Détection et analyse de l'impact des défauts de code dans les applications mobiles

Geoffrey Hecht (INRIA - Université Lille - UQAM)

Membres du jury :

Bram Adams (Polytechnique Montréal) - Rapporteur

Jean-Rémy Falleri (Enseirb-Matmeca) - Rapporteur

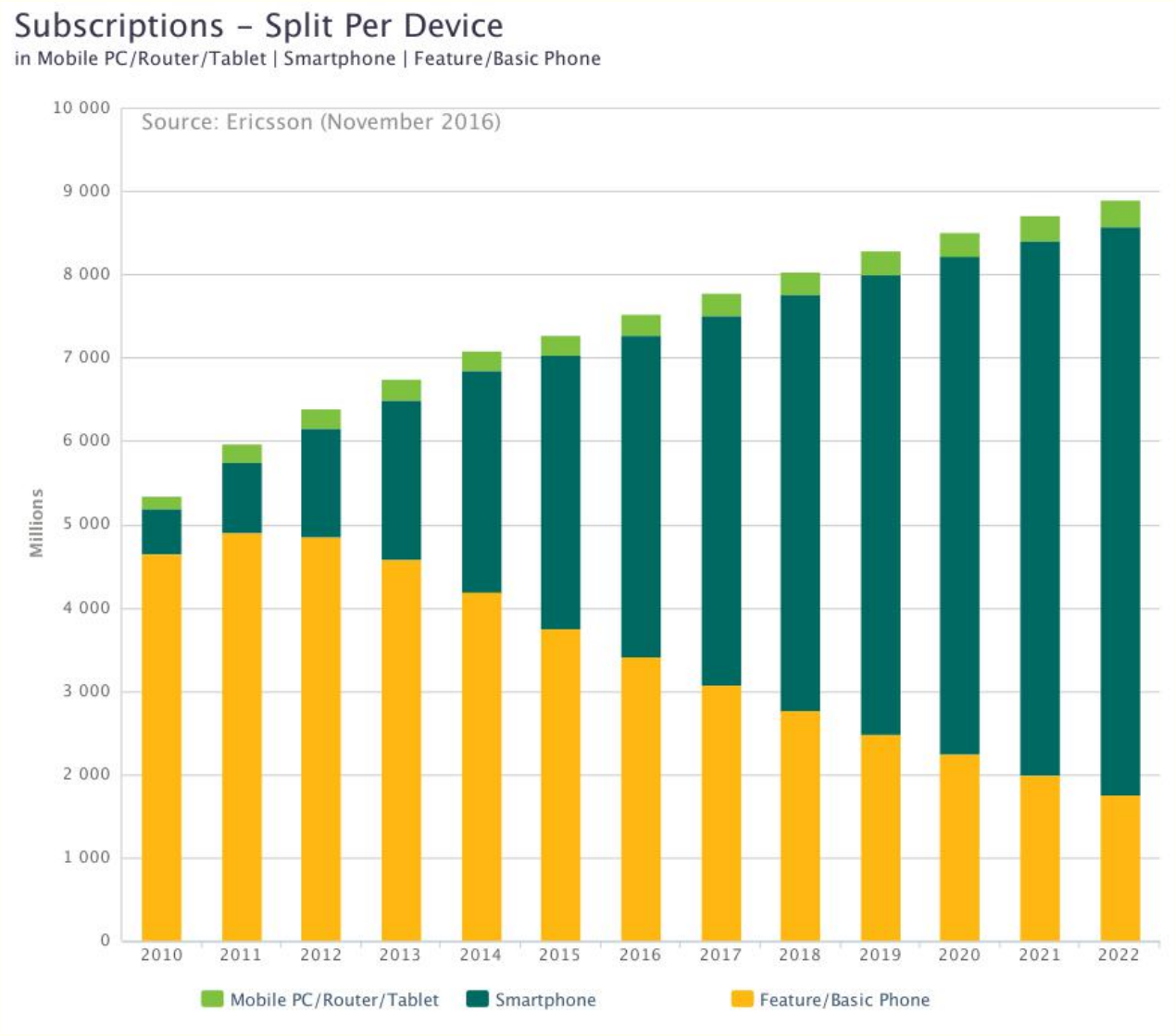
Sébastien Gambs (UQAM) - Examinateur

Pierre-Étienne Moreau (Loria - Mines Nancy) - Examinateur

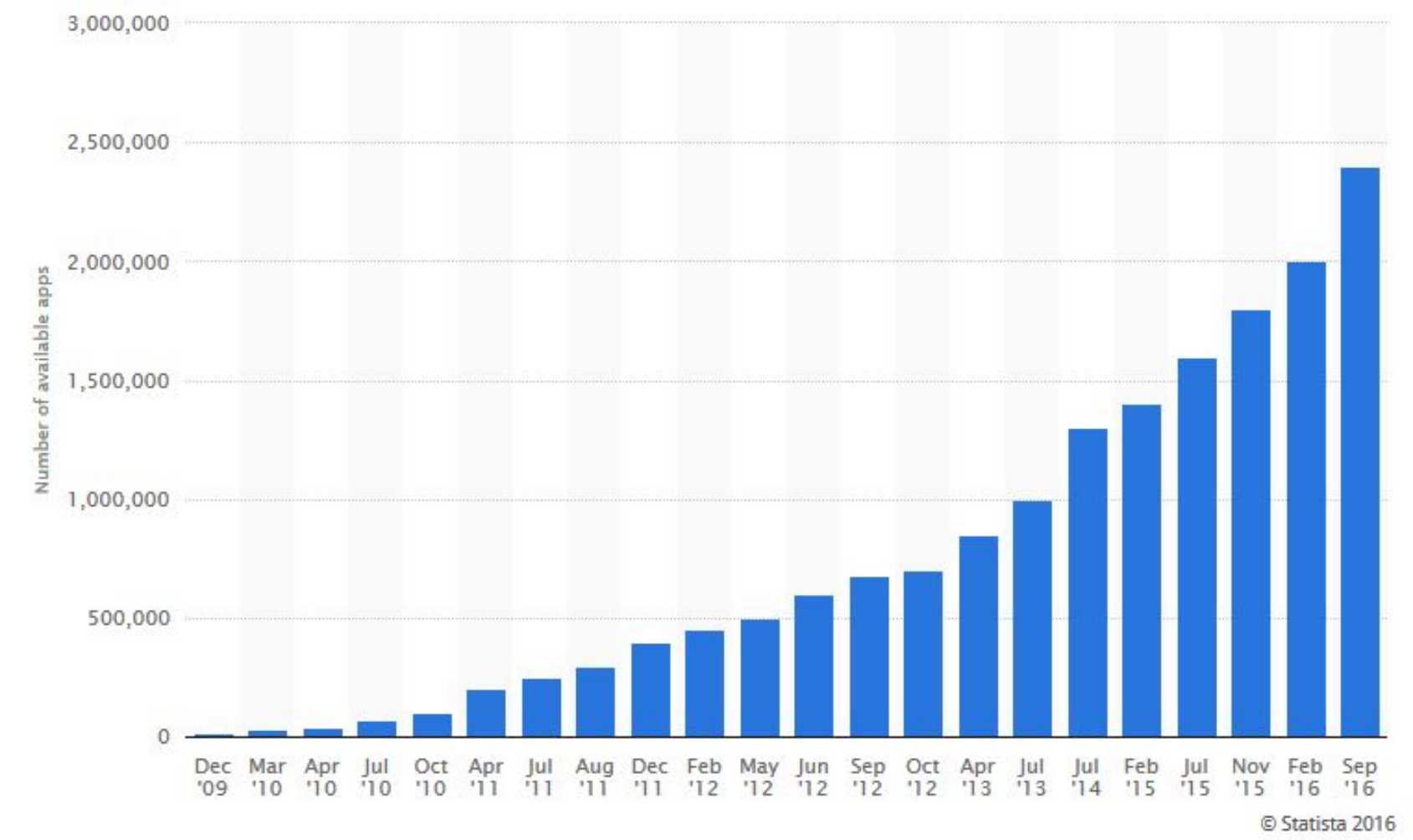
Naouel Moha (UQAM) - Directrice

Romain Rouvoy (INRIA - Université Lille) - Directeur

Succès des smartphones…



et des applications (apps)



Un marché volatile

- **20%** des apps désinstallées le premier jour [localytics, 2014]

Un marché volatile

- **20%** des apps désinstallées le premier jour [localytics, 2014]
- **30%** des apps toujours utilisées après trois mois [localytics, 2014]

Un marché volatile

- **20%** des apps désinstallées le premier jour [localytics, 2014]
- **30%** des apps toujours utilisées après trois mois [localytics, 2014]
- **86%** des utilisateurs ont désinstallé des apps à cause de problèmes de performance [appdynamics, 2015]

Plan

1. Problématique
2. État de l'art
3. Contribution : Classification des défauts de code
4. Contribution : Approche statique de détection
5. Contribution : Approches dynamiques d'analyse de l'impact
6. Conclusion

Problématique

Problématique

- Performance

Problématique

- Performance
- Consommation d'énergie

Problématique

- Performance
- Consommation d’énergie
- Maintenance et évolution

Défauts de code

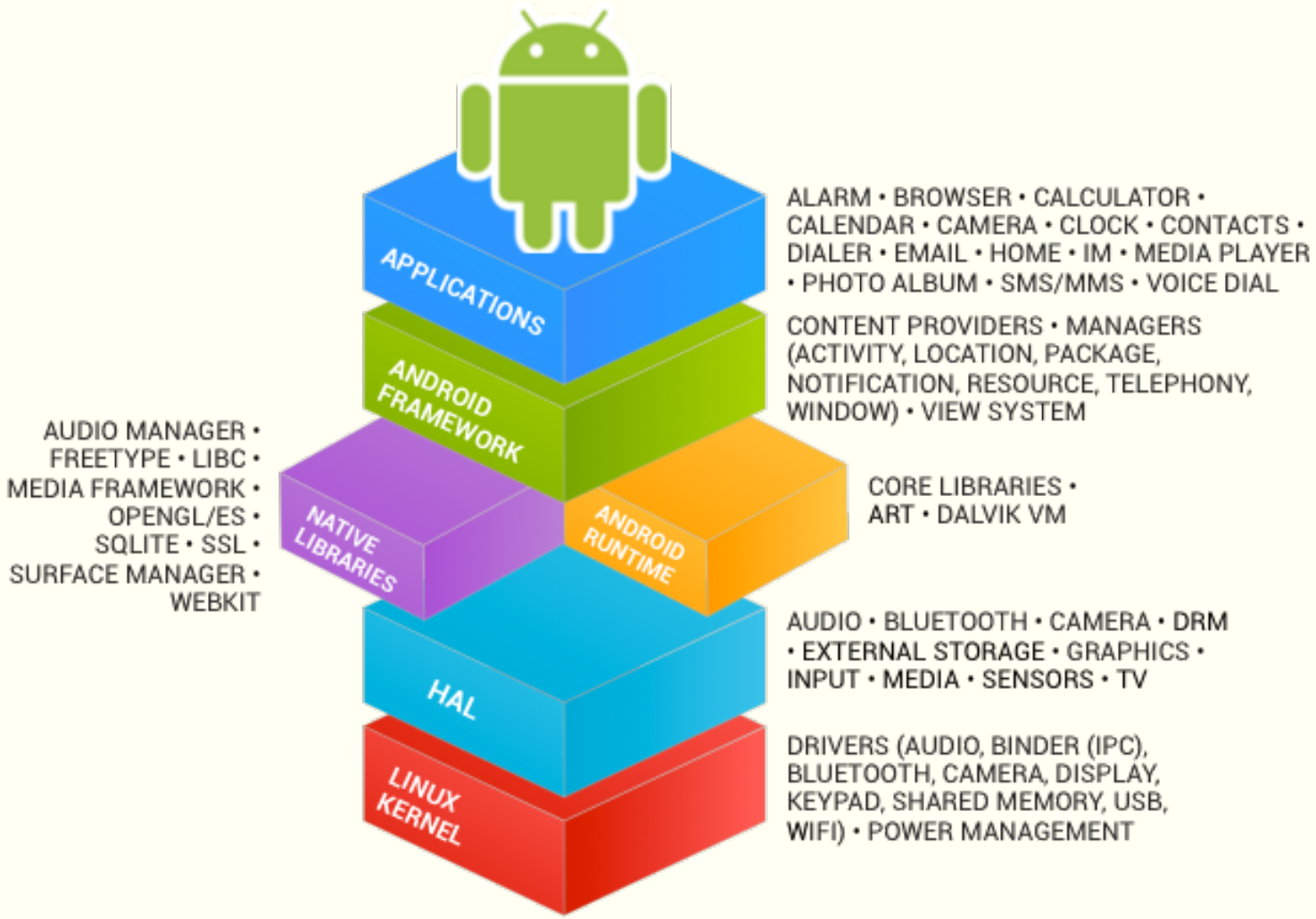
Défauts de code (Code Smells)

Mauvais choix d'implémentation qui peuvent être les symptômes d'importants problèmes conceptuels [1] (Par exemple, *Long Method* et *BLOB*).

