לחפש באגים

מדריך מעשי לבודק התוכנה

מהדורה שלישית

אוריאלה כהן

הוצאת הוד-עמי
ספרי מחשבים

Search for Bugs

By **Uriella Kohn**

עריכה לשונית ועיצוב: **שרה עמיהוד, יצחק עמיהוד**

ת.ד. 6108 הרצליה 46160

טלפון: 09-9564716 פקס: 09-9571582

www.hod-ami.co.il

הודפס בישראל, יולי 2011

מסת"ב ISBN 978-965-361-390-4

לשם שטף הקריאה כתוב ספר זה בלשון זכר בלבד. ספר זה מיועד לגברים ונשים כאחד ואין בכוונתנו להפלות או לפגוע בציבור המשתמשים ו/או המשתמשות.

...קב-ווו אפצף קיצלן, ואפיקו קבלע קפלו
סמפ בפ אוקבפ קסוון וואוסיפ,

תודות

כתיבת הספר הייתה מהנה מאוד כשלעצמה, אבל הביקורת והתמיכה היוו את המרכיבים החשובים ביותר באירוע הוצאתו, ועל כך אני מודה למתי כהן על האמון המלא ביכולות שלי, לארזה הקר ומלכה חי על הביקורת הבונה שתרמה כה רבות לתוכן הספר. תודה מיוחדת לספי גלעזר ולאיתמר היידנפלד, המנהלים שלי ב- V-NESS, על האמון והפירגון.

תוכן עניינים מקוצר

תוכן העניינים

הקדמה

מדוע נכתב הספר

הסיבה לכתיבת הספר נובעת בעיקר מנושאי העיסוק שלי משך השנים, ומההכרה שאין ספרים מתאימים בעברית בנושאי הבטחת איכות ובדיקות תוכנה, ובעיקר – לא ספרים מעשיים, הנותנים את ההיבט המעשי לנושאים אלה, ומענה לשאלה "מה בדיוק אני צריך לעשות כדי לבדוק תוכנה?". **מהדורה שלישית זו היא מהדורה מורחבת הכוללת היבטים נוספים החשובים לבודק, כמו: SQL ועבודה עם או בלי גרסאות.**

אתחיל בהצגה עצמית: את הקריירה שלי בעולם המחשבים התחלתי כמפתחת, כאשר עוד היו בארץ 5 מחשבי-על גדולים, וכאשר חברות קנו שעות שימוש בהם. אחרי 25 שנים בפיתוח – כתוכניתנית, כמנתחת מערכות וכמנהלת פרויקטים על מערכות שונות, מחשבים שונים וטכנולוגיות מגוונות – "נפלתי" במקרה לנושא הבדיקות. זו הייתה תגלית נהדרת עבורי: גיליתי נושא חדש ומלהיב הדורש יצירתיות רבה מאוד.

במשך 15 שנות עבודתי בנושא זה ביצעתי עבודות שונות: בדקתי מערכות, ניהלתי פרויקטי בדיקות והקמתי מערכי איכות בארגונים. כיום אני משמשת כמתודולוגית הראשית באחת מחברות ההי-טק הגדולות בארץ, ומלמדת בקורסים המכשירים בודקי תוכנה. בעקבות העיסוקים המגוונים שלי, נתקלתי לא פעם בעובדה שחסר חומר ראוי כתוב בנושא. קיימת ספרות תיאורטית רבה, אך היא אינה מייצגת את עולם הבדיקות בפועל. הידע של "מה עושים, איך ומתי" עובר בעיקר מפה-לאוזן ונרכש דרך הניסיון. כל אלה תרמו ללידת הרעיון לכתיבת ספר זה.

מבחינת תוכנו של הספר, אני חייבת לציין מספר נקודות חשובות:

1. על כל שאלה בה נתקלתי, מצאתי אמנם אוסף גדול מאוד של מידע, אם בספרים ואם במאמרים רבים באינטרנט, אבל כל אחד שונה מהשני. אין זה מצביע על כך שמקור אחד הוא נכון ואחר שגוי – כולם היו נכונים! הסיבה למגוון הרב של הגישות לבדיקות תוכנה היא בכך שנושא הבטחת האיכות ובדיקות התוכנה איננו, ולא יכול להיות – מוגדר תחת מסגרת חוקי פעולה קבועים ונוקשים. כל העבודה והפעילות בנושאים אלה חייבת להיות מותאמת לאוסף של פרמטרים המגדירים את הארגון, את המערכת, צורת העבודה, הדרישות וכו'. כלומר: כדי להחליט כיצד

לבדוק מערכת מסוימת – חייבים להגדיר תחילה את הפרמטרים המעורבים (כמו: סוג המערכת, האם היא חדשה, מתוחזקת או תוכנת מדף מותאמת וכו'), ולהפעיל הרבה דמיון ויצירתיות כדי לבנות את הדרך היעילה ביותר לבדוק אותה.

2. לאור מה שציינתי קודם – ייתכן שתמצאו בספר זה נושאים אשר התייחסותי אליהם לא תהיה תואמת למקורות מידע אחרים בהם תיתקלו. במקרה זה זכרו רק, שההתאמה היא תמיד לאוסף הפרמטרים המגדיר את צורכי העבודה שלכם. ספר זה נותן לכם את הגישה המעשית, וכלים לביצוע החלטות ושינויים, כאשר נדרש.

