלקי ומנהגי מקטרות, או שימושים אחרים שניתן למצוא עבור ציורי מקטרות (כגון אסתטיקה או טענות על meta-language - השפה בה משתמשים כדי לדון בנושא "שפה".

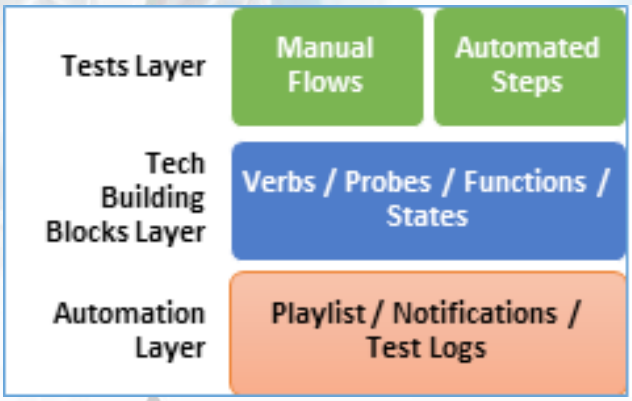
4. "The Tester's Pocket Book". Paul Gerrard 2009 http://bit.ly/TstrPcktBk

5 "The Laws of Software Processes". Phillip Armour, 2003 http://bit.ly/LawsSw
"Value: Why We Have it Backwards" Shmuel Gershon, Goto Conference 2013 http://bit.ly/SwValue

בדיקות חוקרות (ET) במערכות ללא ממשק גרפי | שמואל גרשון

לארגן אוטומציה היא על-ידי הפרדת צרכים שונים לשכבות שונות[1]:

- שכבה בסיסית שמספקת שירותים כלליים (ריצה של רצף צעדים, יצירת יומן, איסוף תוצאות)
- שכבה שנייה שמספקת פעולות בטכנולוגיה הספציפית של המוצר (וממשת פעלים, מַבְדֵקים, וכו')
- ושכבה שלישית לשם צעדי בדיקה "מורכבים" מאבני הבניין של שכבת הטכנולוגיה (איור 6).



איור 6

שים לב שאם הקפדנו על ששת העקרונות הרי שברגע שאתה מוכן לחקור ולבדוק באופן ידני, הטכנולוגיה כבר מוכנה לשימוש גם בתסריטי אוטומציה. שיטה זו חסינה באופן מרענן: בעת שממשקי המוצר מתפתחים, אבני הבניין שבשכבה השנייה מתוכנתים מחדש, בעוד שיש סיכויים טובים שלא יהיה צורך בשינוי בתסריטי הבדיקות; וכאשר כן יש צורך לשנות צעדי תסריט, לא צפוי שום שינוי או השפעה על התסריטים האחרים או על שכבת הטכנולוגיה2.

4.3 אוטופיה בפרקטיקה

בקריירה שלי הייתי עד לבדיקות של סוגים שונים של מוצרים, החל ממאקרו-ים ב-Excel ועד לתוכנה embedded בלתי נראית, והייתי חלק ממאמצי התכנות של מוצרים אלה או פיתוח מכשירי הבדיקה שלהם. כולם (גם אלו שהצליחו וגם אלו שנכשלו) מעידים על ההשפעה של ששת העקרונות האלה בצורה בה הבדיקות נערכו: כאשר מערכת הבדיקה סיפקה מעט פעלים, מַבְדֵקים ושליטה, החקירה-והגילוי היו פחות שוטפים. רובם לא הוסיפו שום קלטים (פעלים) או פלטים (מַבְדֵקים) נוספים מעבר לפונקציות המקוריות של המוצר, והן לא טיפלו טיפול מיוחד שהניב שליטה ושיפוט. אבל חלקן קיבלו ברכה חקירה וגילוי למרות טבעם המוטבע והבלתי נראה.

אני גאה במיוחד על שני מאמצים ליצירת מכשירי בדיקה (אף שלא השתתפתי בהבאת הרעיון או בתכנות שלהם). במאמצים אלה המכשירים תומכים בחקירה וגילוי (ובאוטומציה) על ידי מילוי ששת העקרונות שלמדנו – אם בכוונה או באופן לא מודע.

אחד מהם הוא מכשיר בדיקה למערכת embedded, שהפרוטוקול שלה מממש אוסף של מחלקות ואובייקטים שקשורים מאוד בינם לבין עצמם. הכלי מציג את כל מופעי המחלקות יחד עם הנתונים הפנימיים שלהם, ומציג באופן חזותי את הקשרים עם אובייקטים אחרים. הכלי מאפשר שינוי של כל ערך שהוא, ומעדכן את התצוגה כדי להראות את האפקט של כל פעולה ופעולה.

השני הוא מכשיר המשמש לבדיקת בקר-רשת (איור 7). מעטות המערכות שעבדתי עליהן שהיו כה מרוחקות וכה קשות לאינטראקציה כמו רכיב זה, אשר לא זו בלבד שהוא מוגבל על-ידי מיומנויות שיחה גרועות של חומרה (ביטים, אותות, ורגיסטרים), אלא שגם פעל תחת

1 Layered Architecture, conversations with Bat-Sheva Magen, Jerusalem 2012

Wikipedia on Layered Architectures http://bit.ly/MultiTier

2 זה קצת נאיבי להציג שכבות אלה כבלתי-מחוברות ובלתי-תלויות ביניהן לחלוטין. לבסוף, במציאות יתווספו קשרים והנחות בסיסיות ביניהם, והממשקים יהיו מורכבים. (בדומה לכך מערכת בדיקה ששולטת במערכת בעלת מספר רב של מצבים, תיטה עם הזמן להיות בעלת אותו מספר מצבים כמו המערכת המבוקרת).

> **חקירה וגילוי מאפשרים לצוות לבחור את הצעדים החשובים ביותר לאוטומציה**

מחסום של תוכנה ודרייברים. מכשיר הבדיקה, כאילו בקסם, מייצג חזותית את כל הרגיסטרים והאותות הרלוונטיים, ומגיע אליהם דרך דרייבר שנבנה במיוחד לשם יכולת-הבדיקה. משתנים אלה נקראים בזמן אמת, וניתן להגדיר ולשנות אותם בזמן אמת, בעוד ששאר הנתונים מתעדכנים בהתאם.