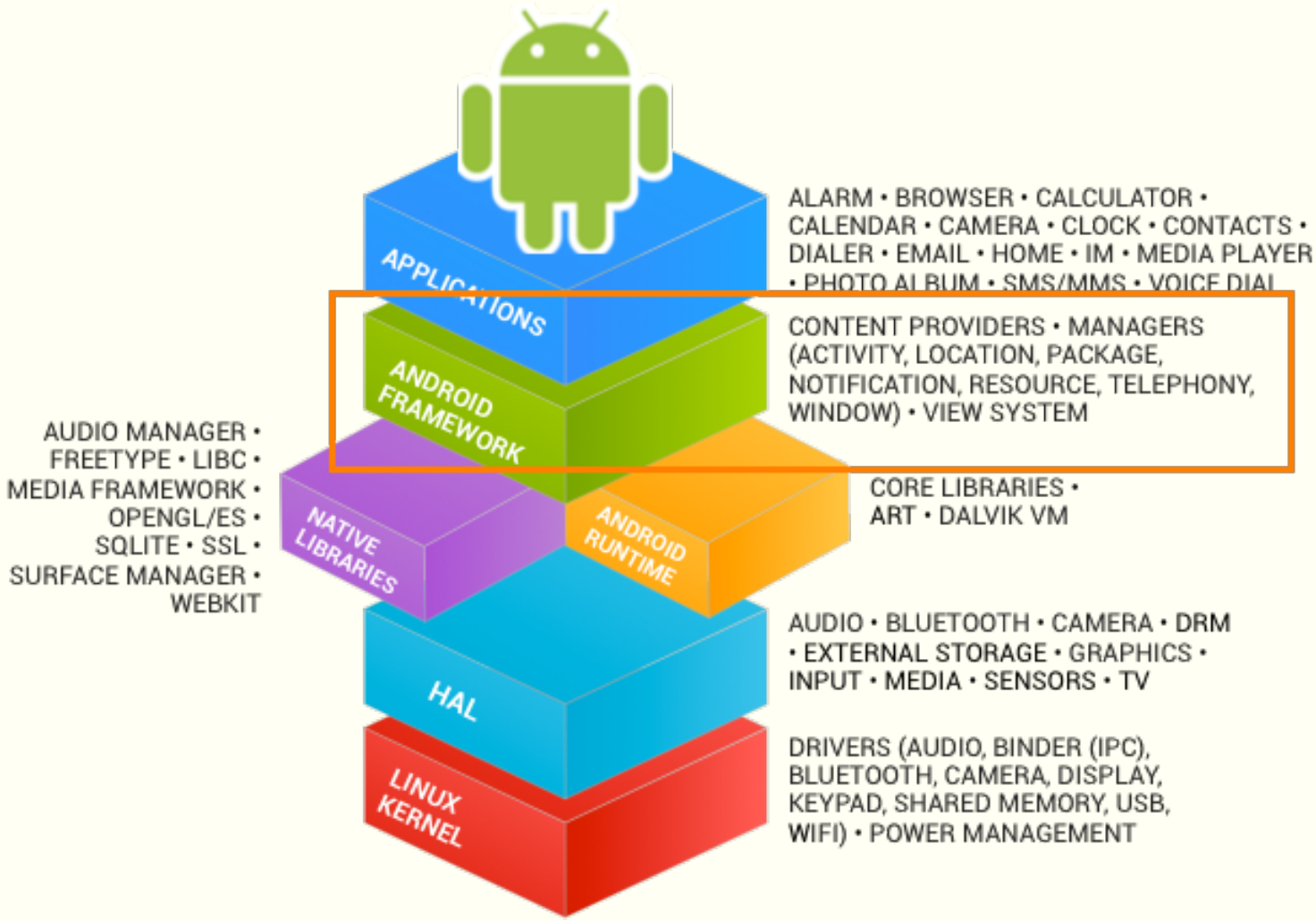
---

1. Martin Fowler et al. *Refactoring : Improving the Design of Existing Code*. Addison Wesley, 1999, p. 1–431.

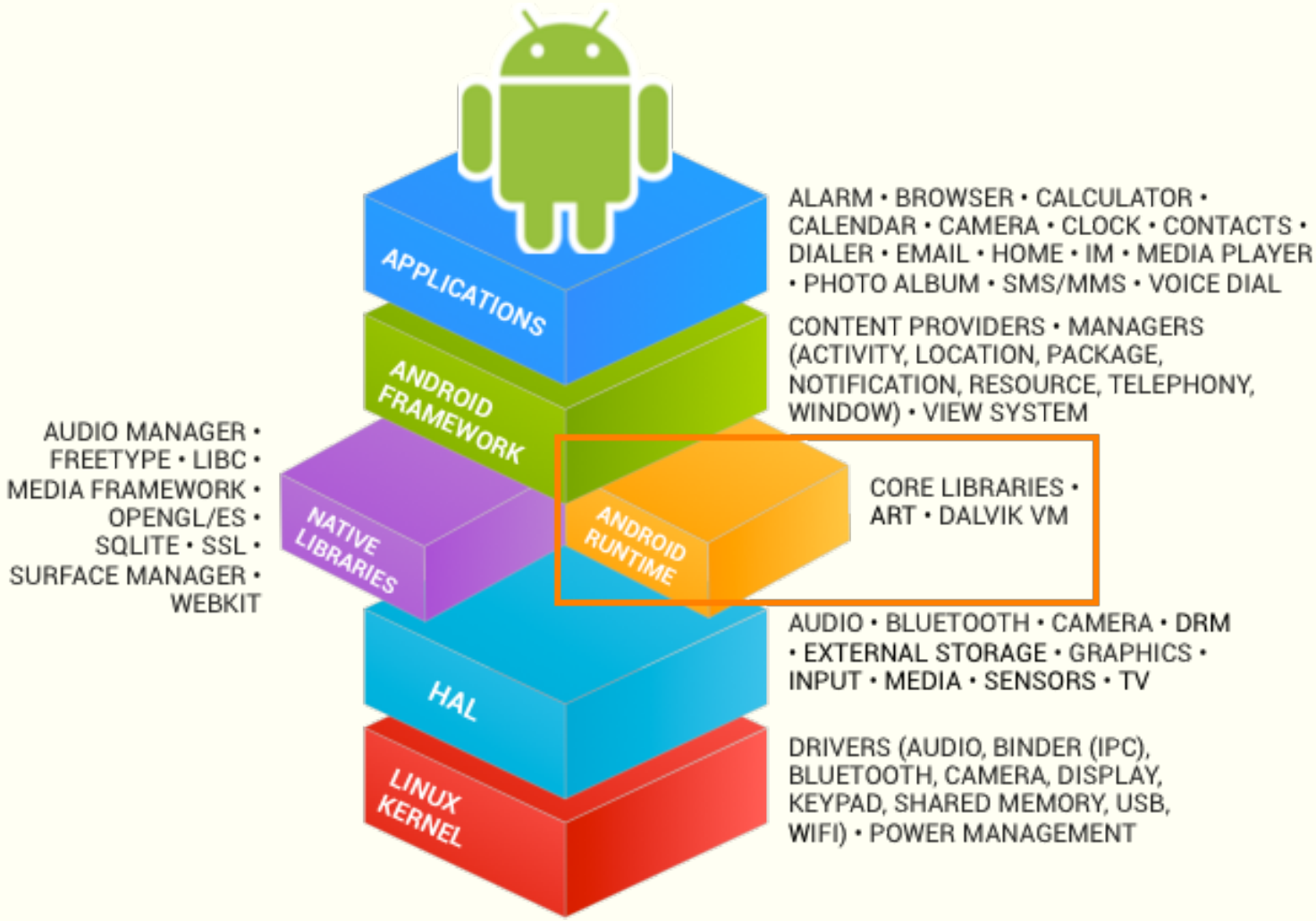
Android


APPLICATIONS
ALARM • BROWSER • CALCULATOR • CALENDAR • CAMERA • CLOCK • CONTACTS • DIALER • EMAIL • HOME • IM • MEDIA PLAYER • PHOTO ALBUM • SMS/MMS • VOICE DIAL
ANDROID FRAMEWORK
CONTENT PROVIDERS • MANAGERS (ACTIVITY, LOCATION, PACKAGE, NOTIFICATION, RESOURCE, TELEPHONY, WINDOW) • VIEW SYSTEM
AUDIO MANAGER • FREETYPE • LIBC • MEDIA FRAMEWORK • OPENGL/ES • SQLITE • SSL • SURFACE MANAGER • WEBKIT
NATIVE LIBRARIES
ANDROID RUNTIME
CORE LIBRARIES • ART • DALVIK VM
HAL
AUDIO • BLUETOOTH • CAMERA • DRM • EXTERNAL STORAGE • GRAPHICS • INPUT • MEDIA • SENSORS • TV
LINUX KERNEL
DRIVERS (AUDIO, BINDER (IPC), BLUETOOTH, CAMERA, DISPLAY, KEYPAD, SHARED MEMORY, USB, WIFI) • POWER MANAGEMENT

Android


APPLICATIONS
ALARM • BROWSER • CALCULATOR •
CALENDAR • CAMERA • CLOCK • CONTACTS •
DIALER • EMAIL • HOME • IM • MEDIA PLAYER
• PHOTO ALBUM • SMS/MMS • VOICE DIAL
ANDROID FRAMEWORK
CONTENT PROVIDERS • MANAGERS
(ACTIVITY, LOCATION, PACKAGE,
NOTIFICATION, RESOURCE, TELEPHONY,
WINDOW) • VIEW SYSTEM
AUDIO MANAGER •
FREETYPE • LIBC •
MEDIA FRAMEWORK •
OPENGL/ES •
SQLITE • SSL •
SURFACE MANAGER •
WEBKIT
NATIVE LIBRARIES
ANDROID RUNTIME
CORE LIBRARIES •
ART • DALVIK VM
HAL
AUDIO • BLUETOOTH • CAMERA • DRM
• EXTERNAL STORAGE • GRAPHICS •
INPUT • MEDIA • SENSORS • TV
LINUX KERNEL
DRIVERS (AUDIO, BINDER (IPC),
BLUETOOTH, CAMERA, DISPLAY,
KEYPAD, SHARED MEMORY, USB,
WIFI) • POWER MANAGEMENT

Android


APPLICATIONS
ALARM • BROWSER • CALCULATOR • CALENDAR • CAMERA • CLOCK • CONTACTS • DIALER • EMAIL • HOME • IM • MEDIA PLAYER • PHOTO ALBUM • SMS/MMS • VOICE DIAL
ANDROID FRAMEWORK
CONTENT PROVIDERS • MANAGERS (ACTIVITY, LOCATION, PACKAGE, NOTIFICATION, RESOURCE, TELEPHONY, WINDOW) • VIEW SYSTEM
NATIVE LIBRARIES
AUDIO MANAGER • FREETYPE • LIBC • MEDIA FRAMEWORK • OPENGL/ES • SQLITE • SSL • SURFACE MANAGER • WEBKIT
ANDROID RUNTIME
CORE LIBRARIES • ART • DALVIK VM
HAL
AUDIO • BLUETOOTH • CAMERA • DRM • EXTERNAL STORAGE • GRAPHICS • INPUT • MEDIA • SENSORS • TV
LINUX KERNEL
DRIVERS (AUDIO, BINDER (IPC), BLUETOOTH, CAMERA, DISPLAY, KEYPAD, SHARED MEMORY, USB, WIFI) • POWER MANAGEMENT

État de l’art

Aperçu

- Manque de spécifications et d'études sur les défauts de code spécifiques aux apps mobiles [2]

---

2. Umme Ayda MANNAN et al. « Understanding code smells in Android applications ». In : *Proceedings of the International Conference on Mobile Software Engineering and Systems*. ACM. 2016, p. 225–234.

Aperçu

- Manque de spécifications et d'études sur les défauts de code spécifiques aux apps mobiles [2]
- Faiblesses des approches de détection disponibles

---

2. Umme Ayda Mannan et al. « Understanding code smells in Android applications ». In : *Proceedings of the International Conference on Mobile Software Engineering and Systems*. ACM. 2016, p. 225–234.

Aperçu

- Manque de spécifications et d'études sur les défauts de code spécifiques aux apps mobiles [2]
- Faiblesses des approches de détection disponibles
- Impact des défauts de code méconnu

---

2. Umme Ayda MANNAN et al. « Understanding code smells in Android applications ». In : *Proceedings of the International Conference on Mobile Software Engineering and Systems*. ACM. 2016, p. 225–234.

État de l'art

	Définitions	Détection/Correction	Étude empirique	Analyse de l'impact perf/conso
Défauts orientés objets	-	-	[Mannan2016]	[Rodriguez2015]
Défauts orientés mobiles	[Reimann2014]	[Reimann2014]	?	[Li2014]

État de l'art

	Définitions	Détection/Correction	Étude empirique	Analyse de l'impact perf/conso
Défauts orientés objets	-	-	[Mannan2016]	[Rodriguez2015]
Défauts orientés mobiles	**[Reimann2014]** [3]	[Reimann2014]	?	[Li2014]

- Catalogue de 30 *quality smells* (pas seulement défauts de code)
- Classification : Accessibilité/sécurité/consommation d'énergie...
- Source principale : Documentation Android

---

3. Jan Reimann, Martin Brylski et Uwe Assmann. « A Tool-Supported Quality Smell Catalogue For Android Developers ». In : *Proc. of the conference Modellierung in the Workshop Modellbasierte und modellgetriebene Softwaremodernisierung – MMSM*. 2014.

État de l'art

	Définitions	Détection/Correction	Étude empirique	Analyse de l'impact perf/conso
Défauts orientés objets	-	-	[Mannan2016]	[Rodriguez2015]
Défauts orientés mobiles	[Reimann2014]	**[Reimann2014]** [4]	?	[Li2014]

- Détection/correction sur modèle EMF
- Code source : quatre défauts détectés
- Pas plus d'informations

4. Jan Reimann, Martin Brylski et Uwe Assmann. « A Tool-Supported Quality Smell Catalogue For Android Developers ». In : *Proc. of the conference Modellierung in the Workshop Modellbasierte und modellgetriebene Softwaremodernisierung – MMSM*. 2014.

État de l'art

	Définitions	Détection/Correction	Étude empirique	Analyse de l'impact perf/conso
Défauts orientés objets	-	-	**[Mannan2016]** [5]	[Rodriguez2015]
Défauts orientés mobiles	[Reimann2014]	[Reimann2014]	?	[Li2014]

- Détection de 20 défauts de code orientés objets
- Comparaison Android/Non-Android
- Différences dans les distributions des défauts de code

5. Umme Ayda MANNAN et al. « Understanding code smells in Android applications ». In : *Proceedings of the International Conference on Mobile Software Engineering and Systems*. ACM. 2016, p. 225–234.

État de l'art

	Définitions	Détection/Correction	Étude empirique	Analyse de l'impact perf/conso
Défauts orientés objets	-	-	[Mannan2016]	[Rodriguez2015]
Défauts orientés mobiles	[Reimann2014]	[Reimann2014]	**?**	[Li2014]

État de l'art

	Définitions	Détection/Correction	Étude empirique	Analyse de l'impact perf/conso
Défauts orientés objets	-	-	[Mannan2016]	**[Rodriguez2015]** [6]
Défauts orientés mobiles	[Reimann2014]	[Reimann2014]	?	[Li2014]

- Correction de défauts de code orientés objets (ex : BLOB)
- Utilisation de scénarios "intensifs"
- La correction de ces défauts peut avoir un impact négatif sur la consommation d'énergie

6. Ana Rodriguez, Mathias Longo et Alejandro Zunino. « Using bad smell-driven code refactorings in mobile applications to reduce battery usage ». In : *16th Simposio Argentino de Ingenieria de Software*. Rosario, Argentina, 2015, p. 56–68.

État de l'art

	Définitions	Détection/Correction	Étude empirique	Analyse de l'impact perf/conso
Défauts orientés objets	-	-	[Mannan2016]	[Rodriguez2015]
Défauts orientés mobiles	[Reimann2014]	[Reimann2014]	?	**[Li2014]** [7]

- Évaluation de bonnes/mauvaises pratiques (requêtes http, utilisation de la mémoire, micro-optimisations)
- Micro-optimisations (correction de défauts) réduisent la consommation d'énergie
- Mesures sur 100 000 itérations d'une boucle

---

7. Ding Li et William GJ Halfond. « An investigation into energy-saving programming practices for android smartphone app development ». In : *Proceedings of the 3rd International Workshop on Green and Sustainable Software*. ACM. 2014, p. 46–53.

Questions de recherche

	Définitions	Détection/Correction	Étude empirique	Analyse de l'impact perf/conso
Défauts orientés objets	-	-	[Mannan2016]	[Rodriguez2015]
Défauts orientés mobiles	[Reimann2014]	[Reimann2014]	?	[Li2014]

- RQ1 : Existe-t-il des défauts de code spécifiques à Android pertinents ?

  Définitions de défauts code spécifiques à Android

Questions de recherche

	Définitions	**Détection/Correction**	**Étude empirique**	Analyse de l'impact perf/conso
Défauts orientés objets	-	-	[Mannan2016]	[Rodriguez2015]
Défauts orientés mobiles	[Reimann2014]	[Reimann2014]	?	[Li2014]

- RQ2 : Pouvons-nous observer des tendances dans la distribution des défauts de code dans les apps Android ?

Détection et études empiriques de la présence des défauts de code

Questions de recherche

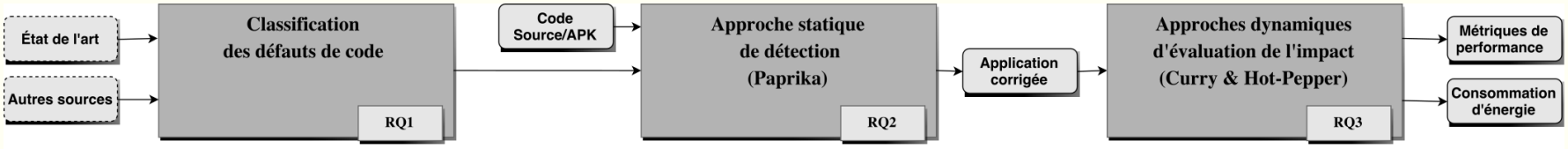
	Définitions	Détection/Correction	Étude empirique	**Analyse de l'impact perf/conso**
Défauts orientés objets	-	-	[Mannan2016]	[Rodriguez2015]
Défauts orientés mobiles	[Reimann2014]	[Reimann2014]	?	[Li2014]

- RQ3 : Est-ce que la correction des défauts de code a un impact sur les performances et la consommation d'énergie ?

  Analyse des performances et de la consommation d'énergie

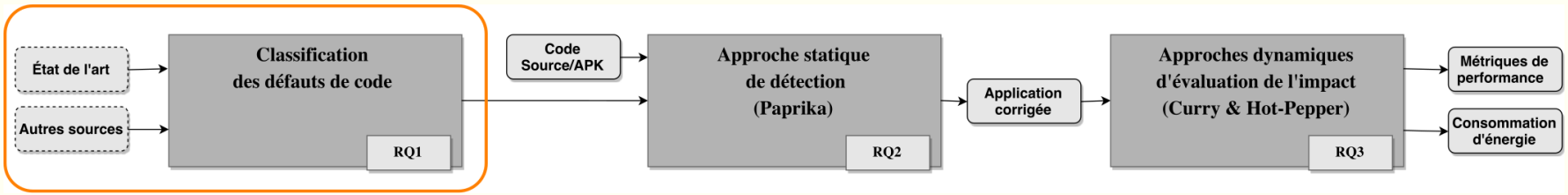
Contributions

Contributions



RQ1 : Existe-t-il des défauts de code spécifiques à Android pertinents ?