3. הספר תוכנן ונכתב בצורה מעשית, שתוכל לא רק לספק מידע על הנושא, אלא גם לתרום לעבודה השוטפת, לכל בודק ואיש הבטחת איכות, בכל ארגון ולכל מערכת. כל אלה – על בסיס ניסיוני האישי בעבודה עם ארגונים שונים ועם מערכות שונות.

מה זה הבטחת איכות ובדיקות תוכנה?

אתחיל דווקא עם כמה עובדות עצובות:

סטטיסטיקות אחרונות טוענות שגלישה של 100-200% בזמני פיתוח ובתקציב הפיתוח של מערכות תוכנה היא גלישה כמעט מקובלת. יחד עם זאת – מפתחים עדיין "מבזבזים" 34% מזמנם על תיקוני תקלות.

מכאן אפשר להבין שני דברים מרכזיים: האחד, שכנראה משהו לא בדיוק תקין בתהליך פיתוח התוכנה, אם הגלישה בלו"ז ובתקציב היא כל כך גדולה; והשני, שאיכות המערכות – למרות הגלישה – עדיין נמוכה, אם שליש מזמן המפתחים "מבוזבז" על תיקוני תקלות.

נושא הבטחת איכות ובדיקות תוכנה מיועד לשפר את שני הליקויים האלה: גם לשפר את תהליך הפיתוח על כל שלביו, וגם להבטיח איכות טובה יותר למערכות המפותחות. ובוודאי תקשו ותאמרו: אם הסטטיסטיקה הזו עדיין שרירה וקיימת – אז אולי המצב לא בדיוק משתפר עם הבטחת האיכות ובדיקות התוכנה?

ובכן, בתקופה בה החלו לפתח מערכות בסדרי גודל משמעותיים, השתמשו במחשבי-על הנחשבים עד היום למחשבים יציבים, בשפות תכנות כמו Cobol שעברה כבר דרך ארוכה של ייצוב, וטכנולוגיות פשוטות יחסית. ואולי יותר מכל – המשתמשים היו מוכנים לקבל כל דבר. באותם זמנים אם הייתה בעיה כלשהי עם התוכנה – ב-99% מהמקרים הדבר

נבע משורת קוד לא תקינה. כל נושא הבדיקות אמנם לא היה קיים באופן רשמי, אבל הבטחת תפקוד המערכות היה פשוט למדי.

כיום המצב מאוד שונה, בשני היבטים עיקריים: האחד, הטכנולוגיות מורכבות יותר, כלי הפיתוח משתנים ומתחלפים בקצב מסחרר, והדבר אינו מאפשר לאף אחד מהם להגיע למצב של יציבות, והלקוחות או המשתמשים דורשים איכות. השני, מערכות תוכנה הפכו לחלק מרכזי כמעט בכל דבר שאנו עושים בחיי היום-יום שלנו – אם זה הטלפון הסלולארי בו אנו משתמשים, הרכבות המנוהלות על ידי מחשב, חשבון הבנק שלנו ואפילו מנוע המכונית. במצב כזה, הנזק אשר מערכות לא תקינות יכול לגרום הוא עצום – כלכלית, חברתית ואפילו בהשפעתו על חיי אדם.

יחד עם כל זה – המצב משתפר (היה בהחלט גרוע יותר...), אבל התהליך איטי, בעיקר בגין החדירה האיטית של הנושא למודעות הארגונים ולמוכנות שלהם להשקיע בו.

כשעזבתי את עבודתי בפיתוח לטובת הבדיקות, ניגשתי לחברה הקודמת בה עבדתי (שפיתחה מערכות לבנקים קטנים) ואמרתי לבוס שם: "מדוע לא תביא את התוכנה שלך לבדיקות? היא הרי מלאה תקלות!" הוא ענה: "בשביל מה? הלקוח יתלונן – אז נתקן!" גישה כזו – כבר אינה מתקבלת כיום, ולקוחות דורשים יותר איכות.

מידע נוסף: כשהתחלתי לעסוק בנושא זה, ההשקעה בבדיקות תוכנה היוותה 5-7% מכלל עלות פרויקט הפיתוח. כיום, 12 שנים מאוחר יותר, הממוצע עומד על 30%, כאשר בארגונים מסוימים ההשקעה מגיעה גם ל-50%. ולדעתי – היד עוד נטויה.

חלוקה ל- QA ו-QC

כפי שניתן לראות מהכותרת – הנושא מורכב משני חלקים: הבטחת **איכות תוכנה** (QA), ו**בדיקות תוכנה** (QC). אלה הם שני מושגים שונים:

הבטחת איכות תוכנה – QA (Quality Assurance) מיועדת לצמצם נזקים בתהליך הפיתוח על כל שלביו: גורמים רבים ושונים מעורבים בפיתוח המערכת, במהלכו מתקבלות החלטות טכניות שונות, עומדים בפני מצבים רבים לא צפויים והרבה מאוד אפשרויות. כל אלה מייצגים מערך שיכול להיות מאוד בעייתי אם אינו מתבצע באופן תקין ומסודר. הבטחת איכות צריכה להבטיח תהליך כזה, והנושא מורכב **מפעולות מנע**, כלומר – פעולות שמכוונות למנוע נזקים – אם לאיכות המערכת הנבנית או ללוחות הזמנים ולתקציב.

