L’efficacité des outils de détection de fautes et de prédiction de refactoring

Auteurs:
Fadi Wedyan
Dalal Alrmuny
James M, Bieman

Présentateurs:
Dao Thuy Hong
Fotsing Sikadie Gervais Sertilange

Plan

Introduction

Méthodologie

Outils utilisés

Résultats

Facteurs remettant en cause l’étude

Conclusion

Plan

Introduction

Méthodologie

Outils utilisés

Résultats

Facteurs remettant en cause l’étude

Conclusion

Introduction

ASA(Automated Static Analysis Tools For Fault Detection)

- Leur But : aider les développeurs

A éliminer les défauts logiciels au début

A produire des logiciels plus fiables à un coût moindre.

- Problème : Ils n'ont pas atteint le succès attendu par leurs concepteurs

- Objectif de l'étude: Évaluer l'efficacité des outils ASA dans la détections de problèmes(code et de refactoring) pour les applications Java.

Introduction

Pour évaluer l'efficacité des outils ASA, on doit répondre aux questions suivantes :

- Quelle est l'efficacité des outils ASA dans la détection des fautes ?
- Quelle est l'efficacité de ces outils dans la détection des refactorings ?
- Quelle est la proportion de faux résultats fournis par ces outils ?
- Quelle est la proportion de fautes et anomalies non détectées par ces outils ?

Plan

Introduction

Méthodologie

Outils utilisés

Résultats

Facteurs remettant en cause l’étude

Conclusion

Méthodologie

- Étudier deux logiciels libres : iText et jEdit.
- Les développeurs de ces systèmes n'ont pas utilisé des outils ASA dans le processus de développement.
- On développe une autre version de iText et jEdit cette fois ci en utilisant les outils ASA.

Hypothèses :

1. Les développeurs utilisent ces outils depuis la première version de chaque système(iText et JEdit).
2. Les développeurs peuvent immédiatement faire des modifications en réponse aux préoccupations signalées par ces outils.

Méthodologie

- On va appliquer 3 outils ASA : FindBugs, IntelliJ IDEA et Jlint pour 13 versions de iText et 7 versions de jEdit.
- Pour chaque version, on va identifier les corrections de fautes et refactorings faites lors du développement de ces systèmes réels et les comparer avec les résultats fournis par les outils ASA.
- Ceci peut être vu comme une comparaison entre la compétence de développeurs expérimentés et la performance des outils ASA.

Méthodologie

Indicateurs de performance des outils ASA

Q1 : Quelle est l'efficacité des outils dans la détection des fautes ?

$$\text{Ratio de détection} = \frac{\text{Nbre de fautes corrigées détectées par l'outil}}{\text{Nbre de fautes corrigées par les développeurs sans outils}}$$

Méthodologie

Indicateurs de performance des outils ASA

Q2 : Quelle est l'efficacité des outils dans la prédiction des refactorings ?

$$\text{Ratio de refactoring} = \frac{\text{Nbre de refactoring recommandés par l'outil}}{\text{Nbre total de refactoring effectués dans le cas réel}}$$

Méthodologie

Indicateurs de performance des outils ASA

Q3 : Quelle est la proportion de faux résultats fournis par ces outils ?

$$\text{Ratio de faux positifs} = \frac{\text{Nbre de faux positifs}}{\text{Nbre total de fautes détectées par l'outil}}$$

Méthodologie

Indicateurs de performance des outils ASA

Q4 : Quelle est la proportion de fautes et anomalies non détectées par ces outils ?

$$\text{Ratio de faux négatifs} = \frac{\text{Nbre de faux négatifs}}{\text{Nbre total de correction effectués dans le cas réel}}$$

Méthodologie

Comment compter le nombre de fautes détectées pour les versions de jEdit et iText développées sans outils ASA ?

- Les données sur les fautes ont été obtenues depuis les répertoires SourceForge et la base de données des reports de bugs des deux projets.
- Dans SourceForge, les utilisateurs reportent les bugs et ensuite, les développeurs corrigent ces bugs ou fournissent une explication en commentaire si le bug n’est pas réel.

