



מיכאל שטאל

מה לשמור?

כאן מתחילה הדילמה: מה שווה לאסוף, באיזו תדירות וכמה מאמץ וזמן מוצדק להשקיע באיסוף כל סוג של נתונים. הכלל הראשון צריך להיות ששומרים רק דברים שיש לנו איזשהו רעיון מה נעשה איתם אחר כך, אחרת פשוט אין לדבר סוף. זה נכון שיש כאן סיכון מסוים: אנחנו עלולים להפסיד את המקרים יוצאי הדופן שבהם פרט שאף אחד לא חשב עליו הוא-הוא החשוב ביותר לצורך אנליזה מסוימת. אבל, כאמור, זה יוצא הדופן וכללית לא משתלם לאסוף טונות של מספרים רק בגלל התקווה שפעם נעשה איתם משהו. הרי בינינו, גם לאותם נתונים שברור לנו מה אפשר להפיק מהם אנחנו לא תמיד מגיעים... מה הסיכוי שנגיע לנתח את אלו שאין לנו מושג מה הם יכולים לתרום לפרויקט?...

ובכל זאת: אם משהו עולה בזול אפשר להחליט לאסוף אותו בלי יותר מידי התחבטויות. נתונים "זולים" הם כאלה שתופסים מעט נפח אחסון, האיסוף לא משנה את זמן הריצה של הבדיקות, לא מעמיס את המערכות ואין צורך במערכות

תמיכה מורכבות על מנת לשמור ולהציג את הנתונים. יש לא מעט נתונים כאלו שזמינים בקלות בזמן הריצה ורק צריך להחליט אם שומרים אותם באופן מסודר. דוגמה טובה לנתון כזה הוא זמן הריצה של כל בדיקה. זה קל לממש ואפשר לשמור את התוצאה באותה רשומה שממילא קיימת לשמירת תוצאת הבדיקה. דוגמה הפוכה היא הריצה של בדיקות במערכת ראייה ממוחשבת. בהרבה מהבדיקות בתחום זה מחשבים מדד כלשהו על כל תמונה שמגיעה ממצלמת הווידאו והתוצאה הסופית של הבדיקה היא ממוצע של הממד על כל התמונות. אז יש לנו את הממד לכל תמונה. לשמור או לא? זה המון מידע! התמונות מגיעות בקצב של שלושים בשנייה ולעיתים אף יותר כך שבקלות מגיעים לאלפי מספרים עבור כל בדיקה.

שמירת הנתונים מחייבת התקנה של בסיס נתונים מהיר ויעיל וכתובת קוד עזר לייבוא של הנתונים לתוך בסיס הנתונים. אחרי שכבר יהיה לנו את כל המספרים... מה נעשה איתם? אם כל מה שנעשה איתם זה לחשב כמה מאפיינים של המידע, כמו ערך מינימלי, מקסימלי וסטיית תקן, הרבה יותר הגיוני לחשב מדדים אלו תוך כדי ריצה ולשמור רק אותם. זו כמובן פשרה. אם נשמור את כל המספרים נוכל לראות איך הערך משתנה בזמן תוך כדי הריצה עיבוד של סדרה ארוכה של תמונות. נוכל לראות למשל שברגע מסויים בסרט יש התדרדרות במדדים, שבחישוב הממוצע לא ממש מורגשת. חקירה ברמת כל תמונה תגלה אולי צוואר בקבוק בעיבוד של סוג מסויים של מידע ויזאלי ותאפשר שיפור משמעותי בביצועים. זה נשמע נהדר... אם אי פעם נגיע לעשות את הניתוח הזה.

ההחלטה של מה לשמור היא אם כך לא קלה. פחות מידי - נצטער על זה אחר כך, יותר מידי - נשלם ביוקר על טיפול במידע שסתם מתאסף בערמות הולכות וגבוהות.