בדיקות תוכנה – QC (Quality Control) מיועדות **לזהות** תקלות בתוצר הסופי: המערכת תוכננה ופותחה, היא כבר קיימת ופועלת לכאורה. תפקיד הבדיקות הוא גם לגלות בה תקלות (פעילות לא תקינה) וגם לזהות אי-התאמות בין התכנון לבין התוצר הסופי הפועל.

בספר זה אתייחס לשני הנושאים הללו, אם כי הדגש יהיה על בדיקות התוכנה.

מבנה הספר

מטרת הספר להסביר בצורה מעשית ומפורטת את נושא הבטחת איכות ובדיקות התוכנה, לעזור לבודק הפוטנציאלי להגיע למצב שיידע בדיוק מה הוא אמור לבצע בתפקידו החדש. הספר מורכב ממספר חלקים, ובכל חלק פרקים אחדים:

חלק 1 – כולל הקדמה חשובה של הגדרות בסיס והסבר על תהליך פיתוח תוכנה. הבנת מערך הפיתוח הכרחית עבור הבודק בפעילותו.

חלק 2 – הוא החלק המרכזי של הספר, ומטפל בנושא בדיקות התוכנה. הוא פותח בהגדרות ובמונחים מעולם הבדיקות. בהמשך הוא מתאר ומלמד את מתודולוגיית הבדיקות. המתודולוגיה מוסברת בעזרת מערכת עזר המוגדרת בנספחים, כאשר כל הדוגמאות מתייחסות אליה לצורך המחשה והדגמה של יישום המתודולוגיה. פרקים אחרים בחלק זה מתארים מצבים שונים הדורשים התאמות למתודולוגיה, בשל סוגי מערכות שונים או שיטות פיתוח המשפיעות על צורת הבדיקות.

פרק נוסף דן גם בנושא התיעדוף. הקדשתי לנושא זה פרק מיוחד עקב החשיבות שאני רואה בו: בדיקות מאופיינות בכך שהן כרוכות בהרבה פשרות, ושיטת התיעדופים מבטיחה שפשרות אלו תהיינה "חכמות", ולא תגרורנה סיכונים מיותרים.

חלק 3 – עוסק בפעולות הבטחת איכות שונות. כל פרק בחלק זה מתאר פעילות הבטחת איכות אחרת. כל הפעילויות קשורות לשלבים השונים של פיתוח תוכנה ובדיקות.

חלק 4 – מסכם את נושאי הבטחת איכות ובדיקות התוכנה, וכולל פרק על הקשר בין תהליך הבדיקות לפעולות הבטחת איכות, הסבר על פעולות האיכות הייחודיות לבדיקות, תיאור קצר של כלי העזר הקשורים לתהליך בדיקות תוכנה, ולסיכום – הגדרת ה"בודק": מיהו, ומהן התכונות הנדרשות ממנו.

בסוף הספר נמצא את חלק הנספחים, אשר כולל את אפיון מערכת העזר אשר שימשה להסבר מתודולוגיית הבדיקות, את מערך הבדיקות המלא עבורה וגם טבלת ריכוז מונחים ומושגים, ואינדקס תרשימים וטבלאות.

חלק 1

הגדרות בסיס

פרק 1: מושגי יסוד

כללי

מהי בעצם "מערכת"? מערכת היא אוסף של תוכניות מחשב, בשילוב עם הגדרות של מאגרי נתונים מתאימים. (על פי ויקיפדיה, תוכנית מחשב היא "רצף של הוראות בשפת תכנות כלשהי, שנועדו לבצע משימה מסוימת, קטנה או גדולה, במחשב").

ומהי "מערכת איכותית"? הגיוני לומר שמערכת איכותית היא מערכת אשר המשתמש (הלקוח) מרוצה ממנה, אבל זו הגדרה כללית מדי, כי מה בעצם יגרום לשביעות רצון הלקוח מהמערכת?

אני מגדירה זאת מארבעה היבטים:

1. המערכת עושה את מה שנתבקשה לעשות: הפעולות שהיא מבצעת תואמות לדרישות, המראה שלה נוח לשימוש וקל ללמוד אותה.
2. אין במערכת תקלות קריטיות או חמורות. שימו לב שלא כתבתי "אין בה תקלות", כי במציאות – מצב כזה כמעט ולא קיים. המצב שאנו שואפים אליו הוא, שאם קיימות תקלות במערכת, הן תהיינה מזעריות וכאלו שאינן מפריעות לעבודה השוטפת עם המערכת ולביצועי המשתמשים.
3. המערכת פותחה בזמן ובתקציב שהוגדרו עבורה. מערכת שעלתה 200% או 300% יותר ממה שהלקוח הקציב, גורמת לו לנזקים, ולכן איננה מוגדרת כאיכותית (כיצד תתייחסו לטלוויזיה שעלתה לכם כפול או פי שלוש ממה שהיא בדרך-כלל עולה? האם לא תצפו שבמחיר ששילמתם תקבלו יותר?).
4. מערכת אשר ניתנת לתחזוקה קלה וזולה, כלומר: אם במהלך השנים יהיה צורך לשנות אותה או להוסיף לה חלקים – זה יהיה קל וישים במחיר סביר.