שתי הדוגמאות מנסות למקסם פעלים ומַבְדֵקים, לא מממשות שום בדיקה (משאירות את השיפוט והשליטה לבודק) ומספקות תגובות מהירות. שתי הדוגמאות גורמות להרגיש כאילו הן מחבקות את המוצר הנבדק בכזו התאמה תפורה, עד שהבודק יכול להתבלבל בין מודל המכשיר לבין המוצר האמיתי. שתי הדוגמאות מעצימות את הבודקים ומעודדות אותם לחקור כמו שמערכות ממשקי משתמש גרפיים היו מאפשרות – אם לא יותר מכך.

5. סיכום

הסיכום האפקטיבי ביותר של תזה זאת הוא שדרך יעילה לאפשר ולטפח חקירה וגילוי במערכות ללא גרפיקה היא ליצור" מערכת אוטומטית עבור בדיקות ידניות".

אם את/ה לוקח/ת על עצמך את האתגר הזה כדי לשפר את יכולות ומיומנויות החקירה והגילוי של הצוות שלך, אני ממליץ על שימוש במוטו הנ"ל כתזכורת במהלך שיחות, ועל שימוש בששת העקרונות כעזרי הנחיה במהלך העבודה.

אפילו אם אינך מצפה לעשות שום שינוי במכשירים או בשיטות, יש תועלת בידע ובמודעות למושגים שהוצגו כשלעצמם. איתם תהיי/ה מצויד/ת טוב יותר כדי להבין, להסביר ולהגן על הפרקטיקה הנוכחית שלך. ואם את/ה מתכוון/ת לעשות או לשנות תהליך בעבודה, עקרונות אלה יעזרו לך לסלול את הדרך ולתקשר ציפיות.

זה יהיה יותר מאשר אתגר טכני, זו תהיה הרפתקה אישית. חקירה וגילוי דורשים מאנשים ומצוותים העזה, והם מהווים זרז ליצירתיות, חידוש ואמ

בהצלחה ב

איור 7

3 "Experience Report on Exploratory Testing". Shmuel Gershon, Oredev 2011

מחפש צרות | מיכאל שטאל

הרגשת בטן

לפני מספר חודשים יצא לי לפגוש חבר ותיק שהוא גם חבר למקצוע בחתונה של מכר משותף. אני יודע שזה לא הזמן ולא המקום, אבל מפה לשם התחלנו לדבר על ענייני המקצוע, והוא חלק איתי את מצבו העגום:

"מחר יש פגישת שחרור לגרסה האחרונה של המוצר שלנו ואני עומד להמליץ שלא לשחרר את הגרסה. אבל יתעלמו ממני".

"ולמה ההמלצה שלא לשחרר?"

"כי יש לי הרגשת בטן ברורה שהגרסה האחרונה שקיבלנו - זו שרוצים לשחרר ללקוחות – דפוקה לגמרי. אבל מנהלי המוצר עובדים לפי רשימה של קריטריונים ליציאה (exit criteria) , המוצר עומד בקריטריונים והרגשת בטן זה לא data".

שאלתי מאיפה תחושת הבטן מגיעה, ואז קיבלתי את כל הסיפור:

לפני שבועיים הקבוצה שלו קיבלה Build לסבב בדיקות אחרון לפני שחרור רשמי של הגרסה. תיאורטית הכל היה אמור להיות חלק ונקי, אבל תיאוריה לחוד ומעשים לחוד: הבודקים גילו כבר ביום הראשון שני באגים קריטיים.

> **הרגשת בטן זה לא data**

צוות הפיתוח מיהר לתקן את הבאגים, ושחרר גירסה חדשה.

מכאן והלאה, במשך שבועיים, הדבר חזר על עצמו: צוות הבדיקות מצא באג או שניים קריטיים, וצוות הפיתוח שחרר גרסה חדשה.

והנה הבעייה שעמדה לפני בן שיחי:

"על פי קריטריוני היציאה שלנו, אין לשחרר מוצר שיש בו באגים קריטיים. אבל צוות הפיתוח מיהר לתקן כל באג שמצאנו, ולכן אין אף באג קריטי בקוד, נכון לגרסה ששוחררה היום. בשבועיים האחרונים קיבלתי 8 גרסאות שונות. נראה לך שהספקנו לבדוק את המוצר כמו שצריך? בהסתמך על איכות הגרסאות בשבועיים האחרונים ברור לי שגם בגרסה של היום מתחבאים לא מעט באגים קריטיים נוספים ואם יתנו לי עוד כמה ימי בדיקות אמצא אותם. אבל תאריך השחרור הוא מחר, ועל הנייר הכל תקין, גם אם הבטן שלי אומרת אחרת".

ובאמת – מה עושים?

במצב אידיאלי, הדרך הנכונה היא להיות פרואקטיביים ולהכניס לרשימת הקריטריונים ליציאה עוד כמה תנאים מעבר רק לספירה פשטנית של מספר הבאגים בגרסה ורמת החומרה שלהם. חייבים לבחון מגמות (trends): לבדוק מה קצב הבאגים החדשים שמדווחים ומה קצב סגירת הבאגים. מה הצפי לעוד באגים, בהסתמך על מספר הבדיקות שעוד נשאר להריץ. הנה דוגמאות אפשריות לקריטריונים אלה:

- מספר הבאגים הקריטיים שנפתחו בסבב הבדיקות האחרון קטן מ-X
- כמות הבאגים שנמצאו בשבועות האחרונים הולכת ויורדת כל שבוע בקצב Y
- לפחות X ימי בדיקות על הגרסה האחרונה ללא באגים קריטיים וללא יותר מ-Y באגים בחומרה "גבוהה".

וכו'.

חיפשתי קצת אם יש ברשת דוגמאות לרשימות של קריטריוני יציאה. לא ממש מצאתי. הנה אחד שחשבתי שהוא רלוונטי:

http://www.pmtraining.com.tw/article/5phase/Phase3-Approval/Completion%20Criteria,Checklists/sw_criteria.doc

כיוון שכך, החלטתי לייצר בעצמי דוגמה... את הוורסיה שלי אפשר להוריד מהקישור הבא: http://bit.ly/QRCTables

הוספת קריטריונים כאלה מראש הייתה מציבה שלט "עצור " ברור במקרה שלנו, ובן שיחי היה יכול ליהנות מהחתונה.

> **רשימת קריטריונים ליציאה צריכה להכיל עוד תנאים מעבר לספירה פשטנית של מספר הבאגים ורמת החומרה שלהם.**

אבל מה לעשות במקרה שמתואר כאן, שבו לא נקבעו מדדים כאלה מראש?