Contributions



Classification

- 17 défauts de code (4 OO - 13 Android)

Classification

- 17 défauts de code (4 OO - 13 Android)
- Sources diverses (8 inédits dans la littérature scientifique)

Classification

- 17 défauts de code (4 OO - 13 Android)
- Sources diverses (8 inédits dans la littérature scientifique)
- Multiples critères de classification

Classification

- 17 défauts de code (4 OO - 13 Android)
- Sources diverses (8 inédits dans la littérature scientifique)
- Multiples critères de classification
- Subjectifs/Objectifs

Classification

- 17 défauts de code (4 OO - 13 Android)
- Sources diverses (8 inédits dans la littérature scientifique)
- Multiples critères de classification
- Subjectifs/Objectifs
- Sévérité : Critique / Problématique / Suspect

Classification

- 17 défauts de code (4 OO - 13 Android)
- Sources diverses (8 inédits dans la littérature scientifique)
- Multiples critères de classification
- Subjectifs/Objectifs
- Sévérité : Critique / Problématique / Suspect
- **AFIC** (Android Framework Inherited Classes)/ Toutes classes

Classification

	Orienté Objet				Android												
	Maintenance				Micro-optimisations			Gestion Mémoire				Affichage IHM			Blocage processus		
	BLOB	CC	LM	HAS	IGS	MIM	IOD	LIC	UCS	HMU	NLMR	UHA	IWR	UIO	HAS	HSS	HBR
Objectif/Subjectif																	
Objectif					✗	✗	✗	✗	✗	✗	✗	✗	✗	✗			
Subjectif	✗	✗	✗	✗											✗	✗	✗
Sévérité																	
Critique					✗	✗											
Problématique	✗	✗	✗				✗	✗	✗	✗	✗	✗					
Suspect				✗									✗	✗	✗	✗	✗
Classes affectées																	
AFIC							✗				✗		✗	✗	✗	✗	✗
Toutes classes	✗	✗	✗	✗	✗	✗		✗	✗	✗		✗					

Blob Class (BLOB)

	Orienté Objet				Android												
	Maintenance				Micro-optimisations			Gestion Mémoire				Affichage IHM			Blocage processus		
	BLOB	CC	LM	HAS	IGS	MIM	IOD	LIC	UCS	HMU	NLMR	UHA	IWR	UIO	HAS	HSS	HBR
Objectif/Subjectif																	
Objectif					✗	✗	✗	✗	✗	✗	✗	✗	✗	✗			
Subjectif	✗	✗	✗	✗											✗	✗	✗
Sévérité																	
Critique					✗	✗											
Problématique	✗	✗	✗				✗	✗	✗	✗	✗	✗					
Suspect				✗									✗	✗	✗	✗	✗
Classes affectées																	
AFIC							✗				✗		✗	✗	✗	✗	✗
Toutes classes	✗	✗	✗	✗	✗	✗		✗	✗	✗		✗					

Internal Getter/Setter (IGS)

	Orienté Objet				Android												
	Maintenance				Micro-optimisations			Gestion Mémoire				Affichage IHM			Blocage processus		
	BLOB	CC	LM	HAS	IGS	MIM	IOD	LIC	UCS	HMU	NLMR	UHA	IWR	UIO	HAS	HSS	HBR
Objectif/Subjectif																	
Objectif					✗	✗	✗	✗	✗	✗	✗	✗	✗	✗			
Subjectif	✗	✗	✗	✗											✗	✗	✗
Sévérité																	
Critique					✗	✗											
Problématique	✗	✗	✗				✗	✗	✗	✗	✗	✗					
Suspect				✗									✗	✗	✗	✗	✗
Classes affectées																	
AFIC							✗				✗		✗	✗	✗	✗	✗
Toutes classes	✗	✗	✗	✗	✗	✗		✗	✗	✗		✗					

No Low Memory Resolver (NLMR)

	Orienté Objet				Android												
	Maintenance				Micro-optimisations			Gestion Mémoire				Affichage IHM			Blocage processus		
	BLOB	CC	LM	HAS	IGS	MIM	IOD	LIC	UCS	HMU	NLMR	UHA	IWR	UIO	HAS	HSS	HBR
Objectif/Subjectif																	
Objectif					✗	✗	✗	✗	✗	✗	✗	✗	✗	✗			
Subjectif	✗	✗	✗	✗											✗	✗	✗
Sévérité																	
Critique					✗	✗											
Problématique	✗	✗	✗				✗	✗	✗	✗	✗	✗					
Suspect				✗									✗	✗	✗	✗	✗
Classes affectées																	
AFIC							✗				✗		✗	✗	✗	✗	✗
Toutes classes	✗	✗	✗	✗	✗	✗		✗	✗	✗		✗					

Question de recherche

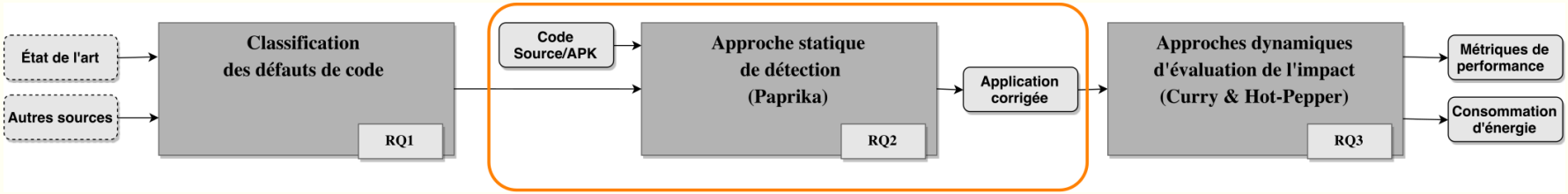
- RQ1 : Existe-t-il des défauts de code spécifiques à Android pertinents ?
- Oui, essentiellement lié à la performance [8].

---

8. Geoffrey HECHT et al. « An Empirical Analysis of Android Code Smells ». In : *En cours de revision pour le journal Transactions on Software Engineering (TSE) - sous réserve d'acceptation*. 2016.

RQ2 : Pouvons-nous observer des tendances dans la distribution des défauts de code dans les apps Android ?

Contributions



Détection de correction : PAPRIKA

- Analyse statique

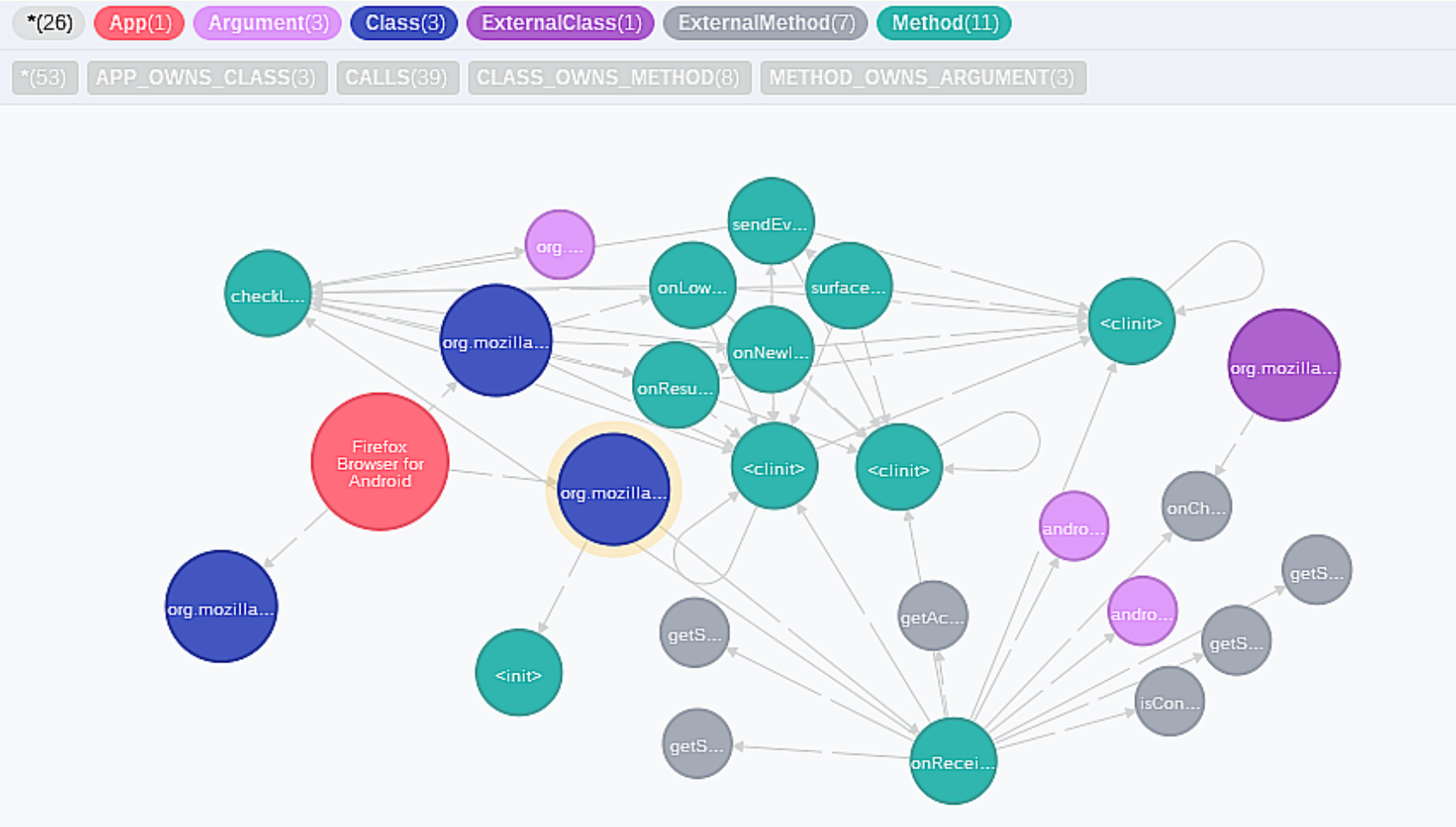
Détection de correction : PAPRIKA

- Analyse statique
- Depuis code source ou APK (paquetage binaire Android)

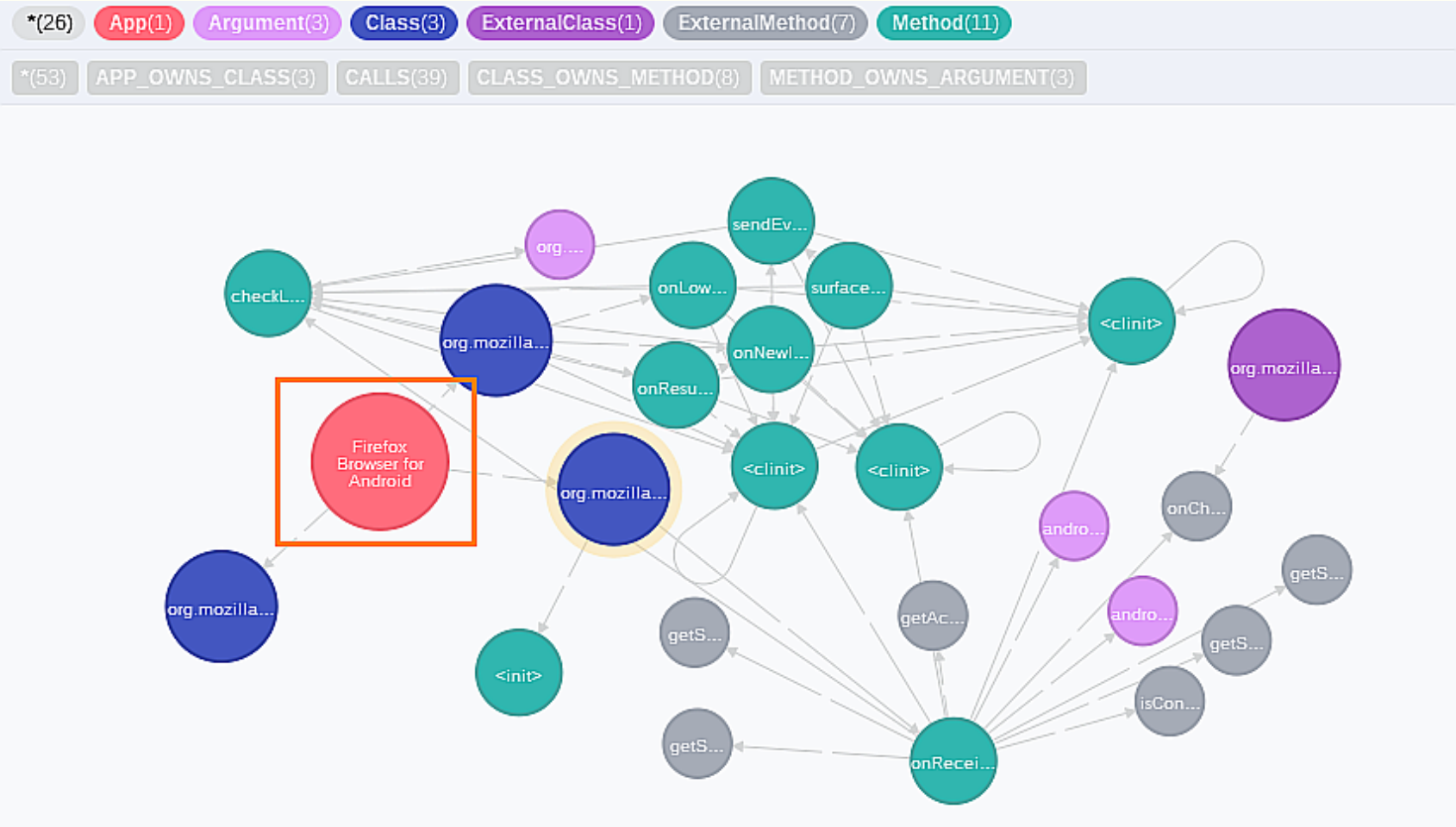
Détection de correction : PAPRIKA

- Analyse statique
- Depuis code source ou APK (paquetage binaire Android)
- Trois étapes : Génération du modèle, conversion en graphe et requêtes

