



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 ובהול"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.

כללי כלים

כשאנחנו מדברים על אוטומציית בדיקות, האסוציאציה הראשונה היא של מערכת מורכבת: בקר להרצת בדיקות, סקריפטים, מוניטורים למיניהם, לוגים, דו"חות, ממשק משתמש ומן הסתם גם איזה בסיס נתונים שתומך בכל העניין. משהו גדול, כבד ומסובך.

אבל בעצם אין שום דבר בהגדרת "אוטומציה" שמגדיר אותה כך. ג'יימס באך, למשל, מגדיר אוטומציית בדיקות כ"כל שימוש בכלים לצורך בדיקות"¹.

דוגמאות? לאחרונה ייצרתי בעצמי, או ביקשתי ממישהו שייצר לי, את הכלים הבאים:

- "תרגום" נתונים מקובץ json לפורמט csv
- איגוד של מספר קבצי csv לתוך קובץ csv אחד
- חישוב גובה של אדם מתוך קובץ המכיל מודל תלת מימדי של אדם
- חילוץ metadata מתוך קובץ וידיאו בפורמט מסויים
- הגדלה או כיווץ מודל תלת מימדי בפקטור נתון

כתיבת כלים היא פעילות מרכזית בעבודתנו, כיוון שכך נכון לשאול איך לנהל אותה נכון.



כשאני חושב על הקבוצה שלי אני משער שיחד אנחנו מייצרים לפחות מאה כלים כאלה בשנה. כיוון שאני לא חושב שאנחנו מי יודע מה יוצאים מן הכלל, אני מניח שהמצב דומה בהרבה בקבוצות בדיקות. כלומר, כתיבת כלים היא פעילות מרכזית בעבודתנו, וכיוון שכך חשוב לשאול איך לנהל אותה נכון.

הדבר שהכי מטריד אותי סביב נושא הכלים הוא שבהכרח יש המון בזבוז משאבים סביב כתיבתם. עיקר הזבוז היא העובדה שיש מעט שימוש חוזר (re-use) בכלים פנימיים בארגון וזאת ממספר סיבות:

בארגון גדול (אפילו לא גדול במיוחד; מספיק שיש כמה מעבדות וכמה קבוצות) אין דרך לבדוק לדעת אם מישהו בקבוצה אחרת כבר כתב כלי שמספק את הצורך שלו בצורה מיידית או בשינויים קלים.

גם אם הבודק גילה שיש כלי מתאים, קשה להשתמש בו: הרבה פעמים הכלי נכתב כמשהו פרטי לשימוש הבלעדי של הבודק שכתב אותו. כותב הכלי לא ראה צורך לתעד אותו ולו גם במעט. עבור משתמש אחר שמנסה להפעיל את הכלי, החוסר בהסבר איך להשתמש בכלי מייצר תסכול מידי, הרגשה ש"זה לא מה שאני צריך", ולנטישת הכלי.

אם צריך לעשות בכלי קיים התאמות קלות, מסתבר שקשה למצוא את הקוד שלו ואם כבר מצאנו אותו אז הוא לא מתקמפל מסיבות שונות. שוב נוצר תסכול שגורם לנטישת הניסיון ל-re-use.

כיף לכתוב כלי חדש. זו משימה יחסית קצרה (אם כי יותר ארוכה משהערכנו ©) עם סיפוק מידי בסופה. אפשר גם לדווח עליה בדו"ח השבועי: "כתבתי כלי...".

התוצאה היא שכלים נכתבים שוב ושוב על ידי אנשים שונים כי ההרגשה הכללית היא שזה ייקח פחות זמן לכתוב אותם מאשר לחפש, להתקשקש אם כלי לא מובן או לריב עם קוד שלא מתקמפל.

אפשר לשפר את המצב על ידי אימוץ של מספר כללים הנוגעים לפיתוח כלים בקבוצת הבדיקות. הכללים מכוונים להתמודד ישירות עם ארבעת הסיבות שמניתי למעלה.

1) באמת אין כלי כזה?

כלל ראשון צריך להיות שלא מתחילים לכתוב כלי בלי להשקיע טיפה מאמץ למצוא אם כבר יש כלי כזה או דומה. קודם כל, חפשו ברשת. אין המון חדש תחת השמש, ויש סיכוי רב שמישהו כבר פתר את הבעיה שלך. גם אם צריך לפעמים לשלם כמה עשרות דולר על רישיון, זה לא בושה לשלם על



1 What is Test Automation? James Bach, <http://www.satisfice.com/blog/archives/118>



משהו באינטרנט, ותאמינו לי - לפתח את זה יעלה לכם יותר. זה יעבוד רק אם תארגנו מערכת מאוד זריזה וקלה לאישור ההוצאה וקניית תוכנות שירות (נגיד, עד \$100). אם קניית כלי תוכנה גוררת ימים של דשדוש ביוזוקרטי באישורים, מציאת סעיף תקציבי ומיילים הלך ושוב עם הקניין, הבודקים שלכם יוותרו על התענוג ויבזבזו את הימים האלה על כתיבת כלי שיכלו להוריד.

אבל גוגל לא ימצא לכם כלים שנכתבו בתוך הארגון. לכן: צריך לארגן רשימה של כלים פנימיים קיימים. הרשימה צריכה לכלול (כמינימום) מה הכלי עושה, מאיפה אפשר להוריד אותו ומי כתב אותו. למרות שתאורטית זה דבר פשוט, למי שכותב כלי לעצמו אין שום תמריץ להשקיע את הזמן (ולו גם 10-15 דקות) לרשום את הכלי באיזושהי רשימה כלל-ארגונית. יש גם חשש שפרסום הכלי יגרום אחריו בקשות תמיכה. לגבי כלים שהקבוצה שלכם כותבת - צריך להגדיר את הרישום ברשימת הכלים הכלל ארגונית כצעד חובה במסגרת תהליכי העבודה שלכם לפיתוח כלים. עבור קבוצות אחרות: אפשר לנסות לייצר תמריצים על ידי פרסים קטנים, ציון לשבח למנהל, הגרלה בין רושמי הכלים וכן הסכמה שפרסום הכלי לא מצוין כל מחויבות לתמיכה: "use as is and at your own risk"...

