Le génie pour l'industrie

Étude de la qualité des systèmes IoT

Nour Khezemi

Directrice: Dr. Naouel Moha
Co-Directeur Dr.Yann-Gaël Guéhéneuc

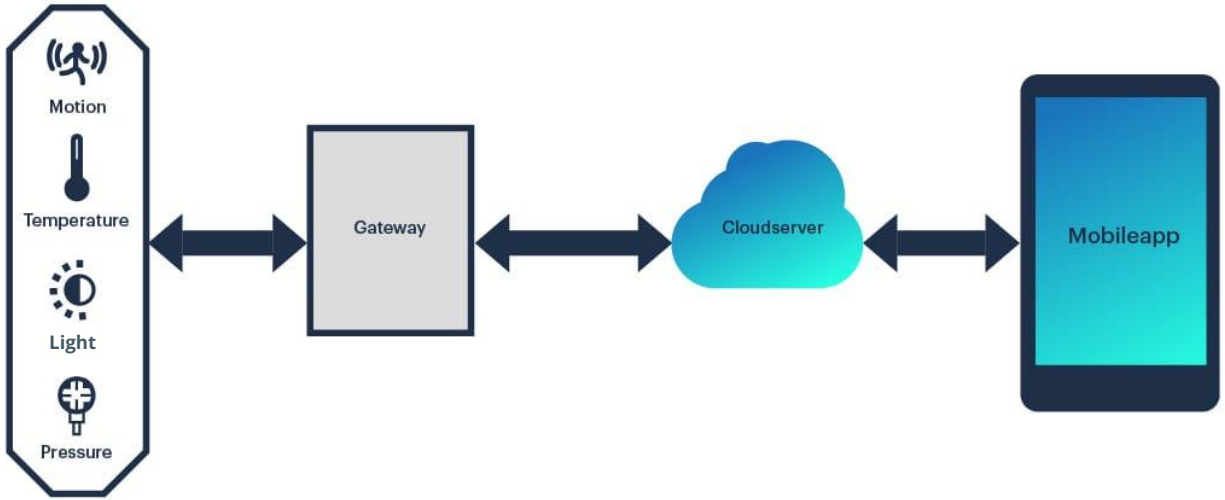
UNIVERSITÉ Concordia UNIVERSITY

PLAN

1. Introduction

Contexte (1/3)

Un système IoT est composé **d'appareils en réseau** dont le but est de **collecter** et de **transmettre** les données collectées pour **fournir des services utiles aux utilisateurs finaux** [1].



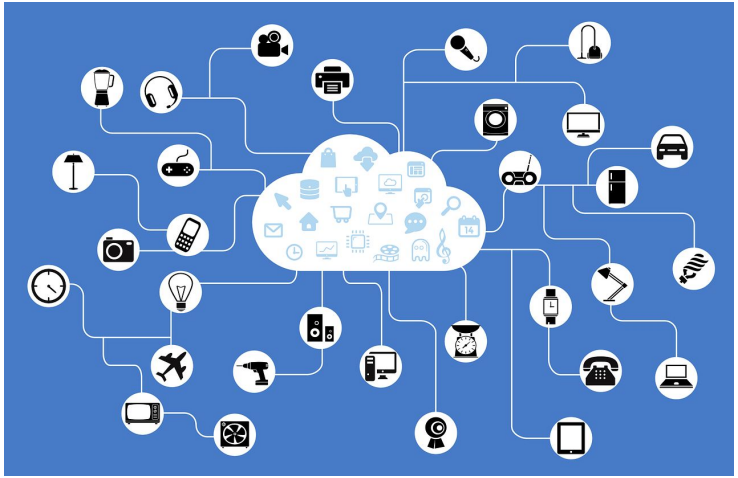
Source: https://www.baianat.com/articles/how-iot-is-transforming-industries

[1] Xabier et al. ; Software engineering for the internet of things ; IEEE Software, 2017

Contexte (2/3)

Faits IoT [2]:

- Plus de **19,8 milliards** d'appareils IoT connectés dans le monde
- Cisco prévoit **500 milliards d'appareils IoT** d'ici 2030
- D'ici 2030, **75 %** de tous les appareils devraient être des appareils IoT
- Le marché de l'IoT vaut actuellement plus de **1,1 milliard de dollars**



Source: https://gladiacteur.com/objets-connectes-iot/

[2] https://explodingtopics.com/blog/iot-stats

Contexte (3/3)

- IoT améliore la vie des individus et automatise les processus pour réduire les coûts.
- La qualité est essentielle pour un système IoT performant.

Exemple: Un système IoT de suivi des véhicules

Source: https://fr.digi.com/blog/post/what-is-connected-vehicle-technology-and-use-cases

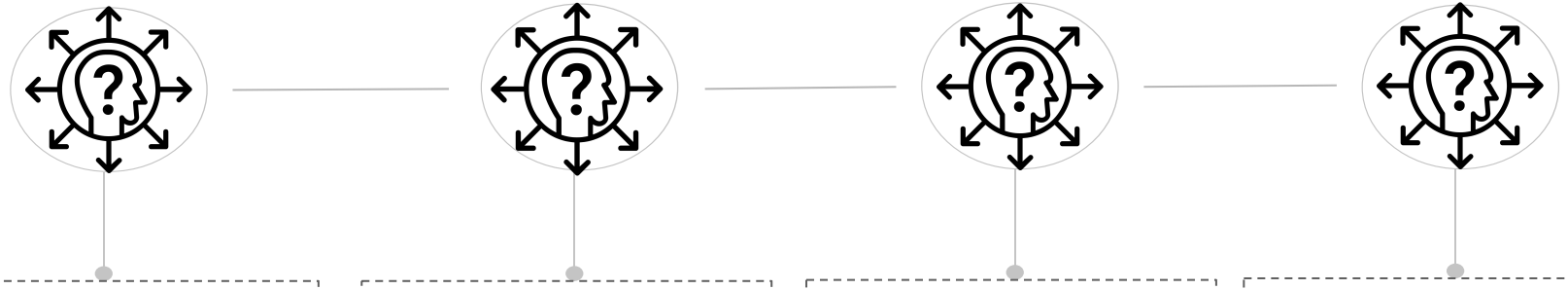
Problème: Le **choix de l'architecture** système IoT.

Conséquences: Un attaquant pourrait **injecter un code malveillant** dans les dispositifs IoT, risquant ainsi le fonctionnement normal des véhicules.

Problématique (1/2)

Les systèmes IoT font face à des défis de qualité liés à leurs aspects logiciels. Parmi lesquels, les choix architecturaux et la qualité du code développé.

Problématique (2/2)



1- Grand volume de recherche sur l'architecture IoT

2- Besoin de répertorier toutes les exigences de qualité et les architectures les plus utilisées dans les systèmes IoT

3- Peu d'études et analyse de la qualité du code des systèmes IoT

4- Peu des travaux étudiant des bonnes pratiques pour le développement de systèmes IoT de qualité

Contribution 1

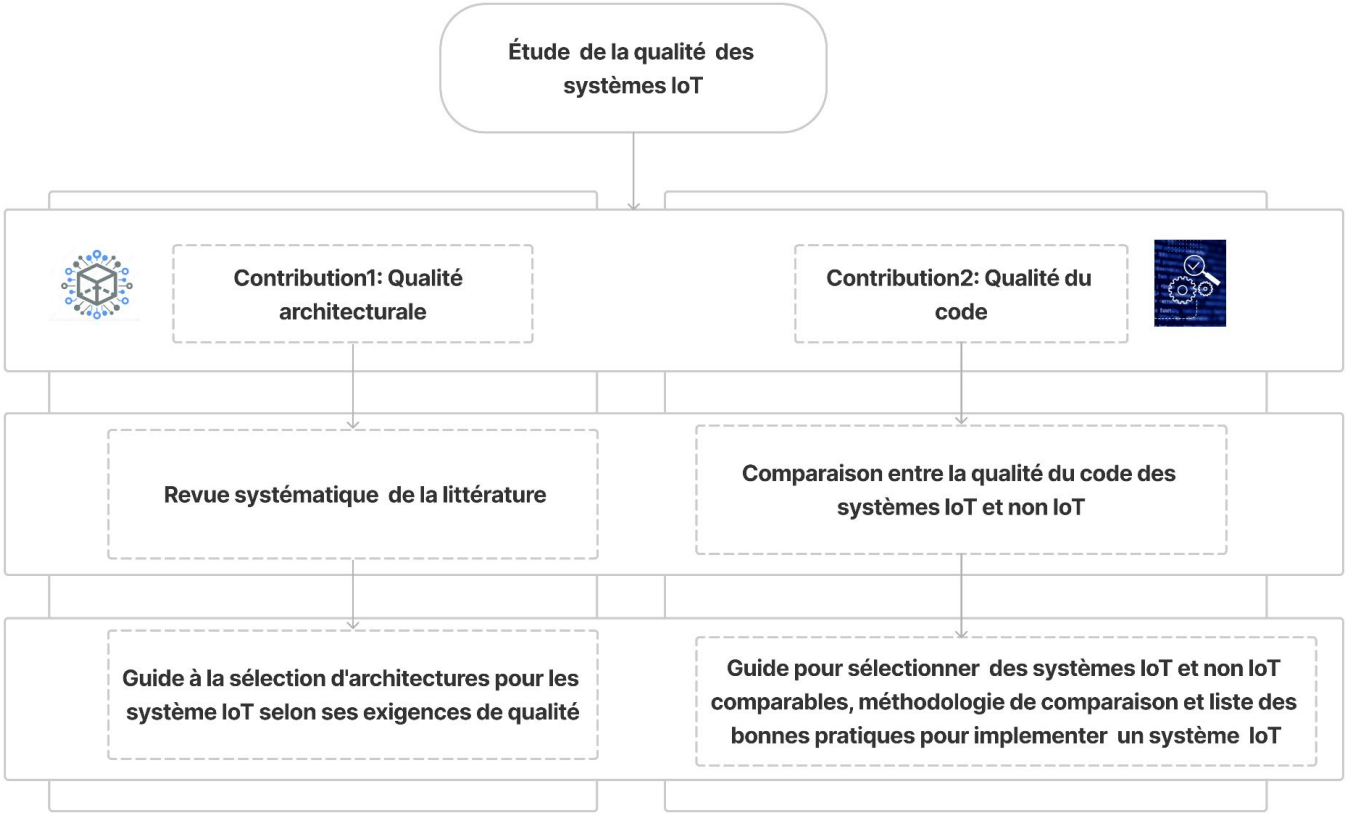
Contribution 2

Publications (1/1)

