

Scalable and Accurate Test Case Prioritization in Continuous Integration Contexts

Ahmadreza Saboor Yaraghi et al, 2022

18T2410 AKAMBA MANI CRESCENCE CATHERINE¹

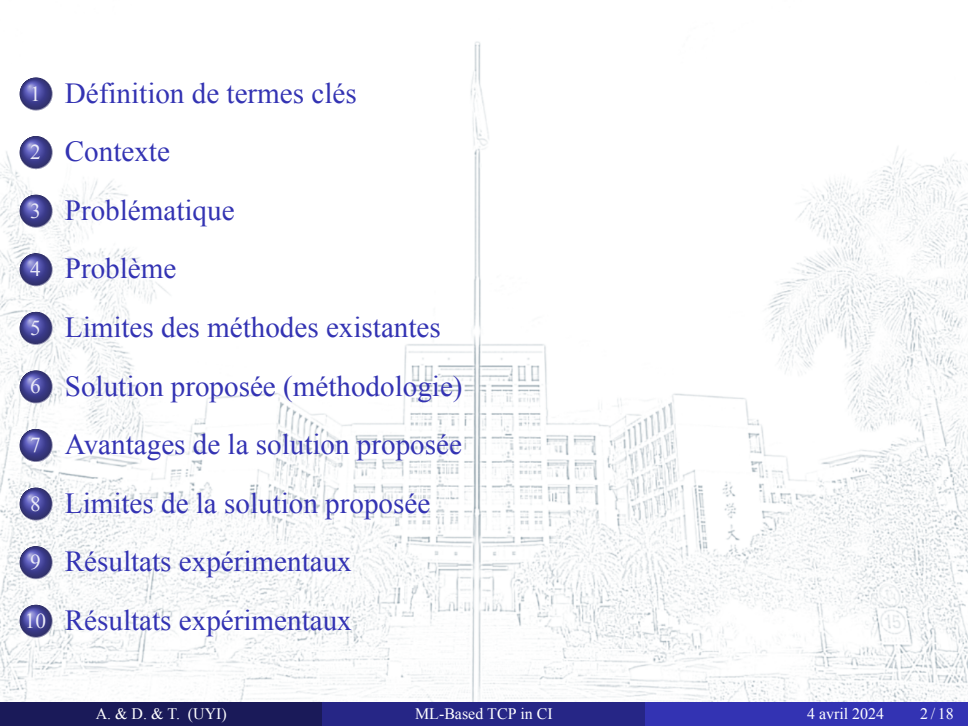
17T2051 DJIEMBOU TIENTCHEU VICTOR NICO¹

19M2364 TEYOU GHOMFO MARTIAL¹

¹Université de Yaoundé I,
INF5029

4 avril 2024



- 
- 1 Définition de termes clés
 - 2 Contexte
 - 3 Problématique
 - 4 Problème
 - 5 Limites des méthodes existantes
 - 6 Solution proposée (méthodologie)
 - 7 Avantages de la solution proposée
 - 8 Limites de la solution proposée
 - 9 Résultats expérimentaux
 - 10 Résultats expérimentaux

- L'Apprentissage Automatique (AA)
- Software Testing (test logiciel)
- Test Case Prioritization (hiérarchisation des cas de test)
- La Test Case Selection (sélection des cas de test)
- Continuous Integration (intégration continue)
- Un Build

