

הנדסת מערכת מנייה - תיאור התחום של בדיקות קבלה פונקציונליות (FAT – Functional Acceptance Tests) בעידן המנייה החכמה

יצחק משיח, אליעזר אברמוביץ, נצח קלמרו

מאמר שני:

מימוש החזון של צמצום משמעותי של נפח העבודה והכנסת יתר סדר בעבודה

1.1 מערכת ממוחשבת למעקב כשלים (Defect Tracking System)

מערכת ממוחשבת למעקב כשלים (באגים), הינה בסיס נתונים שבו חברי צוות הבדיקות פותחים את דיווחי הבאגים שאיתרו במונה. **במודל העבר** נרשמים הבאגים בגליון נייר. מדי תקופה מבצע ראש צוות הבדיקות רישום מסודר של הכשלים, ומדווח למנהל המעבדה. פתיחת כשל נעשית דרך טופס, בממשק הגראפי של התכנה. התוכנה היא אפליקציה אינטרא-נטית: כל חברי צוות פרויקט בדיקות קבלה למונה ספציפי מורשים לצפות בפרויקט, אך אין גישה לאנשים מחוץ לפרויקט ובוודאי מחוץ לח"י. בכל פעם שמתחיל פרויקט אישור דגם מונה, מוגדר פרויקט נוסף במערכת. כאשר נפתח כשל, נשלח דוא"ל לכל הגורמים המוגדרים רלוונטיים. בח"י התוכנה שנעשה בה שימוש היא Quality Center HPTM, של החברה HP (Mercury) שפתחה את תוכנת בדיקות תוכנה Winrunner. תוכנה זו נותנת מעטפת שלמה לכל 5 ווקטורי המחשוב באותה סביבת תוכנה – יתרון משמעותי. תכנה זו עולה **פר רישיון למשתמש פר שנה** בעלות נכרת. לארגונים גדולים – משתלם לרכוש שימוש בכזו תכנה, משום שעלות באג מתרגמת לסכומים גבוהים מעלות רשיונות התכנה. לארגונים קטנים, לפעמים יש תקציב ולפעמים אין כלל. קיימות תוכנות Freeware מקבילות, דוגמת: MantisTM או BugZillaTM. תוכנות אלו כוללות בסיס נתונים וממשק גראפי למשתמש, וממשק אחורי ליצירת דו"חות מפורמטים. תוכנות אלה הן חלק ממבנה הפצה General Public License. הן בשימוש מאות חברות, וקיים עליהן משוב חיובי ברשת. התקנתן **מורכבת** ודורשת איש System Admin – משום שהיא כוללת מספר מכלולים: מנהל סרור, בסיס נתונים, כלי שאילתות גראפי, וממשק תקשורת לתוכנות משלימות. מנהל מעבדה ישאל בוודאי מהי איכות התוכנות, ומה הבסיס הכלכלי לקיומן. התשובה היא שחברות אלו נותנות תוכנה משודרגת בתשלום. זהו בסיס עסקי דומה במקצת לרשתות חברתיות ו-Google. אנו מבינים איפה שהחינמיות אינה מעידה על איכות התכנה. כל התוכנות החינמיות ניתנות עם ספר הפעלה והתקנה מסודר. מדוע

במאמר הקודם הצגנו את תחום הנדסת מערכת מנייה, ואת המורכבות הגדילה בו כיום. בנוסף הדגמנו כיצד טכניקות מסוימות מסייעות לצמצום נפח העבודה המירבי שרואה איש צוות, או שמבוצע בפרויקט. נגענו בטכניקות: (1) מתודולוגיה, (2) מחשוב-אוטומציה, (3) רגולציה. במאמר זה ניצוק תוכן יישומי ע"י הצגת פלטפורמות בדיקה בתעשייה, הזמינות לכל מעבדה – חלקן בעלות רכישה, וחלקן חינמיות. הטכניקות המתוארות כאן, ישימות לכל מעבדה באשר היא.

מתודולוגיות וכלים מתחומי הנדסת מערכת, וורייפיקציה וולידציה – הושאלו ויושמו במעבדת פיתוח אמצעי מנייה. מתחום וורייפיקציה (מכלולי משנה) וולידציה (מערכת) אנו מציעים קבוצת תוכנות המכסה את כל תחום הדרישות שהגדרנו עד כה. תוכנות אלה ממוינות ל 5 נושאים ענייניים המכסים 5 שלבים שונים בעבודה:

