קשיי מתחילים במעקב אחר קוד

Factors in Novice Programmers' Poor Tracing Skills
Vesa Vainio & Jorma Sajaniemi
המאמר השלם ניתן להורדה בכתובת:
http://db.grinnell.edu/sigcse/iticse2007/Program/viewAcceptedProposal.asp?sessionType=paper&sessionNumber=115
הוצג בכנס ITiCSE שהתקיים בדנדי סקוטלנד 2007

המאמר מדווח על מחקר שבדק את הסיבות לקשיים של לומדים מתחילים בביצוע מעקב של קוד. מספר מחקרים שנערכו בעבר הראו שלומדים מתחילים מתקשים במיומנויות מעקב (tracing). המחקר הנוכחי ניסה לבדוק את הסיבות לכך בעזרת ראיונות של סטודנטים מתחילים וכן בעזרת ניתוח פרוטוקולים של הבנת תכניות בג'אווה כדי לזהות בעיות ספציפיות שמשפיעות על היכולת של מתחילים לבצע מעקב. המחקר זיהה 4 גורמים עיקריים לקושי של מתחילים בביצוע מעקב: מעקב ערך יחיד, בלבול בין התפקיד למבנה, חוסר יכולת להשתמש בייצוגים חיצוניים, וחוסר יכולת להעלות את רמת ההפשטה.

הקדמה

הבנת תכנית היא אחת המיומנויות הנדרשות למתכנת (כמו גם תכנון, ניפוי שגיאות). כתיבת תכנית דורשת קריאה והבנה של חלקים כתובים של התכנית. באופן דומה, ניפוי שגיאות דורש הבנת תכנית ומעקב מפורט של ביצוע התכנית.

הנושא של הבנת תכנית לא נלמד כנושא עצמאי למרות החשיבות שלו. מניחים כביכול שהלומדים יפתחו בעצמם מיומנויות להבנת תכנית כתוצאת לוואי של הלימוד לכתיבת תכניות.

לעיתים קרובות נהוג להתייחס להבנת תכנית כתהליך המבוסס על השערות שמטרתו לבנות ייצוג מחשבתי (חלקי) של התכנית. הייצוג הזה מורכב משני חלקים נפרדים (הקוד של התכנית, והמשמעות של התכנית) והקשרים ביניהם. במהלך הנסיון להבנת תכנית, המתכנת יוצר השערות ביחס לחלקים האלה ומנסה לאמת/לדחות את ההשערות האלה. כלומר, מעקב (או ביצוע מחשבתי) של תכנית הוא טכניקה ליצירת השערות ואימותן והוא דרוש גם לצורך הבנה (או תפיסה) של קוד חדש או לצורך מציאת שגיאות.

מעקב אחר קוד ניתן לביצוע בשתי רמות נפרדות: מעקב סימבולי ומעקב קונקרטי. מעקב קונקרטי עוסק בערכים הספציפיים של המשתנים במצב מסוים (למשל, הערך של b הוא 5). הוא מתמקד בנקודה ספציפית בזמן ומספק פחות מידע על מטרת הפקודה מאשר מעקב סימבולי. מעקב סימבולי עוסק ביחסים הקבועים בין משתנים (למשל, b מקבל את הערך c+1). חוסר מיומנות מספיקה והמורכבות של הקוד יכולים למנוע מהמתכנת את היכולת לבצע מעקב סימבולי ולחייב אותו לבצע מעקב קונקרטי.

היכולת לעקוב אחר תכנית היא מיומנות חשובה הדרושה למומחים. מספר מחקרים מצאו שמתחילים מתקשים אפילו במעקב קונקרטי (וכמובן במעקב סימבולי הקשה יותר לביצוע). אבל לא מספיק ידוע על הסיבות לקשיים של מתחילים בביצוע מעקב.