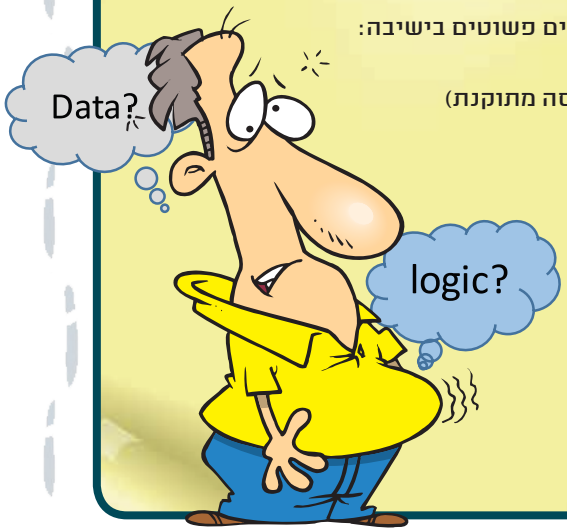
הצעתי לחבר "לתרגם" את הרגשת הבטן שלו לנתונים שהביאו להרגשה זו ולהציג אותם בצורה חדה וברורה: יותר מידי באגים קריטיים נמצאו בקוד ולא היה מספיק זמן לבדוק את הגרסה. אלו עובדות, לא תחושות.

המסר הוא: "זה המצב. על פי נתונים אלה אני מסיק שהגרסה לא באיכות מתאימה לשחרור". לא הרגשה, לא אינטואיציה, לא תחושה – אלא דעה מקצועית המבוססת על עובדות, הבנה בתהליכי פיתוח תוכנה, וניסיון מצטבר.

התקשרתי אחרי כמה שבועות לשמוע איך הלך. לא רע, מסתבר. חברי הציג שלושה גרפים פשוטים בישיבה:

- כמות הבאגים הקריטיים שנמצאו כל יום (2-1 כל יום, ללא מגמת שיפור)
- הזמן שהיה לבודקים לכל גירסה בשבועיים האחרונים (יום-יומיים עד שקיבלו גרסה מתוקנת)
- אחוז הבדיקות שהגרסאות האחרונות עברו (בין 20% ל-30% מהבדיקות)

זה הספיק לגרום לכולם להרגשת בטן לא טובה. תאריך השחרור נדחה.



קריטריונים ליציאה מעבר לספירת באגים (רשימה חלקית מאוד):

- אחוז הבדיקות שבוצע
- מגמות בכמות הבאגים המתגלים מידי שבוע
- תוצאות בדיקות ביצועים, מאמץ ואמינות
- רמת העדכנות של מסמכי המוצר
- רמת המוכנות של ארגוני התמיכה

לרשימה מלאה: http://bit.ly/QRCTables

איך להתחיל לעבוד עם בדיקות רשת (web) אוטומטיות | דייב הפנר

דייב הפנר

דייב הפנר מומחה בעל שם עולמי לפרויקט הקוד הפתוח סלניום ולבדיקות אוטומטיות.
הוא המחבר של "Elemental Selenium" – ניוזלטר שבועי העוסק בטיפים לסלניום, כתב את הספר The Selenium Guide, והיוצר של ChemistryKit (מסגרת עבודה לסלניום מסוג קוד פתוח). בנוסף, הוא המייסד של Selenium Hangout, והוא מרצה מבוקש בכנסים העוסקים בבדיקות קבלה אוטומטיות.

סדרת שילוב מתמשך (CONTINUOUS INTEGRATION) / חלק 1

בסדרת שלוש הכתבות הזו תלמדו כיצד לבנות בדיקות אוטומטיות לבדיקת עמודי אינטרנט, אשר ירוצו על דפדפנים לפי בחירתכם, ואיך לקנפג אותם כך שירוצו בצורה אוטומטית באמצעות שרתי אינטגרציה מתמשכת (CI) .

האתגר

אם אתם חדשים בתחום של בדיקות רשת אוטומטיות, יש רק מספר דברים שעליכם להכיר על מנת להפוך את עבודתכם ליעילה בזמן קצר. עם זאת, חשוב להכיר את הדברים הללו על מנת לא לסגל לעצמכם הרגלי עבודה שגויים ולא לבזבז זמן על מאמצים אשר יניבו תוצאות לא חיוביות בהמשך.

הפתרון

אם תלמדו את היסודות של Selenium (כלי פופולרי המאפשר בדיקה אוטומטית של דפדפנים, ומתבסס על קוד פתוח), תוכלו להריץ במהרה בדיקות רשת אוטומטיות על גבי סוגי דפדפנים שונים. כאשר תשלבו את הבדיקות הללו עם כלי נוסף המבצע בדיקות ויזואליות אוטומטיות כמו Applitools Eyes, תוכלו להגיע לרמות כיסוי גבוהות מאוד בבדיקות שלכם, מבלי להשקיע יותר מדי מאמץ.

דוגמה

Selenium בנוי באופן המאפשר לו לחקות פעולות של משתמש, ולשם כך הוא מתבסס על שני נתונים חשובים: האלמנט בעמוד איתו תרצו לקיים אינטראקציה, והפעולה שתרצו לבצע על האלמנט הזה.

אם ניקח כדוגמה פעולה נפוצה כמו ביצוע התחברות לאתר מסוים, להלן הפעולות ש–Selenium יבצע אשר ידמו את פעולותיו של משתמש אמיתי:

- ביקור באתר
- מציאת שדה הקלט של שם המשתמש והקלדת טקסט לתוכו
- מציאת שדה הקלט של הסיסמא והקלדת טקסט לתוכו
- מציאת כפתור ה– submit והקלקה עליו

כעת נשתמש בדוגמת ההתחברות הלקוחה מהאינטרנט ונכתוב בדיקה של **Selenium** על מנת לבדוק אותה. לשם כך נשתמש בשפת התכנות **Ruby**, מכיוון שהיא נגישה וניתנת-לקריאה בקלות. נשתמש ב-RSpec, שהוא כלי המהווה ספריית קוד פתוח. זה יאפשר לנו לארגן ולבצע חזרה על פעולות מסוימות לפני ואחרי כל בדיקה.