מס	כלי תוכנה	מכפיל כוח ישיר/ מכפיל כוח ע"י ארגון תכנים
1	מערכת ממוחשבת למעקב כשלים (Defect Tracking System).	ארגון תכנים
2	תוכנת ניהול תוכניות בדיקה – בשלב תכנון תוכנית הבדיקה, ותייעוד בשלב הביצוע (Test Program Manager).	ארגון תכנים
3	מערכת ארכוב וניהול גרסאות (Revision Control) – מיצרן ובפיתוח מקומי, מסמכים, תכתובות.	ארגון תכנים
4	תוכנה לניהול תוכנית בדיקה לשם הרצה אוטומטית של הבדיקות (Test Program Execution Manager).	מכפיל כוח ישיר
5	אוטומציה של הבדיקות הספציפיות (Test Automation).	מכפיל כוח ישיר

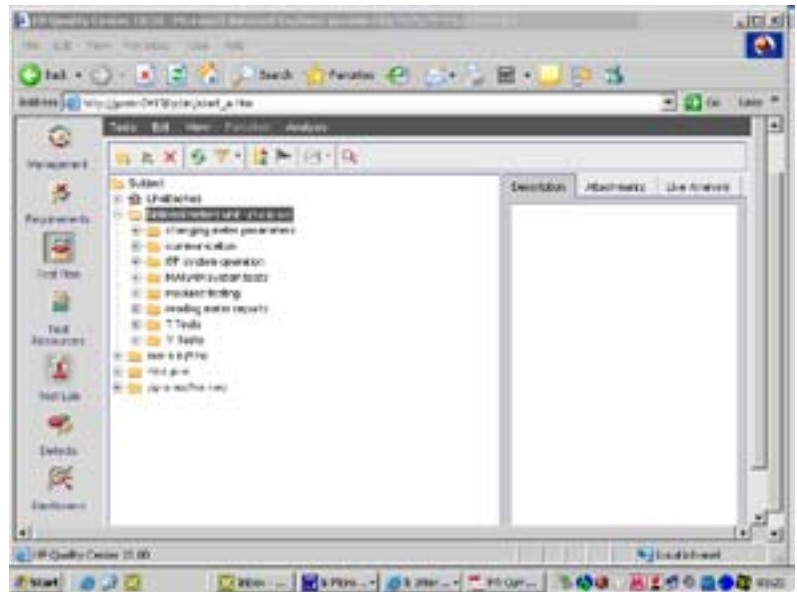
נתאר כעת ביתר פירוט את מנועים מתקדמים אלו: מהם הכלים, דוגמאות לכלים קיימים בתעשייה, המאפשרים ליישם את תכנית המחשוב-אוטומציה, במגבלות החלות על מעבדה ספציפית. המאמר אינו פרסומי, אך אנו מציעים פלטפורמות לדוגמה בתעשייה.

1.2 - תוכנת ניהול תוכניות בדיקה – בשלב תכנון תוכנית הבדיקה, ותיעוד בשלב הביצוע (Test Program Manager).

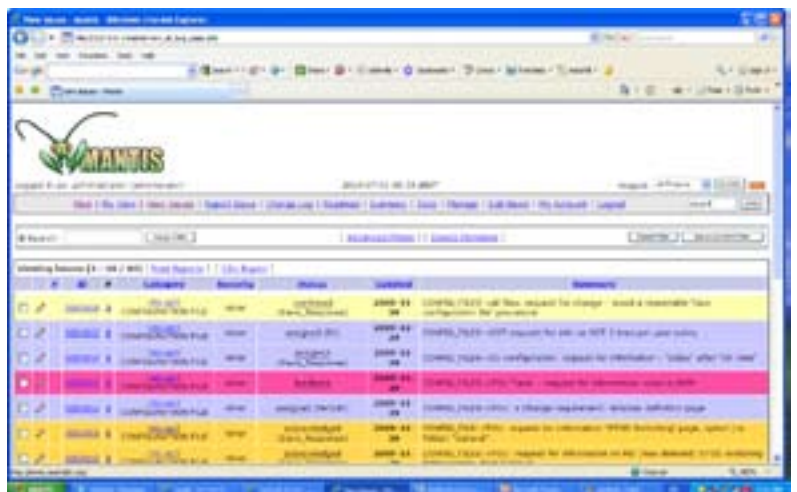
מגיעים מסמכי תוכניות בדיקה ממספר מקורות למונה הספרתי והתוכנות הנלוות: מהיצרן, מהמעבדה. כמות הבדיקות גדולה מאד. מנהל פרויקט בדיקות הקבלה, מפצל את וורטיפיקצית המונה למכלולי משנה נפרדים. פרקי בדיקת מכלולים אלה מפוצלים לפרקי משנה, עד להגעה **לפעולה אטומית** של בדיקה יחידה. מתודולוגית המהנדסים הבכירים יותר מפעילים היררכית בדיקות, שהיא פעולת צמצום של כמות העבודה שכל חבר צוות מנהל. מדי סבב בדיקות "משוחררת" גרסת מונה המורכבת ממספרי גרסה של החומרה, קושחה, כלי תוכנה נלווים. יצרן המונים מבצע תיקונים והתאמות, ומתחיל סבב בדיקות חדש – חלקי או מלא של מערכת מונה. **בעבר** היו אנשי הצוות מקיימים קובץ בדיקות פשוט ומבצעים את הבדיקות על סמך ניסיון. כמות הבדיקות עלתה. ספקנים יאמרו שתהליך QA זה שייך ליצרן. בפועל מונה מותאם לצרכי הלקוח, הוא אינטגרציה של מכלולים. הניסיון בעולם מראה שתהליך זה קיים גם בחברות חשמל שמבצעות אינטגרציה.

