تحلیل ایستای استفاده ناامن از متغیرها در برنامه‌های به زبان جاوا

استاد راهنما: دکتر شیری

ارائه دهنده: سید محمدمهدی احمدپناه

m@ahmadpanah.net

اردیبهشت ۱۳۹۳

فهرست مطالب

- مقدمه
- کارهای مرتبط
- روش پیشنهادی
- آزمایش‌ها و نتایج
- نتیجه‌گیری و کارهای آینده
- مراجع

مقدمه

- ضرورت تولید نرم‌افزار مقاوم و بدون خطا
- استفاده از Exception Handling در برنامه‌ها
- ویژگی‌های زبان جاوا
  - type safe بودن
    - Progress
    - Preservation
- ضرورت تحلیل ایستا برای یافتن خطاهای برنامه‌نویسی
- تعریف استفاده ناامن از متغیرها

فهرست مطالب

- مقدمه
- کارهای مرتبط
- روش پیشنهادی
- آزمایش‌ها و نتایج
- نتیجه‌گیری و کارهای آینده
- مراجع

کارهای مرتبط

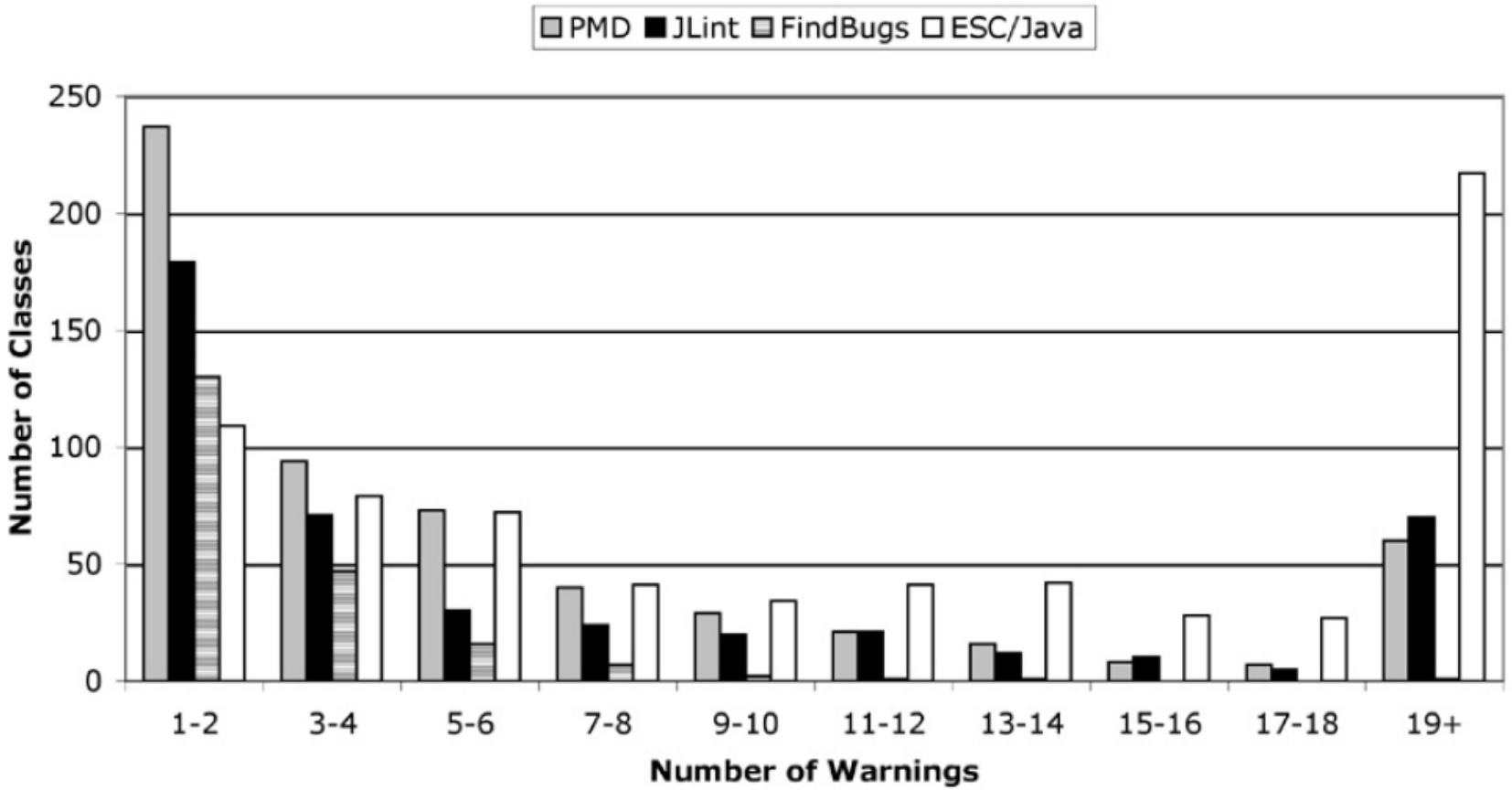
- استفاده از جریان داده، گراف کنترل جریان و متن کد برنامه
- وجود ابزارهای مختلف برای یافتن اِشکال نظیر:
  - PMD
  - FindBugs
  - JLint
  - ESC/Java
  - Bandera