כך נראית הבדיקה הראשונית ב-**Selenium** לאחר כתיבתו:

```
# filename: login_spec.rb
require 'selenium-webdriver'
describe 'Login' do
 before(:each) do
  @driver = Selenium::WebDriver.for :firefox
 end
 after(:each) do
  @driver.quit
 end
 it 'succeeded' do
  @driver.get 'http://the-internet.herokuapp.com/login'
  @driver.find_element(id: 'username').send_keys 'tomssmith'
  @driver.find_element(id: 'password').send_keys
'SuperSecretPassword!'
  @driver.find_element(css: 'button').click
 end
end
```

הערה RSpec עושה שימוש בסינטקס ששווה לשים לב אליו. כל מקרה בדיקה יתחיל עם המילה describe, ואילו הבדיקה עצמה תתחיל עם המילה it ולאחריה נוסיף את שם המבחן (לדוגמה: it 'succeeded' do) ואפשר לציין פעולות שיתבצעו לפני ואחרי כל בדיקה באמצעות before(:each) ו-after(:each). המילים do ו-end הן מילים המציינות ב-Ruby את תחילתו וסופו של כל בלוק של קוד. קבצי בדיקה ב-RSpec נקראים "specs" והם צריכים להסתיים ב-spec.rb_ בשם הקובץ (לדוגמה: **login_spec.rb**).

בחלק העליון של הקובץ יש להוסיף את התלויות (לדוגמה: require 'selenium-webdriver') ולהכריז על מקרה הבדיקה שלנו (לדוגמא: describe 'Login' do).

לפני כל מבחן אנו ניצור מופע (instance) של דפדפן פיירפוקס ונאחסן אותו בתוך משתנה אינסטנס, שבו נשתמש במהלך הבדיקה ולאחריו (לדוגמה: @driver = Selenium::WebDriver.for :firefox).

לאחר כל מבחן יש לסגור את הדפדפן (לדוגמה:@driver.quit).

לאחר מכן, ניצור את מבחן ההתחברות שלנו ונמלא אותו בפקודות של Selenium (פקודות אשר עובדות באמצעות המשתנה @driver אשר נוצר ב- before(:each)).

הפקודה הראשונה במבחן תבקר בעמוד ההתחברות באמצעות הפקודה **get**. שאחריה נוסיף את כתובת האתר כמשתנה מסוג **string** (לדוגמה'http://the-internet.herokuapp.com'). עכשיו נוכל ליצור אינטראקציה עם טופס ההתחברות שבעמוד על ידי מציאת שדה הקלט של שם המשתמש באמצעות **find_element**.

תוך כדי איתור האלמנט (לדוגמה: על ידי הערך (id: 'username')) יחד עם הפעולה אותה נרצה לבצע (לדוגמה: **send_keys**. 'tomssmith').

נחזור על אותה הפעולה עבור שדה הקלט של הסיסמה (למשל: @driver.find_element(id 'password').send_keys 'SuperSecretPassword!')

עבור הפקודה האחרונה נאתר פעם נוספת את האלמנט, כך: **find_element**.

על מנת למצוא את כפתור ה-**submit** (לדוגמה: (css: 'button')), ונקליק עליו באמצעות **click**..

אם נשמור את הקובץ שיצרנו ונריץ אותו (לדוגמה:.rspec login_spec.rb משורת הפקודה), זה יפתח את הדפדפן וישלים את פעולת ההתחברות עבור העמוד הזה, אך זה לא יחזיר לנו משוב לגבי הדרך בה האפליקציה התנהגה. על מנת לבדוק את זה נוכל להשתמש בבדיקות ויזואליות אוטומטיות של Applitools Eyes.

איך להתחיל לעבוד עם בדיקות רשת (web) אוטומטיות | דייב הפנר

Eyes גורמים לזה לקרות

הערה: על מנת להשתמש ב-Applitools Eyes צריך לפתוח חשבון חינמי [מכאן](#) (אין צורך בהתחייבות או בפרטי כרטיס אשראי).
על מנת לשלב את Applitools Eyes בבדיקה שלנו, ראשית צריך לבצע מספר ההתאמות לתחילתה ולסופה של הבדיקה שלנו.

```
# filename: login_spec.rb
require 'selenium-webdriver'
require 'eyes_selenium'

describe 'Login' do

 before(:each) do
  @browser = Selenium::WebDriver.for :firefox
  @eyes = Applitools::Eyes.new
  @eyes.api_key = ENV['APPLITOOLS_API_KEY']
   @driver = @eyes.open(app_name: 'the-internet', test_name:
'login', driver: @browser)
 end
# ...
 after(:each) do
  @eyes.abort_if_not_closed
  @driver.quit
 end
# ...
```

כאשר אנו משתמשים ב-Applitools Eyes Ruby SDK
(לדוגמה: require 'eyes_selenium')
אנו מעדכנים את ההתחלה של הבדיקה שלנו
(לדוגמה:before(:each)) על ידי יצירת מופע (instance) חדש של Applitools Eyes (מאחסנים אותו כמשתנה מופע לשימוש מאוחר יותר), נציין את מפתח ה-API שלנו, ונתחיל התקשרות (session) מול Applitools Eyes (שדורש את שם האפליקציה, שם הבדיקה ואת הדוגמה של Selenium). לאחר כל בדיקה, צריך לוודא שמבטלים את ההתקשרות (session) מול Eyes במידה והוא לא הצליח להיסגר כראוי (לדוגמה: @eyes.abort_if_not_closed) –נדבר על זה בהמשך.
עלינו לבצע זאת לפני שאנו מבטלים את הדוגמה של Selenium (מכיוון שהדוגמה של Eyes מסתמכת על זו של Selenium).

הערה: בדוגמה הנוכחית, מפתח ה-API מועבר על ידי משתנה סביבה. ניתן באותו האופן ליצור בקלות מפתח API משלכם כאן.