דרך הפשוטה יותר לניהול תוכנית בדיקה היא **לתעד בכתב בספר בדיקות קבלה** את כל הבדיקות שמבוצעות במונה ספרתי. זהו שינוי מתודולוגי ואין לזלזל בו כלל. במעבדתנו אנו מחזיקים ספר בדיקות קבלה לאישור דגם מונה ספרתי חדש. הספר כולל 16 פרקי בדיקה, ו 500 בדיקות ומתרחב בהתמדה. בכל אישור דגם מונה, אנו מקפידים שבאבני דרך יצא **ספר דו"ח בדיקות**, שבו חתום צוות הבדיקות, רמ"ד, מנהל מעבדה. בחירת הפרקים שימומשו בכל סבב, תלויה בשינויים שהוכנסו באותה גרסה – באיזה מכלולי מישנה בוצעו שינויים ע"י היצרן. החזקת ספר כתוב ידנית – עשויה להיות הפתרון המספק עבור חלק גדול מהמעבדות.

בשימוש בתוכנת ניהול תוכנית בדיקה, התהליך מבוצע ידני אך ממוחשב. כאשר בודק ביצע בדיקה הוא מסמן **טורם** בתוכנית הבדיקה הממוחשבת. סטאטוס הבדיקה גם מעודכן: **טורם בוצע, נכשל, עבר**. אם הבודק מאתר כשל, הוא פותח באג בתוכנת מעקב הכשלים. בשימוש בתוכנת Quality Center HP™ לדוגמא, תוכנת מעקב הכשלים תוכנת ניהול תוכנית הבדיקה וכל 5 תוכנות המחשוב – פועלות באותה תוכנה. מעבר מעבודה ידנית מליאה, **לתוכנת ניהול תוכניות בדיקה** כרוך בנכונות הארגון להטמיע את השינוי. אם אין כזו – לא יצליח השינוי, משום שהוא כרוך במאמץ ניכר. במידה וחבר צוות הבדיקות מעוניין להפיק את ספר הבדיקות – בתוכנת Quality Center HP™, זו פעולה בלחיצת כפתור. שימוש



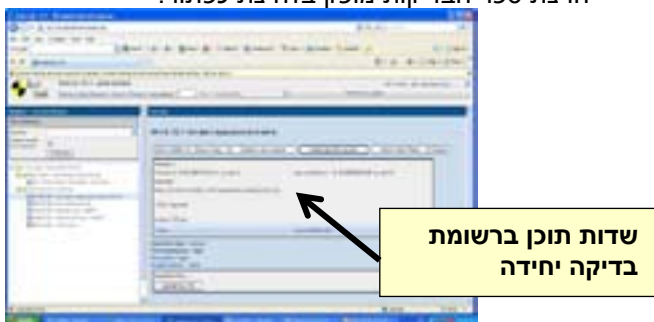
תרשים 1-א: ממשק תוכנת Quality Center HP™. התפריט האנכי משמאל כולל את 5 הווקטורים של המחשוב. העץ במרכז מייצג את תוכנית הבדיקות -



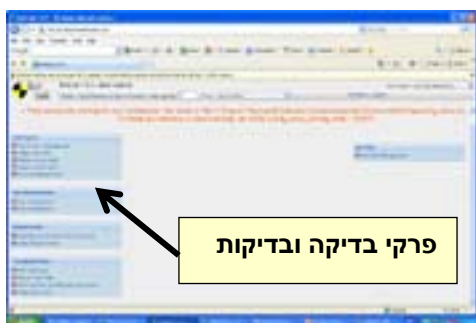
תרשים 1-ב: מערכת מעקב תקלות Mantis™. צבעים שונים מציינים סטאטוס שונה של הבאג (נפתח, אומת, תוקן, נבדק). לחיצה על באג, פותחת את רשומת תיעוד הבאג. -

אם כן **לרכוש** תוכנה בתשלום? קיים יתרון משמעותי במעטפת עבודה **אחת** לכל הווקטורים של המחשוב, כאשר הם מתקשרים ביניהם. מן הסתם איכות התוכנה שנרכשת שונה במשהו מאיכות תוכנת חינוך – אך הפער אינו גדול כפי שחושבים תחילה.