פרק 6: מתודולוגיה קלאסית

כללי

פרק זה מוקדש להסברים ודוגמאות של המתודולוגיה הקלאסית לבדיקות. בנושא בדיקות תוכנה יש ארגונים אשר התאימו לעצמם שיטות עבודה אחרות שאולי יותר נוחות להם מסיבות שונות, אבל ככלל, השיטה הנפוצה ביותר בארץ ובעולם היא השיטה המתוארת כאן. בדרך כלל גם ההתאמות האחרות עדיין מתבססות עליה.

פרויקטי בדיקות

כמעט בכל ארגון – יהיה זה בית תוכנה, בנק או מפעל – הכולל קבוצת פיתוח תוכנה, יש גם קבוצה הנקראת **QA** (Quality Assurance), **הבטחת איכות**. בחלק מהארגונים, קבוצה זו כוללת רק בודקים, וכל הפעילויות שהיא מבצעת קשורות לבדיקת המערכות המפותחות או מתוחזקות. בארגונים אחרים, קבוצה זו מיועדת לביצוע ומעקב אחר פעולות הבטחת איכות כלליות, ויש ארגונים שבהם קבוצת QA עוסקת גם בהבטחת האיכות הכוללת וגם בבדיקות תוכנה.

בדיקות התוכנה בכל אחד מהארגונים האלה יתבצעו עבור כל מערכת, על פי מתודולוגיה כלשהי שהוגדרה שם, בדרגות יעילות שונות. הבדיקות מתבצעות בחלק מהארגונים כהמשכיות ו"נגררת" של תהליך הפיתוח, ולפעמים כפרויקט בפני עצמו.

בדיוק כפי שמנהלים פרויקטי פיתוח, כך צריך גם לנהל פרויקטי בדיקות. פרויקט משמעותו ניהול עצמאי ושלם של התהליך על כל שלביו, כולל בקרה על התהליך והפקת תוצרים.

בפרויקט בדיקות יש מספר גורמים מעורבים, על פי רמת הבדיקות:

1. **בדיקות מערכת (מסירה):**

   בודק: הבדיקות נעשות על ידי בודק אחד או יותר, כתלות בגודל ובמורכבות המערכת. כאשר יש יותר מבודק אחד, קיים גם תפקיד של ראש-צוות או מנהל פרויקט.

מנהל הפיתוח: הוא הלקוח שעבורו מתבצעות הבדיקות.

מתכנת: יש פרויקטי בדיקות הדורשים גם עבודות תכנות לצורך הכנה של נתונים, השוואה בין עיבודים וכו'.

2. **בדיקות קבלה:**

בודק: הבדיקות נעשות על ידי בודק אחד או יותר, כתלות בגודל ובמורכבות המערכת. כאשר יש יותר מבודק אחד, יש צורך גם בתפקיד של ראש-צוות או מנהל פרויקט.

מנהל המוצר או נציג משתמשים: הוא הלקוח שעבורו מתבצעות הבדיקות.

מתכנת: יש פרויקטי בדיקות הדורשים גם עבודות תכנות לצורך הכנה של נתונים, השוואה בין עיבודים וכו'.

למה צריך מתודולוגיה

אתחיל בשתי הבהרות חשובות:

1. המתודולוגיה המתוארת בפרק זה מתייחסת רק לרמות הבדיקה שמעורבים בה הבודקים. כלומר, היא מתייחסת לבדיקות מערכת (מסירה) ולבדיקות קבלה. בדיקות המבוצעות על ידי המפתחים נעשות בדרך שונה, ומתוארות בפרק נפרד.
2. מתודולוגיית הבדיקות מתייחסת גם לבדיקת מערכת חדשה וגם לבדיקת מערכת בתחזוקה. כלומר, בדיקות מערכת בגין שינויים ותוספות.

אז למה בכלל יש מתודולוגיה? מדוע לא להמציא בדיקות לפי העניין לכל מערכת?

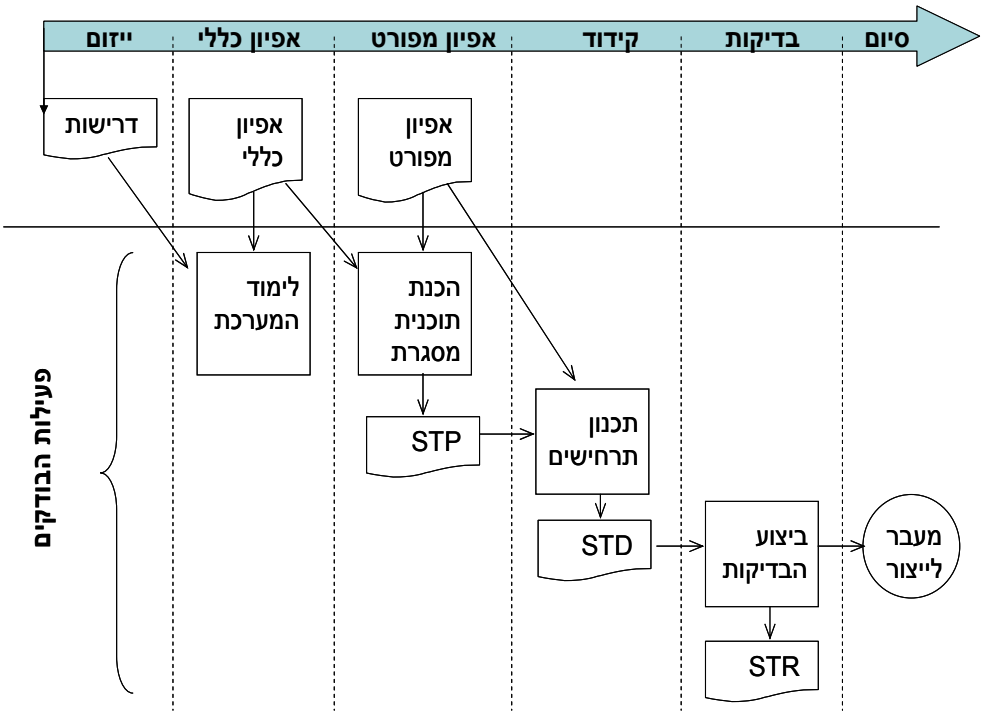
הסיבה המרכזית היא "כיסוי". אם המערכות הנבדקות היו קטנות, בגודל המאפשר ראייה כוללת של כל הפעילויות במבט אחד, אזי – נכון, ייתכן שאין צורך ממשי במתודולוגיה. אבל המערכות כיום הן רחבות היקף, וכוללות כמות עצומה של פעילויות, שכל אחת מהן אפשרית במספר גדול של צירופים. לבדיקת מערכות כאלו אי-אפשר לכתוב בדיקות מבלי להסתכן בכך שלפחות 50% מיכולות המערכת לא ייבדקו.