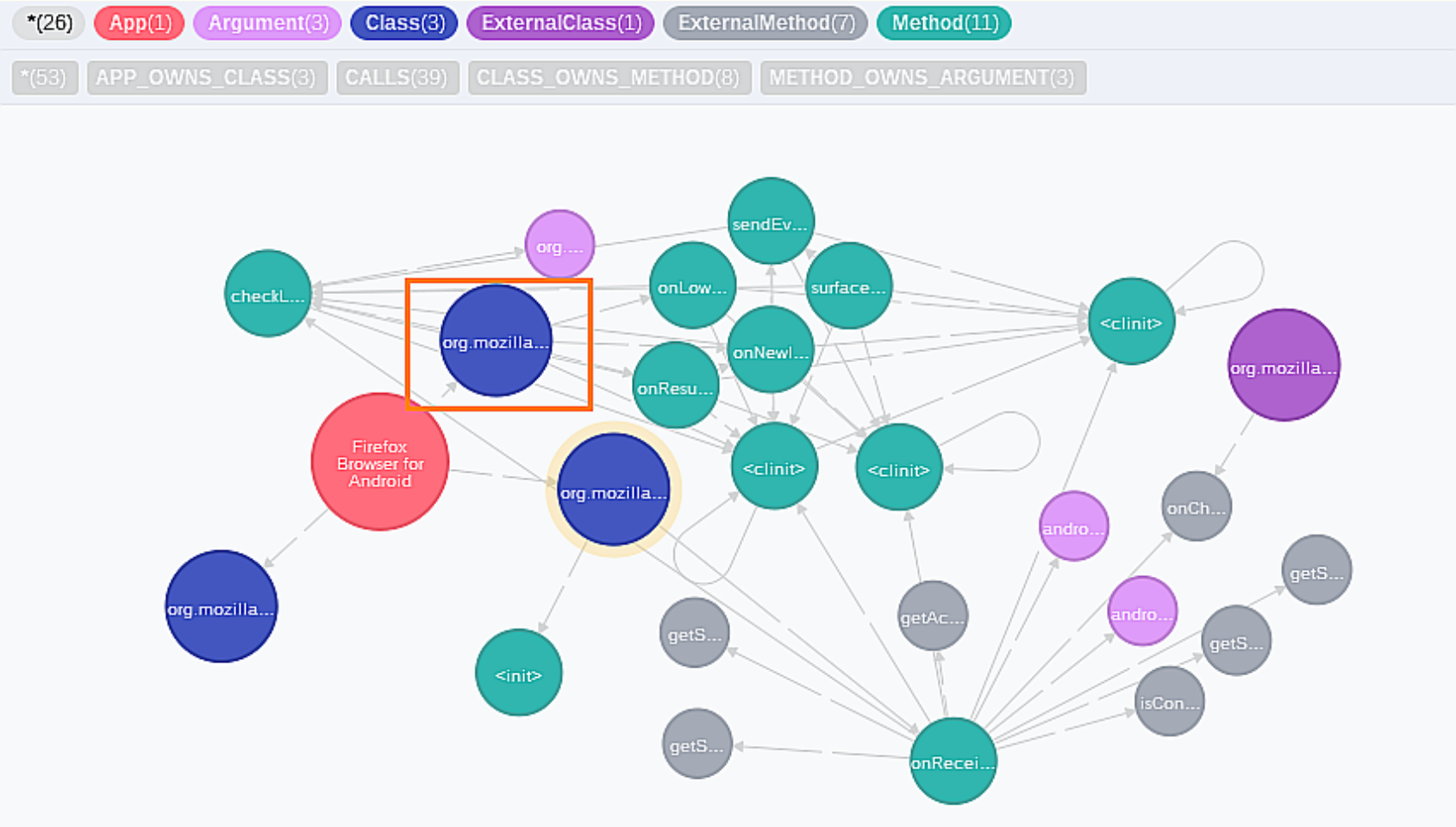
Paprika - Graphe



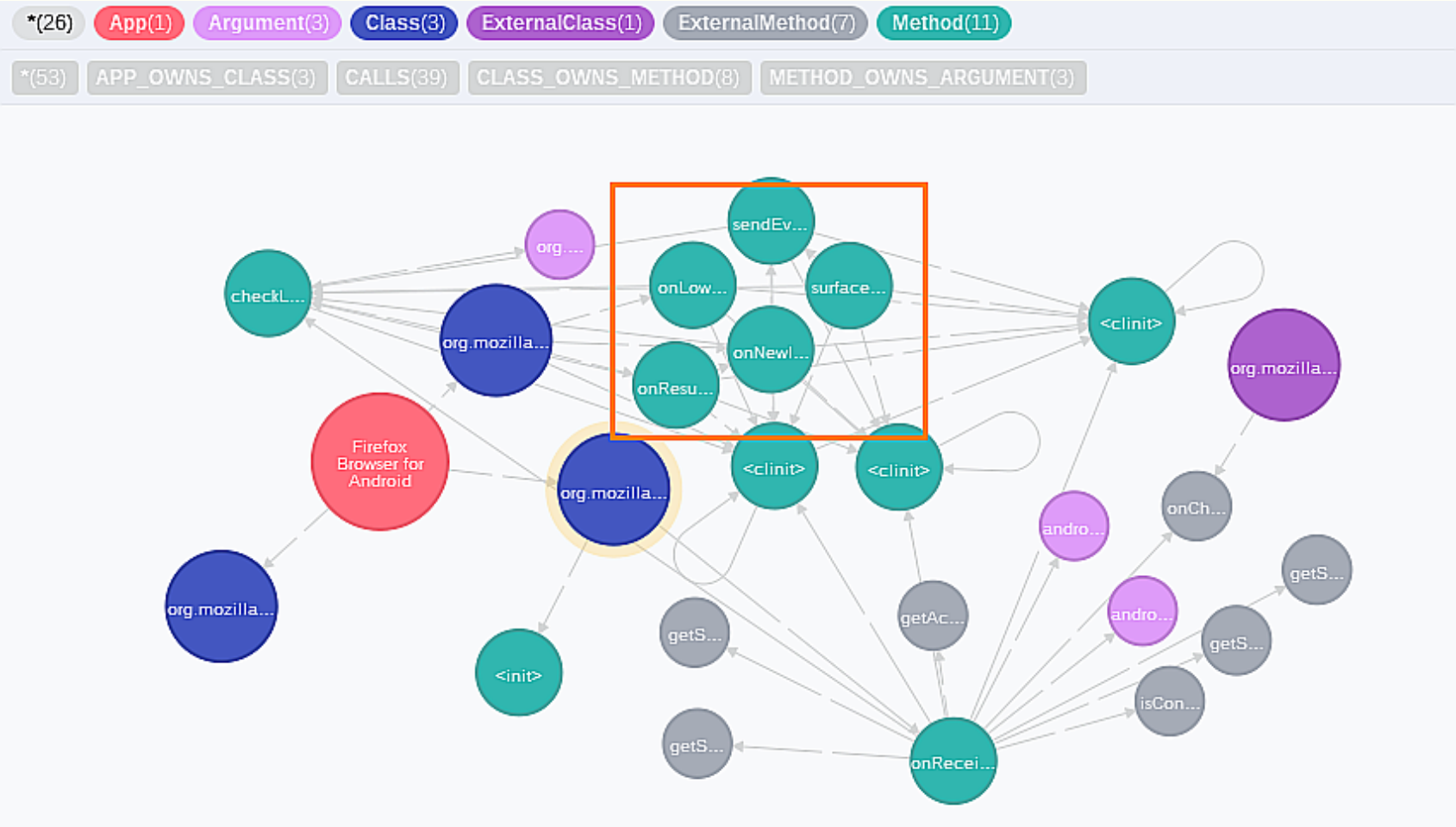
Paprika - Graphe



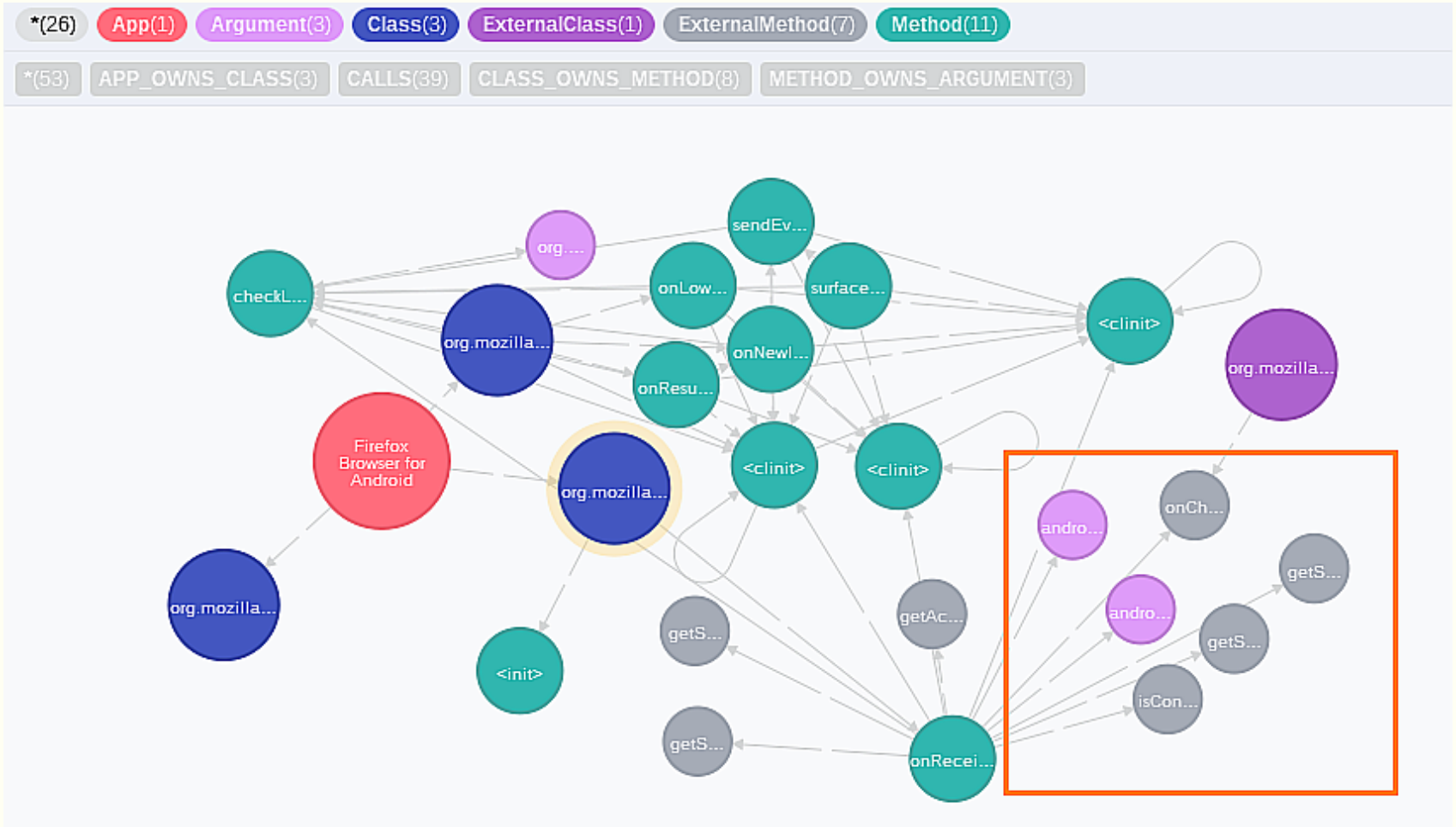
Paprika - Graphe



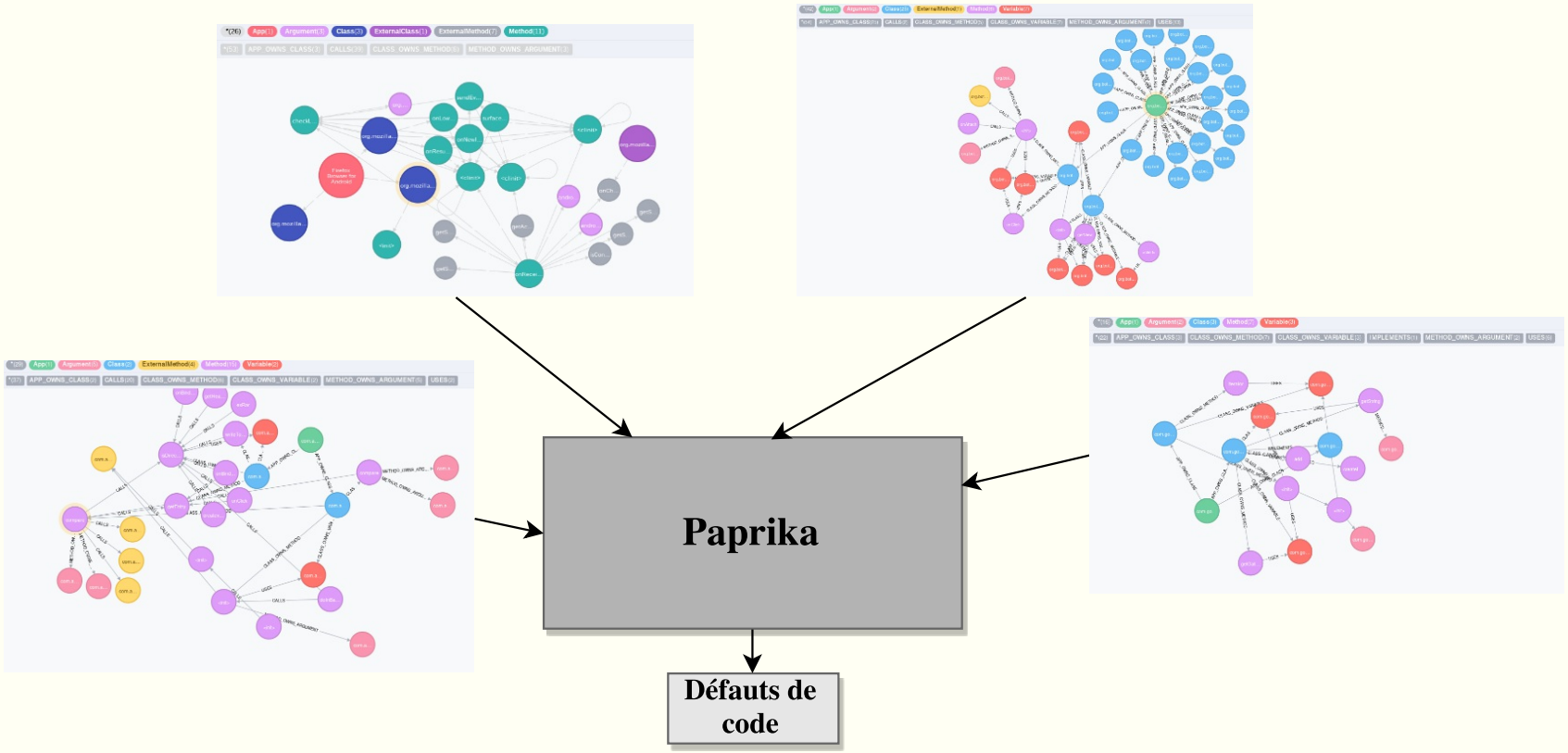
Paprika - Graphe



Paprika - Graphe



PAPRIKA - Utilisation du corpus

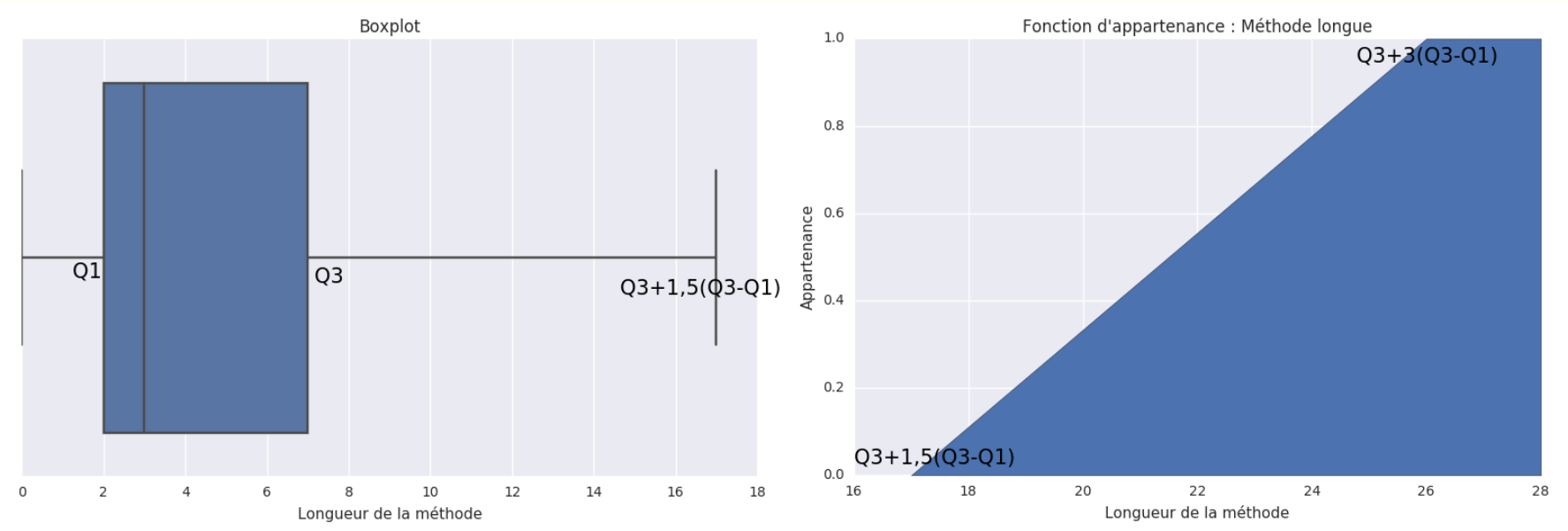


Exemple de requête

Détection de MIM : Pourrait être statique

MATCH (m :Method)
WHERE NOT HAS(m.is_static)
AND NOT HAS(m.is_init)
AND NOT m-[:USES]->(:Variable)
AND NOT m-[:CALLS]->(:Method)
RETURN m

Analyse statistique et logique floue



Mais aussi...

- Correction avec Spoon [9] des défauts de code critiques et objectifs(IGS / MIM / HMU)

---

9. Renaud Pawlak et al. « Spoon : A Library for Implementing Analyses and Transformations of Java Source Code ». In : *Software : Practice and Experience* (2015).

Mais aussi...

- Correction avec Spoon [9] des défauts de code critiques et objectifs(IGS / MIM / HMU)
- Deux études (foule d'apps et évolution des versions dans le temps)

---

9. Renaud Pawlak et al. « Spoon : A Library for Implementing Analyses and Transformations of Java Source Code ». In : *Software : Practice and Experience* (2015).