במעבדת פיתוח אמצעי מניה, ביצענו pilot של התקנת תוכנת Mantis™ לשם בדיקות קבלה פונקציונאליות לתוכנת מרכז מנייה. כיום אנו עוברים משיקולי האחדה בחח"י, עם Quality Center HP™. תהליך מעבר לתכנה ממוחשבת כולל 3 שלבים: (1) התקנה, (2) הדרכה לכלל המשתמשים, (3) **ההטמעה** תפישתית שצריך לעבוד כך, הבאה **מההנהלה**. נדרש מאמץ במעבר, ויש התנגדות בתחילה. יש להסביר את מטרות הכלים. הפקת דו"ח כשלים עם פילוחי כשלים לפי פרמטר היא תהליך בלחיצת כפתור.



א. דף פרוט לבדיקה יחידה.



ב... דף תוכנית בדיקה

תרשים 3: תוכנת TestLink™ לניהול בדיקות

סבב הבדיקות, סומן V ליד בדיקה – שולחת התוכנה, דוא"ל אוטומטי לתזכורת לצוות הפרויקט. דו"ח תוצאות הרצת ספר הבדיקות מופק בלחיצת כפתור.

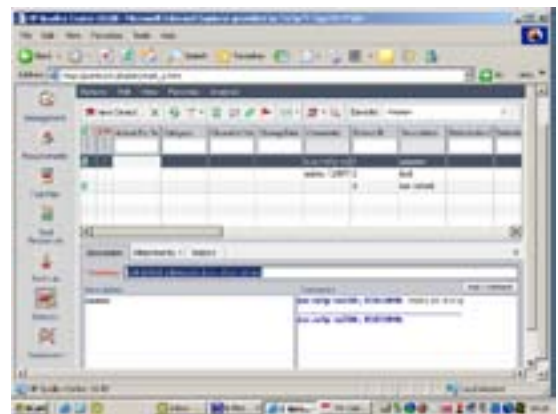
1.3 תכנה לניהול גרסאות משנה (Revision/Sub-Version Control) (System)

בהשוואה לכלי התוכנה הקודם שאיננו מאומץ ע"י כל המעבדות – בגלל היותו "גדול" עליהם, הכלי המוצג כעת קל יחסית ליישום, אינו דורש תהליך הטמעה קשה, ונדרש גם ע"י מעבדות הנשארות בעולם הידני. **הצרכים** של מעבדה, המארכבת גרסאות תוכנה המתקבלות מיצרן מונים, ותוכנות בדיקתיות מפיתוח מקומי, הם שונים מעט להשקפתנו מצרכים של קבוצת תכנות. למעבדת בדיקות, נדרש ממשק יותר קל להבנה למשתמש שאינו מתכנת, ויש פחות דגש על מניעת קריסה של פרויקט תכנה, הנכתב בשיתוף מתכנתים רבים.

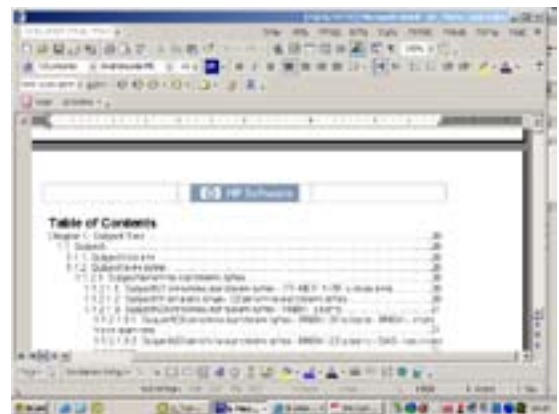
בעבר: בעולם הידני – אירכוב גרסאות תוכנה חדשות המגיעות מהיצרן, מבוצע ע"י פתיחת פולדר חדש עם שם הכולל את מציין הגרסה, בפולדר הראשי של הפרויקט. שיטה זו אינה יעילה בניהול גרסאות מישנה המתחלפות בקצב תדיר, וגרסאות תוכנה נוטות ללכת לאיבוד ולהתבלבל. זה קורה בעיקר משום שלכל הגרסאות **אותו שם** חיכוני Application.exe, וזיהוי מציין הגרסה כרוך בהרצתה במוד מיוחד. שימוש בתוכנת אירכוב מאפשר הצמדת התג "מבחוץ" לגרסת התכנה ורישום הערות לגבי שינויים בין הגרסאות, וכן השוואה טקסטואלית/ בינארית בין קבצים מגרסאות שונות. כל מפרטי היצרן, מפרטי ח"י, וגרסאות התוכנה הנמסרות ע"י היצרן או מפותחות למטרות בדיקות



א. מסך דיווח כשל (פתיחת באג)



ב. מסך תצוגת כל הכשלים



ג. ספר בדיקות ממוחשב בהיקף 100 עמודים מופק בלחיצת כפתור

תרשים 2: ממשקי תוכנת Quality Center™ של HP (Mercury) לניהול בדיקות

בתוכנה חנימית, אנו מציעים את תכנת TestLink™, **לניהול תוכניות בדיקה**, אשר ניתנת להתממשקות לתוכנות **ניהול הכשלים** החנימיות, BugZilla™ או Mantis™. היתרונות והחסרונות של תוכנה נרכשת לעומת תוכנה חנימית – תקפים כאן.