- Khezemi, Minani, Guéhéneuc, Sabir, Moha, El Boussaidi, **A Systematic Literature Review of IoT System Architectural Styles and their Quality Requirements**, soumis dans IEEE IoT Journal (IoT-J), le 23 février 2024.
- Khezemi, Guéhéneuc, Moha, **Comparison of Code Quality and Best Practices in IoT and non-IoT Software**, soumis au Elsevier journal of Information and Software Technology (IST), le 23 février 2024.

2. MÉTHODOLOGIE

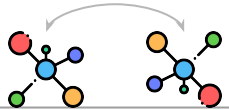
Méthodologie (1/1)



3. Contribution 1: SLR (Qualité architecturale)

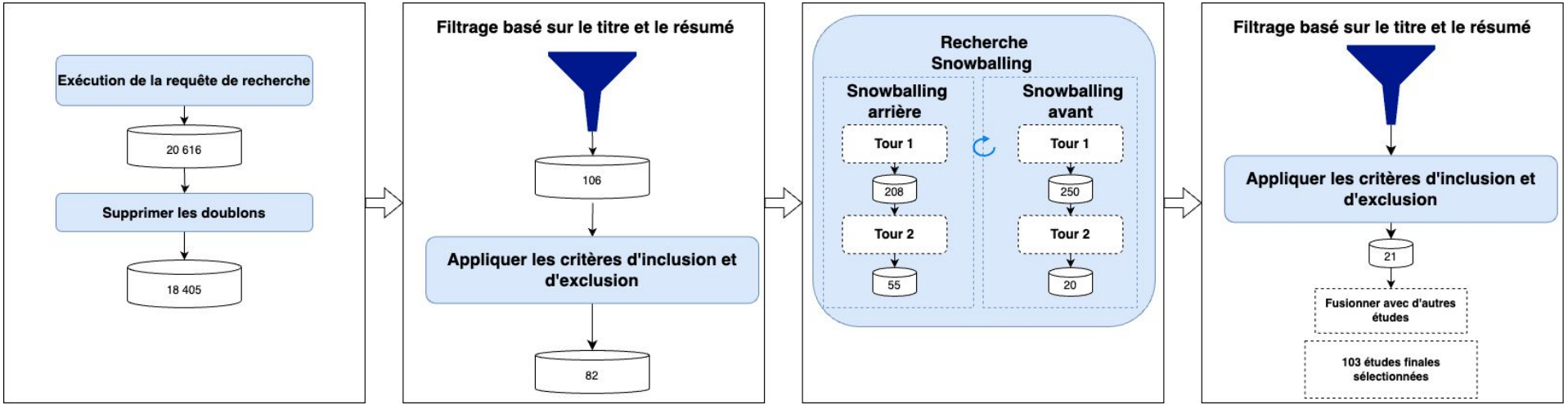
Contribution 1: Étude des styles architecturaux des systèmes IoT et de leurs exigences en matière de qualité

Questions de recherche

 **Q1**	**Quelles sont les exigences de qualité pour les systèmes IoT ?**
 **Q2**	**Quels sont les styles architecturaux des systèmes IoT et leurs avantages et inconvénients ?**
 **Q3**	**Les styles architecturaux identifiés répondent-ils aux exigences de qualité trouvées ?**

Méthodologie (1/1)

Processus PRISMA, Stratégie PICO



Résultat (1/8)

Q1: Quelles sont les exigences de qualité pour les systèmes IoT ?

Performance

Discutée dans **11 PSs.**
La **capacité des systèmes IoT à traiter de grandes quantités de données,** à les traiter rapidement et à fournir des réponses.

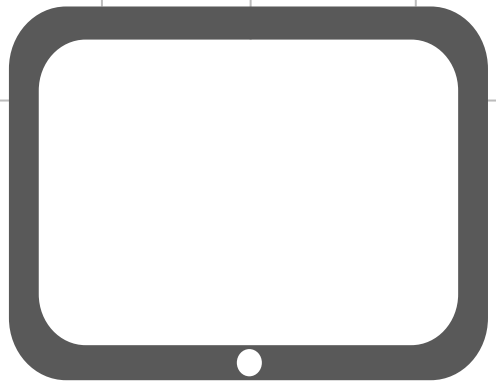
2 Scalabilité

Discutée dans **14 PSs.**
La capacité du système à utiliser, **ajouter ou réduire des ressources** pour traiter les demandes variables de manière optimale.

4 Disponibilité

Discutée dans **10 PSs.**
Garantit que le système IoT **peut accéder aux données et aux services** quel que soit l'état de l'infrastructure.

3



1 Sécurité

Discutée dans **20 PSs.**
Les menaces de sécurité pour les systèmes IoT augmentent davantage de **dispositifs sont connectés aux réseaux**.

5 Interopérabilité

Discutée dans **8 PSs.**
La **capacité** des différents dispositifs et systèmes IoT **à travailler ensemble**.

Résultat (2/8)

Q1: Quelles sont les exigences de qualité pour les systèmes IoT ?

Confidentialité

7

Discutée dans **8 PSs**.
Porte sur la **régulation de l'accès aux données** par les utilisateurs.

Gestion de l'énergie

Discutée dans **6 PSs.**
Implique que **les dispositifs IoT fonctionnent de manière cohérente pendant de longues périodes** sans nécessiter un remplacement fréquent des piles.

Flexibilité

9

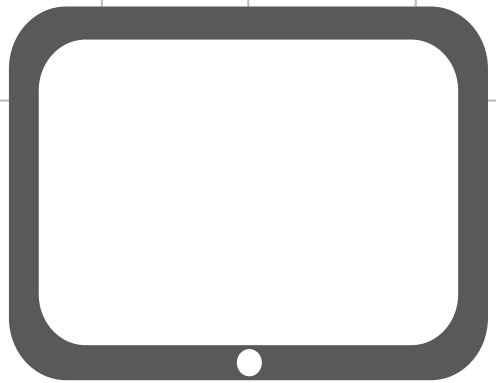
Discutée dans **4 PSs.**
Elle fait référence à **la capacité de s'adapter ou de s'ajuster aux changements** de circonstances ou d'exigences.

8

6

Fiabilité

Discutée dans **8 PSs.**
La capacité du système à **fonctionner de manière constante** et à fournir de manière précise et exacte les fonctionnalités attendues par les utilisateurs finaux.



10

Évolutivité

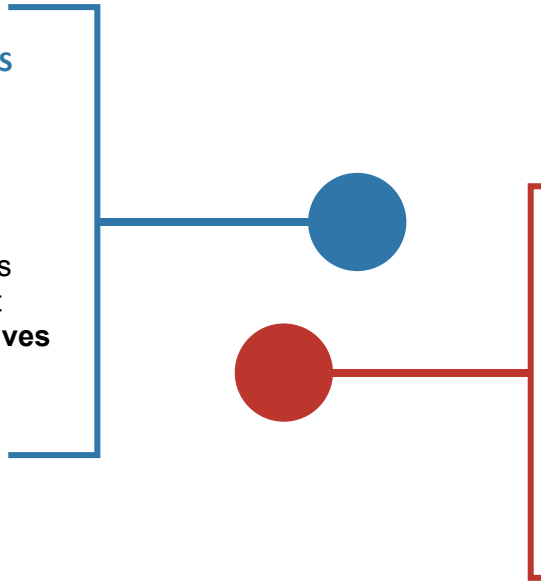
Discutée dans **3 PSs.**
La capacité d'un système à **évoluer et à s'adapter avec le temps**.

Résultat (3/8)

Q1: Quelles sont les exigences de qualité pour les systèmes IoT ?

RECOMMANDATION POUR LES CHERCHEURS

- Étudier **la maintenabilité, la compatibilité et la portabilité**.
- Explorer les exigences de qualité jugées importantes en indiquant celles qui sont **nécessaires** et celles qui sont **facultatives** pour tout système IoT.



RECOMMANDATION POUR LES PRATICIENS

- Prioriser les exigences de qualité en fonction de **cas d'utilisation du système IoT**.
- Analyser les besoins spécifiques de leur système et définir quelles exigences doivent être prises en compte.

Résultat (4/8)

Q2: Quelles sont les styles architecturaux des systèmes IoT et leurs avantages et inconvénients ?