Étude 1 : Analyse d'une foule d'apps

- Validation par des développeurs Android

Étude 1 : Analyse d'une foule d'apps

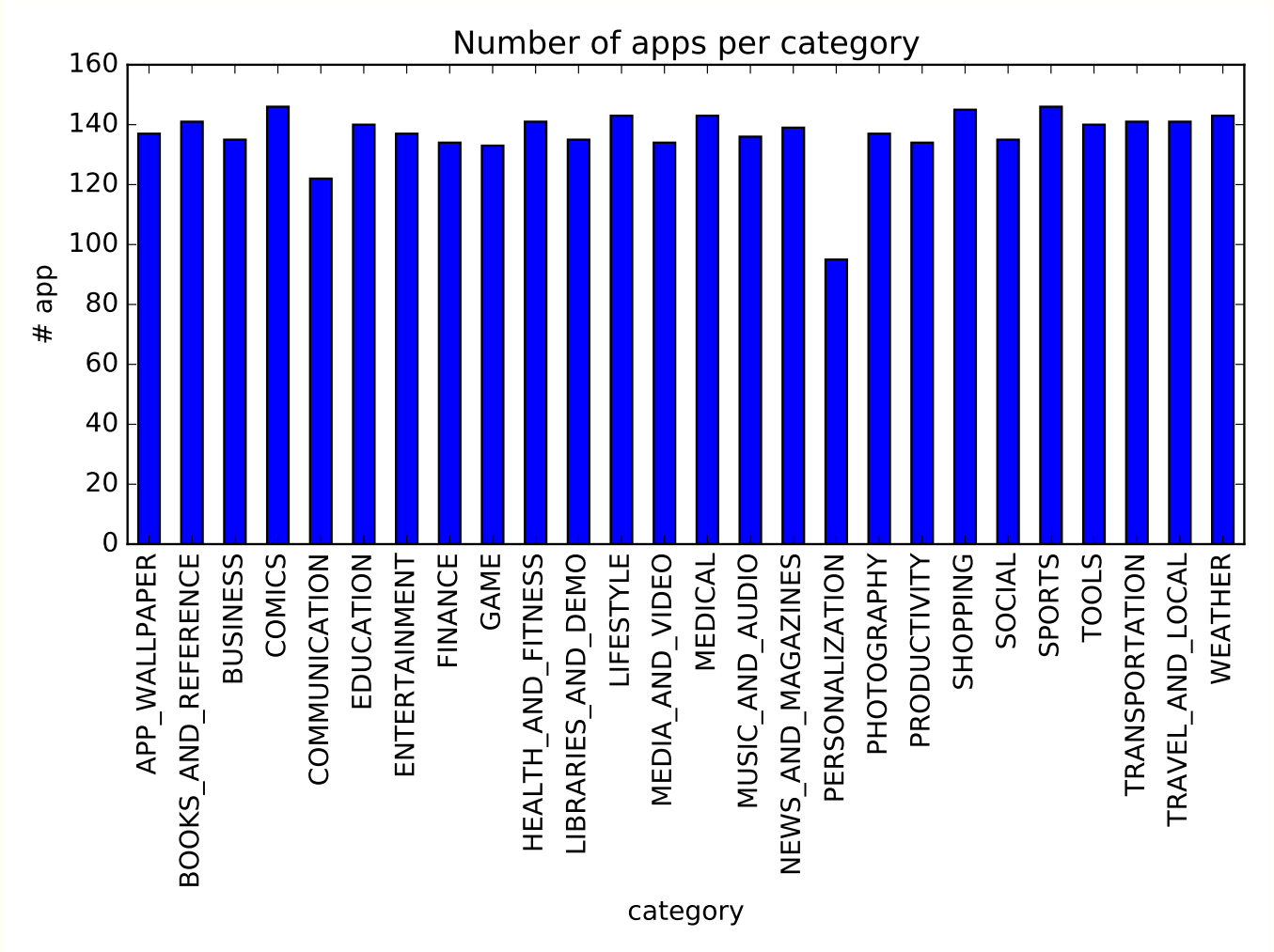
- Validation par des développeurs Android
- Analyse de **3553 apps** populaires avec PAPRIKA

Validation

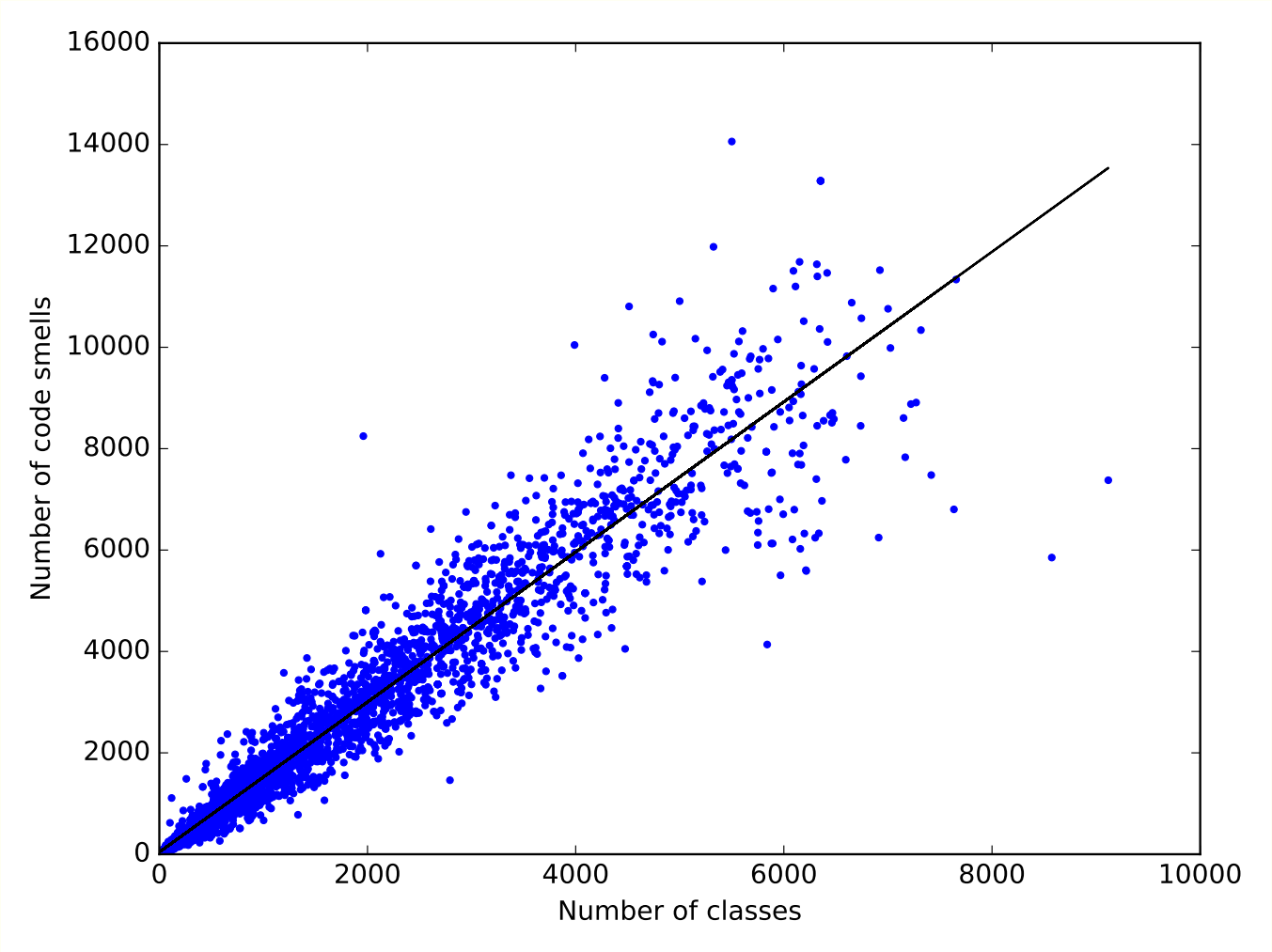
Validation de PAPRIKA par 14 experts sur 7 apps

Exactitude	0.9
Précision	0.88
Rappel	0.93
F1-mesure	0.9

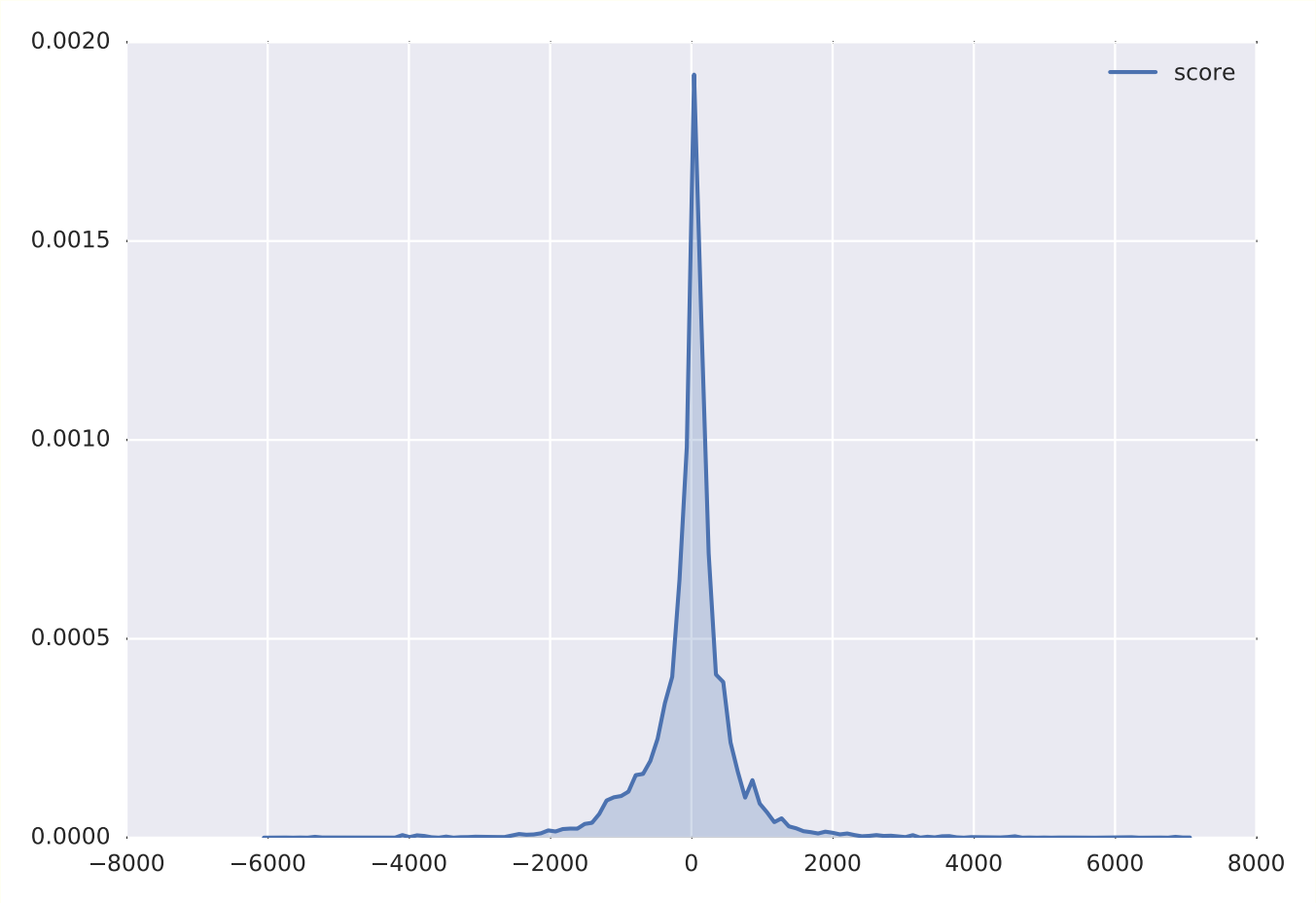
Catégories des apps



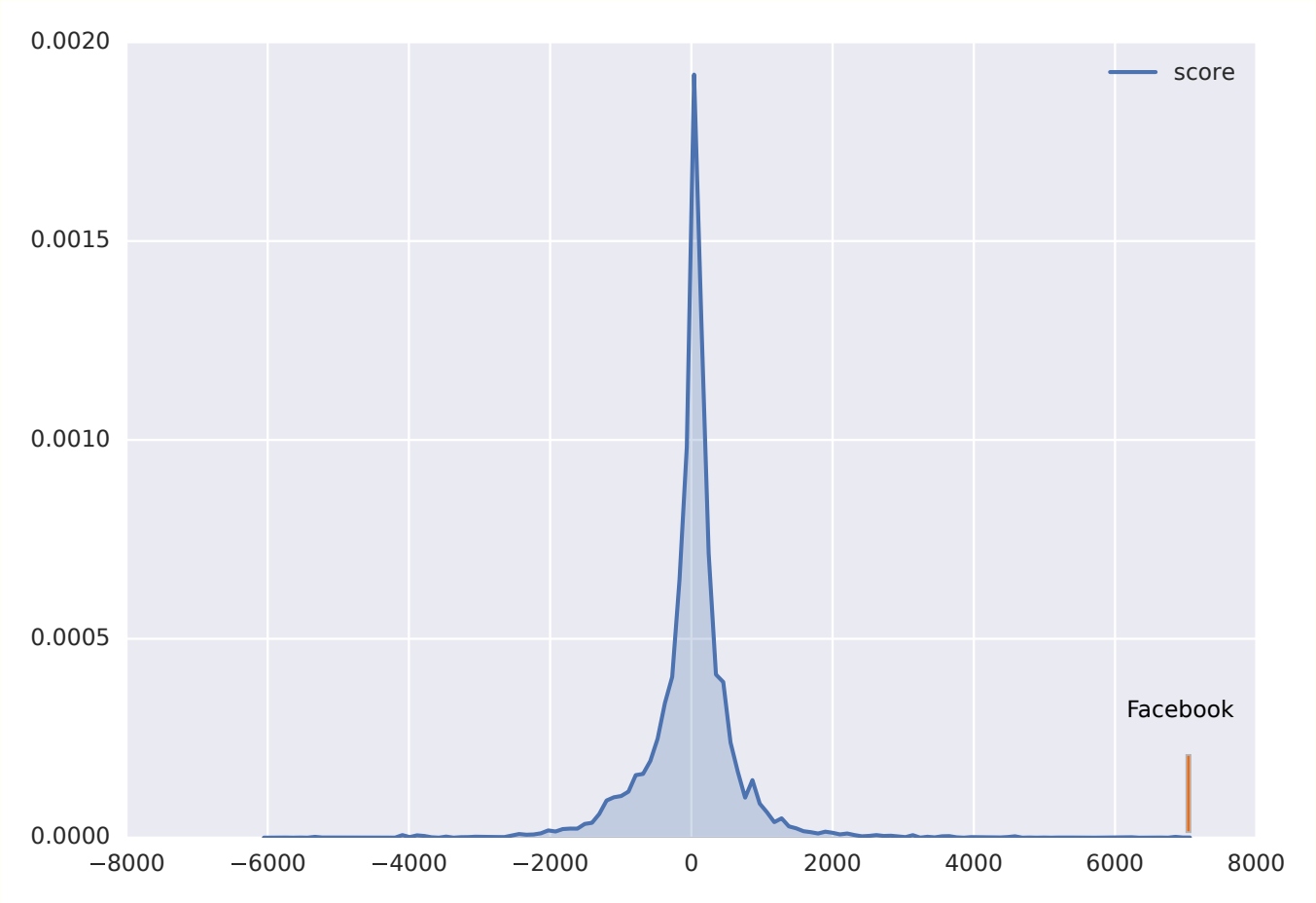
Résultat : Tendance dans la distribution des défauts de code