עכשיו אנחנו מוכנים להוסיף לבדיקה שלנו כמה בדיקות ויזואליות על מנת לוודא שהאפליקציה שלנו עובדת כראוי (ולוודא שהעמוד מרונדר כמצופה).

```
# ...
 it 'succeeded' do
   @driver.get 'http://the-internet.herokuapp.com/login'
   @eyes.check_window('Login Page')
   @driver.find_element(id: 'username').send_keys 'tomsmith'
   @driver.find_element(id: 'password').send_keys
'SuperSecretPassword!'
   @driver.find_element(css: 'button').click
   @eyes.check_window('Logged In')
   @eyes.close
 end
end
```

באמצעות הפקודה @eyes.check_window
אנו מבצעים בדיקות ויזואליות, אפשר גם להוסיף לבחירתכם ערך לפקודה הזו (לדוגמה: ('Logged In'), ('Login Page')). באם תבחרו להוסיף ערך כלשהו, הוא יופיע בתוצאות הבדיקות שלכם ב-Applitools Eyes. הדבר יכול להיות בעל ערך נרטיבי עבורכם או עבור אנשים אחרים בצוות שלכם, ולאפשר לכם לעקוב אחר מה שביצעתם בבדיקה כאשר תעברו על התוצאות לאחר מכן.
הפקודה @eyes.close סוגרת את ההתקשרות מול Applitools Eyes ומאתחלת את יצירת ההשוואה האחרונה של הבדיקות הוויזואליות.

התנהגות מצופה

כשתשמרו את הקובץ שלכם ותריצו אותו (לדוגמה: rspec login_spec.rb משורת הפקודה), להלן רצף הפעולות שיתבצע:

- Selenium יפתח את הדפדפן
- יאותחל התקשרות של Applitools Eyes אשר יתממשק אל האינסטנס של Selenium
- הבדיקה תריץ את הפקודות של Selenium ותיצור בדיקות ויזואליות לאורך הדרך
- התקשרות של Applitools Eyes ייסגר ויבצע ולידציה של הבדיקות הוויזואליות
- Selenium יסגור את הדפדפן
- כישלונות (במידה ויהיו) יוצגו בקונסולה

אם Applitools Eyes ימצא אנומליה ויזואלית, הוא יכשיל את הבדיקה ויספק כתובת URL שיעביר אתכם לתוצאות הבדיקה בקונסולה.
כאשר תבחנו את התוצאות בדשבורד של Applitools Eyes תוכלו לבחור אם לאשר או לדחות את התוצאות, וזה ישפיע על הדרך שבה Applitools Eyes יבצע ולידציה לבדיקות הבאות שלכם.
הערה: הרצת בדיקה חדשה ב-Applitools Eyes תגרום להכשלתו ואתם תקבלו קישור ל-URL אשר באמצעותו תוכלו לעבור על התוצאות. בהחלט מומלץ לבחון את התוצאות, אך זה לא מחייב. התוצאות יישמרו באופן אוטומטי כקו בסיס (נקודת הייחוס) עבור הבדיקות הבאות שלכם.

איך להתחיל לעבוד עם בדיקות רשת (web) אוטומטיות | דייב הפנר

כמה תוספות לסיום

על מנת להפוך את הדוגמה הזו ליעילה עבור יותר מבדיקה אחת, עליכם לבצע שינוי קטן בתחילת הבדיקה שלכם.

```
# filename: login_spec.rb
# ...
  before(:each) do |example|
    @browser = Selenium::WebDriver.for :firefox
    @eyes = Applitools::Eyes.new
    @eyes.api_key = ENV['APPLITOOLS_API_KEY']
    @driver = @eyes.open(app_name: 'the-internet', test_name:
example.full_description, driver: @browser)
  end
# ...
```

בתוך RSpec תוכלו לגשת לכל אחת מהבדיקות שלכם דרך שורת ה-`before(:each)` על ידי הוספת משתנה של בלוק (לדוגמה: `before(:each) do |example|`).

לאחר מכן תוכלו לגרום לבדיקה להשתמש ב- `@eyes.open` כאשר תציינו את שם הבדיקה (לדוגמה: `example.full_description: test_name_`). כעת, כשתוסיפו בדיקה חדשה לקובץ הזה, השם יתווסף באופן דינמי ל-Applitools Eyes.

לסיכום

שילוב הבדיקות של Selenium עם כלי אוטומטי המבצע בדיקה ויזואלית מאפשר לכם לעבוד עם שילוב עוצמתי המעניק לכם רמת כיסוי גבוהה עבור האפליקציה שלכם (כמו הוצאת מאות ולידציות לפועל) תוך כדי שימוש בשורות קוד בודדות. כבונוס, אתם מבצעים אוטומציה למסלול אשר משתמשים רבים מאמינים שלא יכול להיעשות, אלא באופן ידני בלבד (וכתוצאה מכך, הם נוטים לרדת ברמת הבדיקות שלהם בסוף מעגל הפיתוח של התוכנה) – מה שמקדם אתכם ואת הצוות שלכם צעד גדול קדימה לטובת אספקת גרסה מתמשכת (Continuous Delivery) של האפליקציה שלכם.

בראבו!

בודק – רענן הידע הבסיסי

אנו נוטים להניח כי הפקנו את כל אשר יכולנו מהחומר הבסיסי אשר כבר למדנו לגבי המקצוע – אך הנחה זו שגוייה ביסודה,

בכל נקודת זמן נקודת המבט שלנו משתנה בהתאם לידע הנוסף אשר רכשנו, לשינוי בתנאים הסביבתיים, במעמדנו ובתפקיד אותו אנו מבצעים.

לעיתים קרובות קריאה חוזרת של חומר שאנו כבר למדנו לפני שנה-שנתיים, יכולה לפתוח לנו זווית ראייה חדשה – מתי לאחרונה קראתם את הסילבוס של ISTQB? רעננתם את היכרותכם עם TheTestEye SW Quality Characteristics? צפיתם בסרטוני ההדרכה של BBST? בחנתם בשלות תהליכי עבודת הבדיקות של ארגונכם מול תכנית השיפור של TMAP?

אמנם אנו חושבים כי למדנו ושיננו חומר זה בעבר – אך בעצם אנו זוכרים רק אנקדוטות ספציפיות מתוכו שבאותו רגע דיברו אלינו, ויש עוד מגוון רעיונות וידע אשר איננו זוכרים ממנו כלל.

> **קריאה חוזרת של חומר שאנו כבר למדנו לפני שנה-שנתיים, יכולה לפתוח לנו זווית ראייה חדשה**

בכדי לנצל ידע זה עלינו לשמור על ראש פתוח, לקרוא המידע, ולחשוב: כיצד אנו יכולים לנצל ידע ורעיונות אלו בנקודת זמן זו בה אנו נמצאים? מהיכן אנו שואבים את רעיונות הבדיקה שלנו? כיצד נוודא כי נמצה את הבדיקות החשובות? האם אנו יכולים לסמוך על כך שכל אשר למדנו, תרגלנו וחווינו יתבטא בעבודתנו הנוכחית?