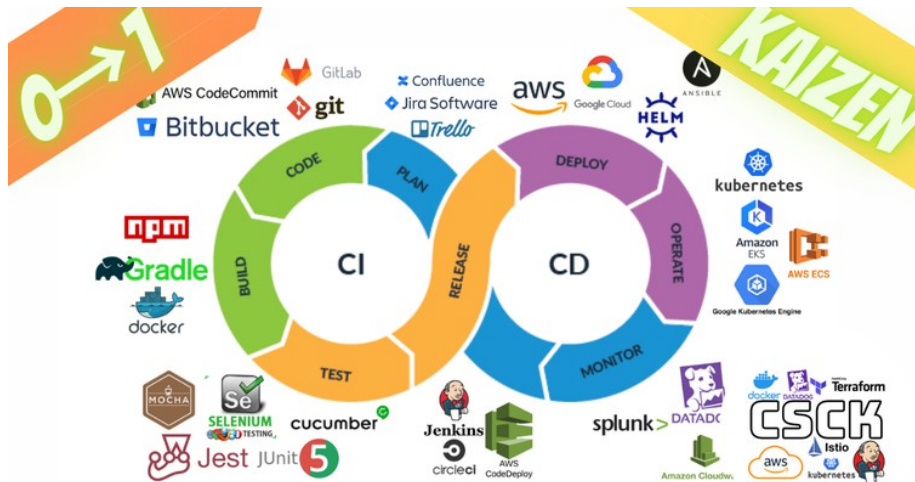


Figure – Image d'un cycle de vie logiciel

Difficultés du domaine

- Nature dynamique de l'environnement d'intégration continue
- Détection rapide des erreurs
- Temps d'exécution des tests
- Gestion des dépendances

Problème que traite l'article

Comment optimiser la priorisation des cas de tests dans un processus d'Integration Continue (IC) en utilisant des techniques basées sur l'apprentissage automatique (AA)?

Limites des méthodes existantes

Heuristiques basées sur la couverture

- Techniques d'analyse dynamique coûteuses en temps
- Dépendance à la qualité des cas de test
- Ne garantit pas la détection de tous les problèmes

Heuristiques basées sur l'historique d'exécution des cas de test

- Biais temporel
- Manque de flexibilité
- Dépendance aux données historiques

TCP basées sur l'apprentissage automatique

- Faible nombre de caractéristiques
- Faible quantité de sujet
- Faible temps d'exécution des test de régression

Solution proposée (méthodologie)

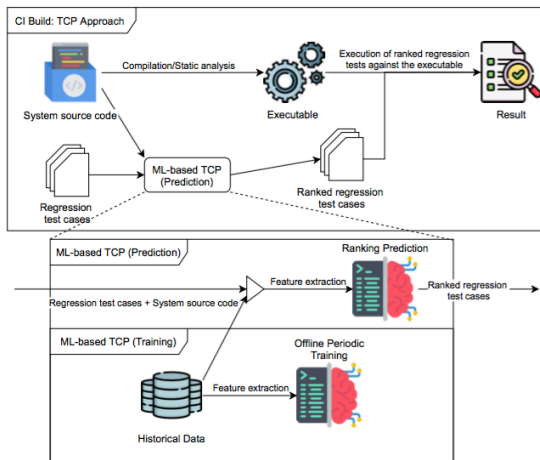


Figure – Modèle TCP basé sur le Machine Learning (ML) dans un contexte CI

Solution proposée (méthodologie)

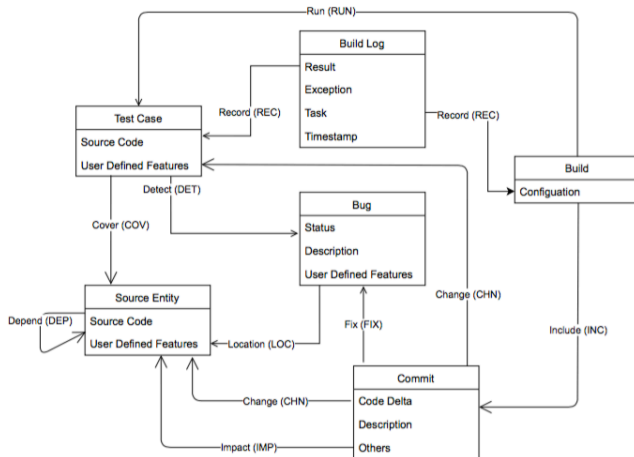


Figure – Modèle de données

Solution proposée (méthodologie)

