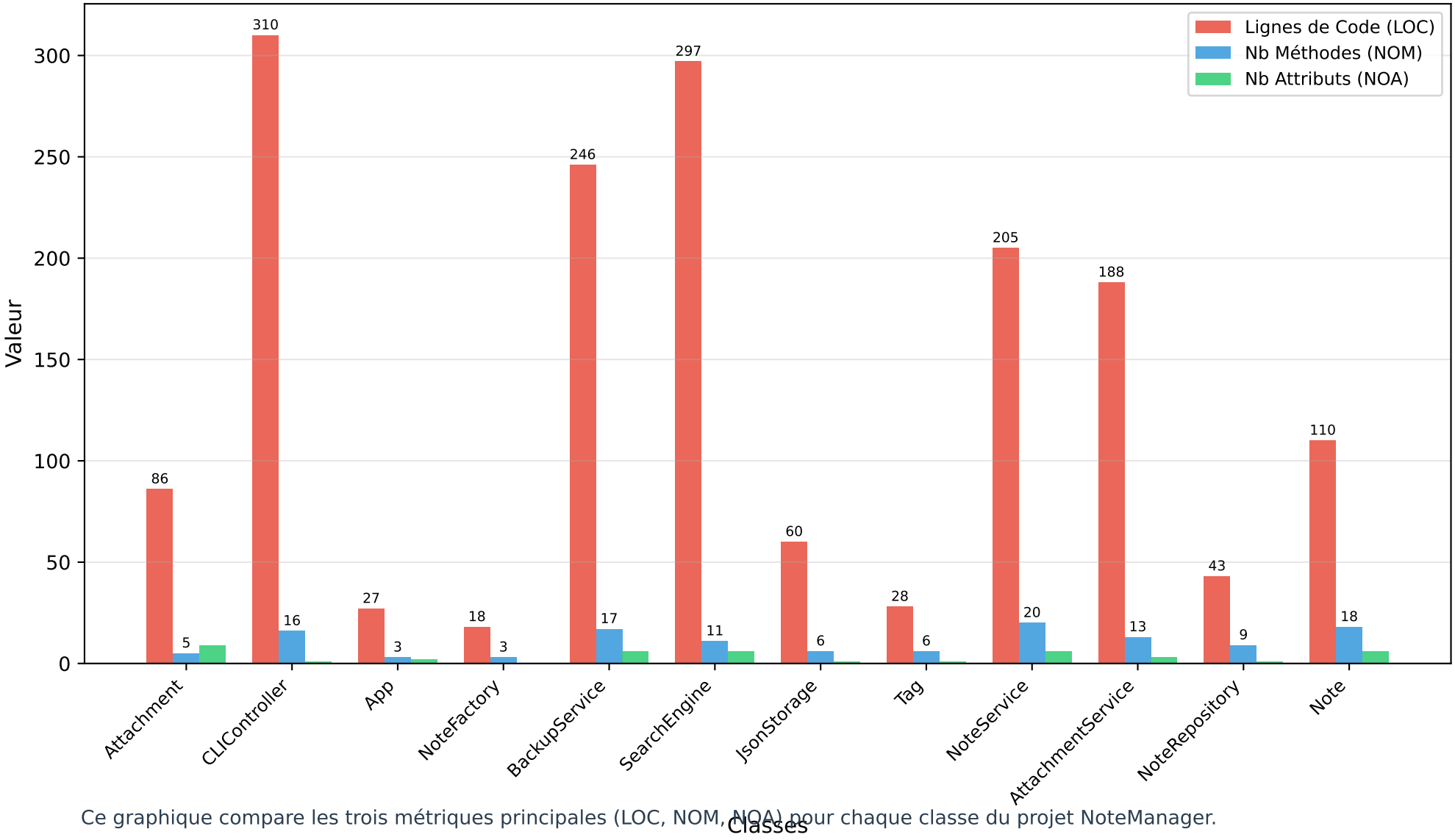
Rapport de Métriques de Qualité

Projet NoteManager — TP2 MGL843

Analyse automatisée par pipeline CI/CD

Généré le 2026-02-21 à 04:19

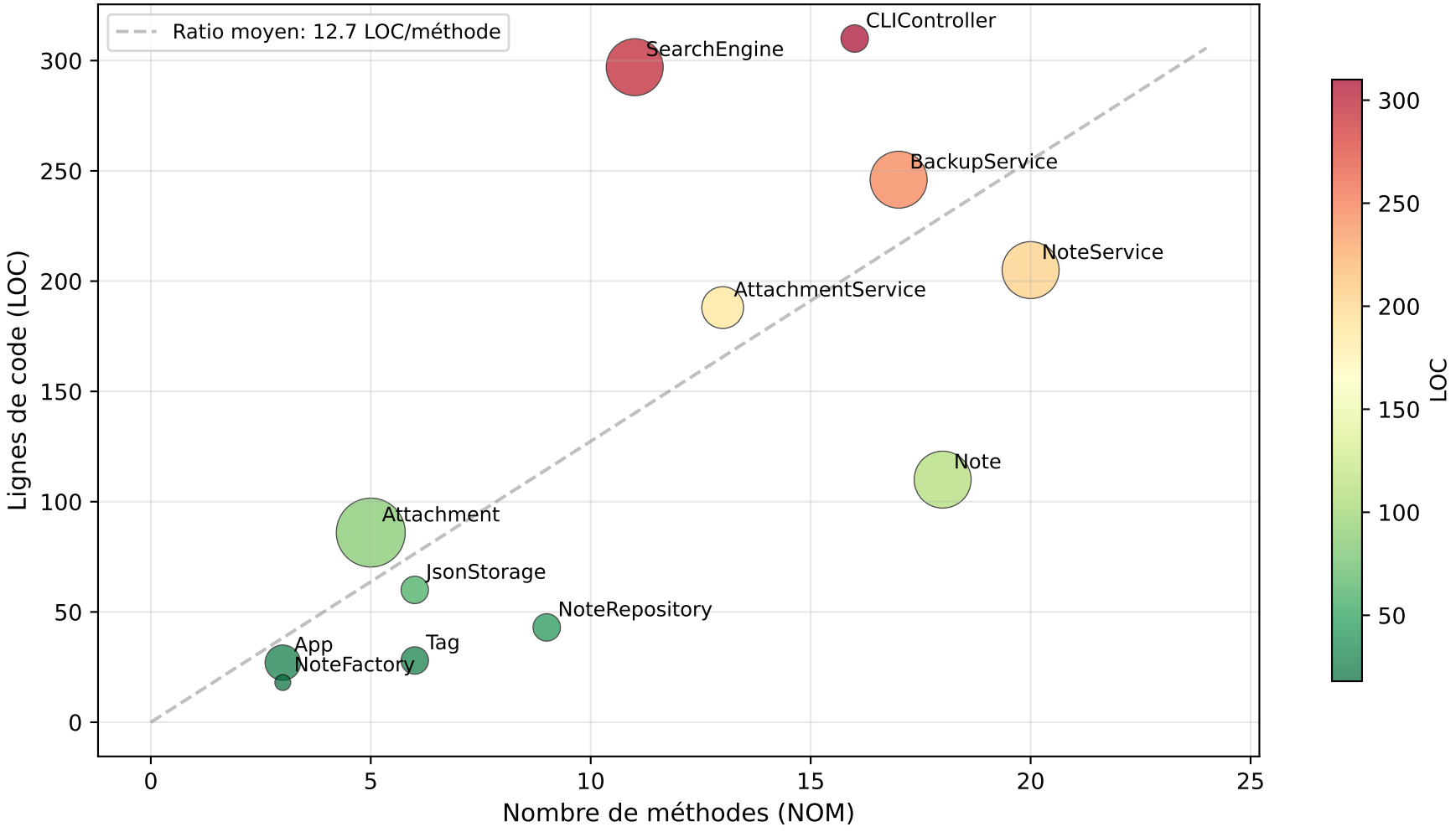
1. Histogramme des métriques par classe



Ce graphique compare les trois métriques principales (LOC, NOM, NOA) pour chaque classe du projet NoteManager.

La classe CLIController domine en termes de lignes de code (310 LOC), tandis que NoteService possède le plus grand nombre de méthodes (20). Un déséquilibre entre LOC et NOM peut indiquer des méthodes trop longues ou un manque de décomposition fonctionnelle.

