כסף זה לא הכל

אחד הדברים שמקבלים תשומת לב רבה מצד הנהלות של ארגונים זה נושא גיוס אנשים: מתי מאשרים גיוס, כמה אנשים מגייסים וכו'. זה עניין שתמיד על הפרק, ובדרך כלל קשה לקבל אישור לגייס אנשים נוספים. לעומת זאת, הרבה יותר קל לקבל אישור להוצאה כספית – במיוחד כשעובדים בחברה מצליחה ומרוויחה. זה לא שהכסף מתגלגל על השטיחים, אבל אם יש הצדקה עסקית להוצאה (הידועה בכינויה "ROI") יש סיכוי סביר לקבל אישור לקנייה.

לעובדה הזאת, לפחות בתחום שלנו, יש תוצאת לוואי לא ממש חיובית.

נניח שבנינו מערכת אוטומציית בדיקות לתפארת. המערכת מסוגלת לשלוט על מספר רב של מחשבים, לקבל רשימת בדיקות ולהריץ אותן במקביל על המחשבים שבשליטתה. קיומה של המערכת מעודד את הבודקים לאטמט עוד ועוד בדיקות, כי זה מפנה זמן לעסוק בהנדסה ממש (לחשוב על מקרי בדיקה מעניינים, לפתח כלים שיאפשרו לבדוק דברים ביתר קלות, להיות פעילים יותר בסקירות קוד ומסמכי תוכן וכו'). חוץ מזה גם יותר כייף לכתוב קוד או סקריפט מאשר להריץ בפעם השבע-עשרה את אותה בדיקה.

קבוצת הבדיקות מריצה על המערכת אלפי מקרי בדיקה בכל סבב בדיקות - גם אם הוא הסבב היומי - כי "למה לקחת סיכון שהיתה רגרסיה, הרי הבדיקות רצות בחינם, אוטומטית, אז נריץ כמה שיותר". זה אפילו מעלה את ה-ROI!

המערכת אמנם מהירה כברק, אבל בכל זאת... כשמנסים להריץ עליה עוד ועוד בדיקות מגיעים בסופו של דבר למצב שבו היא בעומס יתר, בדיקות מתחילות להתעכב והאוטומציה הופכת לצוואר בקבוק. בשלב מסוים מנהלי הפרויקט לא מקבלים תוצאות של סבב הבדיקות בזמן וצועקים געוואלד.

לכולם ברור שצריך לעבור על כל הבדיקות ולהגדיר את הקדימויות מחדש. איזו בדיקה צריכה לרוץ כל יום? מה מספיק להריץ פעם בשבוע? אם יש כבר כמה אלפים טובים של מקרי בדיקה, פעילות כזאת היא מאמץ ניכר, ואין ממש זמן לבצע אותה. כולם עסוקים עד מעל לראש בהרצת הבדיקות שעדיין נותרו ידניות, בחקירת באגים או בסיוע למפתחים בבדיקת פתרונות אפשריים לבאגים. מעבר לצורך לצמצם את מספר מקרי הבדיקה שמורצים, חלק מהבדיקות האוטומטיות לא יעיל ובמאמץ מסויים ניתן היה לקצר את זמן הבדיקה באופן ניכר. אח! אם רק היה לנו זמן! אם רק יכולנו לגייס עוד כמה מהנדסים שיעזרו בהרצות, ויפנו זמן למהנדסים הוותיקים לעבוד על שיפור הבדיקות הקיימות!

> **קשה לקבל אישור לגייס עוד אנשים, הרבה יותר קל לקבל אישור להוצאה כספית**

לגייס כוח אדם נוסף לא בא בחשבון, אבל יש פתרון שיקצר את זמן הבדיקות ב-50% ולמעשה יפתור את כל הבעיות. פשוט מאוד: נכפיל (או נשלש... או נרבע) את כמות המחשבים שמחוברים למערכת הבדיקות. זה פתרון מעשי כי קל להוכיח את ה-ROI (ולו רק בגלל קיצור הזמן להרצת סבב בדיקות שנדרש על מנת לאשר שחרור של גרסה). וחוץ מזה... יותר קל לקבל תקציב לציוד מאשר אישור לגייס עוד מישהו. גם לוקח פחות זמן לפתרון להשיג תוצאה: עובד חדש צריך ללמוד את המוצר, את הטכנולוגיה ואיך מפעילים את מכונת הקפה. מחשב חדש צריך רק חשמל וקו תקשורת והופ לעבודה. [1]

