



מיכאל שטאל

ארכיטקט בדיקות תוכנה באינטל, ישראל. במשך 16 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. במסגרת תפקידו, מיכאל מגדיר שיטות בדיקה ומתודולוגיות עבודה, עוסק הרבה בהדרכה ולפעמים אפילו מרשים לו לבדוק משהו (שזה הכי כיף).

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

עקביות

לפני כמעט עשר שנים יצא לי לבדוק תוכנת התקנה של אפליקציה ל-Windows. ההתקנה הייתה סטנדרטית לחלוטין, כמו שכולנו רגילים. סדרה של מסכים שהשתמש בדרך כלל פשוט בוחר next next next וההתקנה מתבצעת. כל זמן שהשתמשתי בעכבר הכל היה נראה בסדר. אבל אני מעדיף מקלדת על עכבר (זה הולך מהר יותר) אז ניסיתי איך ההתקנה עובדת ללא עכבר. הסתבר שלמרות שהפוקוס בכל חלון נמצא על כפתור ה-next, לחיצה על מקש Enter לא הפעילה אותו.

פתחתי באג. אפילו הגדלתי ראש וציטטתי את מיקרוסופט:

On any Windows Form, you can designate a Button control to be the accept button, also known as the default button. Whenever the user presses the ENTER key, the default button is clicked regardless of which other control on the form has the focus.

מסתבר שלא רק אני יכול לצטט את מיקרוסופט, והמפתח החזיר לי באותה מטבע:

עקביות היא אחת מתכונות המוצר. כל מוצר. גם אם זה לא נאמר בפירוש.



Spacebar, Enter, and Esc keys. The spacebar activates the control with input focus, whereas the Enter key activates the default button. Pressing the Esc key cancels or closes the window.

מה להגיד. על הנייר הוא צודק. מצד שני טענתי, למרות שתיאורטית המפתח צודק, המשתמש מצפה ש-Enter יעבוד, כי ככה זה עובד בהמון תוכנות אחרות. למעשה, דרשתי שהתוכנה שלנו תהיה עקבית עם מה שאר השוק עושה, גם אם מבחינה "הלכתית" המימוש היה נכון.

בסופו של דבר שכנעתי את מנהל הפיתוח שעקביות היא מספיק חשובה, שווה את המאמץ של התיקון ואף גוברת על "ההלכה היבשה" על פי התייעוד של מיקרוסופט.

קל להסכים על החשיבות של עקביות בממשק המשתמש ובדרך שבה המוצר ממש פעולות שדומות אחת לשנייה (למשל: האם פעולות Open ו-Save As פותחות את אותו ממשק לצורך מציאת מיקום על הדיסק?). אבל אני מוצא שיש עוד מקומות שבהם חשוב להקפיד על עקביות (consistency) גם אם אין לכך נראות ברורה כמו ל-GUI. בנוסף, אני רואה את עקביות המוצר כחלק מהדברים שאנו, הבודקים, אחראים לוודא. הנה מספר דוגמאות לתחומים בהם העקביות חשובה ולא תמיד נשמרת:

APIs

ממשקי מערכת - וביחוד אלו שנחשפים כלפי חוץ (public APIs) - הם כר נרחב לטעויות של עקביות. אם התוכנה מציעה API עשיר, הקוד נכתב מן הסתם על ידי מספר רב של מפתחים. לכל אחד יש את "הראש שלו" לגבי שמות הפונקציות ואיך נכון לקרוא לשמות המשתנים שהן מקבלות - והתוצאה היא שהעקביות סובלת. בנוסף יש גם סתם טעויות של חוסר תשומת לב הנובעות מכך שהמפתחים כל כך בתוך הקוד שהם לא שמים לב לפרטים קטנים שיבלבלו מישהו שלא מכיר את הקוד טוב.

הנה כמה דוגמאות שנתקלתי בהן באופן אישי:

- שלוש פונקציות הוגדרו אחת מתחת השנייה באותו h file. שתיים מהם קיבלו משתנה בשם descriptorId והשלישית descriptor_Id
- חתימה של פונקציות שנקראות ללא משתני קלט. בחלק מהמקומות הוגדרה החתימה כך: functionName(void) ובחלק כך: functionName(mode)
- שם המשתנה שהפונקציה מקבלת לא "מסתדר" עם שם ה-type שלו: "Mode" (Mode area) SetArea. הוא ה-type של המשתנה. אז מה הקלט: אופן פעולה (mode) או שטח?
- מספר פונקציות מקבלות שני פרמטרים זהים ושלישי שונה. סדר הפרמטרים לא עקבי: f1(fileName, path, permission) f2(fileName, path, attributeList) f3(inString, path, filename)

כביכול, כל הבאגים האלה הם קוסמטיים אבל הם עושים את השימוש בתוכנה שלכם פחות קל ממה שהיה אפשר וגם פוגעים בדימוי של החברה שלכם כיצרון של תוכנה איכותית.

1 <http://msdn.microsoft.com/en-us/library/ms229629.aspx>
2 <https://msdn.microsoft.com/en-us/library/dn742465.aspx>



אפשר למנוע הרבה מטעויות אלו על ידי מינוי של אדם אחד אחראי להגדרת כל שמות וחתימות הפונקציות של ה-API. זה מעלה את הסיכוי לעקביות בכמה רמות; אבל גם אדם יחיד יתקשה להשתלט על תוכנה שמכילה מאות פונקציות המפוזרות במספר רב של h files, ולכן גם במקרה כזה יש לנו מה לתרום.