לשם כך אנו יכולים לבנות תוך כדי עבודה וקריאה רשימות Checklist ומפות מחשבה אשר יקלו על תהליך ריענון הידע, ו/או להשתמש ברשימות שאחרים כבר יצרו, אך לעיתים קריאה מקוצרת זו אינה מספיקה לשם העלאת רעיונות חדשים, ונדרשת קריאה מקיפה יותר של הרעיונות בכדי לאפשר תהליכי מחשבה מעמיקים יותר.

אני משתדל לחזור אל הסילבוס הבסיסי של ISTQB כל כשנתיים ו/או במעבר תפקיד/חברה – וכל פעם מחדש מוצא רעיונות חדשים לשיפור תהליכי העבודה, לקראת כתיבת תכנית בדיקות לתכונה חדשה – אני מוודא כי אין אספקטים קריטיים עליהם דילגתי ע"י מעבר מחודש על רשימות של נושאים ורעיונות בדיקה בסגנון של SW Quality Characteristics, ושוב ושוב מוצא ערך מחודש בפעילויות אלו.

זכרו – יש המון כוח גם בידע הבסיסי של מקצועינו – אם רק ננצל אותו כראוי.

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

חומר קריאה נוסף:
http://www.mkltesthead.com/2013/08/refresh-on-basics-99-ways-workshop-37.html

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: Ways Workshop 99 - בה מיכאל מרחיב על כל אייטם וגם מספק הנחיות כיצד לתרגל הנושא.

אתם מוזמנים לקרוא טיפים רבים נוספים ב: http://goo.gl/UcW42
ולהוסיף טיפים משלכם לרשימה משותפת זו.

למען הקהילה | ITCB Israeli Testing Certification Board

ITCB מסבסדת הנחה משמעותית למוסמכי ISTQB לכנסים מקצועיים שנערכים ע"י סיגיסט - פורום בודקי התוכנה בישראל(SIGiST)

מבוא לבדיקות עומסים | נחום דימר

נחום דימר

הינו יזם ומנכ"ל TEST4LOAD, חברה מובילה בתחום בדיקות עומסים ושיפור ביצועים של מערכות תוכנה. הוא מעורב בפיתוח, בדיקות וניהול של מגוון פרויקטים בתחום תוכנה ומערכות ארגוניות. יש לו ניסיון עשיר ורקע טכנולוגי מעמיק במגוון תחומים הכוללים הנדסת תוכנה וביצועים, וכן אבטחת איכות של מערכות אינטרנט וטלקום.
TEST4LOAD עוזר לתכנן, ליישם, לבדוק ולמטב את ביצועים של מערכות תוכנה מרובות משתמשים. בין לקוחות החברה נמנים חברות וגופים מובילים בארץ ובעולם. http://cloudbeat.io/

חלק א'

מורכבותם של יישומי אינטרנט, אפליקציות מובייל מודרניים, עבודה בענן והמספר הגדל של המשתמשים, מעלה מדרגה את נושא הבדיקות בכלל ובדיקות ביצועיים בפרט. עם הגידול במהירות הגישה, משתמשים אינם מוכנים עוד להשלים עם מהירות תגובה איטית או חוסר יציבות של היישום. כתוצאה מכך אנו עדים לסטנדרטים גבוהים ולדרישה מוגברת לפיתוח יישומים ואתרים מהירים ויעילים יותר, וכן ניצול מרבי של החומרה ודחיפת הביצועים עד גבול היכולת.

לכן, לא ניתן עוד להסתפק בבדיקות פונקציונאליות בלבד ויש צורך בבדיקות ביצועים בהן ייבדקו זמני התגובה והיציבות של המערכת תחת העומס של המשתמשים. בדיקות ביצועים הן נושא רחב, שתחתיו נכנסים סוגי בדיקות שונים, כגון בדיקות עומסים, בדיקות יציבות, בדיקות השוואתיות, בדיקת ביצועים של קוד (profiling) וכו'. במאמר זה נדבר על בדיקות עומסים שהם חלק חשוב מבדיקות ביצועים של תוכנה.

מה זה בדיקת עומסים?

בדיקת עומסים היא שיטה למידול השימוש הצפוי בתוכנה, באמצעות סימולציה של גישה בו-זמנית של משתמשים רבים אל שרותי התוכנה. סימולציה זו נעשית על ידי כלים אוטומטיים המסוגלים לדמות מספר רב של משתמשים. בדיקות עומסים עוזרות להבין זמני תגובה של המערכת ומסייעת באיתור צווארי בקבוק ונקודות כשל.

למה חייבים לבצע בדיקות עומסים?

לא מעט חברות וארגונים מוותרים על בדיקות עומסים, מתוך מחשבה שאם המערכת תפקדה כראוי בזמן בדיקות ידניות, היא תתפקד בהתאם גם בסביבת הייצור. כמו כן, אנשי ה-IT אוהבים לטעון שאם המערכת תהיה איטית בייצור אז הם יוסיפו עוד שרתים וזה יפתור את כל הבעיות.

כמובן שכל זה לא נכון – ללא ביצוע בדיקות עומסים על מערכות מרובות משתמשים, לא ניתן לגלות בעיות ביצועיים מהותיות ולתקנן מבעוד מועד. המערכת עלולה לקרוס, לא לתפקד או להיות איטית – כל זה יכול לגרום נזק ממשי למשתמשים ולארגון.

אחד מהדוגמאות הבולטות משנים האחרונות היא נפילה של אתר Obamacare – אתר שמאפשר לציבור אמריקאי להירשם לביטוח בריאות בארה"ב אחרי הרפורמה של אובמה. אתר זה לא נבדק כראוי וקרס ביום הראשון להפעלתו. בניית האתר עלתה אז כ-400 מיליון דולר ובדיקות האתר, כולל בדיקות עומסים לא בוצעו כראוי. לקח חודשים להפעיל את האתר מחדש, אבל כמובן הפגיעה המשמעותית ביותר הייתה בתדמיתו של נשיא ארה"ב ומפעיליו של האתר.

כמו במקרים האחרים, גם במקרה זה הוכח שהוספת חומרה לרוב לא תעזור למנוע או לפתור בעיות ביצועיים מהותיות ורק תגרום לבזבוז נוסף של זמן וכסף. לכן חייבים לבצע בדיקות עומסים כחלק אינטגראלי מבדיקות המערכת!