תפקודים נוספים של תכנת ניהול תכניות הבדיקה, בשלב **יישום** התכנית. במידה ובדיקות בוצעו ידנית/אוטומטית ע"י – מסמן הבודק V אלקטרוני, ליד סעיף הבדיקה. במידה וחלפה תקופה מבלי שבמהלך

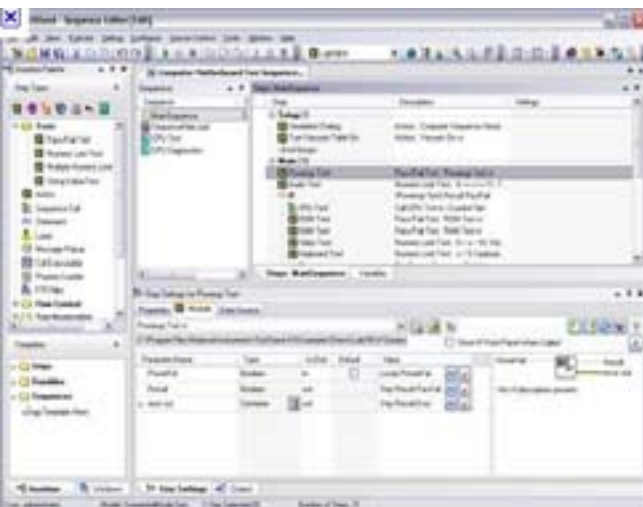
(מערכות) – כתוכנה אחת ולא כאוסף של תוכנות משנה המתאימות כ"א למונה (מערכת) נפרד.

יתרונות המחשוב: מניסיונו הרווח שבאוטומציה גדול, אך מדובר בפרויקט ארוך טווח של מחשוב, שיש לתכנן בו את החזר ההשקעה בתמורה, בשיטת הפונקציונאליות ההדרגתית – בכל רגע נתון קיים רווח. ביישום האוטומציה מרחב הבדיקות שמבוצעות גדל, ע"י הוספת סטטים עם ואריאנטים של פרמטר אחד בכל הרצה. מתגלים כשלים שלא נתגלו בעבר משום שלא הייתה אפשרות להריץ היקף בדיקות כזה. לעתים כשלים אלה הם משמעותיים. המחשוב אינו מייצר את העובדים, אלא הופך אותם למנתחי כשלים בהיקף נרחב, ומפנה אותם לעסוק במשימות ברמת מופשטות גבוהה יותר. כמו כן המחשוב מצמצם את הסיכוי שיברח באג, היות ומרחב פונק. גדול יותר מכוסה. חשוב לציין שיישום אוטומציה בבדיקות אינו טריוויאלי. חייב להיות לדוגמה ברקע מנגנון זיהוי ריצה לא הגיונית (sanity checker) שיאפשר ריצה מושכלת.

אפשרויות יישום: אנו מציגים מגוון אפשרויות המתאימות למעבדות שונות.

רכש פלטפורמה קיימת (Test Bench) מליאה הכוללת תוכנה וחומרה. דוגמא: Test Bench של ZERA לבדיקת מונים. עלות: עד \$150K. פלטפורמה זו מכילה שפת פיתוח בדיקות המתאימה במיוחד למהנדסים שאינם מתכנתים. לחילופין הזמנת חברת פיתוח בדיקות שתפתח מבודק. העלות כאן גבוהה עוד יותר משום שהמבודק ייעודי. גם פלטפורמות אלה כוללות שפת פיתוח המאפשרת מידה של עצמאות אח"כ. יצרני ציוד אחדים, מוסיפים **מודולי תוכנה לבדיקות** ברמת ממשק גראפי. להערכתנו מודולים אלה הם קריטריון מכריע בהעדפת ציוד אחד על משנהו.

תוכנות שפת סקריפטים **ייעודית** לבדיקות, המתאימות למפתחים שאינם מתכנתים. דוגמאות לשפות הן TestShell של חברת Qualisystems, או Easy test – כל איש צוות מרמת טכנאי, בעל גישה לתפעול תוכנות, מסוגל לפתח תוכנית בדיקה. **החסרונות:** עלות גבוהה מאד במחיר רישיון לשנה, חתונה קתולית עם יצרן התוכנה,



תרשים 5: מימשק דוגמא של תכנת ניהול בדיקה בשלב ריצה TestStand NI™.