נתונים אינם נתונים מאליהם

אני מניח שכל אחד שעוסק בבדיקות מקנא לפעמים במפתחים. ולא - אני לא מדבר על העניין הזה שכולנו אוהבים להתכבין עליו, איך אין כבוד למקצוע וכו'. אני מדבר על כך שפיתוח היא לכל הדעות פעולה יצרנית. כותבים קוד; מייצרים מוצר. וכשהוא מצליח, יש סיפוק ישיר. ומה איתנו, הבודקים? מה אנחנו מייצרים? ברגעים מסויימים, כשמרה שחורה יורדת עליו, באמת נראה שכלום. ובכל זאת?

אני חושב שאנחנו מייצרים שני תוצרים עיקריים: דוחות באגים, ונתונים. אלה שני הדברים המוחשיים שאפשר לספור, לסווג ולנתח. כל השאר זה מה שנכנס תחת המטריה הרחבה ובמידת מה לא ממש ניתנת למדידה של "איכות המוצר". את זה אפשר למדוד רק בעזרת מדדים עקיפים וגם כשמדדים אלו משתפרים קשה להוכיח שהשיפור באיכות הושג בזכות עבודת הבודקים. אולי זה בזכות הארכיטקטים שתכננו מוצר נפלא או המפתחים שכתבו קוד כל כך איכותי? (אני מתכוון כמובן לקוד שתיקן את הבאגים שמצאנו 😊).

על החשיבות של כתיבה טובה של באגים נכתב כבר הרבה. גם נאמר הרבה על הנתונים שאנו מספקים לפרויקט - נתונים שמתארים את רמת האיכות של המוצר. פחות נאמר על הנתונים שאנחנו מייצרים עבור עצמנו: נתונים שבעזרתם אנחנו יכולים לשפר את עבודתנו (ראו למשל את מאמרו המצוין של אורי קופפרשמידט [בגיליון 18](#)); עוד פחות נאמר על נקודות שצריך להיות מודע אליהן כשמסתכנים איסוף נתונים ולזה אני רוצה להתייחס במאמר זה.

לא משתלם לאסוף טונות של מספרים רק בגלל התקווה שפעם נעשה איתם משהו

סוגי נתונים

ברמה הפשוטה ביותר, הריצה של בדיקות מייצרת נתון אחד עיקרי: תוצאת הבדיקות. עבר או נכשל. אבל מעבר לזה יש עוד הרבה מאוד נתונים שאפשר לאסוף. הנה רשימה חלקית ביותר; תוסיפו "ועוד ועוד" בסוף כל שורה.

- **מדידות** - הספק, מהירות ביצוע משימה, עומס על המעבד, כמות הזכרון בשימוש
- **פרטי מערכת הבדיקה** - שם המערכת, סוג המעבד, גודל הזכרון, רזולוציית מסך, סוג וגודל הכונן הקשיח, וורסיות של שירותים שונים, של מנהלי התקן (drivers), של קושחות (firmware), פרטי חיבור הרשת, וורסיה של מערכת ההפעלה
- **פרטי המעורבים בדבר** - מי כתב את הבדיקה, מי הריץ אותה, מי ויידא את התוצאות ועדכן אותן
- **פרטי מערכת האוטומציה** - וורסיות של כל חלקי המערכת, פרטי מחשבי הבקרה שמריצים בדיקות בהרצה מרחוק, מספר ההרצות החוזרות
- **פרטים על הבדיקות עצמן** - זמן הבדיקה, לוגים

בודקים מייצרים שני תוצרים עיקריים: דוחות באגים, ונתונים

על זה אחר כך, יותר מידי - נשלם ביוקר על טיפול במידע שסתם מתאסף בערמות הולכות וגבוהות.



איכות

חשוב שהנתונים יהיו נכונים.