3 Architecture microservices

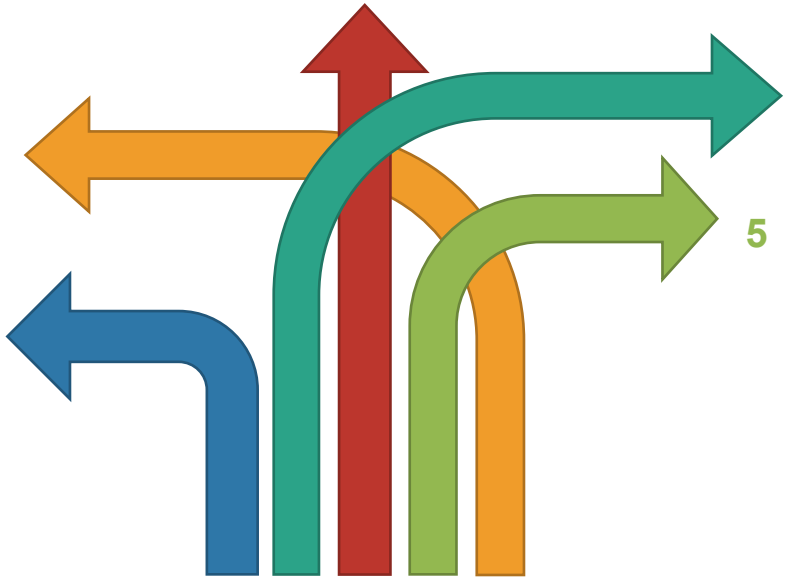
15 PSs.
Organise une application en un groupe de services faiblement couplés qui communiquent à l'aide de protocoles légers.

2 Architecture en couches

21 PSs.
Permet aux développeurs d'organiser les composants ayant des fonctionnalités similaires en couches horizontales.

1 SOA

22 PSs.
Une architecture destinée à la construction de logiciels d'entreprise composée des services.



4 Architecture basée sur le Cloud

6 PSs.
Utilise le cloud computing comme partie centrale de son architecture informatique.

5 Publish Subscribe

5 PSs.
Ce patron architectural composé principalement de deux composants, à savoir les Éditeurs (Publishers) et les Abonnés (Subscribers).

Résultat (5/8)

Q2: Quelles sont les styles architecturaux des systèmes IoT et leurs avantages et inconvénients ?

8 Tubes et Filtres

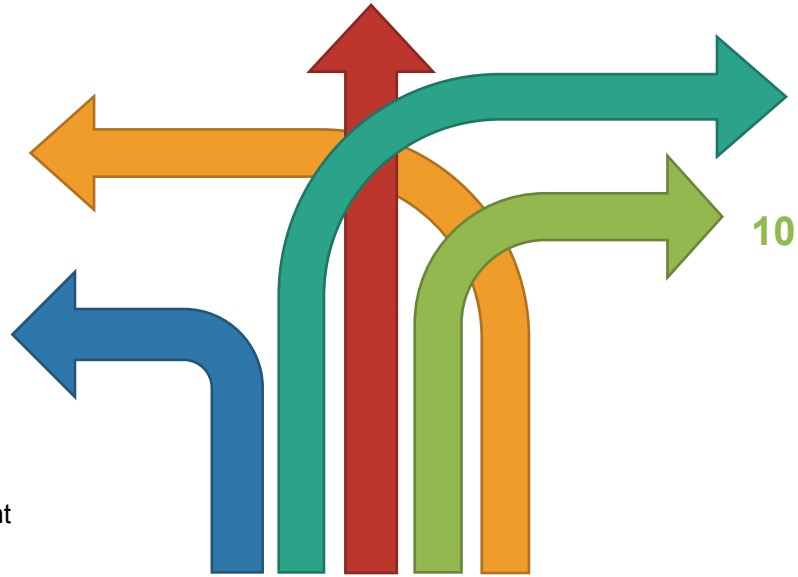
4 PSs.
Une série de composants de traitement, appelés filtres, sont reliés par des canaux appelés tubes pour effectuer des tâches de manière linéaire et séquentielle.

7 Pair à Pair

4 PSs.
Une alternative à l'architecture Client Serveur dans laquelle chaque système a une priorité égale et est soit un serveur, soit un client.

6 Client Serveur

5 PSs.
Une structure d'application distribuée qui répartit les tâches entre les serveurs, qui fournissent des ressources ou des services, et les clients, qui les demandent.



9 Architecture orientée événements

4 PSs.
Met l'accent sur la production, la détection, la consommation et la réaction aux événements.

10 Architecture REST

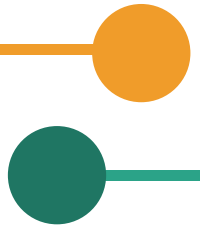
2 PSs.
C'est un style de construction de services Web qui a été appliqué à la conception de systèmes IoT.

Résultat (6/8)

Q2: Quelles sont les styles architecturaux des systèmes IoT et leurs avantages et inconvénients ?

RECOMMANDATION POUR LES CHERCHEURS

- **Data-Centric et Multi-tenancy** ne sont pas mentionnés dans nos PSs.
- **Comparaison empirique** entre les différents styles architecturaux pour évaluer leur influences aux exigences de qualité.



RECOMMANDATION POUR LES PRATICIENS

- Explorer l'architecture microservices car elle prend en charge des **services distribués et interopérables**.
- Se tenir informé **des autres styles architecturaux non explorés** dans la littérature existante.

Résultat (7/8)

Q3 : Les styles architecturaux identifiés dans la Q2 répondent-ils aux exigences de qualité énoncées dans la Q1 ?

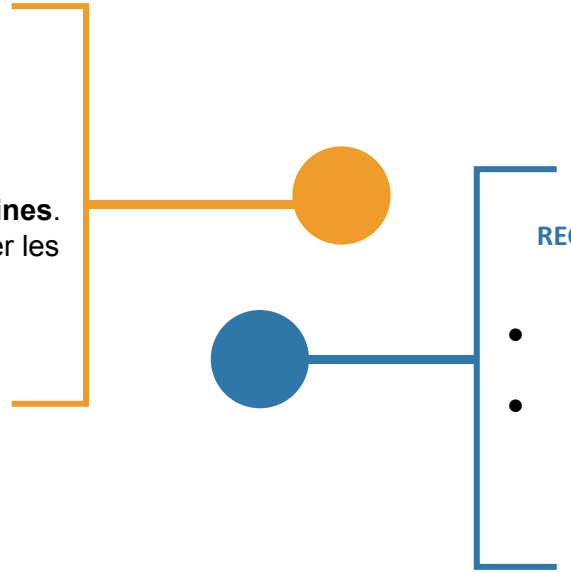
	Scalabilité	*Sécurité*	*Disponibilité*	*Interopérabilité*	*Performance*	*Confidentialité*	*Fiabilité*	*Gestion de l'énergie*	*Flexibilité*	*Évolutivité*	**Total**
Architecture REST	✓	×	×	✓	✓	×	✓	✓	✓	**NA**	6
SOA	✓	×	✓	×	×	×	✓	×	✓	✓	5
Pair à Pair	✓	×	✓	×	×	×	✓	✓	✓	**NA**	5
Architecture basée sur le Cloud	×	×	×	✓	✓	×	✓	×	✓	✓	5
Architecture orientée événements	×	✓	×	×	✓	×	×	✓	×	**NA**	4
Tubes et Filtres	✓	**NA**	**NA**	✓	×	**NA**	**NA**	**NA**	×	✓	3
Publish Subscribe	✓	✓	×	×	×	✓	✓	×	×	**NA**	3
Microservice	✓	×	×	✓	×	×	×	×	✓	**NA**	3
L'architecture en couches	×	✓	×	✓	×	×	**NA**	×	×	**NA**	2
Client Serveur	✓	**NA**	×	×	×	×	✓	×	×	**NA**	2
Total	7	3	2	5	3	1	6	3	5	3	

Résultat (8/8)

Q3 : Les styles architecturaux identifiés dans la QR2 répondent-ils aux exigences de qualité énoncées dans la QR1 ?

RECOMMANDATION POUR LES CHERCHEURS

- Le développement d'une matrice décrivant les **exigences de qualité dans différents domaines**.
- Guide détaillé pourrait être produit pour faciliter les décisions éclairées lors de la sélection de l'architecture la plus appropriée.



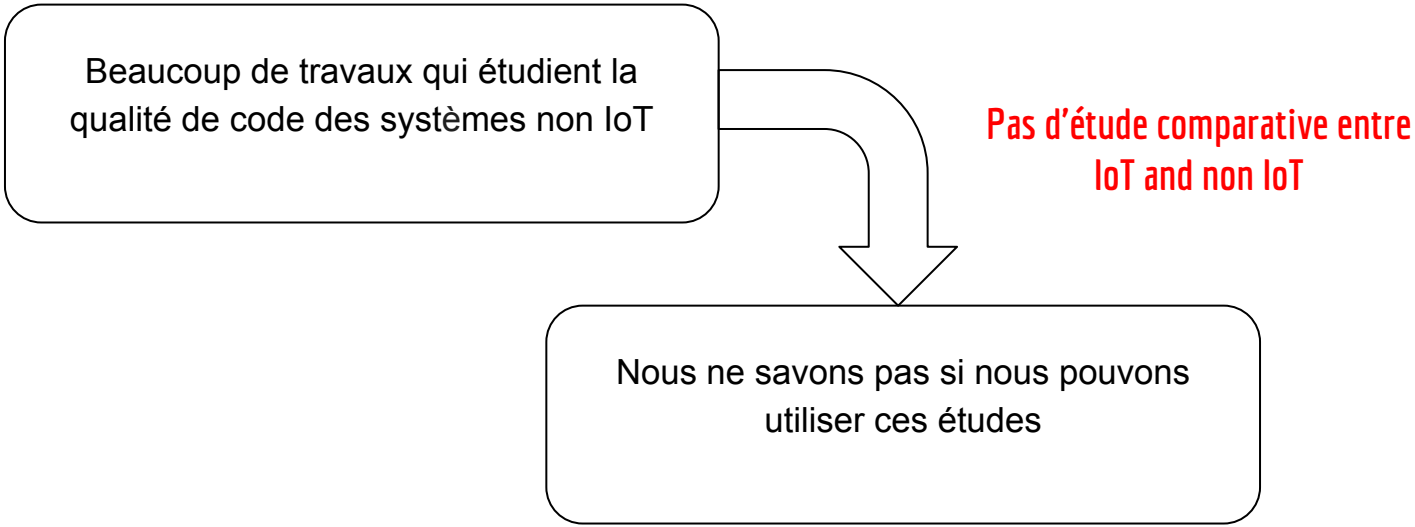
RECOMMANDATION POUR LES PRATICIENS

- Les exigences de confidentialité ont été négligées dans différents styles architecturaux.
- La correspondance peut servir de référence aux praticiens pour choisir la bonne architecture en fonction des exigences de qualité.