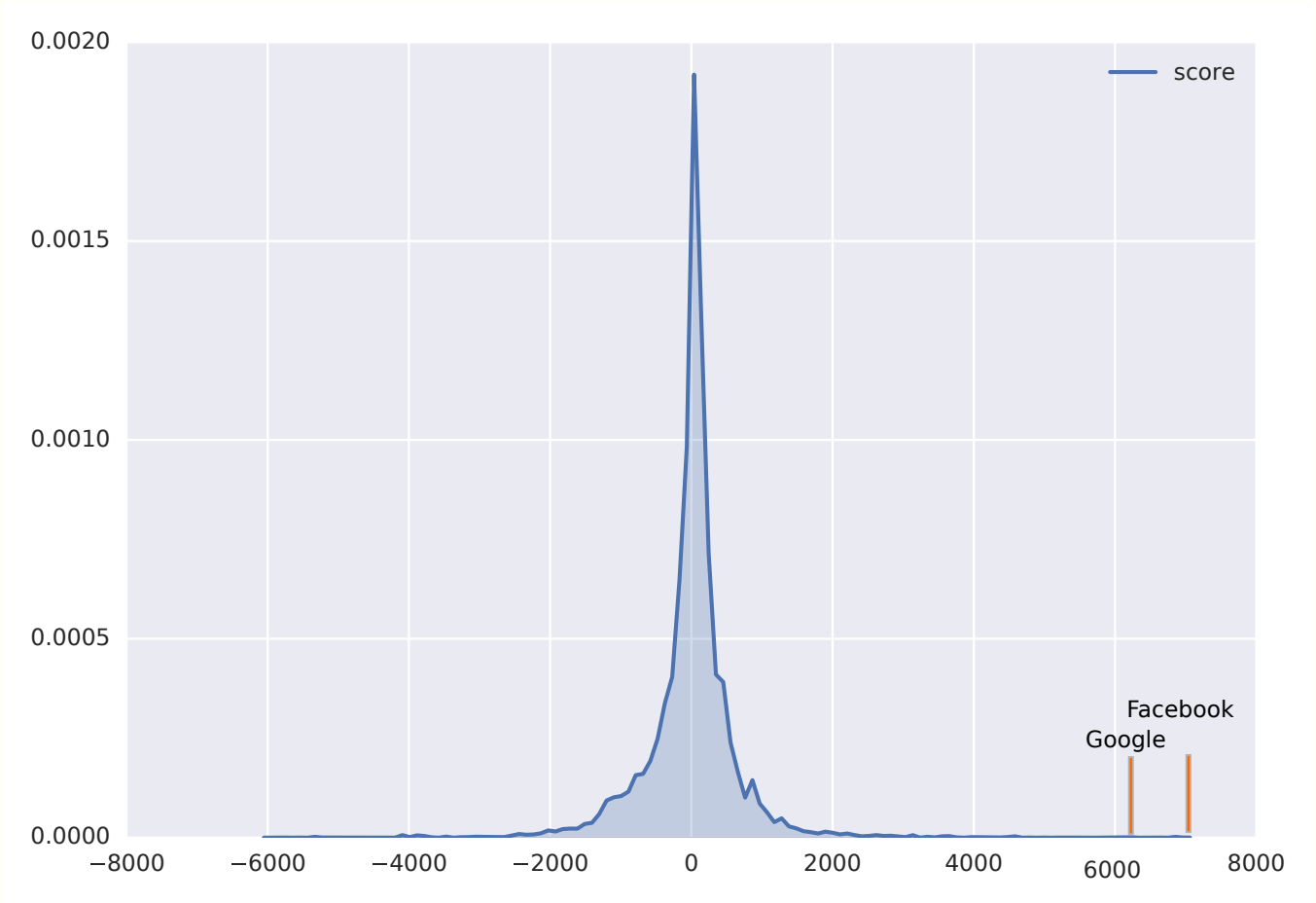
Résultat : utilisation du score



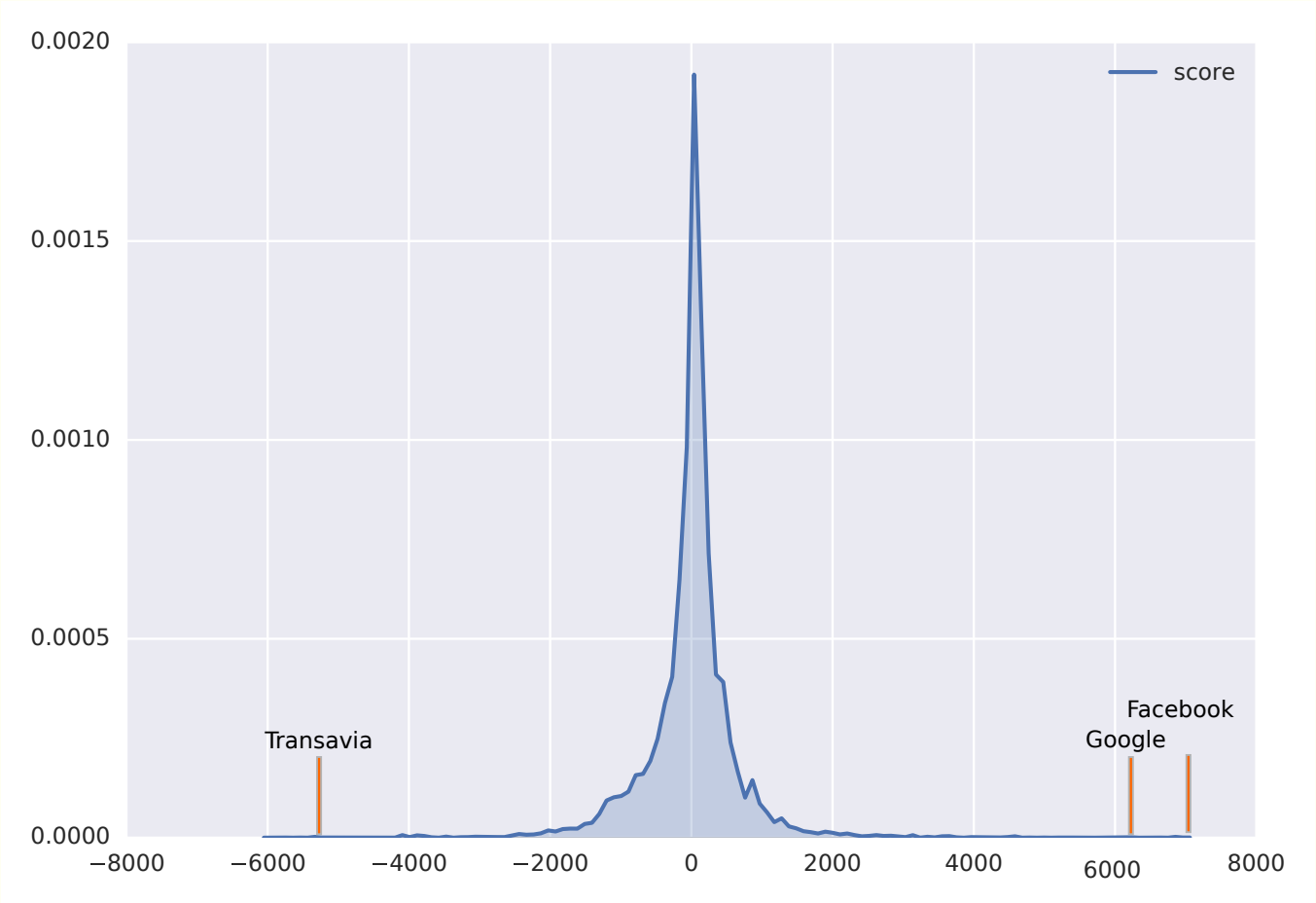
Résultat : utilisation du score



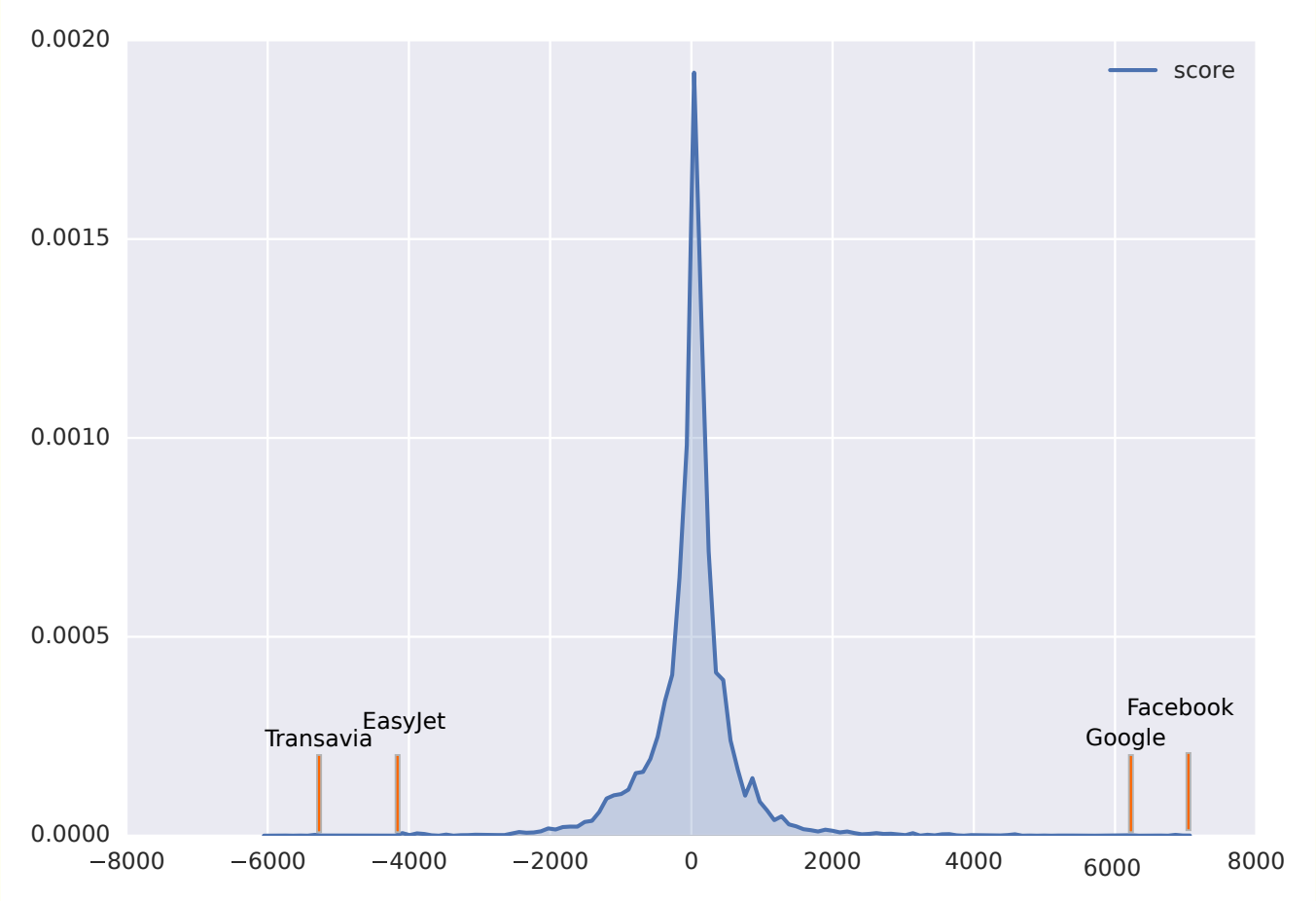
Résultat : utilisation du score



Résultat : utilisation du score



Résultat : utilisation du score



Autres résultats

- Corrélation entre défauts de code et popularité

Autres résultats

- Corrélation entre défauts de code et popularité
- Les classes qui héritent du framework Android ont tendance à contenir plus de défauts

Autres résultats

- Corrélation entre défauts de code et popularité
- Les classes qui héritent du framework Android ont tendance à contenir plus de défauts
- Certains défauts de code apparaissent souvent dans les mêmes classes/méthodes

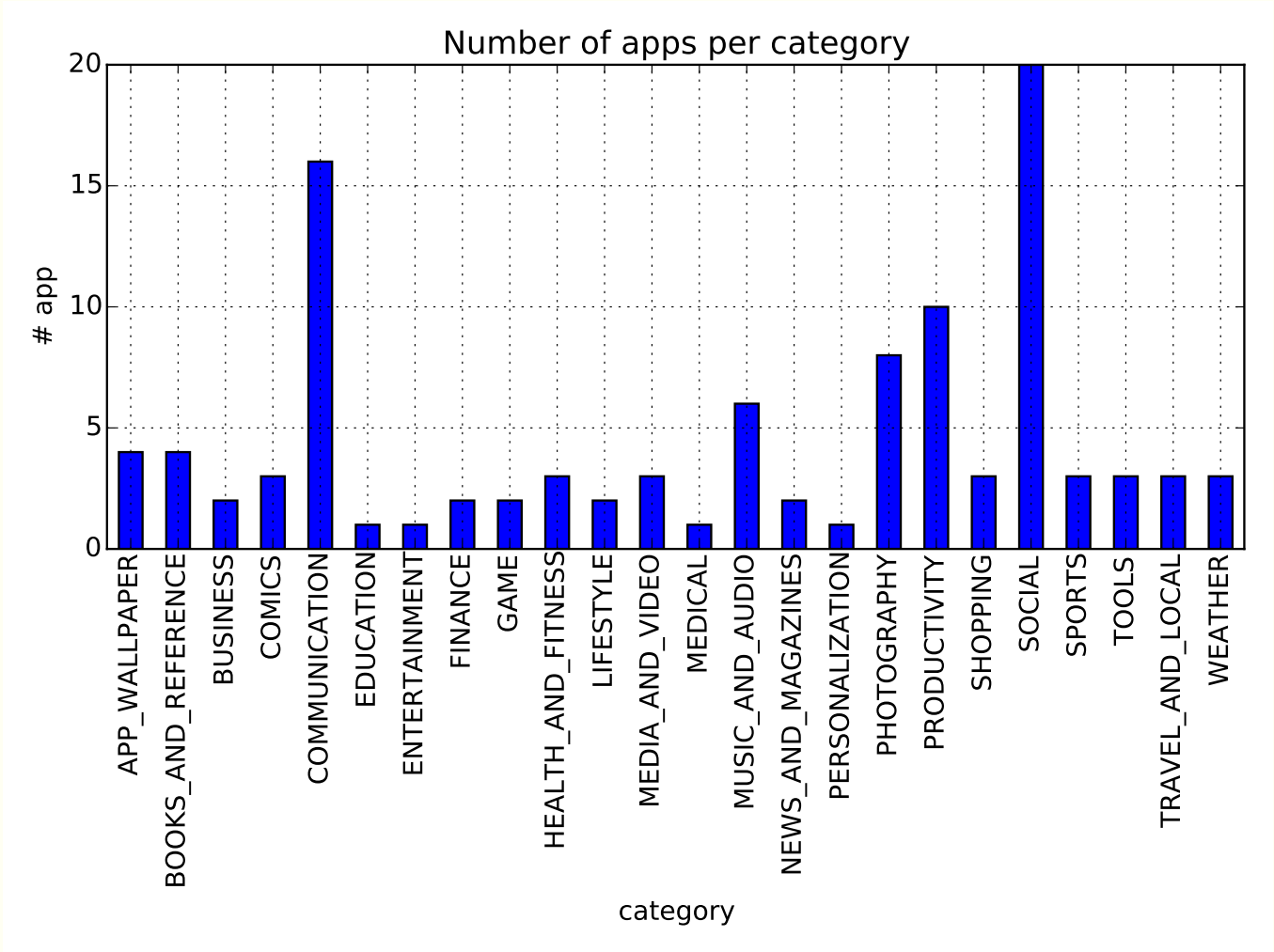
Étude 2 : Évolution des défauts de code dans le temps

- Analyse de **3568 versions** de **106 apps** populaires avec PAPRIKA

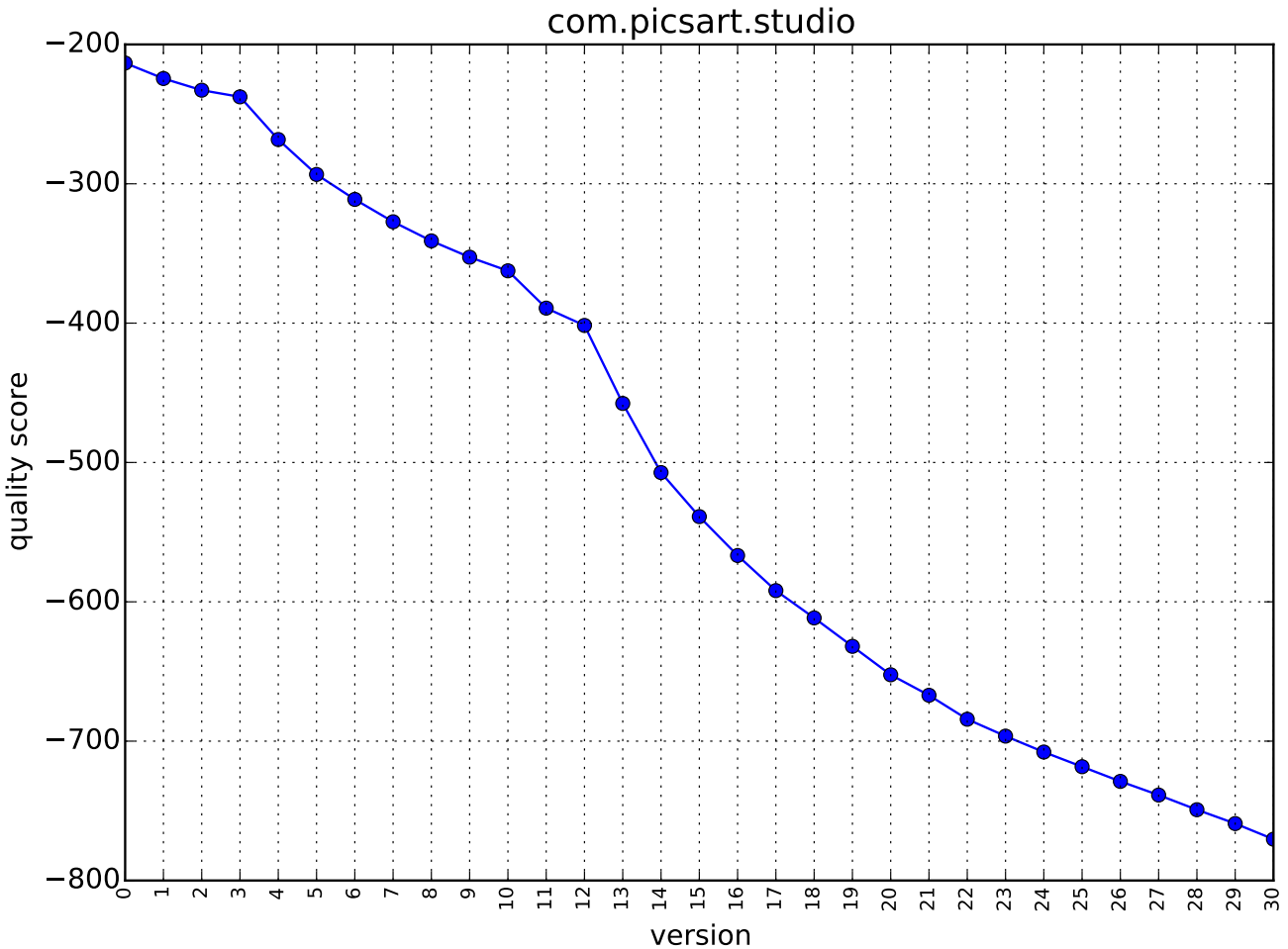
Étude 2 : Évolution des défauts de code dans le temps