כאילו, דא... ברור שצריך, לא? אכן מובן מאליו, אבל לפעמים לא פשוט להשגה. בעיקר בגלל שבאיסוף הנתונים מעורבת בדרך כלל תוכנה. זו, כידוע, לא תמיד עובדת בכל מאת האחוזים (מישהו אמר באגים ולא קיבל?). יותר מזה - מדובר בתוכנה שאנחנו כתבנו, ומשום מה, ברגע שזה קוד שלנו, נראה לנו שלא ממש צריך לבדוק אותו. וגם כשיש קצת בדיקות, ודאי שאף אחד לא בודק אותו כמו את המוצר. התוצאה היא שבמקרים רבים מדי המוצר שלנו (קרי: הנתונים) סובל מבעיות איכות. הנה דוגמאות לבעיות כאלה, בתחומים שהזכרתי ברשימה הקודמת. גם זו, בהכרח, היא רשימה חלקית ביותר:

מדידות - מדידה של הדבר הלא נכון, הגדרה לא נכונה של המדידה (ממוצע במקום מקסימום, או להיפך), מדידה בתדירות הלא נכונה, מדידה שמשנה את הערך הנמדד (probe effect)

פרטי מערכת הבדיקה - אי שמירה של הנתונים החשובים באמת (רשמנו את סוג המעבד, אבל לא את גודל זכרון המטמון (cache), לא שמרנו את פרטי הרזולוציה והם המשפיעים ביותר על הביצועים)

פרטי המעורבים - שמרנו מי אחראי להריץ בדיקה מסוימת אבל לא מי הריץ בפועל

פרטי מערכת האוטומציה - לא חשבנו שזה חשוב בכלל

פרטים על הבדיקות - לוגים ארוכים מידי או כתובים בצורה קריפטית עד כדי כך שאי אפשר להבין אותם

בשביל לעשות משהו מועיל עם הנתונים, צריך שיהיה נוח להגיע אליהם

איסוף הנתונים

נניח שצלחנו את שתי המשוכות הראשונות, כל הנתונים החשובים נאספים בזמן הבדיקות והם באיכות גבוהה. השלב הבא הוא לשמור אותם. אני מדגל כאן על הדיון (שלעיתים לוקח שבועות של וויכוחים לוהטים) באיזה סוג של בסיס נתונים להשתמש ואיזה טכנולוגיה היא הנכונה.

במקום לנסות ולהתעמק בשיקולים הטכניים, אני מציע להשקיע מאמץ בהבנה מדוקדקת של זרימת הנתונים, להגדיר באופן מפורט את השימוש שאנו נעשה במערכת, ולתת לאנשי התמיכה של הארגון (DevOps או system) לשבור את הראש איך לממש. כמובן שאם אתם חברת הזנק קטנה, אז אתם אנשי התמיכה של עצמכם. עדיין, הגדרה טובה של השימוש במערכת קודמת לבחירת הפתרון הטכני.

מאיפה מגיעים הנתונים ומי יטעין אותם למערכת, מה יקרה כשמשוה משתבש, איך נמנע איבוד נתונים, ועד כמה זה באמת נורא אם נאבד נתונים פה ושם.

האם יש צורך לשנות את הנתונים אחרי שעלו לבסיס הנתונים, מי מעורב, כמה פעמים זה יקרה, האם יש לנו צורך בשמירת היסטוריה מדוקדקת של כל השינויים, מי עשה ולמה?

- איך נשלוף את הנתונים, כמה פעמים ביום נצטרך לשלוף את המידע, ידנית או אוטומטית, על איזו כמות נתונים מדובר ועד כמה קריטי שהשליפה תהיה מהירה?

היבט אחר בתכנון מערך הנתונים הוא הקישוריות למערכות ובסיסי מידע אחרים וההתאמה ביניהם. כאן צריך לוודא שאנחנו לא מייצרים מצבים פתולוגיים. למשל:

- מידע שמשוכפל בכמה מקומות, עם יכולת לשנות אותו בכל מקום. האם צריך מנגנון שימנע הבדלים בנתונים בין המערכות?