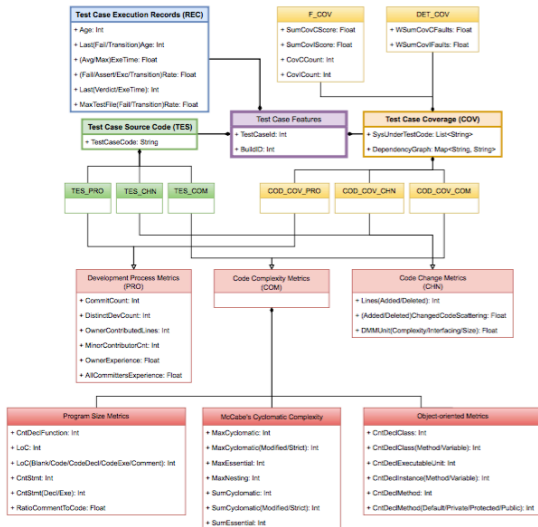


Figure – Modèle de caractéristiques

Avantages de la solution proposée

- Réduction du temps et conservation de la précision de la classification en utilisant le modèle eXtreme Gradient Boosting (XGB)
- La construction des graphes de dépendances permettent d'évaluer l'impact des cas de test
- La modélisation des caractéristiques des données du contexte des IC permet d'extraire les informations plus large dont les variation sont plus susceptibles de faciliter la détecter des défauts.
 - Caractéristiques du code source des cas de test (TES)
 - Caractéristiques des enregistrements de l'exécution des cas de test (REC)
 - Caractéristiques de la couverture des cas de test (COV).
- Large de données représentatives avec 25 sujets étudiés et 25 000 builds

- Surestimation des cas de test (mesure fait sur les fichiers)
- Doublons dans les enregistrements d'exécutions
- Source de données diverses
- Possibilité de défaut dans les outils (Understand) et les ensembles de données (sujets)

Résultats expérimentaux

S_{ID}	Subject	SLOC	Java SLOC	# Commits	Time period (months)	# Builds	# Failed Builds	Failure Rate (%)	Avg. # TC/Build	Avg. Test Time (min)
S_1	JMRI/JMRI	4.56M	1.05M	69.3k	5	1,481	65	4	4364	25
S_2	apache/airavata	1.46M	731k	10.0k	15	236	83	35	49	6
S_3	SonarSource/sonarqube	899k	398k	31.8k	18	4,286	230	5	1309	6
S_4	apache/sling	695k	388k	47.4k	7	1,403	343	24	189	6
S_5	camunda/camunda-bpm-platform	653k	395k	20.6k	34	822	125	15	569	23
S_6	facebook/buck	586k	384k	26.3k	10	846	130	15	663	26
S_7	apache/shardingsphere	422k	165k	29.6k	7	1,049	123	11	789	17
S_8	b2ihealthcare/snow-owl	373k	212k	13.4k	2	277	21	7	46	10
S_9	Angel-ML/angel	336k	204k	3.0k	23	308	124	40	33	20
S_{10}	apache/logging-log4j2	313k	166k	12.7k	13	441	122	27	544	8
S_{11}	eclipse/jetty.project	282k	199k	25.0k	2	192	89	46	137	6
S_{12}	optimatika/ojAlgo	246k	84k	1.6k	22	254	72	28	136	9
S_{13}	yamcs/Yamcs	229k	123k	5.6k	24	504	61	12	114	6
S_{14}	eclipse/steady	221k	98k	2.0k	13	675	51	7	81	7
S_{15}	Graylog2/graylog2-server	182k	85k	22.3k	53	3,668	124	3	110	20
S_{16}	CompEvol/beast2	159k	83k	3.0k	85	415	115	27	65	6
S_{17}	EMResearch/EvoMaster	158k	25k	4.1k	7	583	109	18	100	12
S_{18}	apache/rocketmq	135k	100k	2.0k	16	536	56	10	182	17
S_{19}	zolyfarkas/spf4j	125k	79k	32.6k	37	587	180	30	116	7
S_{20}	spring-cloud/spring-cloud-dataflow	104k	54k	3.5k	9	408	19	4	115	17
S_{21}	cantaloupe-project/cantaloupe	98k	77k	4.5k	29	450	65	14	148	11
S_{22}	thinkarelius/titan	85k	40k	5.1k	25	384	41	10	45	48
S_{23}	apache/curator	84k	58k	3.1k	21	517	65	12	115	67
S_{24}	jcabi/jcabi-github	61k	32k	2.8k	29	788	6	< 0.01	176	14
S_{25}	eclipse/paho.mqtt.java	61k	34k	1.0k	16	378	77	20	37	15