בחח"י – מאורכבים בתוכנת ניהול גרסאות מישנה. זו תוכנה מוכרת לכל העוסקים בפיתוח תוכנה – משום ששם הפרויקט (קוד התוכנה) הוא בסיס נתונים משותף המפותח ע"י כל המתכנתים. בחח"י מוכרת תוכנת NATA – כתוכנה תקנית לארכוב מסמכים. חסרונה, לפחות באופן שבו היא מותקנת בחח"י, שזהו בסיס נתונים משותף לציבור רחב החורג בהרבה מצוות הפרויקט, וכן אי התאמה לארכוב תוכנות. תוכנות צריכות להפרש עם כל ספריות התוכנה הנלוות (DLL), ולפעמים בגרסת התקנה. תוכנה לארכוב מסמכים בלבד, מתקשה לתמוך בארכוב מבנה תוכנה. אנו מזכירים 2 **דוגמאות** כלים, לא למטרות פרסום, אלא ע"מ לתת פתרונות יישומיים (קיימות דוגמאות נוספות בתעשייה). תכנת Visual Source Safe מתוצרת Microsoft™. על תכנה זו קיימת ברשת ביקורת, בעולם התכנות של פרויקטים מסיביים, אך בכל הנוגע לארכוב גרסאות כלי תוכנה המגיעים מיצרן – היא מצוינת, וכן בנוגע לפשטות הממשק ובהירותו. עלות רישיון תוכנה זה סבירה ואפשרית לארגונים קטנים כגדולים. קיימת תוכנה חנימית מצוינת בשם TortoiseSVN™. הממשק שלה פשוט להבנה, אך הוא פחות מתאים להשקפתנו למשתמשים שאינם מתכנתים. תוכנה זו מקבלת ביקורות מצוינות ברשת בכל הנוגע לעולם התכנה. שם היא מצטיינת במניעת קריסה, של בסיס נתונים המפותח ע"י קבוצה, וביכולת שחזור במקרה קריסה.



תרשים 4: ממשק תוכנה לארכוב קבצים לדוגמא Tortoise™ SVN משולב טבעי במערכת ההפעלה Windows™ המוכרת –

I.4 אפליקציה לניהול תכניות בדיקה אוטומטיות בשלב ההרצה (Automatic Test Program Manager)

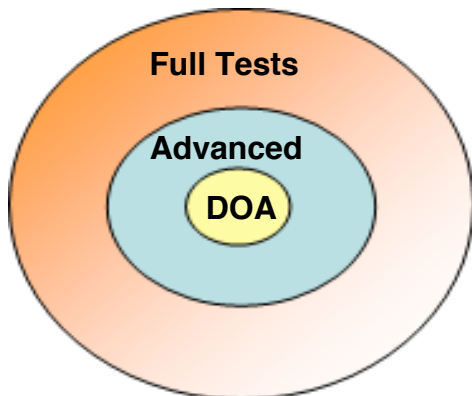
הווקטור החמישי שאנו מציעים ליעול העבודה, הינו אימוץ פלטפורמה לאוטומציה תהליכי בדיקה. 4 הווקטורים הקודמים שהוצגו, ניתנים ליישום גם כאשר תכנית הבדיקה ידנית – ונותנים הגברת תפוקה בעלות סבירה של השקעת משאבים: כסף, זמן, כוח אדם. הווקטור החמישי הוא הקשה ביותר ליישום ואינו תנאי הכרחי ל 4 הווקטורים הקודמים. במאמר הקודם הצגנו כבר כיצד אוטומציה של תכנת בדיקה, מצמצמת את נפח הבדיקה בקיום N דגמי מונים, מ $O(N)$ ל $O(1)$.

הדגש העיקרי להשקפתנו לגבי מעבדה שמחליטה לבצע אוטומציה, הינו ליישם תוכנה אחת שמתאימה לכל סוגי המונים

כשלים, ומתוקנים ע"י היצרן. מתקבלת גרסת מונה חדשה – בעיקר שינויי קושחה (firmware) וכלי תוכנה נלווים. מבוצע סבב נוסף, עד להתכנסות לגרסה יציבה ונקייה מכשלים מהותיים. בסבב בדיקות, בעיקר מישני, ובעיקר בתהליך בדיקה ידני, להערכתנו לא ניתן לבדוק הכול תמיד, ביעדי הזמן. יש להשקיע בתבונה בבחירת כיסוי הבדיקות, ע"מ לקבל אותו כיסוי באגים בפחות זמן.