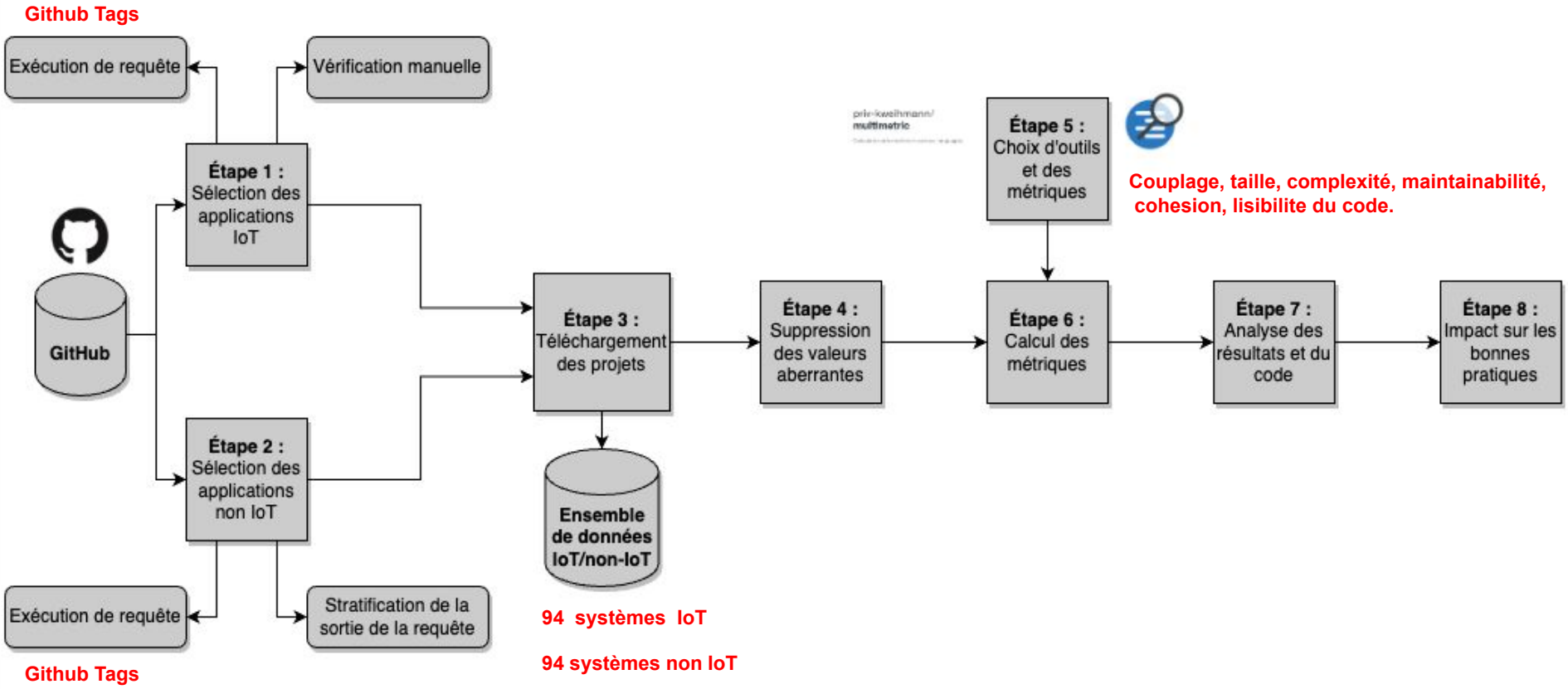
4. Contribution 2: Comparison du code IoT et non IoT (Qualité du code)

Contribution 2: Comparison du code IoT et non IoT

Pourquoi nous avons besoin d'une comparaison?



Méthodologie (1/2)



Résultats (1/5)

Calcul statistique des métriques

			IoT	non IoT
Couplage	**CBO**	**Médiane**	**188**	40
		Moyenne	4252,41	715,72
	RFC	**Médiane**	**505**	98
		Moyenne	22334,82	2956,8
Taille	**LOC**	**Médiane**	**4574**	3713
		Moyenne	62960,13	31374,58
	#Classes	**Médiane**	**116,5**	33
		Moyenne	975,89	373,76
	#Fichiers	**Médiane**	**115**	47,5
		Moyenne	984	356,24
Complexité	**WMC**	**Médiane**	**217**	22
		Moyenne	4550,46	1667,51
	CC	**Médiane**	**462032,02**	465
		Moyenne	26953,31	2358,08
	HV	**Médiane**	2016	**283436,35**
		Moyenne	7283,76	5334,77
Maintainabilité	**MI**	**Médiane**	3645,21	**4666,48**
		Moyenne	24854,31	36403,86
Cohesion	**LCOM**	**Médiane**	**0,72**	0,75
		Moyenne	0,70	0,74
Lisibilité du code	**CP**	**Médiane**	**44,87**	17,91
		Moyenne	2200,13	168,49

Résultats (2/5)

Interprétation du calcul statistique des métriques

Les systèmes IoT présentent un code plus vaste (**LOC, #Classes, #Fichiers**).

Les systèmes IoT sont plus complexes que les systèmes non IoT (**CC, RFC, WMC**).

Les systèmes IoT présentent une plus grande interdépendance que les systèmes non IoT (**CBO, LCOM**).

Les systèmes IoT sont plus difficiles à entretenir et à modifier (**MI, HV**).

Résultats (3/5)

Analyse approfondie du code

Langage	IoT	non IoT
Java	eclipse-ditto/ditto	kymjs/CJFrameForAndroid
JavaScript	rwaldron/johnny-five	uuidjs/uuid
C	timmbogner/Farm-Data-Relay-System	unbit/spockfs
C++	project-chip/connectedhomeip	zeek/zeek
C#	renode/renode	madskristensen/MiniBlog
Python	DT42/BerryNet	JohnHammond/msdt-follina

⇨ **Confirmation de nos observations générales.**

Résultats (4/5)

Analyse approfondie du code

Comment cette complexité se manifeste dans la base de code?

Code Java pour la transformation Msg dans un système IoT

```
thingTemplate = configuration.findProperty
(THING_TEMPLATE).orElseThrow(()->
 ↪  MessageMapperConfigurationInvalidException.
 ↪  newBuilder(THING_TEMPLATE).build());
commandHeaders=configuration.findProperty(
 ↪  COMMAND_HEADERS, JsonValue::isObject,JsonValue::
 ↪  asObject).filter(configuredHeaders->!
 ↪  configuredHeaders.isEmpty()).map(configuredHeaders->
 ↪  { /* ... */ }).orElseGet(()->DittoHeaders.newBuilder
 ↪  () /* ... */);
```

Résultats (5/5)

Analyse approfondie du code

Comment cette complexité se manifeste dans la base de code?

Code JavaScript pour les interactions entre les composants matériels dans le système IoT

```
var scanner = new five.Servo({ pin: 12, range: [0, 170] });

var ping = new five.Ping(7);
```

Implications Pratiques pour le Développement IoT (1/3)

Défis du monde réel et observations 1

- IoT implique des **interactions complexes** entre le matériel et le logiciel, et les communications de données complexes.
- Les métriques et l'analyse du code mettent en évidence la complexité du code des systèmes IoT.

Implications 1

- Les développeurs devraient cultiver des compétences spécialisées :
  - Une compréhension approfondie des aspects matériels et logiciels
  - Une expertise dans des protocoles de communication

Implications Pratiques pour le Développement IoT (2/3)

Défis du monde réel et observations 2

- Une interdépendance entre différents modules et une faible maintenabilité dans les systèmes IoT posent des défis pour apporter des modifications et maintenir la base de code.
- LCOM, CBO, HV, et MI.