Figure – Liste des sujets prise en compte

Résultats expérimentaux

S_{ID}	COD_COV_COM			COD_COV_PRO			DET_COV			COD_COV_CHN			F_COV			TES_COM			TES_PRO			REC			TES_CHN		
	P	M	T	P	M	T	P	M	T	P	M	T	P	M	T	P	M	T	P	M	T	P	M	T	P	M	T
S_1	508.9	0.4	509.3	455.8	5.5	461.3	452.8	4.1	456.9	451.4	6.1	457.5	451.4	0.0	451.4	57.5	0.1	57.6	4.4	28.9	33.3	0.4	139.1	139.4	0.0	5.6	5.6
S_2	61.6	0.0	61.6	23.9	0.2	24.2	23.6	0.1	23.7	23.3	2.2	25.5	23.3	0.0	23.3	38.3	0.0	38.3	0.6	0.4	1.0	0.0	0.5	0.5	0.0	0.4	0.4
S_3	262.7	0.1	262.8	244.2	0.8	245.0	244.1	0.5	244.6	243.8	4.5	248.4	243.8	0.0	243.8	18.9	0.0	18.9	0.4	5.1	5.5	0.1	21.9	22.0	0.0	4.2	4.2
S_4	98.6	0.0	98.6	74.3	0.2	74.5	74.1	0.1	74.3	74.1	1.1	75.2	74.1	0.0	74.1	24.6	0.0	24.6	0.2	0.9	1.2	0.0	1.3	1.4	0.0	1.1	1.1
S_5	130.0	0.1	130.1	107.9	0.7	108.6	107.7	0.4	108.1	106.8	1.3	108.1	106.8	0.0	106.8	23.2	0.0	23.2	1.1	3.6	4.7	0.1	6.6	6.7	0.0	1.2	1.2
S_6	69.7	0.1	69.9	49.4	0.7	50.1	49.2	0.3	49.5	48.9	1.6	50.5	48.9	0.0	48.9	20.9	0.0	20.9	0.5	3.3	3.8	0.1	5.6	5.7	0.0	1.5	1.5
S_7	110.3	0.1	110.3	93.8	0.6	94.3	93.2	0.3	93.5	92.5	1.6	94.1	92.5	0.0	92.5	17.7	0.0	17.8	1.2	3.3	4.5	0.1	6.2	6.3	0.0	1.5	1.5
S_8	47.5	0.0	47.5	33.5	0.3	33.8	32.8	0.2	33.0	31.2	0.7	31.9	31.2	0.0	31.2	16.2	0.0	16.2	2.3	0.5	2.8	0.0	0.6	0.6	0.0	0.6	0.6
S_9	29.4	0.0	29.4	21.0	0.1	21.1	20.8	0.1	20.9	19.6	0.3	20.0	19.6	0.0	19.6	9.8	0.0	9.8	1.4	0.2	1.6	0.0	0.3	0.3	0.0	0.3	0.3
S_{10}	41.0	0.0	41.1	30.2	0.2	30.4	30.1	0.1	30.1	29.8	0.5	30.2	29.8	0.0	29.8	11.2	0.0	11.3	0.5	2.5	3.0	0.1	3.7	3.8	0.0	0.4	0.4
S_{11}	65.3	0.0	65.3	52.2	1.1	53.2	52.0	0.8	52.9	51.6	0.6	52.3	51.6	0.0	51.6	13.7	0.0	13.7	0.5	1.7	2.2	0.1	1.4	1.4	0.0	0.6	0.6
S_{12}	22.4	0.0	22.4	16.2	0.3	16.6	16.0	0.1	16.1	14.3	0.9	15.1	14.3	0.0	14.3	8.1	0.0	8.1	2.0	0.8	2.8	0.0	1.1	1.1	0.0	0.5	0.5