1 שימו לב שמיקבול הבדיקות על מכונות רבות עובד בקלות יחסית כשמדובר בתוכנה שעומדת בפני עצמה ללא אינטגרציה עם מערכות אחרות. למשל: עבור יישום למערכת משובצת מחשב (embedded). הדברים מסתבכים

אז קנינו עוד מחשבים וחיברנו אותם ואכן צוואר הבקבוק נעלם. אבל אחרי כמה חודשים, עם העליה במספר הבדיקות, שוב יש צוואר בקבוק.

אחת הסיבות לכך נובעת מעקרון הטרגדיה של נחלת הכלל (Tragedy of the Commons). העקרון, שנקבע על ידי הכלכלן הבריטי ויליאם פוסטר לויד ב-1833, זכה לעדכון ופרסום מחודש על ידי גארט הרדין ב-1968. על פי ויקיפדיה זהו "סוג של מלכודת חברתית, בעלת זיקה כלכלית בדרך כלל, העוסקת בקונפליקט הנוצר בין הפרטים ובין טובת הכלל בעת ניצול משאבים על ידי אותם פרטים". במערכת אוטומציה ארגונית, כאשר כל קבוצה יכולה להעביר למערכת ההרצה כמות בדיקות בלתי מוגבלת, העקרון מסביר למה אין לאף אחד אינטרס להיות יעיל.

> **כמעט תמיד הפתרון של "נזרוק כסף על הבעיה" הוא ההגיוני ביותר לשעתו**

אסביר בעזרת דוגמה:

בפרויקט מסויים יש חמש קבוצות בדיקה ולרשותן מערכת הרצת בדיקות אוטומטיות משותפת. המערכת, כצפוי, עמוסה, דבר שגורם לאיחור בהרצת הבדיקות. עקרונית, כל הקבוצות מבינות שהן צריכות להשקיע מאמץ לקיצור זמן ההרצה. אבל באופן טבעי נוצר מנגנון שמונע הליכה בכיוון זה. נניח שהרצת הבדיקות של כל אחת מהקבוצות, אם היינו נותנים לקבוצה זו להשתמש בכל המחשבים שבמערכת, לוקחת 12 שעות. כלומר, הרצת סבב בדיקות של כל הקבוצות במקביל, על כמות המחשבים הקיימת, לוקחת 60 שעות. נניח שקבוצה א' לקחה ברצינות את הבעייה הארגונית והחליטה להשקיע כל מה שידרש על מנת לקצר את זמן הבדיקות שלהם ב-50%, ל-6 שעות. הם קיצרו את זמני ההרצה של כל הטסטים; הם הורידו בדיקות מיותרות ווידאו שהבדיקות יציבות ואמינות. כמה זמן ידרש עכשיו להשלמת סבב הבדיקות? 54 שעות. כלומר: מאמץ-העל שקבוצה א' עשתה, הפחית את זמן הסבב הכולל ב-10% בלבד. כיוון ששאר הקבוצות ממשיכות לעבוד כמו קודם, הרי שהזמן היקר שקבוצה א' חסכה, "יאכל" במהרה על ידי הקבוצות האחרות. די מאכזב...

ביישומים על הרשת או בתצורת שרת-לקוח שבהן פעולה של בדיקה אחת משפיעה על בדיקות אחרות. ראו מאמר מעניין על כך:
https://techbeacon.com/app-dev-testing/parallelizing-test-automation-read-first

מיכאל שטאל

ארכיטקט בדיקות תוכנה באינטל, ישראל
עוסק בעיקר בבדיקות מערכות משובצות מחשב.
במסגרת תפקידו, מיכאל מגדיר שיטות בדיקה ומתודולוגיות עבודה, עוסק בהדרכה ולפעמים אפילו מרשים לו לבדוק משהו (שזה הכי כייף).
מיכאל מציג תכופות בכנסים בארץ ובחו"ל ומלמד בדיקות תוכנה בפקולטה למדעי המחשב באוניברסיטה העברית.
ניתן לראות חלק מהמצגות והמאמרים שלו באתר www.testprincipia.com.