סקר ספרות על קשיים בביצוע מעקב אחר קוד

יש מעט מאד מחקרים על קשיי מתחילים במעקב והם מתמקדים בעיקר בהבנת תכנית (ולא בביצוע מעקב). מצד שני, ההתנהגות של מתכנתים מומחים בביצוע מעקב נחקרה לעומק. ממחקרים אלה (ואחרים על מומחים) ניתן ללמוד שנסיון ספציפי בתחום ממלא תפקיד מרכזי אצל מומחים (וכמובן שלמתחילים אין נסיון). אפשר לסכם את עניין הנסיון כך – מתכנת מומחה צבר ידע על חלקי תכניות שכתב בעבר והידע הזה מאוחסן בזכרונו. הידע הזה יכול להיות מורכב מחלקי תכניות או תבניות מופשטות יותר של תכניות. כאשר הוא מתכנן תכנית, המומחה משתמש בחלקים (fragments) האלה כאבני בניין מוכנות ואינו צריך לבנות אותן. בתהליך של הבנת תכנית, המומחה מזהה את התבניות המוכרות האלה. בזכות הנסיון, המומחה יכול לקבוע תמונה מחשבתית של התכנית ולעקוב ביעילות אחר הקשרים בין חלקים שונים בתכנית. מומחים יכולים גם לייצר השערות טובות מהם ההבטים הרלבנטיים ולהתעלם מההבטים הלא רלבנטיים.

כדי שסימולציה מחשבתית תהיה מוצלחת, היא צריכה להיות מאד חסכנית ולכלול מעט מאד חלקים בכל רגע: רק את החלקים הרלבנטיים ברמה הנכונה של ההפשטה.

הואיל ומעקב סימבולי מספק יותר מידע ברמה גבוהה, מומחים מעדיפים אותו (על פני מעקב קונקרטי שבו משתמשים רק כמוצא אחרון כאשר קשה להבין אחרת).

לעומת זאת, למתחילים יש מעט מאד נסיון ולכן הם חייבים לבנות את ההבנה שלהם על התכנית על פי מילים בודדות (אבני הבניין שלהם עוד לא קיימות) והם אפילו לא מבינים טוב את המילים הבודדות (את השפה) כי הידע שלהם שביר. הידע של מתחילים מבוסס על מעט מאד דוגמאות שהספיקו לראות ויש להם קשיים לבחור חלקים נכונים לצורך הסימולציה.

תאור המחקר

המחקר התבסס על ראיונות עם סטודנטים שלמדו את הקורס הראשון בתכנות (בשפת ג׳אווה) במסגרת סמסטר קיץ מרוכז. הקורס נמשך 6 שבועות (56 שעות הרצאה, 24 שעות מעבדה, ושיעורי בית).

במחקר השתתפו 6 סטודנטים (גבר אחד ו- 5 נשים). כל סטודנט רואיין פעם בשבוע במהלך הקורס וסה״כ התקיימו 32 ראיונות. הראיונות כללו שאלות פתוחות, הגדרות למושגים, משימות של מעקב ומשימות תכנות שנפתרו בכתב (ללא מחשב). כל ראיון נמשך כשעה.

ממצאים:

במחקר נמצאו 4 גורמים עיקריים לקשיים של מתחילים בביצוע מעקב: מעקב ערך יחיד, בלבול בין התפקיד למבנה, חוסר יכולת להשתמש בייצוגים חיצוניים, וחוסר יכולת להעלות את רמת ההפשטה.

1. מעקב ערך יחיד - Single Value Tracing

במאמר מוגדרים שני סוגים של משתנים בתכנית: משתנים שערכיהם ישירים (למשל a=15) וניתן לראות אותם בקוד, ומשתנים שמקבלים את ערכיהם כתוצאה מחישובים ולא ניתן לראות אותם ישירות בקוד. המשתנים מהסוג הראשון נקראים משתנים טריביאליים (trivial) והמשתנים מהסוג השני נקראים משתנים לא טריביאליים (nontrivial).