המתודולוגיה מיועדת להבטיח שכל פעילות וכל צירוף של פעילות – תבוא לידי ביטוי בבדיקות, כלומר: שיהיה כיסוי מלא.

מטרה נוספת, שאינה קשורה בגודל או במורכבות המערכת, היא שמירת אחידות בשיטת הבדיקה. דבר זה מאפשר לאנשים רבים להשתמש בה ללא השקעת זמן בהבנת הנעשה. מערך של בדיקות מערכת לא נועד לשימוש חד-פעמי, אלא לשימוש חוזר למשך כל חיי המערכת.

מחזור החיים של הבדיקות

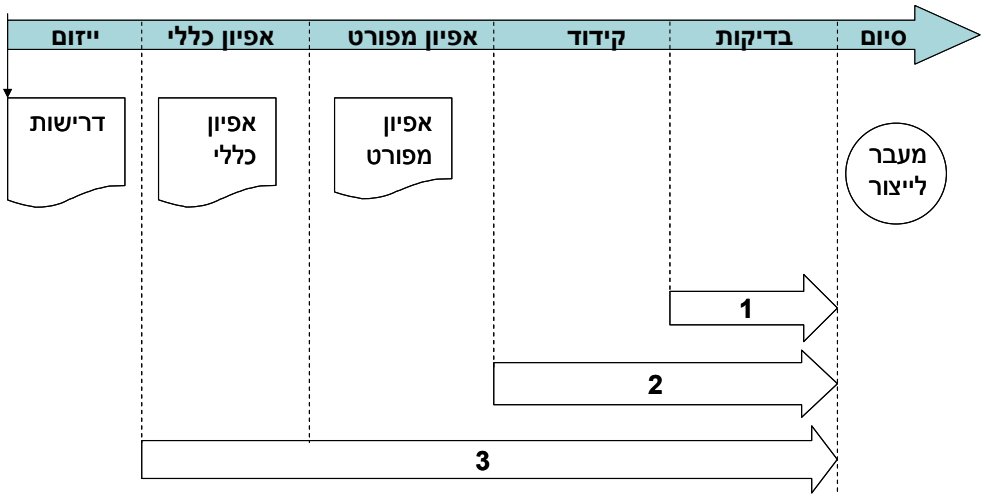
במקביל לשלבי מחזור החיים של מערכת, נמצאים גם שלבי מחזור החיים של הבדיקות, הכוללים מספר צעדים התלויים זה בזה (תרשים 8).



תרשים 8: שלבי מתודולוגיית הבדיקות

הערה: התרשים למעלה מתייחס למערכת המפותחת לפי מודל המפל (ראו חלק 1 "הגדרות בסיס"). כאשר מודל הפיתוח שונה, עדיין מתבצעות כל פעילויות הבודקים כפי שמוגדר כאן, אבל נדרשת התאמה. לפירוט ההתאמות, ראו פרק 9, "השפעת מתודולוגיות הפיתוח על הבדיקות".

מתי בדיוק מתחילה הפעילות של הבודק? לפני שנים היה נהוג להמתין עם הבדיקות עד לאחר שיש מערכת מקודדת, ורק אז להתחיל בתהליך תכנון וביצוע הבדיקות (מצב 1 בתרשים 9). אחר-כך החליטו שתהליך כזה גוזל זמן רב מדי, ולכן גם גרם לקיצורים ופשרות מסוכנות בתהליך הבדיקות. הגישה החדשה קבעה שהבודק יתחיל את תכנון הבדיקות כאשר מסמכי המערכת מוכנים ומאושרים (מצב 2 בתרשים 9). על פי הגישה העדכנית, יש לשתף את הבודקים גם בשלב אפיון המערכת (מצב 3 בתרשים 9).



תרשים 9: שילוב בודקים במחזור חיי מערכת

לבודק יש חשיבות רבה בתהליך, ובמיוחד הגישה הביקורתית והיכולת לשאול תמיד: "ומה אם...?". שיתוף הבודק בתהליך אפיון המערכת מאפשר לגלות תקלות אפיון בשלבים מוקדמים ביותר ולהבטיח שהאפיון יהיה מגובש ושלם יותר.