דוקומנטציה

חלק מתפקיד הבודקים זה לעבור על התיעוד שאנו משחררים עם התוכנה ולוודא שהוא נכון. הסברים ברורים, תמונות מסך מעודכנות ודוגמאות שאכן עובדות. כולנו מכירים כמה זה מתסכל ומרתיח כשהוראות ההפעלה לתוכנה בה אנחנו משתמשים מתייחסות לכפתור שלא קיים, אופציה חסרה או נותנות דוגמה ברורה ומפורטת אבל שממש לא עובדת.

עקביות המוצר היא חלק מהדברים שאנו, הבודקים, אחראים לוודא



אז דיוק וודאי חשוב, אבל תחום נוסף שצריך לשים לב אליו הוא העקביות. הנה כמה דוגמאות:

האם השימוש בפונטים זהה לכל אורך התיעוד (למשל: האם אותיות מודגשות תמיד מציינות אותו דבר?)

האם משתמשים באותו מונח באופן זהה לאורך כל המסמך?

האם הסגנון בו אובייקטים נקראים הוא עקבי? (דוגמה לחוסר עקביות: Filename לעומת user_name)

עקביות לא עוצרת כאן אלא ממשיכה הלאה, לתחום שמן הסתם יראה לכותב התיעוד, ביחוד אם זה מפתח ולא כותב טכני, כנדנד: שימוש עקבי במספרים (מילים או ספרות); כן או לא לשים נקודה בסוף כל bullet ברשימה; שמירה על אותם כללים בשימוש באותיות גדולות וקטנות (upper and lower case) לאורך המסמך; איות בכתיב חסר או מלא; באנגלית אמריקאית או בריטית וכו' וכו'.

מילת אזהרה: דוקומנטציה מסודרת ונקייה מבעיות קוסמטיות היא חשובה ומצדיקה השקעת מאמץ; אבל אם יש מחסור בזמן או משאבים וצריך להחליט על מה מוותרים, על בדיקות פונקציונליות של המוצר או על בדיקות רמת הגימור של הדוקומנטציה... ברור מה כאן יותר קריטי. ההצעות כאן נאמרות בהנחה שהוגדרה פעילות "בדיקות דוקומנטציה" ושמשהו מבצע את זה. במקרה כזה מוצדק לשים לב לא רק לתוכן התיעוד אלא גם על איך התיעוד נראה.

מסמכי בדיקות

לא רק המפתחים טועים בעניין עקביות. גם במה שאנחנו כותבים צריך להשקיע ולהיות עקביים. הכללים שהזכרתי בקשר לתיעוד המוצר תקפים גם למסמכים פנימיים שאנו כותבים.



לפני שנים עבדתי בבדיקת רכיבי WiFi. אחד מהתכונות של המוצר הייתה האפשרות לכבות את המשדר האלחוטי. היום כולם קוראים לזה "airplane mode", ואילו אז קראנו לזה... הנה רשימה של השמות השונים שנתקלתי בהם במסמכי הבדיקות: "RF Kill, RF Disable, RF Off". כאשר במהלך הבדיקה היה צריך לכבות את המשדר, הרי שההוראות היו:

Turn the RF Off; Set RF Kill to ON; Set the RF Kill to RF Disable ... וכו' וכו'.

מובן שזה מבלבל ומעלה את הסיכוי לטעות במהלך ביצוע הבדיקה.

קשה יותר לקרוא וקשה יותר להבין תיעוד עם עקביות נמוכה. קרל וויגרס שכתב מספר ספרים על ניהול דרישות, מדגים את הבעיה היטב. הדבר נכתב לגבי מסמכי דרישות אבל הרעיון רלוונטי לכל מסמך:

"מסמכי דרישות רבים עושים שימוש מעורב ואקראי במונחים הבאים:

shall, must, should, could, would, is recommended, is desirable, is requested, will, can, may and might.

בשיחה יום-יומית זה לא בעיה כי השומע מסתמך על ההקשר ועל היכולת לשאול שאלות הבהרה על מנת לוודא את כוונת הדובר אבל זה הופך מבלבל כשמדובר במסמך כתוב. הקורא תוהה: האם יש הבדל דק אך בעל חשיבות בין המילים השונות? האם must בעל משמעות שונה מ can? האם might זה אותו דבר כמו may?

לסיכום עקביות היא אחת מתכונות האיכות של המוצר. כל מוצר. גם אם זה לא נאמר בפירוש במסמכי הדרישות וגם אם המפתחים מתעצבנים כשמעירים על זה.

דייב ווינר, יזם ומפתח אמריקאי, **כתב כך**:

"על מנת ליצור תוכנה שימושית צריך להילחם על כל תיקון, כל התאמה שתעזור לעוד אדם אחד להשתמש במוצר. אין קיצורי דרך. זה יקרה כי נלחמת על כל סנטימטר."

זה רלוונטי לכל באג, אבל רלוונטי ביותר לבאגים של עקביות; חוסר עקביות היא אחד הדברים שפוגעים ישירות בשימוש הנוח במוצר.

<http://www.intelligentediting.com/resources/the-top-10-consistency-mistakes/> 3

RF = Radio Frequency 4

More about Software Requirements - Thorny Issues and Practical Advice, Wiegers, Karl E. 2006 p.118 5