کارهای مرتبط (ادامه)

Name	Version	Input	Interface	Technology
Bandera	0.3b2 (2003)	Source	Command Line, GUI	Model Checking
ESC/Java	2.0a7 (2004)	Source	Command Line, GUI	Theorm Proving
FindBugs	0.8.2 (2004)	Bytecode	Command Line, GUI, IDE, Ant	Syntax, Dataflow
JLint	3.0 (2004)	Bytecode	Command Line	Syntax, Dataflow
PMD	1.9 (2004)	Source	Command Line, GUI, IDE, Ant	Syntax

جدول ۱ - مقایسه ابزارهای مختلف اشکال‌یابی [۱]

کارهای مرتبط (ادامه)



شکل ۱ - مقایسه ابزارهای مختلف اشکال‌یابی [۱]

فهرست مطالب

- مقدمه
- کارهای مرتبط
- روش پیشنهادی
- آزمایش‌ها و نتایج
- نتیجه‌گیری و کارهای آینده
- مراجع

روش پیشنهادی

- تعریف استفاده ناامن بر پایه گسترش عملگرها بر روی متغیرهای استفاده‌شده در تحلیل جریان داده
  - عملگرهای define
  - عملگرهای sDef
  - عملگرهای eDef
  - عملگرهای use
  - عملگرهای kill

روش پیشنهادی (ادامه)

- یافتن زوج‌های eDef با استفاده از گراف کنترل جریان
  ۱. ساختن گراف کنترل جریان
  ۲. تحلیل جریان داده سلسله‌مراتبی
- الگوریتم تشخیص استفاده ناامن
  - تولید مجموعه متغیرها
  - تولید ردهای عملگر برای هر متغیر
  - تشخیص جفت‌های eDef-use در هر رد و تعیین استفاده ناامن

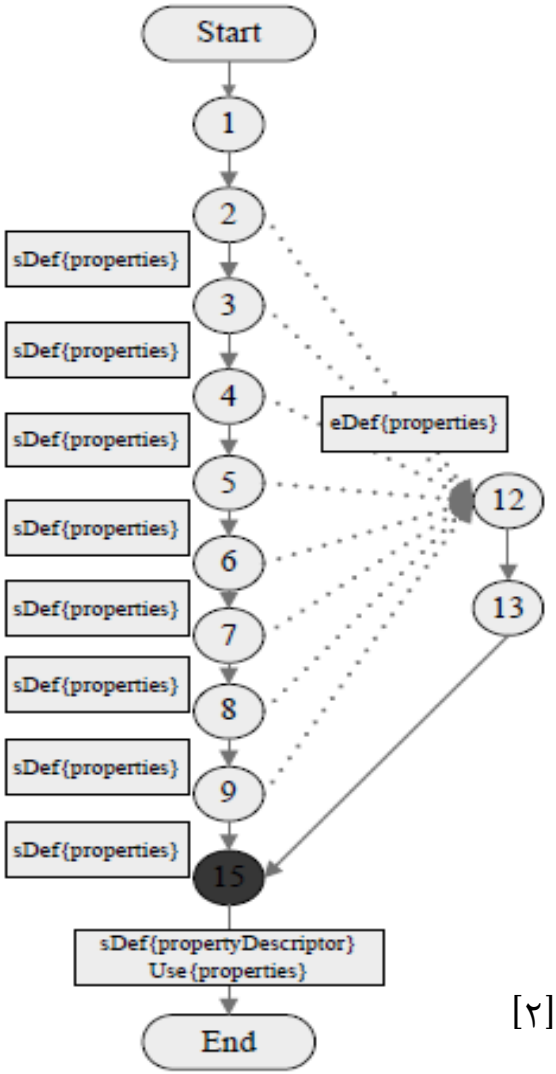
روش پیشنهادی (ادامه)

```
Algorithm: detect unsafe use of variables for method M
Input:     Control flow graph of M
Output:    Nodes where unsafe use of variables occur

Begin
/* Step1: Generate variable set for the operator of sDef,
eDef, Use and Kill */
        For (each node in control flow graph){
                Divide each variable into the set of sDef, eDef,
          Use and Kill.
        }
/* Step2: Generate operation traces for each variable */
        Traverse the control flow graph to generate the
  operation traces for each variable;
/* Step3: Detect unsafe use on each trace */
        For (each trace) {
                Detect all appearance of EU pairs.
                For (each appearance of EU in the trace) {
                        Locate unsafe use node in program;
        }}
End
```