מתי מבצעים בדיקות עומסים?

בדיקת עומסים מצריכה מערכת יציבה ובדוקה היטב ולכן לרוב מבצעים בדיקות עומסים בשלב טרום ייצור או בשלבים מאוחרים של תהליך הפיתוח. בנוסף, מומלץ לבצע בדיקת עומסים בסביבה דומה ככל שניתן לסביבת הייצור – דבר שבדרך כלל זמין לקראת עליה לאוויר של המערכת.

כמובן שביצוע בדיקות עומסים זמן קצר לפני עליה לאוויר מעלה סיכון ולעיתים אף עלול לגרום לעיכובים משמעותיים. לכן מומלץ להשתדל לבצע בדיקות עומסים בשלבים מוקדמים יותר של הפיתוח. מומלץ להתחיל את בדיקת עומסים על חלקים יציבים יותר של המערכת ואז לקראת הסוף לבצע בדיקה כוללת של כל החלקים על המערכת השלמה. אפשר גם לשלב תהליך בדיקות עומסים עם Continues Integration ואף Unit Testing. ככל שמבצעים בדיקות עומסים בשלבים מוקדמים יותר, כך היכולת להבין את מהות הבעיה ולתקן אותה היא קלה ומהירה יותר. כמו כן, מחיר תיקון התקלה בשלבים מוקדמים יהיה נמוך משמעותית ממחיר התיקון רגע לפני העלייה לאוויר, או גרוע מכך בסביבת הייצור.

איך לתכנן בדיקת עומסים?

תכנון של בדיקת עומסים נגזר מדרישות הביצועים של המערכת. לכן כדאי להתחיל לתכנן בדיקת עומסים יחד עם הגדרת דרישות הביצועים של המערכת. בלא מעט מקרים, דרישות הביצועים לא מוגדרות מראש בשלב התכנון ולא מופיעות בדרישות המערכת. זה עלול לגרום לוויכוחים אין סופיים וגלגול אחריות בשלבים מאוחרים יותר בין גורמים שונים בארגון, בהם אנשי פיתוח, QA, IT, מנהלי מוצר, לקוח סופי ועוד רבים אחרים. כדאי למנוע מצב זה ע"י חשיבה משותפת של כל הגורמים הרלוונטיים והגדרה מסודרת של הדרישות עוד בשלב הגדרת הדרישות לכל המערכת הנבדקת.

דרישות ביצועיים בדרך כלל כוללות דברים כמו כמות משתמשים מקסימלית שהמערכת אמורה לתמוך בה, הגדרה של זמני תגובה מקובלים, עמידה בפיקים, הגדרת יכולת גדילה של המערכת (scale) וכו'. לאחר שדרישות אלו הוגדרו, אפשר להתחיל בתכנון של בדיקות העומסים. לרוב התכנון יכלול את הדברים הבאים:

- **תסריטי בדיקה** – פירוט של תסריטי הבדיקה (הרחבה להלן).
- סוגי הבדיקות – סוגי הבדיקה הרלוונטיים לדרישות (הרחבה להלן).
- **מודל העמסה** – כמות משתמשים מרבית, זמן עליית המשתמשים ומשך הבדיקה בהתאם לסוגי הבדיקות. בניית מודל העמסה המייצג את המציאות הוא לעיתים המשימה הכי קשה בבדיקה. בבסיסו של המודל עומדת הגדרת כמות משתמשים המקסימאלית שהמערכת צריכה לתמוך בה. אך צריך גם לקחת בחשבון נתונים כמו כמה זמן משתמש עושה שימוש רציף במערכת (session time), בכמה שעות ביום יש פעילות מוגברת ובכמה שעות אין כמעט כל פעילות. במקרה של מערכת קיימת, כדאי להסתכל על נתונים של גוגל אנליטקס או כלים דומים ולחשב את מודל העמסה בהתאם למספר משתמשים בשעת השיא (מדד sessions) ואורך הכי קצר של המופע (session time).
- **סביבת הבדיקה** – הגדרת סביבת הבדיקות על כל מרכיביה. חשוב לבחור סביבת הבדיקות מקבילה או קרובה ככל שניתן לסביבת הייצור.
- **נתוני הבדיקה** – הגדרת סוג וכמות הנתונים הנדרשים לביצוע הבדיקה. לדוגמא, אם מדובר במנוע חיפוש – כדאי לתת רשימה רחבה של מילות מפתח שמכסות מקרי קצה שונים מבחינת המנוע (משפטים קצרים, ארוכים וכו'). אם מדובר ברישום של המשתמש, צריך להקצות מספיק שמות משתמשים ייחודיים כך שהבדיקה לא תיעצר על שם משתמש שכבר קיים. לא פעם צריך להגדיר החרגות מיוחדות לטובת הבדיקה, כגון מספר כרטיס אשראי לא אמיתי שיתקבל ע"י המערכת וכו'.
- **לו"ז** – לוח זמנים מפורט של הבדיקה, כולל זמני פיתוח התסריטים, הרצות וזמן תיקונים משוער.
- גורמים המעורבים וחלוקת אחראיות – רשימה של הגורמים המעורבים בבדיקה כגון DBA, אנשי מערכות מידע, מנהל הפרויקט וכו'

מבוא לבדיקות עומסים | נחום דימר

בחירת תסריטי בדיקה

בניגוד לתסריטי בדיקה ידניים שמטרתם לכסות את מרבית הפונקציונליות, בבדיקות עומסים מטרה העיקרית היא לבדוק תהליכים שהשפעתם ישירה או מקיפה היא קריטית לביצועי המערכת.

יש לבחור את תסריטי הבדיקה לפי הכללים הבאים:

המסלול הקריטי – בחירת תהליכים שהשפעתם היא קריטית מבחינה עסקית לארגון או מהותית מבחינת המשתמש

אפקט הדומינו – בחירת תהליכים שאולי השפעתם הישירה היא לא קריטית לארגון או מערכת, אך עצם הפעלתם יוצרת סדרת פעולות שעלולה לגרום לעומס יתר של המערכת. דוגמא טובה לסוג זה של תהליך היא הפקת דוחות.

סוגי הבדיקות

ישנם 5 סוגי העמסה בסיסיים שכדאי להכיר. המטרה ואופן הביצוע של כל סוג הם שונים.