מחפש צרות | מיכאל שטאל

התוצאה היא שלאף קבוצה אין ממש תמריץ להשקיע בייעול הבדיקות.

בעייה אחרת היא שריבוי בדיקות גם גורר ריבוי תקלות: פה ושם יש בדיקות אוטומטיות שנופלות סתם וצריך לתקן אותן. כמות האירועים האלה תלויה במיומנות של הקבוצה בכתיבת אוטומציית בדיקות, אבל תמיד, באופן סטטיסטי, אחוז קטן של הבדיקות יכשל סתם. פעם זה דיסק קשיח שקרס, פעם זה מחשב שהתחמם יתר על המידה. במקרים רבים הרצה חוזרת של הבדיקה, ואפילו מספר הרצות, פשוט עוברות. ככל שיש לנו יותר מחשבים, וככל שאנו מריצים יותר בדיקות, גם המספר של הנפילות שדורשות טיפול וזמן הנדסה יעלה (לא אחוז הנפילות! הוא ישאר אותו דבר).

ראיתי את התהליך הזה קורה לא רק בבדיקות אלא גם בפעילות של DevOps. כל check-in של קוד למערכת ניהול התצורה מחייב לעבור בדיקות CI. גם CI רץ על מערכת אוטומטית וגם שם יש לעניינים נטייה להתארך. זה קצת קשור לכך שעם ההתקדמות בפיתוח, יש במוצר יותר יכולות ולכן יותר בדיקות. זה יכול להיות כי נוסף פרויקט חדש במקביל לקיים. כך או כך, כיוון שהמערכת עמוסה, מתחילים האיחורים. תוצאות CI, שבתחילת הדרך לקחו כמה דקות, מגיעות אחרי חצי שעה. ניחא. המפתחים מתרגלים ש-check-in עושים לפני שיוצאים לצהריים. אחרי כמה חודשים, זמן הארוחה של תן-ביס במסעדה הקרובה כבר לא ארוך מספיק, ותוצאות ה-CI לא מגיעות גם אחרי שחוזרים. משנים טקטיקה ומתרגלים לעשות check-in לקראת סוף היום. אבל אחרי כמה זמן... הבנתם את העניין וגם DevOps פותרים אותה בקניית מחשבים נוספים.

> **כיוון שקשה לתקן את הבעייה ברגע שהיא כבר קיימת, צריך להשתדל למנוע אותה.**

> **האם אתם קורבנות הטרגדיה של נחלת הכלל?**

ברגע שהתהליך שתיארתי קורה, קשה מאוד לעצור אותו. הלחץ של העבודה השוטפת והצורך להתקדם מהר אומר שכמעט תמיד הפתרון של "נזרוק כסף על הבעיה" הוא ההגיוני ביותר לשעתו. מידי פעם כן מצליחים לעשות "בליץ" של יעילות, שהשפעתו מחזיקה מעמד כמה חודשים, ואחר כך שוב יש בעיות.

כיוון שקשה לתקן את הבעייה ברגע שהיא כבר קיימת, צריך להשתדל למנוע אותה.