- מידע מפוזר בין כמה מערכות כך שקבלת תמונה מלאה של הנתונים מחייבת אחזור ושילוב מידע מכמה מקומות. האם התכנון מאפשר לעשות זאת בקלות?

ברגע שיש קשר בין מערכות, יש כר נרחב לטעויות קטנות ומיותרות שאם לא מונעים אותן כבר ברגע הראשון ימשיכו לעצבן ולהפריע לארגון לאורך שנים. הנה שתי דוגמאות שנתקלתי בהם בעבר:

- בבסיס נתונים אחד, תוצאות של בדיקה נשמרו בשדה בשם "תוצאה". בבסיס נתונים אחר, אותו מידע נקרא "סטטוס". השימוש בשני שמות לאותו מידע יצר בלבול ואי הבנות.

- בשדה מסוים בדוחות בבאגים, הבודק בחר את שם המודול שבו נמצא הבאג מתוך רשימת בחירה המכילה את שמות כל המודולים בפרויקט. במערכת ניהול הבדיקות, הבודק קבע עבור כל מקרה בדיקה איזה מודול הוא מכסה. גם במערכת זו שמות המודולים פורטו ברשימת בחירה. אני נותן לכם ניחוש אחד אם הרשימות היו מסונכרנות ואם היה שם אחד ויחיד לכל מודול.

אנשי התשתיות שמקימים את המערכות השונות לא תמיד מודעים לקשרים אלו. זה התפקיד שלנו, בתור המשתמשים, לשים לב לפרטים הקטנים האלה ולמנוע אותם. באופן כללי: ככל שנצליח לצמצם את מספר המערכות השונות שמחזיקות את המידע שלנו, נמנע מבעיות של אי התאמות וצורך בתקשורת בין המערכות. אפשר לחסוך הרבה צרות אם מצליחים לאגד את כל תוצרי הבדיקות במערכת אחת.

נגישות

איסוף הנתונים הוא כמובן רק עוד צעד בדרך להשתמש ב"מוצר" שלנו - במידע. אחרי איסוף המידע, צריך לעשות אתו משהו מועיל. ובשביל לעשות משהו מועיל, צריך שיהיה נוח להגיע לנתונים. אפשר להגיד בוודאות די גבוהה שאם קשה להגיע לנתונים, אף אחד לא ישתמש בהם.

ההחלטה על אופן אגירת הנתונים משפיעה על הגישה אליהם. למשל, מימוש שיעיל מבחינת ניצול שטח האחסון, יכול להיות לא יעיל מבחינת האחזור, ולגרום לאיטיות רבה בשליפת אותם נתונים שדווקא הם אלו שמשמש מעניינים אותנו. גם בגלל שיקול זה חשוב להגדיר מראש, עד כמה שניתן, מה אנחנו מתכננים לעשות עם הנתונים, עוד לפני ההחלטה על הפתרון הטכני. הנה כמה דוגמאות נוספות.

ראיתי מקרים שבהם המידע מאורגן ככה שדי מסובך להגיע לנתונים מסוימים. למשל, אם בסיס הנתונים לא מחזיק גיורסה חדשה לרשומה עבור כל שינוי שנעשה בה אלא רק רישום של היסטוריית פעולות, הרי שיהיה קשה מאוד לשלוף את ערך הרשומה בזמן כלשהו בעבר. תרגיל קצר: כמה קל לשלוף מבסיס המידע שלכם את כמות הבאגים שתוקנו, אבל בבדיקת אימות נפתחו מחדש? כמה קל לשלוף את כל הבדיקות שנפלו בהרצה ראשונה אך עברו אחרי הרצה נוספת?



סיכום

איסוף וטיפול במידע של הבדיקות היא פעילות שמחייבת מחשבה מראש והשקעת משאבים באופן קבוע. זה לא קורה מעצמו, וניהול לא נכון של תחום זה פוגע ישירות באיכות הנתונים הנאספים על ידי קבוצת הבדיקות, שהם, כאמור, אחד משני התוצרים העיקריים של פעילות הבדיקות.