קביעת שם הכלי היא כר נרחב ליצירתיות ותוספת שעשוע לסביבת העבודה.



עוד הערה קטנה בעניין: צריך להשקיע טיפת מחשבה בקביעת השם של כלי. השם צריך לרמוז על הפונקציונליות של הכלי על מנת להקל למצוא אותו ברשימת כלים. השם צריך להיות ייחודי, אחרת תמצאו ששמות כל הכלים שלכם הם ווריאציות לא מי-יודע-מה מקוריות על "AutoTest". שם "גנרי" מכביד על התקשורת כי הוא אינו מאפשר להתייחס לכלי רק בשמו ("שתמש ב-RunTool; אבל בזה של יואב, לא בזה של שלומית..."). שם הכלי הוא גם הזדמנות להנציח את המפתח על ידי בחירת מושג בעל משמעות אישית למפתח; או השחלת ראשי התיבות של שם המפתח לשם הכלי; או ראשי תיבות שהם גם בדיחה פנימית... בקיצור - יש כאן כר נרחב ליצירתיות ותוספת שעשוע לסביבת העבודה.

(2) איך משתמשים בזה?

גם כאן יש לכם שליטה רק על עצמכם. אבל במסגרת קבוצת העבודה שלכם אפשר להסכים: לכל כלי חייב להיות readme קצר שמסביר איך להפעיל אותו. בנוסף, צריכה להיות דוגמה עובדת, כך שמי שמוריד את הכלי יכול לראות מה בדיוק הכלי עושה בלי להשקיע זמן בארגון קלט מתאים. אם הכלי דורש שמשוה נוסף יהיה קיים במחשב (תוכנה אחרת - כמו visual studio runtime) זה צריך להיות כתוב בפירוט ב-readme. אם זה רק dll אחד או שניים - פשוט לשים את הקבצים האלה באותה תיקייה שממנה ניתן להוריד את הכלי. המטרה היא שאם כבר מישוה גילה עניין בכלי, ניתן יהיה להפעיל את הכלי באפס מאמץ.

(3) איפה הקוד?

זה בעצם כל כך טריוויאלי שלא צריך לדבר על זה: הקוד של כל כלי שאתם מפתחים צריך להיות במערכת ניהול תצורה. אני לא מעליב אתכם ואניח שאתם יודעים למה זה דבר כל כך בסיסי. ברשימה הארגונית של כלים צריך להוסיף איפה הקוד נמצא. או לרשום זאת ב-readme שמלווה את הכלי.

שמירה ונגישות לקוד זה הצעד הראשון. הצעד השני לעידוד re-use זה לוודא באופן חד משמעי שמי שיווריד את הקוד יוכל לקמפל אותו ללא התברברות. הקוד צריך להיות ללא hard-wired paths (תלויות) תלויות (dependencies) צריכות להיות מבוססות על משתני מערכת שהגדרתם מוסברת בפרטנות ב-readme file; אם צריך dll או lib ספציפיים, הוסיפו אותם לקבצים שבמערכת ניהול הקוד; וכו'. הבדיקה האולטימטיבית היא לקחת מחשב שרק עתה הותקן מחדש, להוריד את הקוד ולקמפל. זה צריך להצליח.

(4) מה עושים עם הרצון לכיף?

אם אתם באמת מעוניינים לקבל יותר שליטה, צריך לקבוע כלל שלא מפתחים כלי חדש בלי לקבל אישור מהמנהל. המנהלים, צריכים לייצר מסלול מאוד מהיר לאישור הזה. בסך הכל לשאול שתי שאלות: חיפשת בגוגל? חיפשת ברשימה הארגונית? אם התשובה היא לא, אז אין אישור. אם כתוצאה מכך הבודק חיפש ומצא כלי קיים - תנו מילת עידוד על כך. אם הבודק החליט לכתוב כלי בלי לשאול - חצי מהתמריץ לכתוב אותו (דיווח וגאווה) לא רלוונטיים כי בעייתו לפרסם שכתבת משהו שהמנהל היה אמור לאשר קודם. זה לא פותר את הבעיה לגמרי אבל משנה את המשוואה ומעלה את הסיכוי לשימוש חוזר.

לפני זמן מה ייצרתי רשימה של כללים (checklist) לפיתוח כלים פנימיים. אפשר לראות עותק כאן.

אני מתריע מראש שכלל לא טריוויאלי לעמוד בכל המשימות שברשימה. למשל, הדרישה שלכלי יהיו בדיקות מינימליות (ואוו... איזה רעיון חדשני!... כלי תוכנה צריך להיבדק!) דורשת לא מעט השקעה. מעבר על הקוד (code review) גם הוא לא משהו קל. לעומת זאת יש ברשימה לא מעט דברים שקל ליישם במאמץ קטן. הרשימה אינה "הכל או לא כלום". כל פרט ממנה שתבצעו יגדיל במעט את הסיכוי לשימוש חוזר. אפשר להחליט מה הם הכללים שאתם מאמצים כ"הרג ובל יעבור", מה הם אלה שמומלצים בחום, ומה הם אלה שלא נראים לכם ולכן לא יופיעו ברשימה שלכם. וכמובן שאפשר להוסיף כללים אחרים שלא חשבתי עליהם (ואשמח אם תשלחו לי אותם!).

בהצלחה!