شکل ۲ – الگوریتم تشخیص استفاده ناامن از متغیرها برای یک متد [۲]

روش پیشنهادی (ادامه)



(a)

Variable	Path	Trace
properties	1,2,3,4,5,6,7,8,9,15	SSSSSSSSU
	1,2,12,13,15	EU
	1,2,3,12,13,15	SEU
	1,2,3,4,12,13,15	SSEU
	1,2,3,4,5,12,13,15	SSSEU
	1,2,3,4,5,6,12,13,15	SSSSEU
	1,2,3,4,5,6,7,12,13,15	SSSSSEU
	1,2,3,4,5,6,7,8,12,13,15	SSSSSSEU
	1,2,3,4,5,6,7,8,9,12,13,15	SSSSSSSEU

(b)

شکل ۳ – گراف کنترل جریان پردازش‌شده [۲]

فهرست مطالب

- مقدمه
- کارهای مرتبط
- روش پیشنهادی
- آزمایش‌ها و نتایج
- نتیجه‌گیری و کارهای آینده
- مراجع

آزمایش‌ها و نتایج

- اجرا روی HSQLDB، FreeCS و JWhoisServer
- پیاده‌سازی به کمک چارچوب تحلیل Soot

Project Name	Packages	Classes	Lines of code
HSQLDB1.8.3	14	258	143291
FreeCS1.3.2	13	140	29949
JWhoisServer0.3.3	4	28	7603