הכלל הראשון הוא **תעדוף הבדיקות** כך שבכל שלב נתון יש תמונה רוחבית של מצב ווריפיצקית המונה. אנו מתעדפים את הבדיקות ב 3 רמות חומרה:

1. **Dead-Or-Alive**. זו המעטפת שחובה להריץ תמיד. היא כוללת בדיקת פונקציונאליות בסיסית שבלעדיה אין טעם להמשיך בבדיקה. אנו ממליצים להכניס במעטפת זו 20%-10 מכלל הבדיקות.
2. בדיקות מתקדמות **Advanced Tests** – מעטפת בדיקות שבהיקף מצומצם יחסית – כ 40%, נותנות רמת סמך יחסית גבוהה – כ 70% לגבי הפונקציונאליות הכוללת של המערכת.
3. בדיקות מליאות **Full Tests** – מכלול הבדיקות. בנוסף אנו מציעים לחלק את אישורי גרסת מישנה של מונה (=מערכת) בסבבי בדיקות ביניים, לפני אישור סופי, **לשתי רמות קושי:**



- **תרשים 7:** מעגלי בדיקה בעדיפויות שונות

1. גרסה עם שינויים מהותיים ביחס לגרסה הקודמת (**Major Revision**). הקריטריון המגדיר פשוט: או שבוצע שינוי מהותי בלפחות אחד ממכלולי המישהנה, או שיש אוסף של שינויים לא מהותיים רבים. במקרה כזה של שינוי המוגדר מהותי, אנו מבצעים סבב בדיקות **מלא** או כמעט מלא (**Advanced Tests**).

2. גרסה עם שינויים קטנים ביחס לגרסה הקודמת (**Minor Revision**). גרסה עם שינוי במכלול מישהנה, כשהשינוי אינו משפיע על פונקציונאליות אחרת. בנוסף כמות שינויים קטנה. במקרה כזה איננו בודקים הכול, אלא רק את תוכנית הבדיקה של מכלול המישהנה ומכלולי מישהנה מושפעים.

מיפוי בדיקות לפי מהות השינויים בגרסת מישהנה למערכת. בסיום אנו מציעים שיטה לבחירת הבדיקות שיש לבצע בהתייחס לשינוי מסוים:

1. יש לכלול פרקי בדיקה של המכלול שהוכנס בו שינוי.

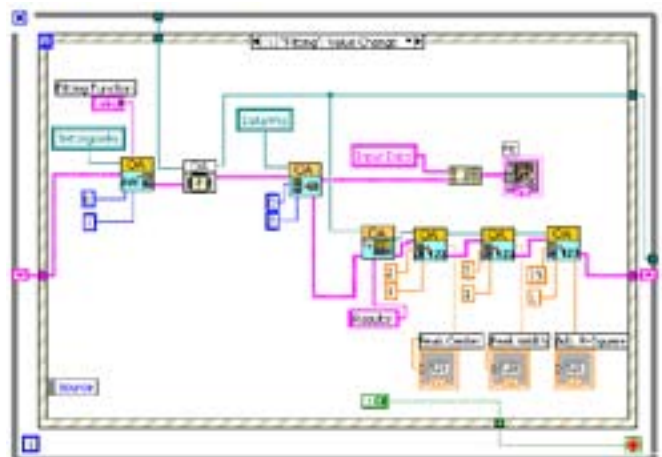
נטייה להזדקק לשרות תמיכה עקב דרייברים הנשברים בשינויי תוכנת בדיקה.

שימוש בתוכנת ניהול תכניות בדיקה לשלב ההרצה, כדוגמת **TestStand™** של חברת **National Instruments**. זוהי פלטפורמה בעלות מצומצמת יחסית, ונקייה מבאגים משמעותיים, המעניקה מגוון תהליכים שהאדם שמנהל את תכנית הבדיקה אינו רוצה לעסוק בפיתוחם: ממשק גראפי לסימון הבדיקות – באיכות של תוכנת הדוא"ל **Outlook**, בסיס נתונים לשמירת התוצאות, דו"חות מפורמטים אוטומטית, הרצה חוזרת של כשלים בלבד. מנהל תוכנית הבדיקה צריך "להחזיק" מפתח קוד שמתמחה בכתיבת **בדיקות** בלבד – הפעולות האוטומיות, אך לא בפיתוח ממשקים, דו"חות ובסיסי נתונים.

V.5 מימוש אוטומציה של הבדיקות הספציפיות

השלב האחרון הינו **בחירת השפה** שבה נכתבים הטסטים הבודדים. המנהל צריך לעסוק במה ובאיך ברמות מופשטות גבוהות. המתכנת גם הוא צריך סביבת יישום שתאפשר לו להשאיר ברמות מופשטות גבוהות. קיים מגוון שפות מתאימות – מותאם לפי יכולות הבודקים שקיימים במעבדה.

- שפת **.NET (Visual Basic ו C#)**. אנו ניסינו את **.NET**. ומצאנו שלאחר הכשרה קצרה, כל קבוצת המהנדסים היו מסוגלים לפתח קוד בדיקתיות בקלות. שפת **CVN** ייעודית. השפות קולחות, קיימות ספריות שרות רבות מספור, עם תיעוד מרשים ודוגמאות קוד חיות להורדה מהרשת.
- שפת **LabView** – שפה סימולטיבית בעלת יכולות מרשימות, בדומה ל **Matlab Simulink**. החיסרון: מצריך מתכנת מיומן. יתרון כתיבת טסטים בודדים בשפת תכנות: עלות רישון בלבד, והפצה חינומית של הבדיקות ללא עלות. בשפה כגון **NET** עלות הרישון נמוכה, ובניגוד לדיעה הראשונית – קלות השימוש רבה.