סיבה נוספת לצירוף הבודקים כבר בשלב תכנון וכתיבת האפיון היא עלות תיקוני התקלות, או מה שמקובל לכנות "חוקיות 1:10:100". שפירושו: כאשר מגלים איזו שהיא תקלה בשלב תכנון ואפיון המערכת, העלות שלה תהיה יחידה אחת (1); אם מגלים את התקלה בשלב הקידוד או הבדיקות, עלותה תהיה 10 יחידות; ואילו תקלות המתגלות לאחר מעבר המערכת לייצור – עלות התיקון שלהן תהיה 100.

הבודק מתבסס בתכנון הבדיקות שלו על מסמכי מערכת. בעדיפות ראשונה נמצאים מסמכי הדרישות שהלקוח או המשתמש העתידי הגדירו. אם לא קיים מסמך כזה, או

כתיבת התרחישים במסמך STD

כתיבת התרחישים מתבצעת על פי המידע הבא:

1. עץ-הנושאים אשר הוגדר במסמך STP.
2. רשימת מקרי הבדיקה (Test Cases) של כל נושא, אם הוגדרו.
3. סוגי הבדיקות אשר הוגדרו בפרק האסטרטגיה במסמך STP.

מספר נקודות לתשומת לב:

- בדיקות מסוג GUI ובדיקות פונקציונליות יוגדרו עבור **כל הנושאים** אשר בעץ-הנושאים, ולכל נושא תוגדר לפחות בדיקה אחת. יש להקפיד על כך שבדיקות אלו יכללו בדיקות חיוביות ובדיקות שליליות.
- בדיקות תהליכיות ואחרות לא ניתנות להגדרה על פי הנושאים, מכיוון שהן חוצות מערכת וכוללות מספר נושאים בכל בדיקה.
- לכתיבת בדיקות פונקציונליות ניתן להיעזר גם ברשימת הודעות השגיאה המתוכננות למערכת. רשימה כזו יכולה לתרום להבטחת כיסוי כל המצבים הלא-תקינים הצפויים.

הסעיף הבא יכלול דוגמאות למימוש על פי סוגי הבדיקות השונים. לפני הצגת דוגמאות אלו, ברצוני להוסיף מספר חוקי "אל תעשה".

<u>דוגמת "אל תעשה" 1</u>:

זיהוי בדיקה: XXX
מטרת הבדיקה: בדיקת משיכת כספים מעל יתרת האשראי.
תנאים מקדימים: מספר חשבון פרטי קיים ופעיל, ללקוח פעיל וישראלי, עם יתרה < 100₪, אחוז עמלה למשיכה מחשבון פרטי ללקוח ישראלי = 0.2%.

#	תיאור צעד	תוצאה צפויה
1	היכנס למסך טלר, הקש מספר חשבון תקין.	פרטי החשבון מוצגים
2	הקש סכום של 100₪. בחר פעילות "משיכה" בסניף החשבון על ידי בעלי החשבון. לחץ על לחצן "בצע".	הפעולה בוצעה.

<u>הערות</u>: משפט כמו "הפעולה בוצעה" (תוצאה צפויה לצעד 2) אינו משמעותי, ואינו בר-ביצוע, שהרי מה בדיוק קורה? הבודק המבצע צריך לעקוב אחר ההוראות בכל צעד, לבצע אותן ולהשוות את התוצאות שקרו בפועל לאלו שמוגדרות תחת "תוצאות צפויות". מה ישווה הבודק עם משפט כמו "הפעולה בוצעה"?

דבר שני: האם זו אכן התוצאה? האם המערכת לא עשתה שום דבר בעקבות ביצוע המשיכה? אין כל פלט? חייבים להגדיר ב"תוצאה הצפויה" מה בדיוק ביצעה המערכת כתוצאה מהפעילות ב"תיאור הצעד", ומה הן משמעויות הפלטים של פעילות זו.

<u>דוגמת "אל תעשה" 2</u>:

זיהוי בדיקה: XXX

מטרת הבדיקה: בדיקת משיכת כספים מעל יתרת האשראי.

תנאים מקדימים: חשבון עם יתרה < 100₪, אחוז עמלה למשיכה מחשבון פרטי ללקוח ישראלי = 0.2%.

#	תיאור צעד	תוצאה צפויה
1	היכנס למסך טלר, הקש מספר חשבון תקין.	פרטי החשבון מוצגים.
2	הקש את הסכום 100₪. בחר פעילות "משיכה" בסניף החשבון על ידי בעלי החשבון. לחץ על לחצן "בצע".	אם סטטוס החשבון פעיל – המשיכה תבוצע, ויווצרו 4 תנועות כספיות. אם סטטוס הלקוח מעוקל – המשיכה לא תאושר.
3	היכנס לשאילתת תנועות, ובדוק האם התנועות נרשמו במאגר לפי ערך יום הפעולה.	אם סטטוס החשבון פעיל, התנועות קיימות ותקינות. חיוב חשבון העו"ש ב-100₪. זיכוי סעיף מזומנים ב-100₪. חיוב חשבון העו"ש בסכום 0.2₪. זיכוי סעיף עמלות בסכום 0.2₪.