- **בדיקת עומס רגיל** (workload) – הדמיה של עומס יום-יומי של המערכת. בדרך כלל נגזר מפעילות ממוצעת של משתמשים לאורך היום.
- **בדיקת עומס קיצוני** (stress) – הדמיה של עומס קיצוני על המערכת. נגזר מפעילות או מצפי לפעילות המערכת בשעות השיא עם מכפיל ביטחון (כ-30-50%). בדרך כלל עומס קיצוני הוא כפול או משולש מעומס היום-יומי.
- **בדיקת יציבות** (stability) – הדמיה של פעילות ממושכת של המערכת ברמת עומס יום-יומי. בדיקת יציבות בדרך כלל נמשכת בין 48-12 שעות. מטרת הבדיקה היא לגלות זליגת משאבי השרתים, כגון זיכרון, וגילוי בעיות יציבות נוספות.
- **בדיקת הרחבה** (scale) – מטרת הבדיקה היא לגלות את יכולת ההרחבה של המערכת – כלומר מה תהיה רמת הביצועים של המערכת במקרה של הרחבת החומרה והגדלה של העומס. בדיקה זו מאוד חשובה במערכות שנדרשות להתרחב בצורה אוטומטית (auto-scale) או שיש צפי לגדילה משמעותית של כמות המשתמשים בתוך זמן קצר לאחר עלית המערכת לאוויר.
- **בדיקת קיבולת** (capacity) – מטרת הבדיקה היא למצוא קיבולת של המערכת, כאשר ביצועי המערכת נשמרים ברמה המקובלת ובהתאם לדרישות. כלומר, אם המערכת אמורה לעמוד בזמן תגובה של 5 שניות וצריכת המשאבים בשרתים לא אמורה לעבור רף של 80%, אז בדיקת קיבולת תגלה מה היא כמות משתמשים מקסימלית שבה ניתן לעמוד ביעד זה.

הבדיקות הכי נפוצות הן workload ו-stress אך כמובן כדאי לבצע לפחות את שלושת סוגי הבדיקות הראשונים, כולל בדיקת יציבות. בדיקת קיבולת נגזרת בלא מעט מהמקרים מבדיקת עומס קיצוני (stress)) מכיוון שלרוב בדיקה זו תביא את המערכת לקצה היכולת. בדיקת הרחבה (scale) תתאים יותר למערכות שיש בהם מנגנון auto-scale, כלומר הוספה אוטומטית של שרתים או משאבים בהתאם לעומס או ניצול המשאבים של המערכת. אך כמובן שכדאי לבצע בדיקת scale גם במקרים בהם צוות התשתיות רוצה להבין מה תהיה ההשפעה של הוספת חומרה נוספת על ביצועי המערכת.

סיכום

כתבה זו היא ראשונה בסדרת הכתבות בנושא בדיקות עומסים. בכתבה זו הבאתי בפניכם את המבוא והצד התאורטי יותר של בדיקות עומסים. בכתבה הבא נדבר על הצד הפרקטי יותר של הבדיקה – נבחן את הכלים לבדיקות עומסים ונדבר על ביצוע הבדיקה בפועל.

לכל שאלה, הערה או המלצה – אתם מוזמנים לכתוב לי ל:

ndimer@test4load.com.

להתראות בגיליון הבא!

אתר Obamacare לא נבדק כראוי וקרס ביום הראשון להפעלתו, הפגיעה המשמעותית ביותר הייתה בתדמיתו של נשיא ארה"ב ומפעיליו של האתר.

פתרון השאלה

השאלה הזו בוחנת ידע בפרק 1.2: "היבטים של גישות אג'ייל". יעד הלימוד (שממוספר FA-1.2.2) אומר שעל הבודק לדעת לכתוב סיפורי משתמש שניתן לבדוק (testable) בשיתוף עם המפתחים והנציגים העסקיים.
בשאלה מתואר דיון שמתרכז בשני היבטי איכות של יישום סיפור המשתמש: שימושיות (usability) וביצועים (performance). בפיתוח בגישת אג'ייל סיפורי משתמש נכתבים בשיתוף פעולה של חברי הצוות: מפתחים, בודקים ונציגים עסקיים.

חשוב שכל אחד יביא לדיון את נקודת מבטו וביחד יגדירו את המדדים (קריטריונים) שיאשרו את קבלת הסיפור ואיכותו.

נעבור על התשובות

א – למרות שחשוב לקחת בחשבון שיקולים של אוטומציה, הרי שעיצוב המערכת כך שתצמצם את מאמצי הבדיקות לא בהכרח תספק פתרון שיהיה טוב ללקוח. וודאי גם ששיקול זה הוא משני לעומת שיקולים של שימושיות וביצועים. תשובה זו לא נכונה.

ב – מוביל המוצר (product owner) הוא זה שמתעדף בין מדדי האיכות, ולא הבודק. תשובה זו לא נכונה.

ג – קריטריון הקבלה של מדד הביצועים יוגדר בדרך כלל על ידי מוביל המוצר (product owner). תשובה זו לא נכונה.

ד – התרומה של הבודק תהיה בכך שידאג שהצוות ייצור קריטריון קבלה עבור סיפור המשתמש. תשובה זו היא הנכונה.

נתראה בגיליון הבא

SIGiST ISRAEL

Save the date

בין המרצים אשר ישתתפו בכנס:

- Richard Bender (USA) - Bender RBT
- Alon Linetzki - QualityWize
- Michael Stahl - Intel
- Igal Levi - Nice Actimize
- Meira Diskin - Ceragon
- Dr. Ronny Deutsch - Sela

בין הנושאים החמים שיועברו בכנס:

- Requirements Based Testing
- Optimizing Test Design
- Measuring People
- Cyber Security
- Automation in Agile
- Out of the Box Thinking
- Finding Ambiguities in Requirements

מס' המקומות מוגבל – התקשר עכשיו ושריין הרשמתך!

ליצירת קשר והרשמה: 03-6176066 | www.sigist.org.il | info@sigist.org.il

ISTQB International Software Testing Qualifications Board | ITCB Israeli Testing Certification Board | SIGiST ISRAEL | SELA | 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 ירושליםTeleknowledge , 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 ISTQB® Agile Tester – Foundation Level Extension, SIGiST Israel – The Israeli Testing Forum, מייסד שותף של ITCB® – Israel Testing Certification Board. היה אחד מכותבי הסילבוס של ISTQB®, 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.