- **תרשים 6-א:** שפת כתיבת בדיקות סימולטיבית **LabView™**. מצריך מתכנת מיומן.

V. כללים לניהול בדיקות יעיל

כל אישור מערכת פונקציונאלית המשלבת חומרה ותוכנה, כדוגמת מונה, כולל מספר רב של סבבי הרצה. במהלך סבבי הרצה מתגלים



איציק משיח – מנהל מעבדת פיתוח אמצעי מניה. החל את עבודתו בחברת החשמל בשנת 1986. אחראי על נושא מניית אנרגיה אלקטרונית. בחברה, בכלל זה מונים אלקטרוניים. לתעשייה, לצרכנות הביתית ופרויקטים מיוחדים. בעל תואר ראשון בפקולטה להנדסת חשמל מהמכון הטכנולוגי בחולון. תואר שני במינהל עסקים במכללת דרבי שלוחה של אוניברסיטת דרבי באנגליה. בוגר קורסים מקצועיים רבים וקורסי ניהול בחברת החשמל.

אלי אברמוביץ – רמ"ד מכרזים במעבדת פיתוח אמצעי מניה. אחראי להגדרת מפרטי מכרזים, ולתהליך בדיקות קבלה פונקציונאליות של מגוון מונים במעבדה. בעל ידע טכנולוגי מומחה בכל תחומי הנדסת מערכת מנייה. משרת בחברה 23 שנה. בעל תואר ראשון בהנדסת חשמל ואלקטרוניקה מאוניברסיטת תל-אביב.



ניצח קלמור - מהנדס מערכת במעבדת פיתוח אמצעי מניה. בעל תואר ראשון ושני בהנדסת חשמל מהטכניון. החל דרכו בחיל אויר כקצין תחזוקה בשנים 1989-1995 ושימש כמפקד מעבדה וסגן מפקד ב"מ שם. עבד כעשור בחברת אינטל בתפקידי וורייפיקציה וולידציה ועוד כחמש שנים בחברת אלספק בתכן ובדיקותיות מכשור. הצטרף לחברת החשמל 2009 כיום במעבדת פיתוח אמצעי מנייה, בקליטת מונים חדשים ובסיוע בהגדרת מכרזים למונים חדשים. בעל מומחיות טכנית במגוון נושאים. הנדסת מערכת, אוטומציה של תהליכים, מערכות הספק ואחר.

2. יש לכתוב וליישם תכנית בדיקה ייעודית לשינוי שהוכנס במכלול, הכוללת **בדיקות חדשות**. יש להכניסן לספר הבדיקות בקביעות.

3. יש לכלול פרקי בדיקה של מכלולים המושפעים מהשינוי שבוצע במכלול אחר. לדוגמא: שינוי קושחה (firmware), מקרין על כל מכלולי המערכת. כמו כן ההסתברות ששינוי בקושחה יפגע בפונקציונאליות אחרת גבוהה. אם הקושחה נכתבת במקור בשפה תחתית (Assembler) – הנחת היסוד היא שהכול עלול להשתנות. אם הקושחה נכתבת במקור בשפה עילית ומהודרת (compiled) לשפת Assembler, הנחת היסוד היא שבד"כ רק פונקציונאליות ישירה לקוד משתנה. מידע כזה ניתן להשיג מהיצרן, או לא להניח כלום.

VI. שיקולים בהגדרת מפרטי מכרזים למונים חדשים

בהסתמך על ניסיונו הרלוונטי לכל מערכת באשר היא הפקנו כללי יסוד:

1. במכרזים עתידיים נשאף למוצר מדף ככל האפשר, עם כמות שינויים המבוצעים לטובת חח"י קטנה יותר מבעבר. להערכתנו זה יצמצם את כמות אי ההתאמות.

2. נשאף להאחדה של פרוטוקול התקשורת ל **DLMS/COSEM** עבור מונים, ול **ModBus** עבור מנטרי איכות חשמל. האחדת הפרוטוקול לא רק מביאה למונה קרוב יותר ל Plug-n-Play, אלא חלק ניכר מהבדיקות הפונקציונאליות, כוסה ע"י היצרן בקבלו את הסרטיפיקט לפרוטוקול זה. לכן אנו סבורים שהמונה יהיה נקי יותר מאי-התאמות.

קוראים המעוניינים במידע נוסף (מצייני מקום ברשת להורדת תוכנות ומדריכי הפעלה) מוזמנים ליצור איתנו קשר בכתובת המייל: ui703@iec.co.il