מעקב ערך יחיד הוא אסטרטגית מעקב פשוטה שנמצאה במחקר. לפי האסטרטגיה הזו, הסטודנט מחזיק בראשו מקסימום ערך לא טריביאלי אחד עבור כל המשתנים של התכנית. כלומר, הסטודנט משתמש באותו ערך לכל משתנה לא טריביאלי שהוא נתקל בו במהלך המעקב (בלי לשים לב לאיזה משתנה הוא זוכר את הערך). במקרה כזה, הערך הלא טריביאלי היחיד הוא תמיד האחרון שמוצב לאיזשהו משתנה בתכנית. ולכן, לסטודנט יש "משבצת זכרון" (memory slot) יחידה לכל המשתנים והוא משתמש בה בכל פעם שהערך לא נראה ישירות בקוד.
האסטרטגיה הזו יכולה לעבוד די טוב בתכניות קטנות שהן אופייניות לקורס הראשון בתכנות. במיוחד בתכנות מונחה עצמים, כאשר בתוכניות הדוגמה יש הרבה ערכים טריביאליים ומעט מאד ביטויים מתמטיים. ולכן, לסטודנטים אין הזדמנות להיווכח שהאסטרטגיה הזו היא בעייתית.
השימוש באסטרטגיה הזו לא יאפשר לסטודנט לבצע מעקב אם יהיו בתכנית שני משתנים לא טריביאליים (לפחות) ויהיה צריך למשל להשוות ביניהם.

2. בלבול בין תפקיד ומבנה - Confusing Function and Structure

לפי תיאוריה של De Kleer & Brown מודלים מחשבתיים צריכים למלא מספר עקרונות כדי שיהיו יציבים. אחד העקרונות המרכזיים נקרא העקרון של מבנה ללא פונקציה (no-function-in-structure principle). לפי עקרון זה, בלבול הידע על מבנה המערכת עם הידע על תפקיד המערכת יכול להוביל להנחות שגויות. דוגמה להנחה שגויה כזו בתכנות היא שאם ערך של משתנה גדל (או משתמשים בו) בתוך לולאה, אזי המשתנה תמיד יאותחל לאפס בתחילת הביצוע של הלולאה. בתוכניות דוגמה נוהגים לאתחל משתנה לאפס בתחילת הלולאה ואז ערכו גדל תוך כדי ביצוע הלולאה. זהו תפקיד טיפוסי ללולאת for אבל כמובן שאינו הכרחי – כלומר, זה לא נדרש על ידי המבנה של לולאת for. דוגמה להנחה שגויה כזו נצפתה במחקר כאשר סטודנט עקב אחר הקוד הבא

```
int num = 64;
int counter = 0;
for (int n = 0; num > 2; num /= 2) {
        for (int k = 0; k < num; k++) {
                counter++;
                System.out.println(counter);
        }
}
```

תוך כדי המעקב, הסטודנט התעקש שהמונה counter יאותחל לאפס בכל ביצוע של הלולאה החיצונית. לטענת מחברי המאמר, התופעה הזו מדגימה שהסטודנט דמיין פעולה שלא קיימת בתכנית ולטענתם זה קרה בגלל הסטודנט בלבל בין התפקיד הטיפוסי של for לבין המבנה של הלולאה.

3. חוסר יכולת להשתמש בייצוגים חיצוניים - Inability to Use External Representations

לפי מחקר של Thomas סטודנטים מתחילים מתקשים להחצין את הידע שלהם על הנייר והם מופתעים כשמראים להם ששרטוטים כאלה (כמו דיאגרמה של מבנה נתונים למשל) יכולים לסייע בפתרון בעיות. אבל לא ברור האם השימוש בשרבוטי הנייר הוא זה שמוביל להבנה טובה של

התכנית? או שאולי רק אותם סטודנטים שמסוגלים מלכתחילה להבין טוב את התכנית הם אלה שהשתמשו בשרבוטים?

הנושא הזה נבחן במחקר הנוכחי בעזרת תכנית שהיו בה 5 עצמים מאותה מחלקה. למחלקה היו 2 שדות (אחד היה מחרוזת והשני מצביע לעצם מאותה מחלקה). כאשר הסטודנטים התבקשו להשתמש בעט ונייר כדי לתאר איך הם מבינים את ביצוע התכנית הם התקשו בכך למרות שהמרצה של הקורס השתמש בהרבה דוגמאות (למשל, הסטודנטים הצליחו לייצג מצביע ריק אבל לא הצליחו לייצג מצביע לא ריק).