- Analyse de **3568 versions** de **106 apps** populaires avec PAPRIKA
- Calcul du score pour chaque défaut de code et suivi dans le temps

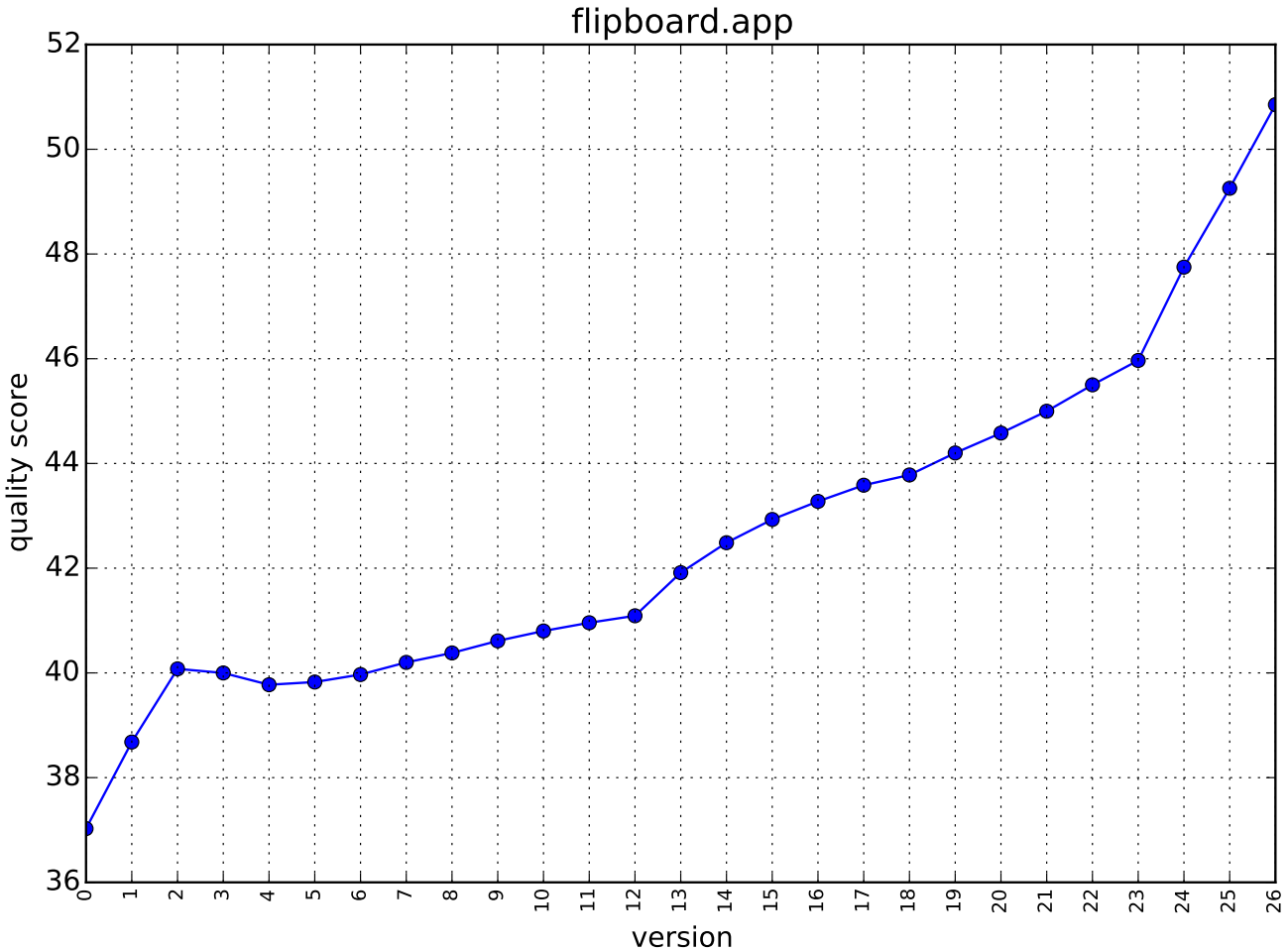
Catégories des apps



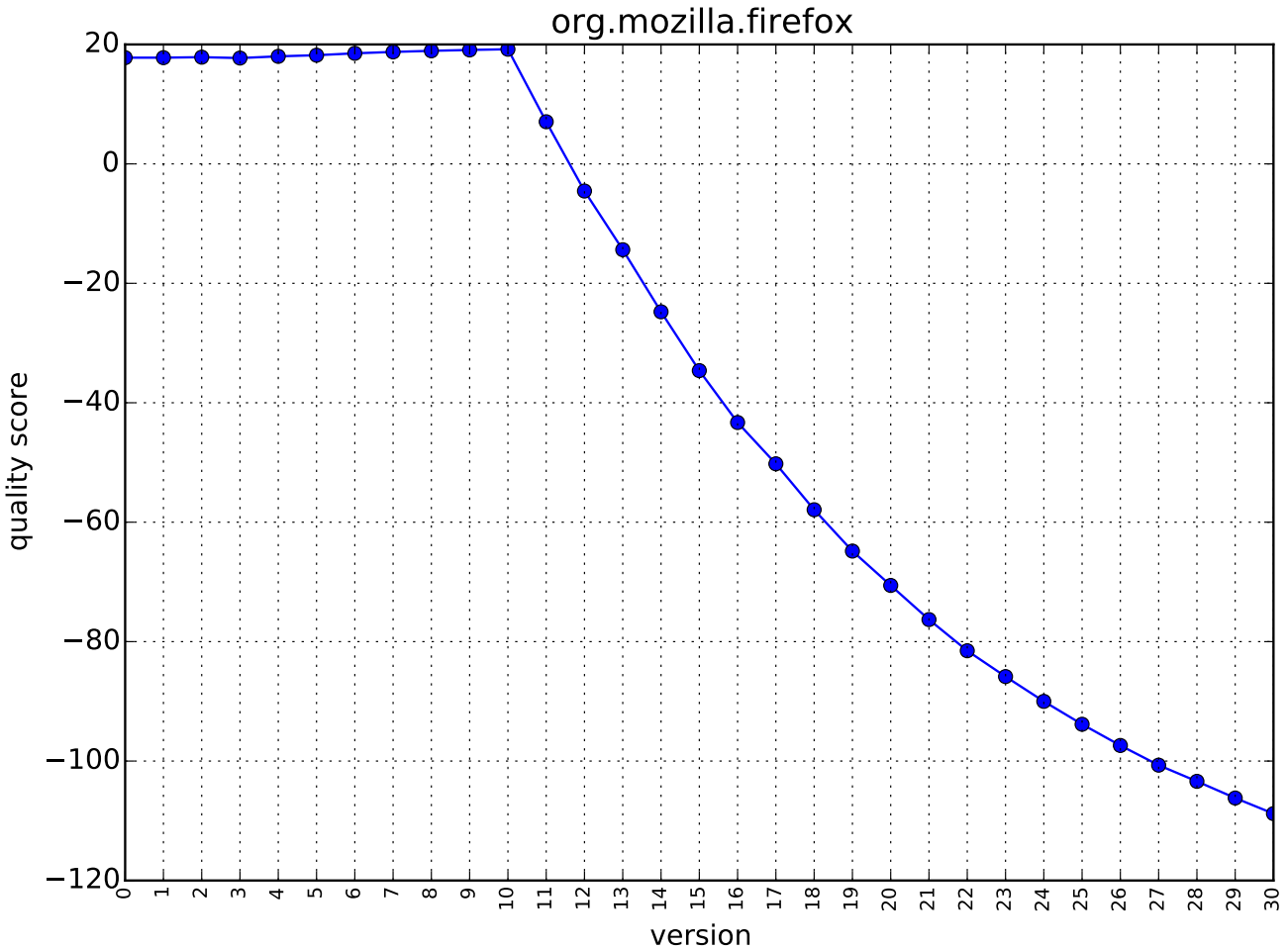
Résultat : baisse constante



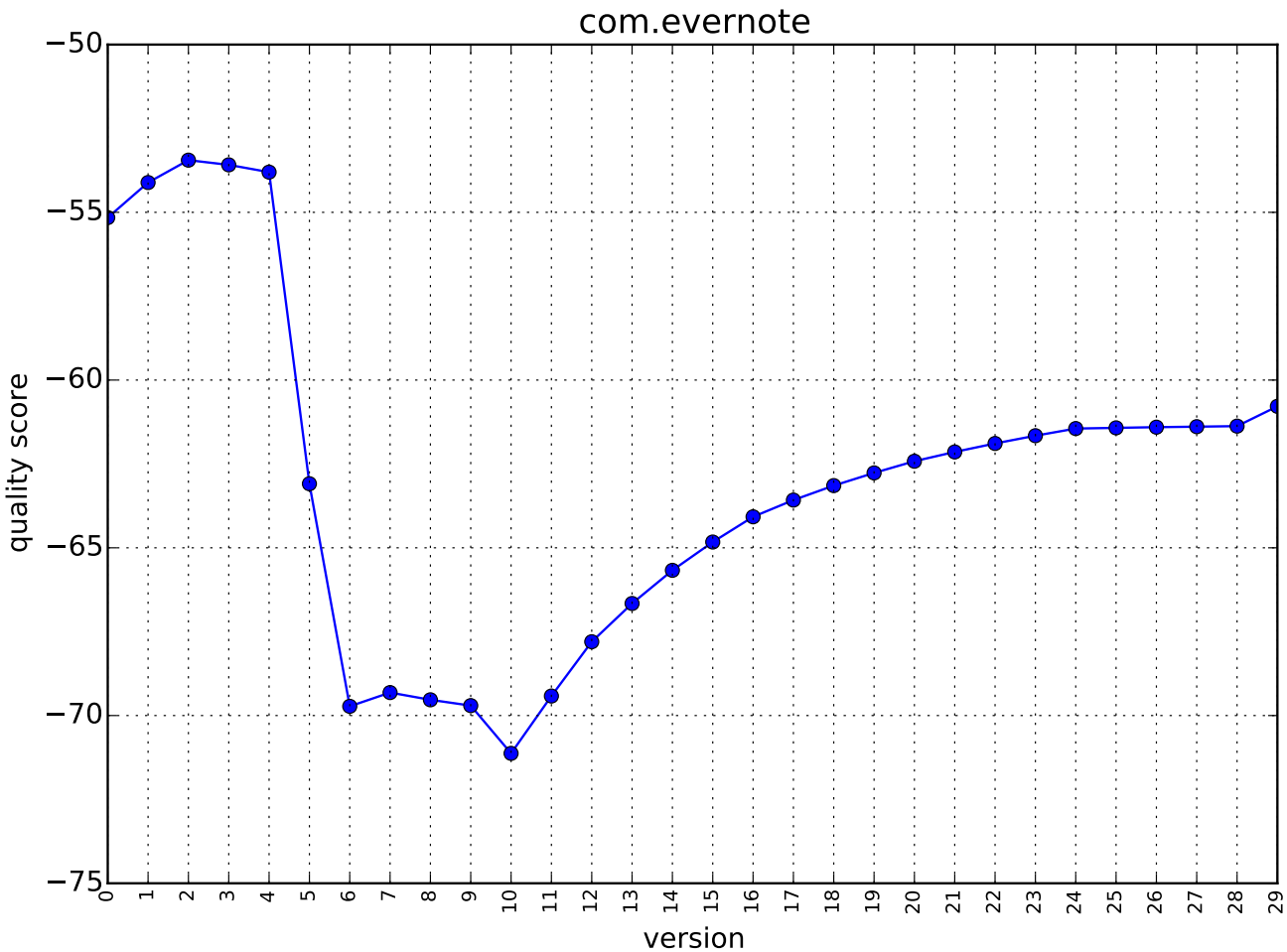
Résultat : augmentation constante



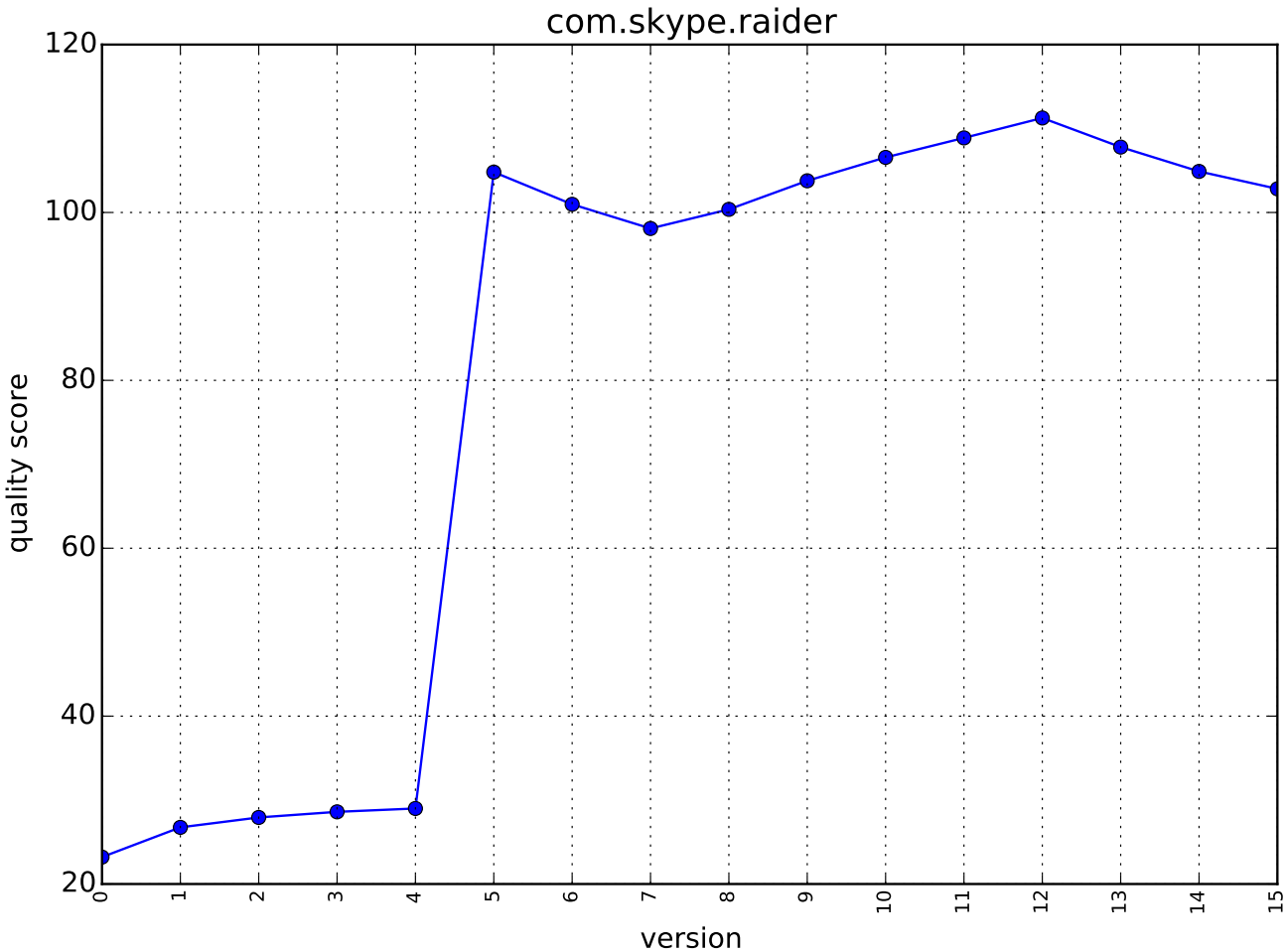
Résultat : stabilité



Résultat : baisse soudaine



Résultat : augmentation soudaine



Question de recherche

- RQ2 : Pouvons-nous observer des tendances dans la distribution des défauts de code dans les apps Android ?
- Oui, il existe une norme pour toutes les apps [10]
- Il existe aussi des tendances dans le temps [11]

---

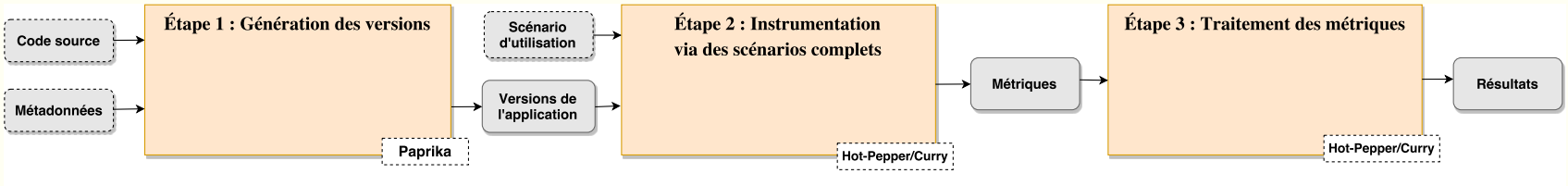
10. Geoffrey Hecht et al. « An Empirical Analysis of Android Code Smells ». In : *En cours de revision pour le journal Transactions on Software Engineering (TSE) - sous réserve d'acceptation*. 2016.