S_{13}	59.0	0.0	59.0	16.8	0.3	17.1	16.4	0.1	16.5	14.9	10.4	25.3	14.9	0.0	14.9	44.1	0.0	44.1	1.9	0.7	2.6	0.0	1.0	1.1	0.0	1.0	1.0
S_{14}	70.5	0.0	70.5	59.4	0.2	59.6	59.0	0.0	59.1	50.3	0.6	50.9	50.3	0.0	50.3	20.2	0.0	20.2	9.1	0.5	9.5	0.1	0.6	0.6	0.0	0.4	0.4
S_{15}	7.1	0.0	7.1	4.5	0.2	4.7	4.4	0.2	4.6	4.2	0.8	5.0	4.2	0.0	4.2	2.9	0.0	2.9	0.2	0.4	0.7	0.0	0.4	0.5	0.0	0.8	0.8
S_{16}	12.9	0.0	12.9	6.3	0.1	6.4	6.2	0.0	6.3	6.2	0.2	6.4	6.2	0.0	6.2	6.7	0.0	6.7	0.2	0.4	0.6	0.0	0.6	0.6	0.0	0.1	0.1
S_{17}	6.0	0.0	6.0	2.8	0.1	2.9	2.7	0.0	2.8	2.5	0.2	2.7	2.5	0.0	2.5	3.5	0.0	3.5	0.3	0.5	0.8	0.0	1.1	1.1	0.0	0.2	0.2
S_{18}	29.8	0.0	29.8	18.4	0.3	18.7	18.3	0.0	18.3	17.7	0.4	18.1	17.7	0.0	17.7	12.1	0.0	12.1	0.7	0.9	1.6	0.1	1.3	1.4	0.0	0.3	0.3
S_{19}	21.1	0.0	21.1	13.6	0.7	14.3	12.6	0.5	13.1	12.5	1.6	14.0	12.5	0.0	12.5	8.6	0.0	8.6	1.2	1.1	2.3	0.0	1.6	1.6	0.0	1.5	1.5
S_{20}	23.9	0.0	23.9	29.0	0.2	29.2	27.3	0.1	27.3	16.9	3.6	20.5	16.9	0.0	16.9	7.0	0.0	7.0	12.1	0.6	12.7	0.0	1.0	1.0	0.0	3.5	3.5
S_{21}	10.1	0.0	10.1	4.5	0.2	4.7	4.4	0.0	4.4	4.2	0.3	4.6	4.2	0.0	4.3	5.9	0.0	5.9	0.2	0.8	1.0	0.0	1.3	1.3	0.0	0.3	0.3
S_{22}	10.6	0.0	10.7	8.7	0.3	9.0	8.4	0.2	8.6	6.0	0.5	6.6	6.0	0.0	6.0	4.6	0.0	4.6	2.7	0.5	3.2	0.0	0.6	0.6	0.0	0.5	0.5
S_{23}	10.3	0.0	10.3	5.4	0.1	5.5	5.3	0.0	5.4	5.3	0.1	5.4	5.3	0.0	5.3	5.0	0.0	5.1	0.1	0.5	0.6	0.0	0.7	0.7	0.0	0.1	0.1
S_{24}	7.3	0.0	7.3	5.2	0.1	5.3	5.1	0.1	5.2	5.0	0.2	5.1	5.0	0.0	5.0	2.3	0.0	2.3	0.2	0.7	1.0	0.0	1.3	1.3	0.0	0.1	0.1
S_{25}	6.7	0.0	6.7	3.2	0.1	3.3	3.1	0.0	3.1	2.8	0.1	2.9	2.8	0.0	2.8	3.8	0.0	3.8	0.3	0.2	0.6	0.0	0.3	0.4	0.0	0.1	0.1
Avg	83.9	0.0	84.0	68.5	0.5	69.0	68.1	0.3	68.4	67.3	1.6	68.9	67.3	0.0	67.3	16.6	0.0	16.6	1.2	2.3	3.5	0.1	7.2	7.3	0.0	1.3	1.3