<u>הערות</u>: יש לשים לב לתנאים המקדימים, שאינם מגדירים מספר חשבון מסוים וגם לא את סטטוס החשבון. כעת הביטו בתוצאה הצפויה של צעד 2 שבו יש בחירה של תוצאה שונה בהתאם לסטטוס החשבון. אבל מה יקרה בפועל בזמן ביצוע הבדיקה? רק מצב אחד ייבדק: **או** משיכה תקינה לחשבון פעיל, **או** סירוב למשיכה מחשבון מעוקל. כלומר, אחד מהמצבים האלה **לא ייבדק**!

במקרה כזה, הדרך הנכונה היא להגדיר שתי בדיקות נפרדות: אחת עבור חשבון בסטטוס פעיל, ואחת עבור חשבון בסטטוס מעוקל.

<u>דוגמת "אל תעשה" 3</u>:

זיהוי בדיקה: XXX
מטרת הבדיקה: בדיקת משיכת כספים מעל יתרת האשראי.

#	תיאור צעד	תוצאה צפויה
1	היכנס למסך טלר והקש מספר חשבון תקין.	פרטי החשבון מוצגים.
2	הקש את הסכום 100₪. בחר פעילות "משיכה" בסניף החשבון על ידי בעל החשבון. לחץ על לחצן "בצע".	המשיכה מבוצעת ונוצרות 4 תנועות כספיות. (אימות הפעולה נעשה בצעד הבא)
3	אימות: היכנס לשאילתת תנועות. בדוק אם התנועות נרשמו במאגר לפי ערך יום הפעולה.	התנועות קיימות ותקינות. חיוב חשבון העו"ש ב-100₪. זיכוי סעיף מזומנים ב-100₪. חיוב חשבון העו"ש בסכום 0.2₪. זיכוי סעיף עמלות בסכום 0.2₪.

<u>הערות</u>: יש לשים לב לתנאים המקדימים שאינם מוגדרים כלל. כלומר, הבדיקה תתבצע על כל חשבון שהוא, עם כל תנאי שהוא, סטטוס כלשהו, יתרה כלשהי. במקרה זה אי אפשר לדעת מראש מה תהיה התוצאה הצפויה, ולכן גם לא ניתן לוודא שהתוצאה בפועל תקינה!

זכרו: אתם בודקים את התוכנה! אתם יודעים כיצד היא **אמורה** להתנהג בתנאים שונים ועם נתונים שונים, ואתם רוצים לוודא שאכן היא מתנהגת כפי שתוכננה!

שאלות ותשובות

שאלה: האם חובה לכתוב תמיד "מקרי בדיקה" (Test Cases)?

תשובה: אין קביעה אחת בנושא, והדבר תלוי בשני גורמים:

- תקן העבודה בארגון בו אתם עובדים. אם שיטת העבודה מחייבת הכנת מקרי-בדיקה, אז כמובן שיש לכתוב אותם.
- אם אין דרישה להכנת מקרי-בדיקה, צריך לברר אם הנושא שרוצים לבדוק מסובך ומורכב?

האם הצגת רשימת המקרים לבדיקה תעזור להבטיח שאכן בודקים כל נושא מכל הכיוונים האפשריים?

שאלה: באיזו רמה יש לכתוב את צעדי התרחישים?

תשובה: אין חוקיות לגבי רמת הפירוט של צעדי התרחישים, ולכן צריך לברר: מי עומד לבצע את הבדיקות בשלב הזה? מי יבצע את הבדיקות בעתיד? האם אלה אנשים מנוסים המכירים את המערכת? האם תחלופת האנשים בארגון גדולה, וצריך יהיה ללמד אותם שוב לקראת כל בדיקה? גישת הביניים היא: כתיבת תרחישים בצורה שתהיה ברורה לחלוטין וחד-משמעית, כדי שגם אדם חיצוני עם רקע ממוצע במערכת, ידע בבירור מה לעשות – ללא שאלות, ללא התלבטויות לגבי הכוונה.

שאלה: האם ניתן לשלב מספר הוראות ביצוע בצעד אחד?

תשובה: אין חוקיות להכנת ההוראות, ולכן צריך לברר: האם לכל הוראת ביצוע יש תוצאה צפויה שונה אותה יש לבדוק? אם כן, רצוי להפריד ולכתוב כל הוראה בצעד נפרד, כי בזמן ביצוע ההוראות משווים את התוצאה הצפויה לתוצאה שאירעה בפועל, ומסמנים את הצעד כ"עבר" או "נכשל" על פי ההשוואה הזו.

שאלה: מה עושים אם מסמך האפיון אינו כולל צילומי מסכים והמערכת עדיין לא קיימת?

תשובה: כשאין תיאור למסכי המערכת לא ניתן לכתוב בדיקות GUI. גם את הבדיקות הפונקציונליות נכתוב ברמה קצת שונה. במקום לכתוב בצעד 1:

"היכנס למסך טלר והקש מספר חשבון" ובצעד 2: "סמן פעולה = משיכה, סניף = חשבון, מבצע = בעל חשבון, נכתוב ברמה כוללת יותר: "היכנס למסך טלר ובצע משיכה על ידי בעל חשבון, בסניף החשבון".

נספח א׳ – אפיון מערכת עו״ש

1. תיאור המערכת

מערכת העו״ש החדשה מיועדת להחליף מערכת ישנה.