11. Geoffrey Hecht et al. « Tracking the Software Quality of Android Applications along their Evolution ». In : *30th IEEE/ACM International Conference on Automated Software Engineering*. IEEE. 2015, p. 12.

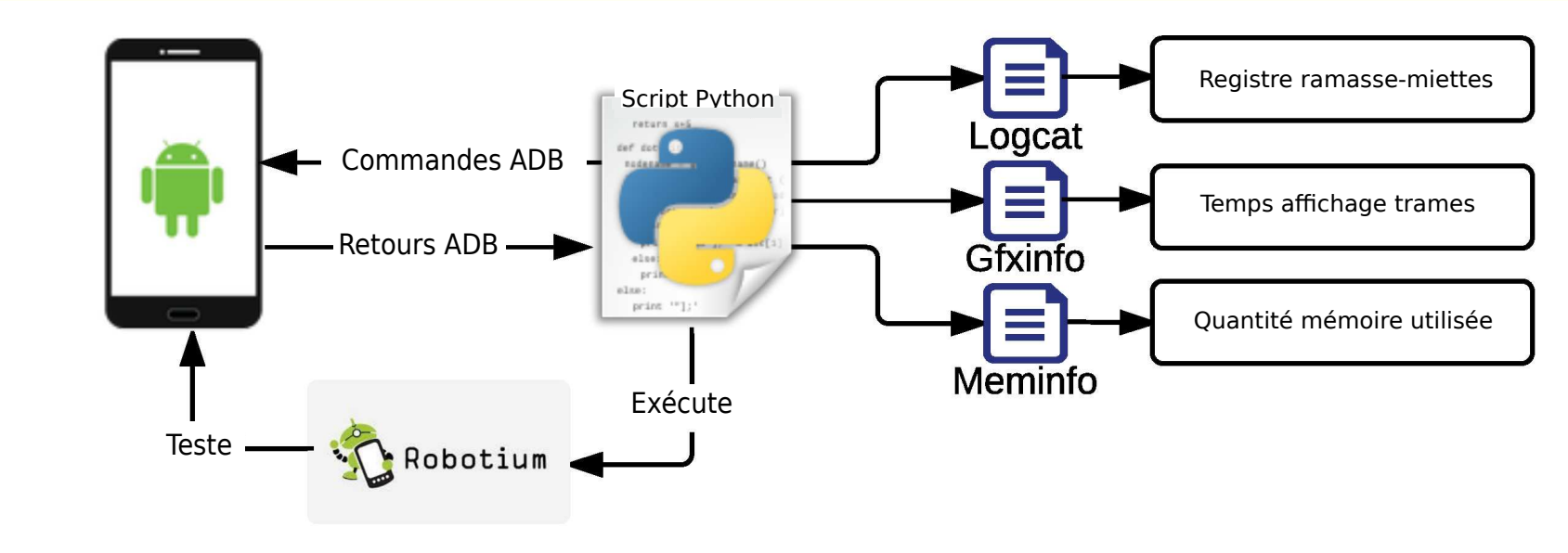
RQ3 : Est-ce que la correction des défauts de code a un impact sur les performances et la consommation d'énergie ?

Contributions

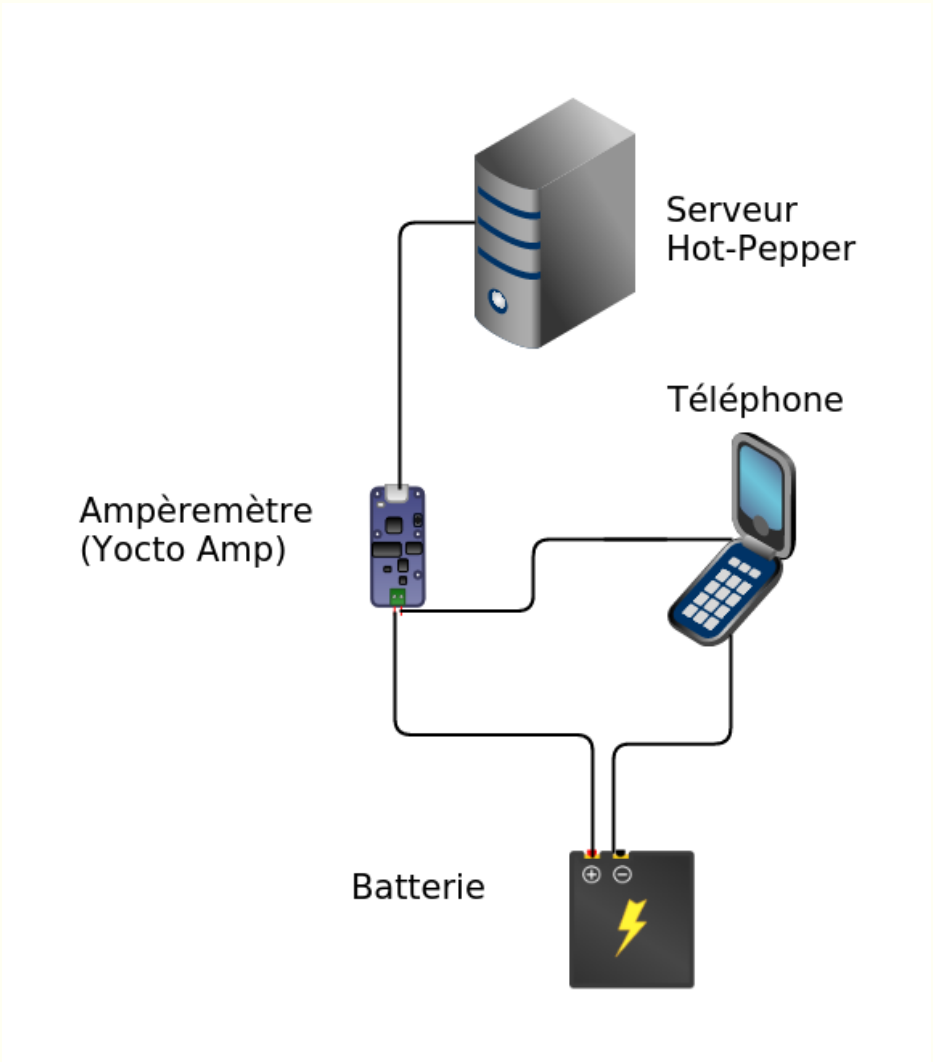
Curry/Hot-Pepper : deux approches similaires



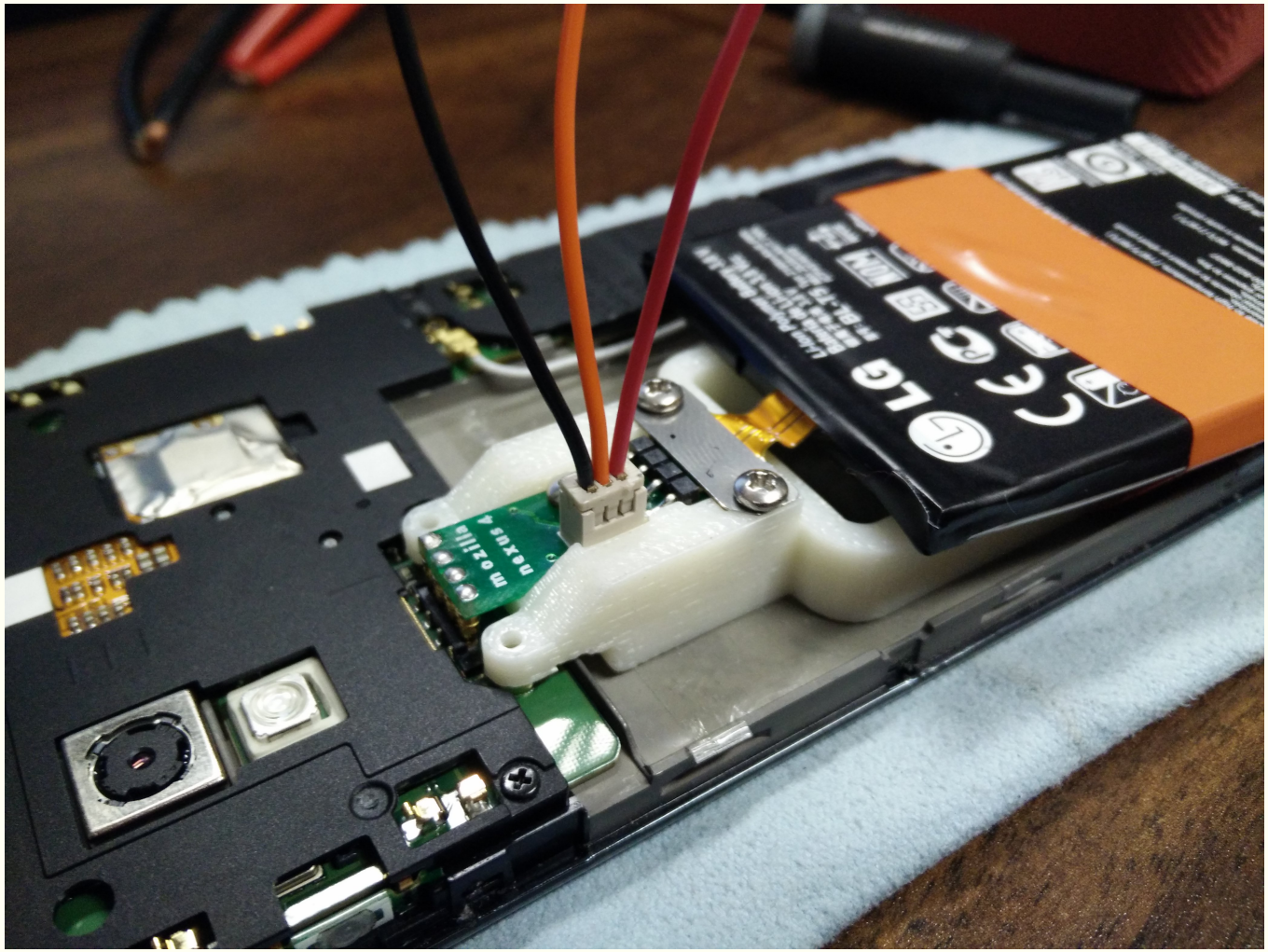
CURRY : impact sur les performances



Hot-Pepper : impact sur la consommation d'énergie



Hot-Pepper : impact sur la consommation d'énergie



Étude 3 : Impact des défauts de code sur la performance

- Trois défauts de code (IGS, MIM, HMU) avec CURRY

Étude 3 : Impact des défauts de code sur la performance

- Trois défauts de code (IGS, MIM, HMU) avec CURRY
- Deux apps libres (Soundwaves Podcast et Terminal Emulator)

Étude 3 : Impact des défauts de code sur la performance

- Trois défauts de code (IGS, MIM, HMU) avec CURRY
- Deux apps libres (Soundwaves Podcast et Terminal Emulator)
- Scénarios complets

Résultat : performance

	IGS	MIM	HMU	Tous
Temps trame	✗	✗	✗	✗
Trames différées	✓	✓	✓	✓
Mémoire	✗	✗	✗	✓
Ramasse-miettes	?	?	✓	✓

Étude 4 : Impact des défauts de code sur la consommation d'énergie

- Trois défauts de code (IGS, MIM, HMU) avec HOT-PEPPER

Étude 4 : Impact des défauts de code sur la consommation d'énergie

- Trois défauts de code (IGS, MIM, HMU) avec HOT-PEPPER
- Cinq apps libres (Soundwaves Podcast, Aizoban, Calculator, Web Opac et ToDo)

Étude 4 : Impact des défauts de code sur la consommation d'énergie

- Trois défauts de code (IGS, MIM, HMU) avec HOT-PEPPER
- Cinq apps libres (Soundwaves Podcast, Aizoban, Calculator, Web Opac et ToDo)
- Scénarios complets

Résultat : consommation d'énergie

	IGS	MIM	HMU	Tous
Consommation	✓	✓	✓	✓

Questions de recherche

- RQ3 : Est-ce que la correction des défauts de code a un impact sur les performances et la consommation d'énergie ?
- Oui pour les performances [12]
- Oui pour la consommation d'énergie [13]

---

12. Geoffrey Hecht, Naouel Moha et Romain Rouvoy. « An Empirical Study of the Performance Impacts of Android Code Smells ». In : *IEEE/ACM International Conference on Mobile Software Engineering and Systems*. T. 1. 1. IEEE. 2016.

13. Antonin Carette et al. « Investigating the Energy Impact of Android Smells ». In : *IEEE 24rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*. 2017.

Conclusion

Conclusion

- RQ1 : Existe-t-il des défauts de code spécifiques à Android pertinents ? **Oui** (Classification)

Conclusion

- RQ1 : Existe-t-il des défauts de code spécifiques à Android pertinents ? **Oui** (Classification)
- RQ2 : Pouvons-nous observer des tendances dans la distribution des défauts de code dans les apps Android ? **Oui, entre apps et dans le temps** (PAPRIKA)

Conclusion

- RQ1 : Existe-t-il des défauts de code spécifiques à Android pertinents ? **Oui** (Classification)
- RQ2 : Pouvons-nous observer des tendances dans la distribution des défauts de code dans les apps Android ? **Oui, entre apps et dans le temps** (PAPRIKA)
- RQ3 : Est-ce que la correction des défauts de code a un impact sur les performances et la consommation d'énergie ? **Oui** (CURRY et HOT-PEPPER)

Perspectives

Court terme

- Améliorer classification

Court terme

- Améliorer classification
- **Améliorer PAPRIKA**

Court terme

- Améliorer classification
- **Améliorer PAPRIKA**
- Impact des défauts de code avec ART

Court terme

- Améliorer classification
- **Améliorer PAPRIKA**
- Impact des défauts de code avec ART
- **Détection/correction autres OS (premier pas iOS)**

Long terme

- **Correction plus complexe**

Long terme

- **Correction plus complexe**
- Exploiter les bases de données

Long terme

- **Correction plus complexe**
- Exploiter les bases de données
- Analyse dynamique

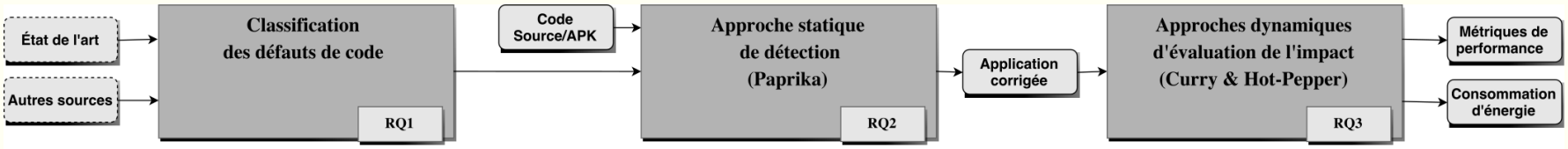
Long terme

- **Correction plus complexe**
- Exploiter les bases de données
- Analyse dynamique
- Anti-patrons d’interaction

Long terme

- **Correction plus complexe**
- Exploiter les bases de données
- Analyse dynamique
- Anti-patrons d’interaction
- **Collaboration avec des professionnels**

Questions



- Geoffrey Hecht et al. « An Empirical Analysis of Android Code Smells ». In : *En cours de revision pour le journal Transactions on Software Engineering (TSE) - sous réserve d'acceptation*. 2016
- Geoffrey Hecht et al. « Tracking the Software Quality of Android Applications along their Evolution ». In : *30th IEEE/ACM International Conference on Automated Software Engineering*. IEEE. 2015, p. 12
- Geoffrey Hecht, Naouel Moha et Romain Rouvoy. « An Empirical Study of the Performance Impacts of Android Code Smells ». In : *IEEE/ACM International Conference on Mobile Software Engineering and Systems*. T. 1. 1. IEEE. 2016
- Antonin Carette et al. « Investigating the Energy Impact of Android Smells ». In : *IEEE 24rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*. 2017