אם רוצים לעשות אנליזה המשתמשת בשיטות של נתוני עתק (big data) על הנתונים, מדובר בשליפה של מיליוני רשומות. כמה קל יהיה לעשות את זה? באנליזות מסוג זה אנחנו מחפשים בין השאר גם קשרים לא אינטואיטיביים בין הנתונים השונים, לכן נרצה לשלוף את כל המידע שיש לנו (כלומר, גם את כל אותם השדות שמכילים מידע שהגיונית לא אמור להיות קשור לפרמטרים אחרים או להשפיע על תוצאות ההרצה). האם המערכות שמתוכננות עבורנו מחביאות חלק מהנתונים, כי אנשי ה-system חשבו שאלה נתונים לא חשובים?

מעבר לכלים של שליפת נתונים, צריך גם שתהיה גישה לכלים המאפשרים עיבוד מידע בכמויות גדולות. כל מי שניסה לעשות ניתוח עם אקסל על קובץ עם מיליון רשומות כבר הגיע למסקנה שצריך כלי אחר. זמינות של כלים שמאפשרים לנתח כמויות מידע גדולות בעילות ומהירות היא קריטית לניצול משמעותי של הנתונים אותם אספנו.

חשוב לקבוע איך בקשות לשינויים מתנהלות ואיך מוודאים שהנתונים הנאספים אינם סתם זבל

תחזוקה

גם אם עשינו את כל הניתוחים כמו שצריך, והשתדלנו להגדיר את הדרישות עד הסוף, תמיד יתברר לנו אחרי זמן מה שלא הכל הוא כמו שחשבנו. נתונים שהערכנו שיהיו בעלי משמעות, מתבררים כחסרי ערך ממשי. נתונים שמוזנים ידנית (כמו למשל בנתוני באגים) לא מעודכנים כמו שקיוונו, ערכים שהכנסנו לרשימות בחירה והיו נראים לנו אינטואיטיביים וברורים מתגלים כדו-משמעיים עבור המשתמשים. שדות אחרים נשארים ריקים או עם ערך ברירת המחלל שלהם. גם בנתונים שנאספים באופן אוטומטי יתכנו כשלים: טעות בהגדרות גורמת לאיבוד דיוק במדידות (הכנסת ערך real לשדה שהוגדר כ integer), ערכים אחרים בכלל לא מצליחים להטען או מלכתחילה אינם בנתונים הגולמיים וכו'. בנוסף, תוך השימוש במערכת, מתגלים צרכים חדשים ואנשי ה-system מוצפים בבקשות לתוספת שדות וערכים.

כל זה צריך להיות מנוהל. מישהו צריך לבדוק (די בהתחלה, ואחר כך מידי פעם) שנתונים אכן נטענים כמו שצריך ובלי טעויות. מישהו צריך לדון בכל הבקשות ולהחליט מה אכן נדרש ומה ניתן לספק ללא שינויים בכלים ובבסיסי הנתונים. מישהו צריך לעבור על השדות שהוגדרו ולהעיף מבסיסי הנתונים שדות שאף אחד לא משתמש בהם. בארגון שלנו יש שתי קבוצות כאלה: אחת שדנה ספציפית במערכות הקשורות לבדיקות והאחרת שדנה במערכות שתומכות בכל תהליך הפיתוח (דרישות, DevOps, באגים). כיוון שזה ארגון גדול, הרי שכמעט כל שבוע יש על מה לדון. אני מניח שבחברות קטנות זו פעילות פחות תדירה, אבל חשוב מאוד להגדיר אותה, לקבוע מי אחראי עליה, איך בקשות לשינויים מתנהלות ואיך מוודאים שהנתונים שאנחנו מייצרים ושומרים אינם סתם זבל.