Figure – Temps moyen de prétraitement (P), de mesure (M) et de collecte totale (T) des données (en secondes) pour tous les groupes de caractéristiques chez tous les sujets. Pour chaque colonne, la valeur maximale est indiquée en gras.

Résultats expérimentaux

S_{ID}	Subject	SLOC	Java SLOC	# Commits	Time period (months)	# Builds	# Failed Builds	Failure Rate (%)	Avg. # TC/Build	Avg. Test Time (min)
S_1	JMRI/JMRI	4.56M	1.05M	69.3k	5	1,481	65	4	4364	25
S_2	apache/airavata	1.46M	731k	10.0k	15	236	83	35	49	6
S_3	SonarSource/sonarqube	899k	398k	31.8k	18	4,286	230	5	1309	6
S_4	apache/sling	695k	388k	47.4k	7	1,403	343	24	189	6
S_5	camunda/camunda-bpm-platform	653k	395k	20.6k	34	822	125	15	569	23
S_6	facebook/buck	586k	384k	26.3k	10	846	130	15	663	26
S_7	apache/shardingsphere	422k	165k	29.6k	7	1,049	123	11	789	17
S_8	b2ihealthcare/snow-owl	373k	212k	13.4k	2	277	21	7	46	10
S_9	Angel-ML/angel	336k	204k	3.0k	23	308	124	40	33	20
S_{10}	apache/logging-log4j2	313k	166k	12.7k	13	441	122	27	544	8
S_{11}	eclipse/jetty.project	282k	199k	25.0k	2	192	89	46	137	6
S_{12}	optimatika/ojAlgo	246k	84k	1.6k	22	254	72	28	136	9
S_{13}	yamcs/Yamcs	229k	123k	5.6k	24	504	61	12	114	6
S_{14}	eclipse/steady	221k	98k	2.0k	13	675	51	7	81	7
S_{15}	Graylog2/graylog2-server	182k	85k	22.3k	53	3,668	124	3	110	20
S_{16}	CompEvol/beast2	159k	83k	3.0k	85	415	115	27	65	6
S_{17}	EMResearch/EvoMaster	158k	25k	4.1k	7	583	109	18	100	12
S_{18}	apache/rocketmq	135k	100k	2.0k	16	536	56	10	182	17
S_{19}	zolyfarkas/spf4j	125k	79k	32.6k	37	587	180	30	116	7
S_{20}	spring-cloud/spring-cloud-dataflow	104k	54k	3.5k	9	408	19	4	115	17
S_{21}	cantaloupe-project/cantaloupe	98k	77k	4.5k	29	450	65	14	148	11
S_{22}	thinkarelius/titan	85k	40k	5.1k	25	384	41	10	45	48
S_{23}	apache/curator	84k	58k	3.1k	21	517	65	12	115	67
S_{24}	jcabi/jcabi-github	61k	32k	2.8k	29	788	6	< 0.01	176	14
S_{25}	eclipse/paho.mqtt.java	61k	34k	1.0k	16	378	77	20	37	15

Figure – Durée moyenne d'exécution de tous les cas de test par version (durée moyenne des tests) comparée à la durée moyenne de collecte des données pour tous les groupes de fonctionnalités par version (durée moyenne de collecte des données) pour tous les sujets. Les durées sont exprimées en minutes et la valeur maximale de chaque colonne est indiquée en gras.

Dataset	BERT	XGBoost
Zafar et al. [13]	92.2%	89.2%
Levin et al. [12]	78.0%	75.1%
Berger et al. [19]	81.8%	81.6%
Berger et al. [19] (subset [13])	91.8%	90%

Figure – Comparaison du modèle XGB de l'article au modèle BERT

Merci de votre attention

Catherine AKAMBA ¹

Martial TEYOU ¹

Victor DJIEMBOU ¹

¹ *Université de Yaoundé I,
Faculté des Sciences,
Département d'Informatique,
Étudiant Master SD*

References



Wikipedia, “Machine learning,” 2021, accessed : 2nd April 2024.



——, “Software testing,” 2021, accessed : 2nd April 2024.



S. Yoo, M. Harman, G. Fraser, and H. Gross, “Test case prioritization techniques : a systematic literature review,” in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE, 2019, pp. 1105–1116.



S. Yoo, M. Harman, and G. Fraser, “Test case selection techniques : a systematic literature review,” *IEEE Transactions on Software Engineering*, vol. 42, no. 6, pp. 499–528, 2016.



Wikipedia, “Continuous integration,” 2021, accessed : 2nd April 2024.



A. S. Yaraghi, M. Bagherzadeh, N. Kahani, and L. C. Briand, “Scalable and accurate test case prioritization in continuous integration contexts,” *IEEE Transactions on Software Engineering*, vol. 49, no. 4, pp. 1615–1639, 2022.