המערכת מטפלת במספר פעילויות בנקאיות, הן במטבע ישראלי (₪) והן בדולרים ($).

- משיכת כספים מתוך חשבון העו״ש,
- הפקדת כספים לחשבון העו״ש של הלקוח.

המערכת כוללת שני מאגרי מידע מרכזיים (בסיסי נתונים):

1. מאגר חשבונות עו״ש ללקוחות, המתעדכן בעקבות פעילות המערכת. המאגר כולל, בין היתר:

   יתרה
   גובה אשראי
   סוג חשבון
   פרטי
   עסקי
   סטטוס חשבון:
   פעיל
   מעוקל
   מוקפא
   בעלי חשבון
   יחיד
   שותפים (יותר מאחד)

2. מאגר פרטי לקוחות, משמש לתצוגת מידע בלבד, וכולל:

   ת.ז. לקוח
   שם וכתובת
   תושבות
   ישראלי
   זר

שטחים

סטטוס לקוח

פעיל

נפטר

מעמד

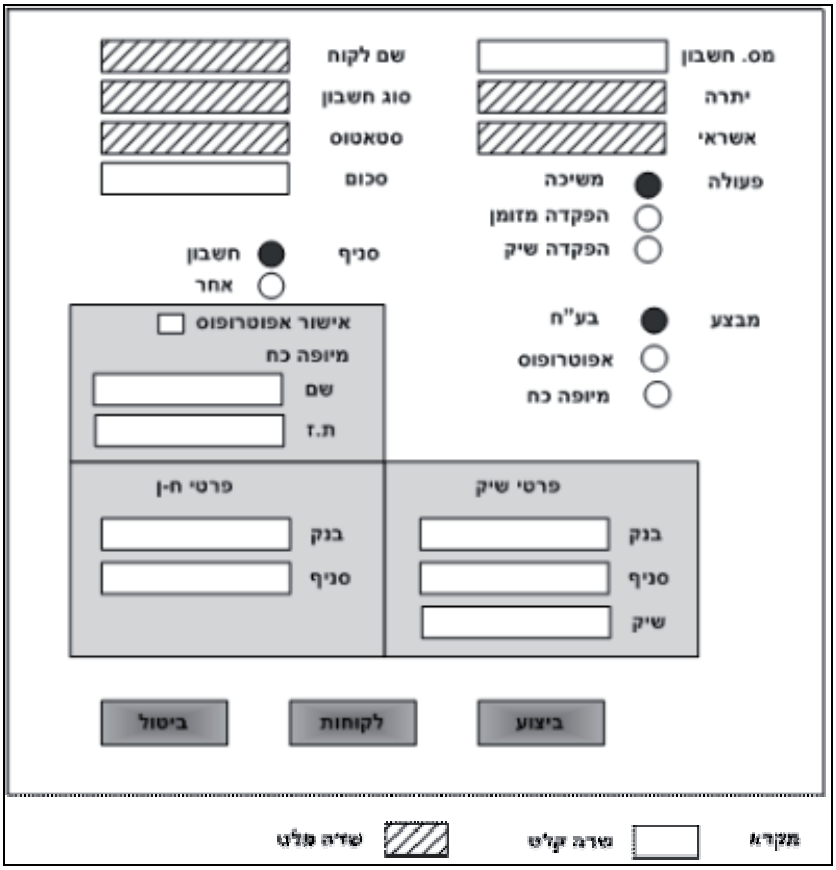
בעל החשבון

שותף בחשבון

אפוטרופוס

2. מסך הטלר

הטלר מבצע את פעולות העו"ש בעזרת מסך כללי אחד, שנראה כך:



מאפייני השדות:

שם שדה	פורמט	מגבלות
מס' חשבון	9(N)	חובת הקשה
הפעולה	RadioButton	ברירת מחדל = "משיכה"
סכום הפעולה	7(N).NN	חובת הקשה, חייב להיות סכום חיוב > 0
סניף	RadioButton	ברירת מחדל = "חשבון"
מבצע	RadioButton	ברירת מחדל = "בע"ח" (בעל חשבון)
אישור אופוטרופוס	CheckButton	חובת הקשה כאשר המבצע הוא אפוטרופוס
פרטי מיופה כח: שם	A30	חובת הקשה כאשר המבצע הוא מיופה כח
פרטי מיופה כח: ת.ז.	9N	חובת הקשה כאשר המבצע הוא מיופה כח
פרטי חשבון: קוד בנק	NNN	קוד בנק תקין מתוך טבלת קודי הבנקים. חובת הקשה בפעולה דרך סניף אחר
פרטי חשבון: סניף	NNN	קוד סניף תקין מתוך טבלת קודי סניפים חובת הקשה בפעולה דרך סניף אחר
פרטי שיק להפקדה: קוד בנק	NNN	קוד בנק תקין מתוך טבלת קודי הבנקים. חובת הקשה בפעולת הפקדת שיק
פרטי שיק להפקדה: סניף	NNN	קוד סניף תקין מתוך טבלת קודי סניפים חובת הקשה בפעולת הפקדת שיק
פרטי שיק להפקדה: מס' שיק	7(N)	חובת הקשה בפעולת הפקדת שיק

החלון הראשי הזה ישמש את כל הפעולות:

לאחר הקשת מספר החשבון, יוצגו אוטומטית כל פרטיו: שם הלקוח, יתרה, סוג חש', אשראי וסטטוס.