בהתבסס על הממצאים האלה, מניחים מחברי המאמר שיש הבדל בתיאור או בייצוג החיצוני של מושגי תכנות שונים בהתאם לרמת ההפשטה של המושגים האלה. משתנה מקומי מטיפוס פרימיטיבי הוא מושג פחות מופשט מאשר הפנייה לעצם. סטודנטים מכירים מספרים ומחרוזות עוד לפני שלמדו תכנות ולכן קל להם לבטא ערכים פרימיטיביים על נייר. אבל, הפנייה לעצם כרוכה ברעיונות מופשטים שהסטודנטים לא הכירו לפני לימוד התכנות ולכן קשה להם לתאר זאת על נייר. במקרה אחד המראיין ביקש מסטודנט לצייר והסטודנט עשה זאת אבל אמר שזה לא עוזר לו להבין טוב יותר. מדוגמה זו ניתן ללמוד כי לסטודנטים יש שני סוגי בעיות עם ייצוג חיצוני: מצד אחד הם מתקשים לייצר דיאגרמות שמתארות את מצב התכנית, ומצד שני הם לא יכולים להיעזר בדיאגרמות (גם אם הן ניתנות להם) לצורך מעקב. התופעות האלה קשורות זו בזו אבל יש להן כנראה מקורות שונים. בעוד שחוסר היכולת לצייר מבוססת כנראה על מחסור בייצוגים דיאגרמיים לרעיונות מופשטים, הרי שלחוסר היכולת להשתמש בדיאגרמה נתונה כדי לבצע מעקב יש סיבה אחרת (שמחברי המאמר לא יודעים מהי ומציינים שנדרש לשם כך מחקר נוסף).

4. חוסר יכולת לעלות ברמת ההפשטה - Inability to Raise Abstraction Level

לפי מחקר של Klein, שבדק סימולציה מחשבתית בתחומים לא תכנותיים, נטען שהיכולת לבצע סימולציה מחשבתית היא מוגבלת ולא ניתן לבצע אותה על יותר מ- 3 עצמים בו זמנית. כדי לבצע סימולציה מחשבתית מוצלחת יש צורך בנסיון ומיומנות והאתגר הוא בבחירה נכונה של רמת ההפשטה ובחירה נכונה של העצמים עליהם מבצעים את הסימולציה.

דוגמה לרמת הפשטה עולה היא בהחלפה בין שני אברי מערך, תוך שימוש במשתנה עזר שלישי. אם משתנה העזר לא מעורב בפעולות אחרות, הרי ששלוש הוראות ההשמה ניתנות להפשטה כפעולה מחשבתית יחידה שהופכת את אברי המערך. אבל אם מנסים לבצע מעקב אחרי אלגוריתם המיון ללא ההפשטה הזו, הרי שיש יותר מידי עצמים/משתנים שצריך לעקוב אחריהם ולכן לא ניתן לבצע זאת בזכרון האנושי.

השלכות להוראה

חלק מהממצאים במחקר מצביעים על השפעות של חוסר הנסיון של מתחילים, בעוד שחלק מהממצאים מצביעים על בעיות ספציפיות אחרות שקשורות כנראה לחוסר הבנה בתכנות. האבחנה בין שני סוגי הגורמים האלה (חוסר נסיון מול חוסר הבנה) היא קריטית למורה.

מחברי המאמר מביאים מספר המלצות, כמו ההמלצה הבאה: מורה יכול להצביע בפני תלמידיו על טעויות הנובעות מחוסר נסיון ועל ידי כך למנוע אותן. אבל לא מספיק להראות בכיתה מעקב על הלוח. צריך גם להדגיש בפני התלמידים שלא משתמשים בכך רק להדגמה אלא כיוון שמעקב מחשבתי אחר מספר ערכים נידון לכשלון מלכתחילה בגלל מגבלות הזכרון האנושי.