Implications 2

- Favoriser la conception modulaire et l'organisation efficace du code

Implications Pratiques pour le Développement IoT (3/3)

Défis du monde réel et observations 3

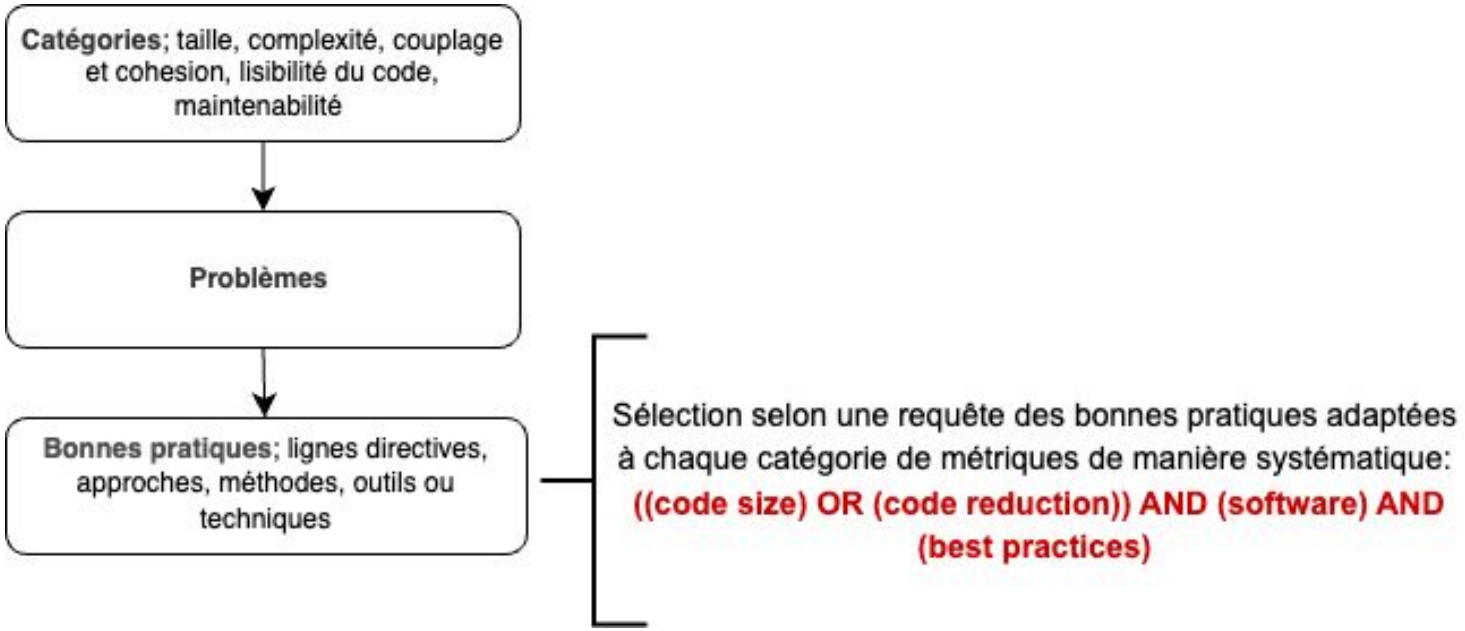
- Comprendre et maintenir la qualité du code dans des projets IoT évolutifs est un défi.
- Comprendre le compromis entre des métriques telles que RFC, CBO, LCOM, CC et WMC dans les systèmes IoT.

Implications 3

- La surveillance continue des métriques de code pour adapter le développement en fonction des résultats de ces métriques.

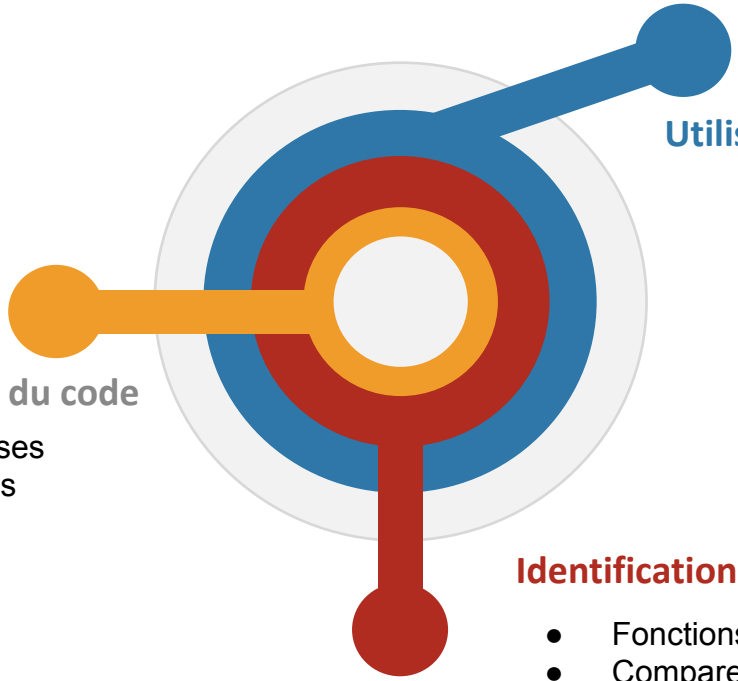
Les bonnes pratiques (1/6)

Origine des bonnes pratiques:



Les bonnes pratiques (2/6)

Taille



Utilisation de la décompression à l'exécution

- La décompression sélective, la décompression dynamiques et la décompression conditionnelle

Techniques d'optimisation du code

- Opérations coûteuses
- Inlining de fonctions
- Code dupliqué
- Cppcheck

Identification et consolidation des fonctions similaires

- Fonctions structurellement similaires
- Compare les signatures de fonctions et les graphes de flux de contrôle

Les bonnes pratiques (3/6)

Complexité

Application de la modularité

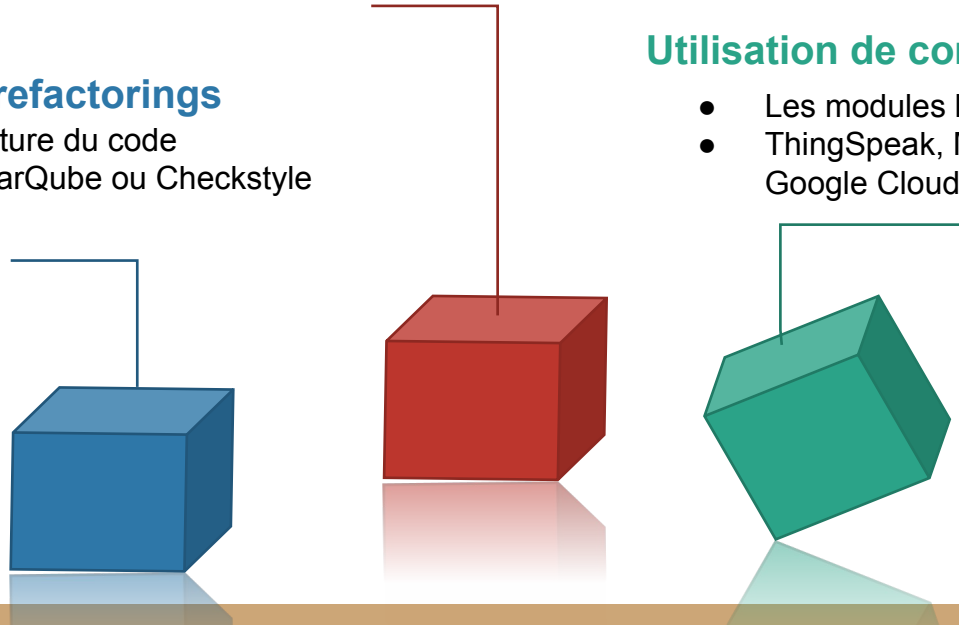
- Modules plus petits et plus gérables
- Les principes d'encapsulation et d'abstraction
- Outils de conteneurisation comme Docker

Application des refactorings

- Simplifier la structure du code
- Eclipse IDE, SonarQube ou Checkstyle

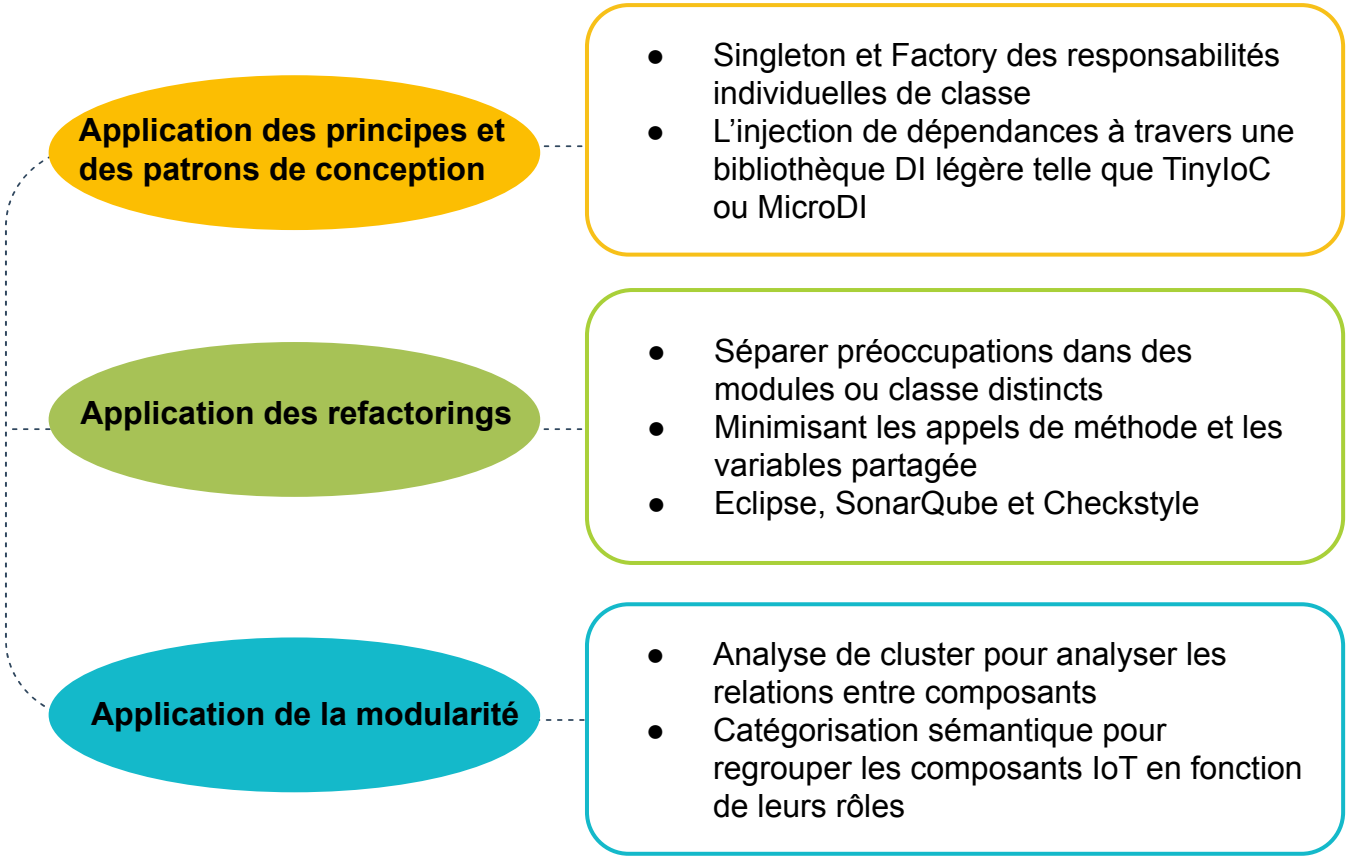
Utilisation de composants logiciels empaquetés