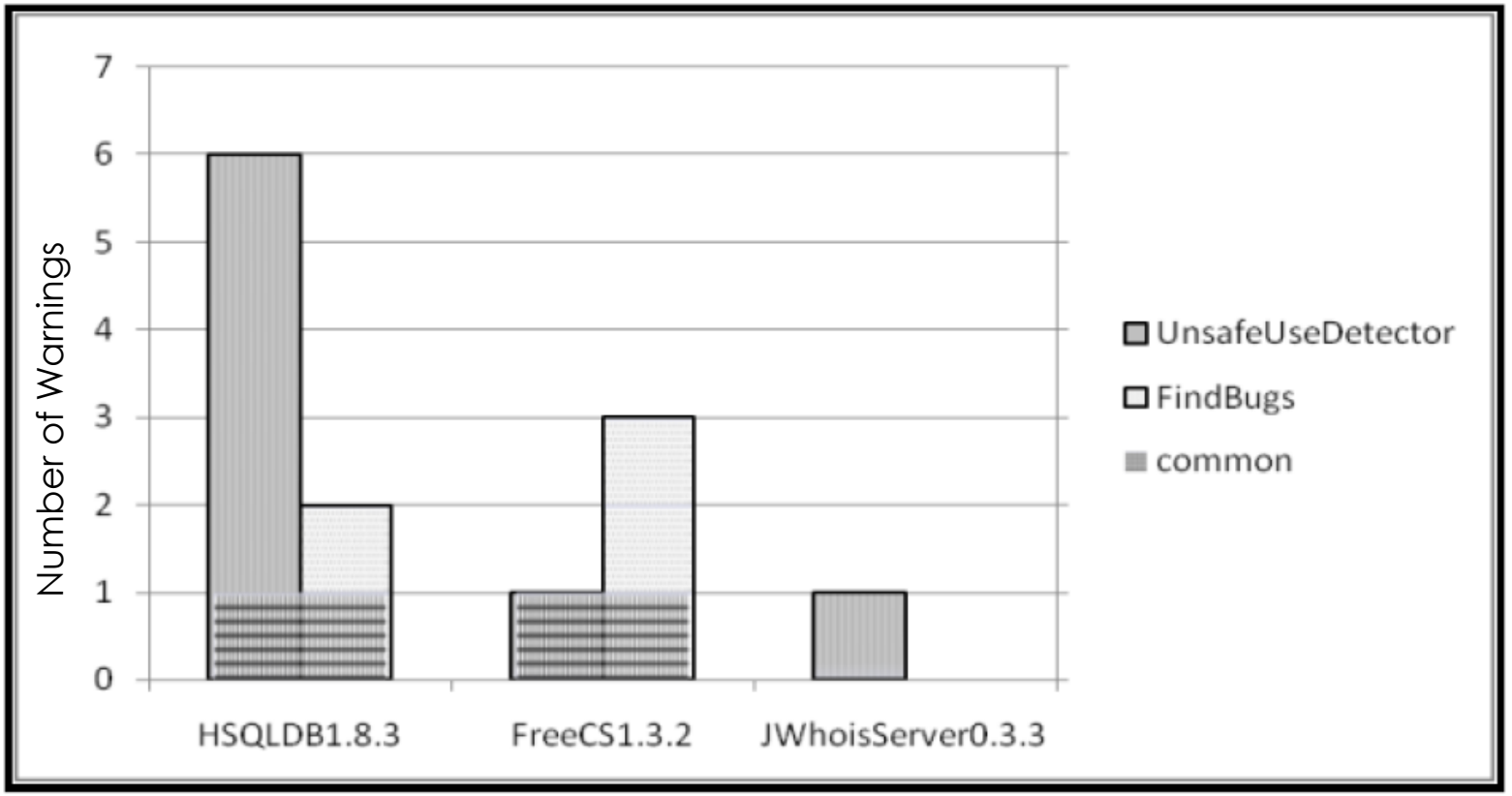
جدول ۲ – برنامه‌های استفاده‌شده در آزمایش [۳]

آزمایش‌ها و نتایج (ادامه)

Project Name	Warnings	Bugs	False Positives
HSQLDB1.8.3	11	6	5
FreeCS1.3.2	1	1	0
JWhoisServer0.3.3	1	1	0

جدول ۳ – نتایج آزمایش بر روی برنامه‌های استفاده‌شده [۳]

آزمایش‌ها و نتایج (ادامه)



شکل ۴ – نمودار مقایسه روش پیشنهادی با سایر روش‌ها [۳]

فهرست مطالب

- مقدمه
- کارهای مرتبط
- روش پیشنهادی
- آزمایش‌ها و نتایج
- نتیجه‌گیری و کارهای آینده
- مراجع

نتیجه‌گیری و کارهای آینده

- راه‌حل ارائه شده، راه‌حلی مناسب برای مسئله
- ایجاد هشدار کاذب (False Alarm) به دلیل محافظه‌کارانه بودن الگوریتم
- کاهش هشدار کاذب و کاهش زمان اجرای الگوریتم
- کار روی Exceptionهای بررسی نشده، نام مستعار متغیرها (alias) و محدود کردن پویا (dynamic binding)
- بهره‌گیری از روش‌های فرمال امنیت زبان-مبنا

فهرست مطالب

- مقدمه
- کارهای مرتبط
- روش پیشنهادی
- آزمایش‌ها و نتایج
- نتیجه‌گیری و کارهای آینده
- مراجع

مراجع

[1] N. Rutar, et al., "A Comparison of Bug Tools for Java", Software Reliability Engineering, 2004. ISSRE 2004. 15th International Symposium on, 2004, pp. 245-256.

[2] J.Wei, et al. Static Detection of Unsafe Use of Variables in Java, Ubiquitous Intelligence & Computing and 7th International Conference on Autonomic & Trusted Computing (UIC/ATC), pp. 439-443, 2010

[3] X.Wu, et al., "Static Detection of Bugs Caused by Incorrect Exception Handling in Java Programs", 11th International Conference On Quality Software, 2011.

[4] W. Wosgerer, "A Survey of Static Program Analysis Techniques", Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on , Volume:27, Issue: 7 , 2005.

سوال؟!

با سپاس از توجه شما! ☺