צעד ראשון: להשקיע מאמץ כך שלא יכנסו בדיקות "שבירות" למערכת. אלה הבדיקות שבהמשך נופלות סתם ומאלצות השקעת זמן הנדסה לוודא אם מדובר בבאג אמיתי או בסתם טסט לא יציב. כל בדיקה אוטומטית צריכה אם כן לעבור סף מסוים של איכות לפני שהיא מתקבלת להרצה קבועה במערכת האוטומציה. ההצעה שלי היא לייצר checklist שכל בדיקה אוטומטית צריכה לעמוד בו. צריך להזהר שלא יהיה ארוך מידי (לפני הרבה שנים כתבתי רשימה כזאת שבפועל התברר שלקח יומיים לבצע אותה... מיותר לציין שהיא לא הייתה בשימוש נפוץ). משהו שלוקח חצי-שעה עד שעה לבצע נראה לי סביר. למעשה, מרגע שמכניסים checklist כזה, אני מניח שרוב הדרישות יבוצעו תוך כדי הפיתוח של הבדיקות – בייחוד כל הדרישות לגבי צורה נכונה של פיתוח הקוד.

צעד שני: להיות מודע למשמעות של זמן בדיקה ארוך, ולהשקיע מחשבה ומאמץ בקיצור זמן הריצה של כל בדיקה. בדיקות יכולות להיות לא יעילות מכל מיני סיבות, לדוגמה, כשבדקתי כרטיסי Wi-Fi, כמעט כל הבדיקות התחילו בשלב הכנה של "בצע התחברות לרשת". כיוון שכך, היה מאוד הגיוני לכתוב פונקצית ספריה של בדיקת החיבור וכמעט כל בדיקה קראה לפונקציה זו בתחילת ההרצה. הפונקציה אמנם הבטיחה חיבור טוב, אך עשתה זאת על ידי העברת קובץ הלוך ושוב מהרשת למחשב ומהמחשב לרשת. עניין קטן, שלקח זמן שעל פניו נראה למהנדס שפיתח את הפונקציה כסביר: 15 שניות. עבור בדיקה אחת זה נסבל אם כי גם זה בזבוז זמן (יש דרכים מהירות הרבה יותר לוודא חיבור). כשמריצים אלף בדיקות בסבב, זה שורף (סתם!) מעל 4 שעות בכל סבב בדיקות. דוגמה נוספת היא הכנסת wait לקוד כדי לתת למערכת לסיים פעולה או להתייצב לפני שממשיכים. אם זמן המתנה של שנייה הוא מספיק, הרי שזמן המתנה של 10 שניות – כשהוא קורה גם במאה בדיקות אחרות, בסופו של דבר יקנה לעצמו עוד מחשב או שניים. ההזדמנות הטובה ביותר לייעל כל בדיקה ולקצר אותה ככל שניתן זה בזמן הכתיבה שלה. אחר כך, קשה מאוד לעבור על כל הבדיקות אחת אחת ולנתח אותן.

צעד שלישי: צעד זה אפשר לבצע כצעד מונע אבל גם כצעד יעיל לפתרון בעיית צוואר הבקבוק אחרי שהיא כבר קיימת. צעד זה, המטפל בטרגדיה של נחלת הכלל הינו פשוט והוכח כיעיל: הגדרה מראש כמה זמן בדיקות כל קבוצה מקבלת. באופן מעשי הדבר נעשה על ידי הקצבת מכונות לכל קבוצה. כלומר, אם המערכת האוטומטית שולטת על 30 מחשבים, נקצה 6 מחשבים לכל קבוצה. עכשיו, שיסתדרו. בדוגמה הקודמת, אם מקודם לקח לסבב של כל קבוצה 12 שעות ריצה על 30 מחשבים, הרי שעכשיו סבב הבדיקות לכל קבוצה הוא 60 שעות (על 6 מחשבים). עבור קבוצה א', המאמץ להיות יעילים משתלם עכשיו בגדול: סבב הבדיקות מסתיים ב-30 שעות במקום 60. קודם כל, זה נראה נפלא בדוחות. חוץ מזה, כל שעה שנחסכה היא שעה שעומדת לרשות קבוצה א' להריץ בדיקות חדשות. נוצר מצב שבו יש תמריץ לכל קבוצה להשקיע בהתייעלות.

אז בפעם הבאה שאתם מתפתים לפתור בעייה כלשהי על ידי הוצאת כסף, כדאי לעצור לרגע ולשאול אם זה הכיוון הנכון, האם אתם קורבנות לטרגדיה של נחלת הכלל והאם הוצאת הכסף תפתור את שורש הבעיה.