- Les modules logiciels préconstruits et prêts
- ThingSpeak, Microsoft Azure IoT Suite, Google Cloud IoT, etc.



Les bonnes pratiques (4/6)

Couplage & cohesion



Les bonnes pratiques (5/6)

Lisibilité du code

Utilisation de caractéristiques textuelles

- Lignes courtes, l'indentation cohérente et de l'utilisation des commentaires, utilisation des lignes vides, etc.

Améliorer l'entropie du code

- Calculer à partir du nombre de termes
- Le code avec des éléments plus variés (opérateurs et opérandes) est plus facile à comprendre
- Surveiller de manière constante la valeur de l'entropie lors du développement

Les bonnes pratiques (6/6)

Maintainabilité

Utilisation des conventions et normes de code source

- Les conventions de nommage et la structure syntaxique
- FindBugs, Checkstyle et Jtest

Utilisation de l'architecture pilotée par les modèles (MDA)

- Facilite les modifications au niveau des exigences, les propageant automatiquement aux modules concernés

Utilisation des patrons de conception

- Factory Method: objets sans spécifier de classes concrètes
- Decorator: l'ajout dynamique de responsabilités aux objets

Intégration continue et déploiement continue (CI/CD)

- Automatiser les processus de test et de déploiement
- Azure IoT Edge, CircleCI et Jenkins

Application des refactorings

- Directives
- Encapsulation

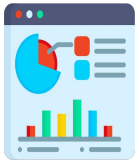
5. Conclusion

Conclusion

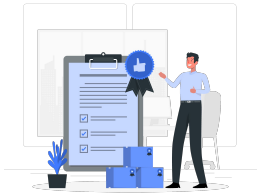
- Notre recherche porte sur la **qualité logicielle des systèmes IoT**, en se penchant sur **l'architecture et le code.**
- L'identification des **architectures** les plus utilisées, avec les **exigences de qualité** IoT, fournit un **guide** essentiel pour le développement de systèmes IoT.
- Les systèmes IoT présentent **une complexité accrue, un couplage élevé, une taille importante et une maintenabilité moindre.**
- Une liste revisitée des **bonnes pratiques.**

Limitations

Manque de représentativité des PSs



Les PSs peuvent ne pas être matures, affectant la généralisabilité de la conclusion



Le choix des métriques et des outils

L'expérience des développeurs

Contribution 1

Contribution 2

Travaux futurs

Comparer les styles architecturaux

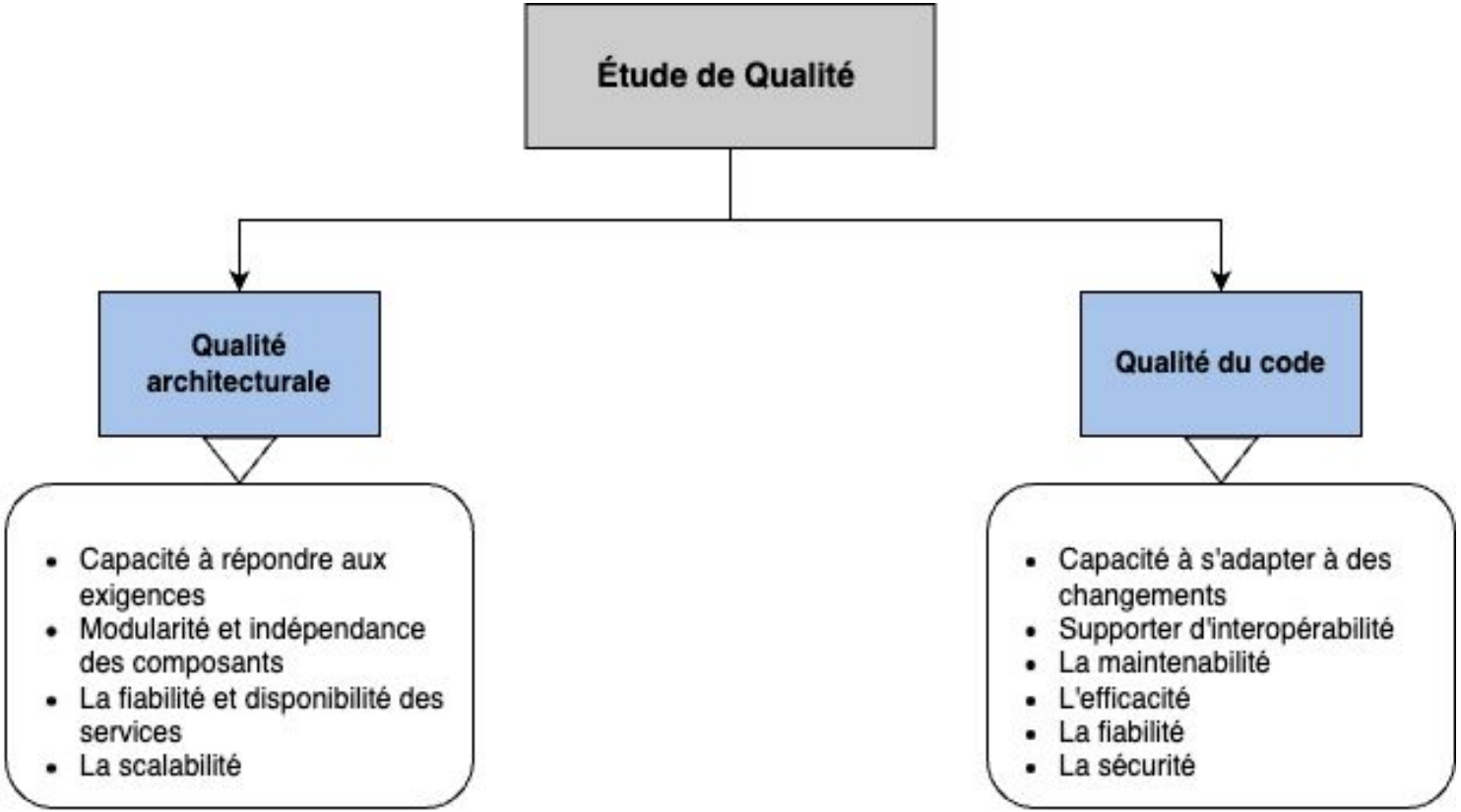
Étudier la maintenabilité, la compatibilité et la portabilité

Inclure au-delà des systémes GitHub

Mise en œuvre des bonnes pratiques

Merci pour votre attention

Contexte (3/3)


Étude de Qualité
Qualité architecturale
Qualité du code
Capacité à répondre aux exigences
Modularité et indépendance des composants
La fiabilité et disponibilité des services
La scalabilité
Capacité à s'adapter à des changements
Supporter d'interopérabilité
La maintenabilité
L'efficacité
La fiabilité
La sécurité

Qualité d’architecture IoT

- La qualité de l'architecture des systèmes IoT, peut être évalué a travers les exigences de qualité, telles que la scalabilité, fiabilité, la sécurité, l'évolutivité, la maintenabilité, la performance et d'autres aspects non fonctionnels.

Qualité du code d'IoT

- Peut être évaluée par les mesures des métriques.
- Peut être représenté par des modèles de qualité.