Plan

Introduction

Méthodologie

Outils utilisés

Résultats

Facteurs remettant en cause l’étude

Conclusion

Outils ASA utilisés

- IntelliJ IDEA
- FindBugs
- Jlint

Plan

Introduction

Méthodologie

Outils utilisés

Résultat

Facteurs remettant en cause l’étude

Conclusion

Résultats

Table 5. Fault corrections and refactoring performed in 13 releases of *iText* along with the corresponding coding concerns found by FindBugs and IDEA

Release	Fault Corrections	Corrected faults found by FindBugs	Corrected faults found by IDEA	Refactorings Performed	Refactorings correctly predicted by FindBugs	Refactorings correctly predicted by IDEA
0.87	34	1	1	0	0	0
0.9	3	0	0	2	0	0
0.91	6	0	0	4	3	3
0.92	22	1	1	29	0	26
0.93	6	0	0	16	0	6
0.93b	1	0	0	3	0	1
0.94	5	0	0	2	0	1
0.95	7	0	0	3	0	0
0.96	15	0	0	3	0	2
0.97	5	0	0	0	0	0
0.98	5	1	1	4	0	0
0.99	6	0	0	6	2	1
1.00	21	0	0	29	2	26
Total	136	3	3	101	7	66

Résultats

Table 4. Fault corrections and refactorings performed in 7 releases of *jEdit* along with the corresponding coding concerns found by FindBugs and IDEA

Release	Fault Corrections	Corrected faults found by FindBugs	Corrected faults found by IDEA	Refactorings Performed	Refactorings correctly predicted by FindBugs	Refactorings correctly predicted by IDEA
3.0	18	0	1	0	0	0
3.0.1	2	0	0	0	0	0
3.0.2	26	1	1	25	6	22
3.1	22	0	1	31	4	24
3.2	2	0	0	0	0	0
3.2.1	3	0	0	0	0	0
3.2.2	39	0	0	21	3	15
Total	112	1	3	77	13	61

Résultats

Table 8. FindBugs False Positives (%)

	iText	jEdit	Both Projects
Faults	98.73	98.61	98.70
Refactoring	92.85	65.79	85.29
Faults & Refactorings	97.01	93.67	95.68

Table 9. IDEA False Positives (%)

	iText	jEdit	Both Projects
Faults	99.43	99.12	99.31
Refactoring	98.47	96.93	97.99
Faults & Refactorings	98.58	97.25	98.15

Résultats

Table 10. FindBugs False Negatives (%)

	iText	jEdit	Both Projects
Faults	97.79	99.11	98.38
Refactoring	93.07	83.12	88.76
Faults & Refactorings	95.78	92.59	94.37

Table 11. IDEA False Negatives (%)

	iText	jEdit	Both Projects
Faults	97.79	97.32	97.58
Refactoring	34.65	20.77	28.65
Faults & Refactorings	70.89	66.13	68.78

Plan

Introduction

Méthodologie

Outils utilisés

Résultats

Facteurs remettant en cause l’étude

Conclusion

Facteurs remettant en cause l’étude

- Validité de construction
- Validité de contenue
- Validité externe

Validité de construction

- Se rapporte à la signification même des mesures.
- Est ce que les mesures quantifient vraiment ce pour quoi elles ont été crées ?
- Nombre de faux positifs par exemple.

Validité de contenue

- Se rapporte à la pertinence de l’échantillonnage du contenue.
- Est ce que les les mesures de corrections de défauts réels couvrent la totalité des défauts identifiables ?
- Non les développeurs n’ont corrigé que les fautes qu’ils ont jugé les plus importantes dans le développement réel.

Validité externe

- Se rapporte à la façon dont les résultats de l'étude peuvent être généralisés en dehors de la portée de l'étude.
- Cette étude porte sur deux projets Java.
- Elle utilise seulement 3 outils ASA.

Plan

Introduction

Méthodologie

Outils utilisés

Résultats

Facteurs remettant en cause l’étude

Conclusion

Conclusion

- Les outils étudiés ne sont pas efficaces pour détecter les défauts de code.
- Les outils sont plus efficaces pour détecter les erreurs de refactoring.
- Le coût d’utilisation de ces outils est très élevé en raison du nombre trop grand de faux positifs.

MERCI POUR VOTRE BIEN AIMABLE ATTENTION