Énoncé du mémoire

- **Étude des exigences de qualité des systèmes IoT et les architectures les plus utilisées en proposant des conseils pour la sélection d'architectures répondant aux exigences de qualité.**
- **Étude de la qualité du code des systèmes IoT à travers une comparaison et analyse du code des systèmes IoT à celle des systèmes non IoT. Ces comparaisons orientent nos recommandations des bonnes pratiques pour le développement des systèmes IoT.**

Travaux existants:

Étude	Année	Concentration		
		EQ	SA	CAEQ
Sethi & Sarangi (2017)	2017			
Gill *et al.* (2017)	2017			
Muccini & Moghaddam (2018a)	2018			
Fahmideh & Zowghi (2018)	2018			
Muccini *et al.* (2018)	2018			
Alshohoumi, Sarrab, AlHamadani & Al-Abri (2019)	2019			
Washizaki *et al.* (2020)	2020			
Razzaq (2020)	2020			
Stojanov & Dobrilovic (2021)	2021			
Stefanova-Stoyanova *et al.* (2021)	2021			
Alfonso, Garcés, Castro & Cabot (2021)	2021			
Siddiqui, Khendek & Toeroe (2023)	2023			
Márquez, Astudillo & Kazman (2023)	2023			
Our study	2024			

EQ - Exigences de qualité,
SA - Style architecturale,
CAEQ - Correspondance Architecture/Exigences de Qualité,
NA - Non Disponible

Stratégie de recherche

P Études englobant les systèmes IoT et leurs styles architecturaux

I Discussions, évaluations ou analyses liées à l'architecture IoT

O Comparaisons des avantages, inconvénients et évaluations des styles architecturaux IoT

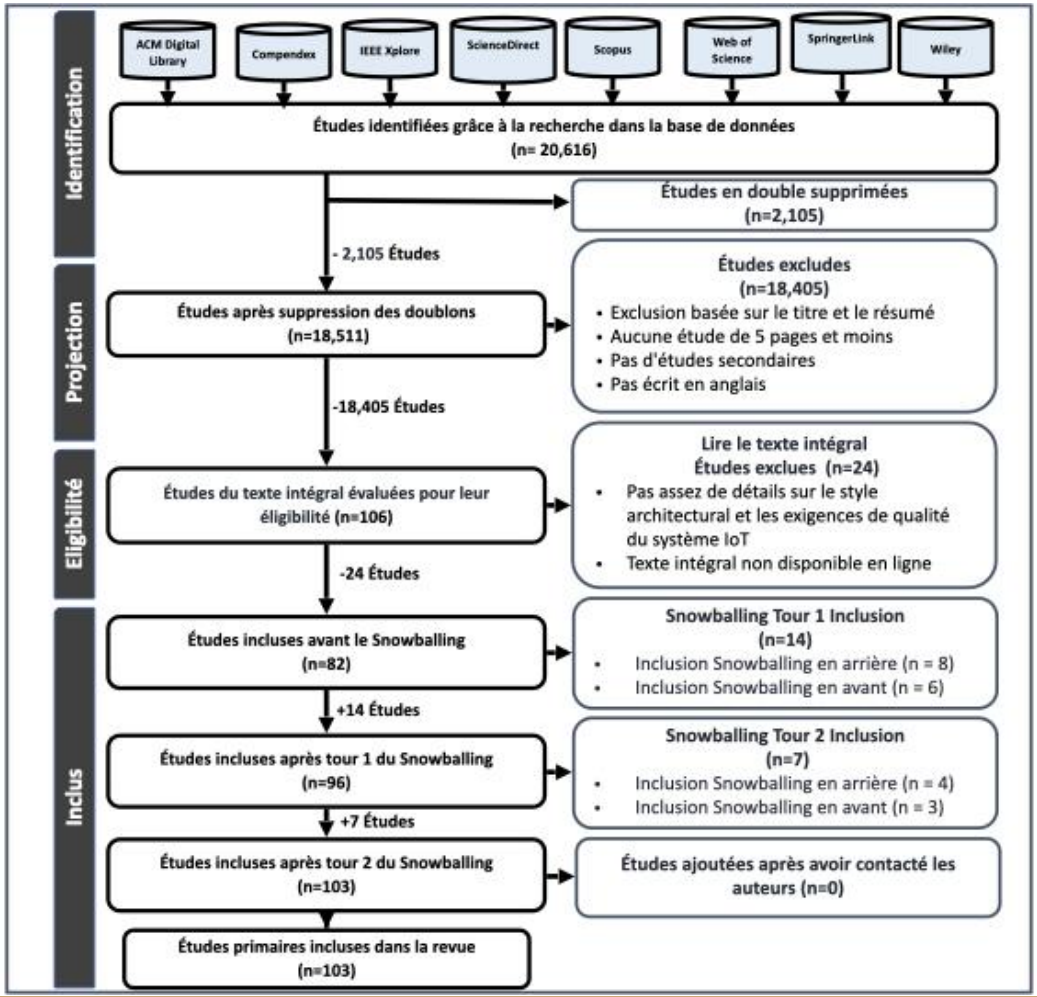
C Identification des exigences de qualité des systèmes IoT et des styles architecturaux IoT

Requête de recherche

((IoT) OR (internet of things) OR (internet-of-things)) AND ((Software architecture) OR (architecture) OR (architecting) OR (architectural) OR (design)OR (designing) AND ((disadvantages) OR (inconvenient) OR (drawbacks) OR (negative side) OR (problems)) OR ((advantages) OR (benefits) OR (added value))) AND ((IoT) AND ((evaluation) OR (evaluating) OR (quality) OR (quality attributes) OR (requirements) OR (quality characteristics)))

50

Processus PRISMA:



Méthodologie (2/2)

Les métriques à calculer

Catégorie	Métrique
Taille	Lignes de code (LOC)
	#Classes
	#Fichiers
Complexité	Complexité cyclomatique (CC)
	Volume Halstead (HV)
	Weighted Methods per Class (WMC)
Couplage	Réponse pour classe (RFC)
	Couplage entre objets (CBO)
Cohesion	Manque de cohésion des méthodes (LCOM)
Lisibilité du code	Ratio de commentaires par rapport au code (CP)
Maintenabilité	Indice de maintenabilité (MI)

Sélection (1/2)

Critères d'inclusion

- L'étude se concentre sur les styles architecturaux IoT et les exigences de qualité des systèmes IoT ;
- L'étude est publiée dans une conférence, un journal, un atelier ou un chapitre de livre ;
- L'étude est rédigée en anglais ;
- L'étude est primaire ;
- L'étude comporte au moins 5 pages.

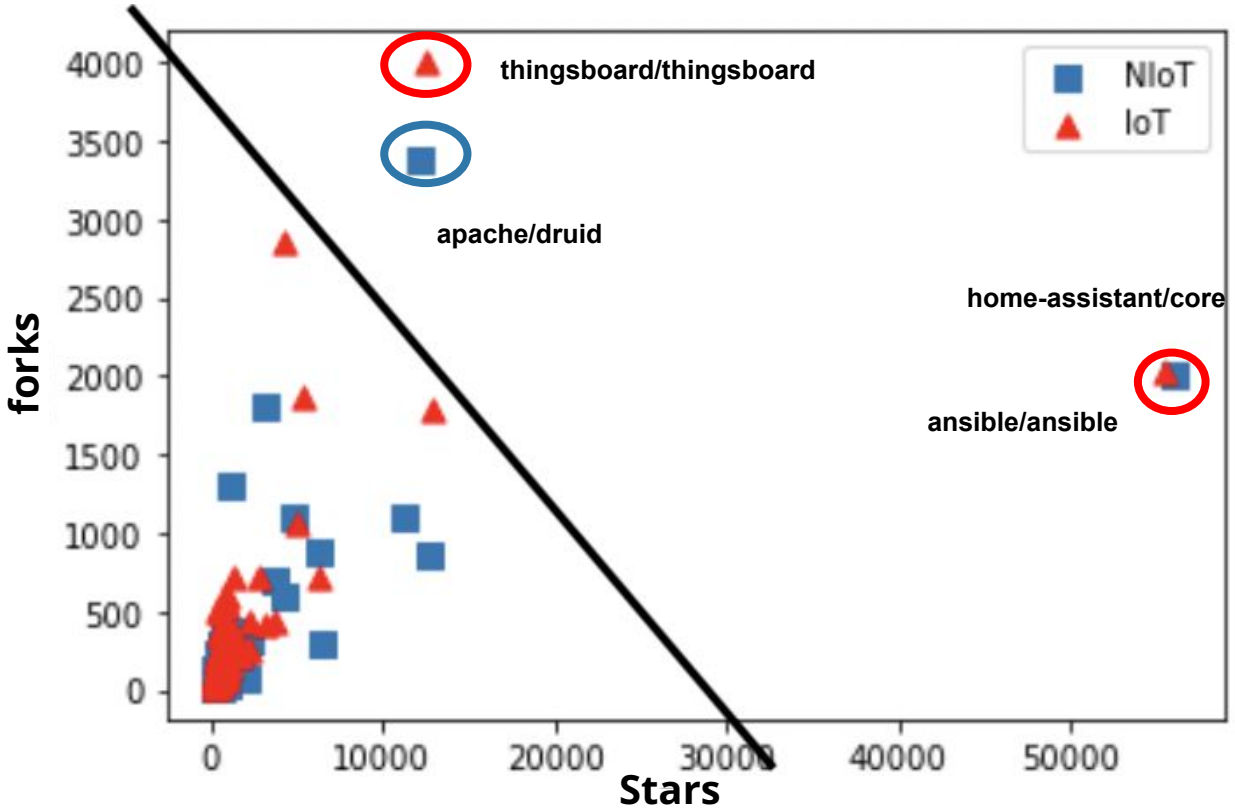
Critères d'exclusion

- L'étude n'est pas rédigée en anglais ;
- L'étude n'est pas disponible en ligne ;
- L'étude comporte moins de 5 pages (y compris les références) ;
- L'étude ne fournit pas suffisamment de détails sur le style architectural des systèmes IoT et les exigences de qualité.

82 Études primaires

Résultats (1/7)

Valeurs aberrantes



Résultats (2/7)

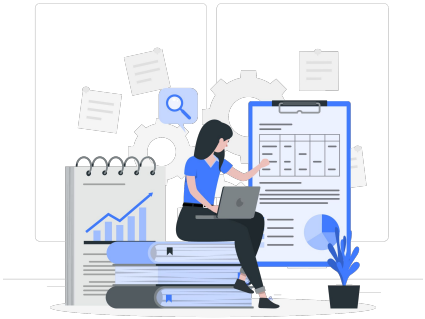
Valeurs aberrantes

Avant d'éliminer les systèmes aberrants

	IoT		non IoT	
	Nom du Projet	**Valeur**	**Nom du Projet**	**Valeur**
RFC	Samsung/TizenRT	475185	Apache/Druid	191718
CBO	eclipse-ditto/ditto	83212	Apache/Druid	22423
CC	espressif/esp-mqtt	167	quarnster/SublimeGDB	339
HV	Samsung/TizenRT	7539487.092	quarnster/SublimeGDB	137964.39
MI	eclipse-ditto/ditto	368880.47	Apache/Druid	475261.25
LOC	Samsung/TizenRT	2009696	Apache/Druid	1003619
WMC	project-chip/connectedhomeip	83347	Apache/Druid	16057
LCOM	rwaldron/johnny-five	0.95	rthenica/ffmpeg-kit	0.93
CP	ARMmbed/mbed-os	70251.58	Apache/Druid	4705.6

Sélection (2/2)

Snowballing



Snowballing	Cycle	Récupéré	Inclus
Arrière	Tour 1	208	5
Avant	Tour 1	250	9
Arrière	Tour 2	55	4
Avant	Tour 2	20	3
Total		533	21

→ **103** Études primaires

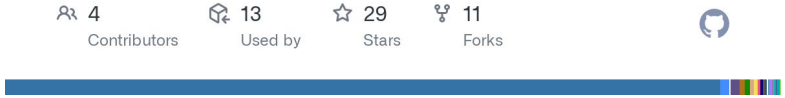
Méthodologie (2/5)

Les outils utilisés pour le calcul des métriques

Understand™ by SciTools

priv-kweihmann/
multimetric

Calculate code metrics in various languages



Méthodologie (3/5)

Les métriques à calculer

Catégorie	Métrique
Taille	Lignes de code (LOC)
	#Classes
	#Fichiers
Complexité	Complexité cyclomatique (CC)
	Volume Halstead (HV)
	Weighted Methods per Class (WMC)
Couplage	Réponse pour classe (RFC)
	Couplage entre objets (CBO)
Cohesion	Manque de cohésion des méthodes (LCOM)
Lisibilité du code	Ratio de commentaires par rapport au code (CP)
Maintenabilité	Indice de maintenabilité (MI)

Méthodologie (4/5)

Selection des systemes IoT

- Les langages du référentiel sont pris en charge par les outils d'analyse utilisés
- Le nombre d'étoiles est supérieur à 200
- Le nombre de forks est supérieur à 20
- Un référentiel actif avec la dernière mise à jour datant d'au moins 6 mois
- Un référentiel mature créé entre 2012 et 2022

(Internet of things OR IoT OR Internet-of-things OR EIoT OR IIoT OR Industrial Internet of Things OR internet of everything OR Entreprise internet of things) AND (stars:>200 language:C language:Java language:Python language:C++ language:JavaScript language:C# pushed:>2022-04 created:>2012 created:<2022)

Méthodologie (5/5)

Sélection des systèmes non IoT

Étape 1 : Requête de Recherche (basée sur les résultats de la recherche sur l'IoT)

created:>=2012-10-01 created:<2022-10-01 stars:>200 stars:<6500 forks:>18 forks:<=20216 pushed:>2022-04-01 language:C language:C# language:C++ language:C++ language:Java language:JavaScript language:Python

Étape 2 : Principe du Front de Pareto

1. Nous traçons les deux ensembles de données (IoT sélectionné et non-IoT) sur le même graphique.
2. Détermine la fonction objectif pour chaque ensemble de données. Ces fonctions objectif représenteront les deux objectifs considérés, qui sont le nombre d'étoiles (#stars) et le nombre de forks (#forks).
3. Identifiez le Front de Pareto, qui est l'ensemble de points représentant le compromis optimal entre les deux ensembles de données.
4. Compare les deux ensembles de données et sélectionnez les applications qui rendront le deuxième ensemble de données le plus similaire à l'autre.

Résultats (3/9)

Valeurs aberrantes

Après l'élimination des systèmes aberrants

	IoT		non IoT	
	Nom du Projet	**Valeur**	**Nom du Projet**	**Valeur**
RFC	Samsung/TizenRT	475185	DarthFubuMVC/fubumvc	64016
CBO	eclipse-ditto/ditto	83212	DarthFubuMVC/fubumvc	22423
CC	flomesh-io/pipy	75256385.40	mgba-emu/mgba	26505
HV	greghesp/assistant-relay	86796	dachev/node-cld	240076813.4
MI	eclipse-ditto/ditto	368880.47	wmira/react-icons-kit	811513
LOC	Samsung/TizenRT	2009696	dachev/node-cld	551449
WMC	project-chip/connectedhomeip	83347	rthenica/ffmpeg-kit	78624
LCOM	rwaldron/johnny-five	0.95	rthenica/ffmpeg-kit	0.93
CP	ARMmbed/mbed-os	70251.58	UnknownShadow200/ClassiCube	1413.95

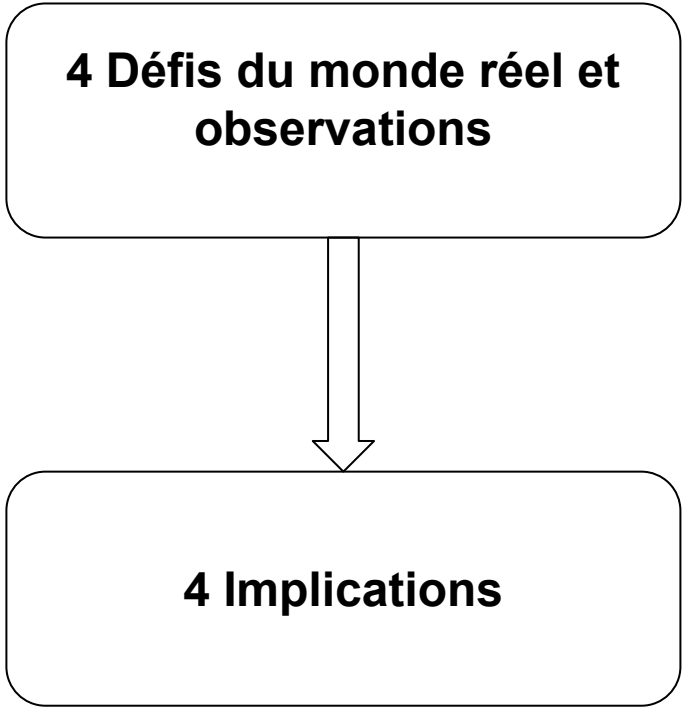
Résultats (7/9)

Analyse approfondie du code

	IoT **eclipse-ditto/ditto**	**non IoT** **kymjs/CJFrameForAndroid**
#Étoiles	414	412
#Forks	147	157
#Classes	**7573**	32
#Fichiers	**4917**	25
RFC	**130215**	328
CBO	**83212**	254
CC	**17056558.06**	94
HV	13423	**72196.47**
MI	368880.47	2043
LOC	**363467**	2040
WMC	**41499**	238
LCOM	0.62	0.67
CP	5702.18	17.23

⇨ **Le système IoT est plus complexe.**

Implications Pratiques pour le Développement IoT



Implications Pratiques pour le Développement IoT(1/4)

Défis du monde réel et observations 1

- IoT implique des interactions complexes entre le matériel et le logiciel, et les communications de données complexes.
- LOC, #Classes, #Fichiers, CC, RFC, WMC et l'analyse du code mettent en évidence la complexité du code des systèmes IoT.

Implications 1

- Les développeurs devraient cultiver des compétences spécialisées;
  - Une compréhension approfondie des aspects matériels et logiciels
  - Une expertise dans des protocoles de communication de données efficaces

Implications Pratiques pour le Développement IoT (2/4)

Défis du monde réel et observations 2

- Une interdépendance entre différents modules et une faible maintenabilité dans les systèmes IoT posent des défis pour apporter des modifications et maintenir la base de code.
- Les métriques LCOM, CBO, HV, et MI.

Implications 2

- Favoriser la conception modulaire et l'organisation efficace du code

Implications Pratiques pour le Développement IoT (3/4)

Défis du monde réel et observations 3

- Le nombre accru de classes et de fichiers dans les systèmes IoT est induit par leur nature distribuée.
- LOC, #Classes, #Files révèlent que les systèmes IoT présentent un code plus étendu.

Implications 3

- Les développeurs devraient privilégier la mise en œuvre de pratiques pour la réduction du code;
  - L'élimination de segments de code redondants et l'optimisation des bibliothèques de fonctions.

Implications Pratiques pour le Développement IoT (4/4)

Défis du monde réel et observations 4

- Comprendre et maintenir la qualité du code dans des projets IoT évolutifs est un défi.
- Comprendre le compromis entre des métriques telles que RFC, CBO, LCOM, CC et WMC dans les systèmes IoT.

Implications 4

- La surveillance continue des métriques de code pour adapter le développement en fonction des résultats de ces métriques.