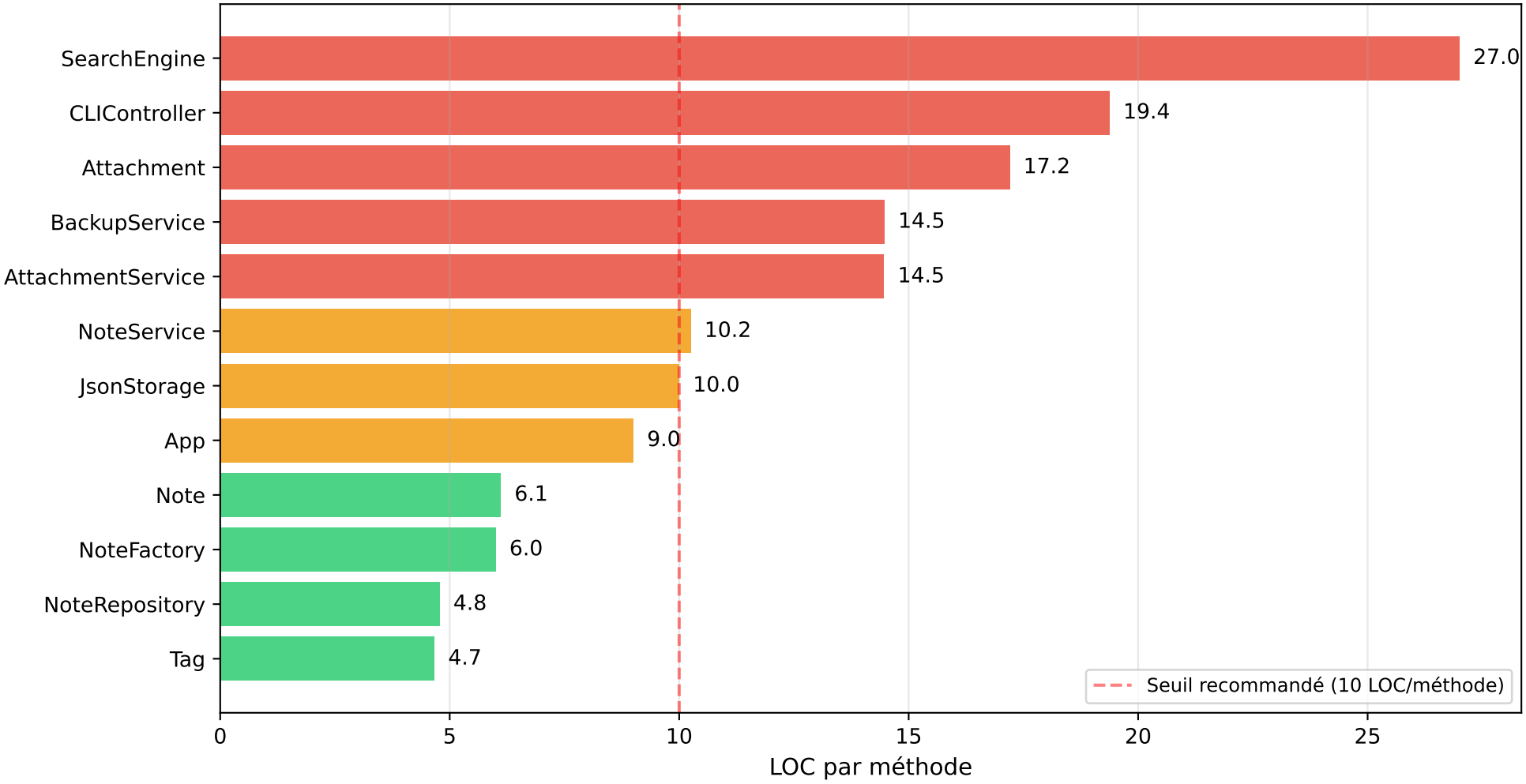
2. Diagramme de dispersion — LOC vs NOM



Ce diagramme positionne chaque classe selon son nombre de méthodes (axe X) et ses lignes de code (axe Y). La taille des cercles est proportionnelle au nombre d'attributs (NOA). La ligne pointillée représente le ratio moyen de 12.7 LOC par méthode.

Classes au-dessus de la moyenne (méthodes plus longues) : Attachment, CLIController, BackupService, SearchEngine, AttachmentService.
Classes en dessous (méthodes plus concises) : App, NoteFactory, JsonStorage, Tag, NoteService, NoteRepository, Note.

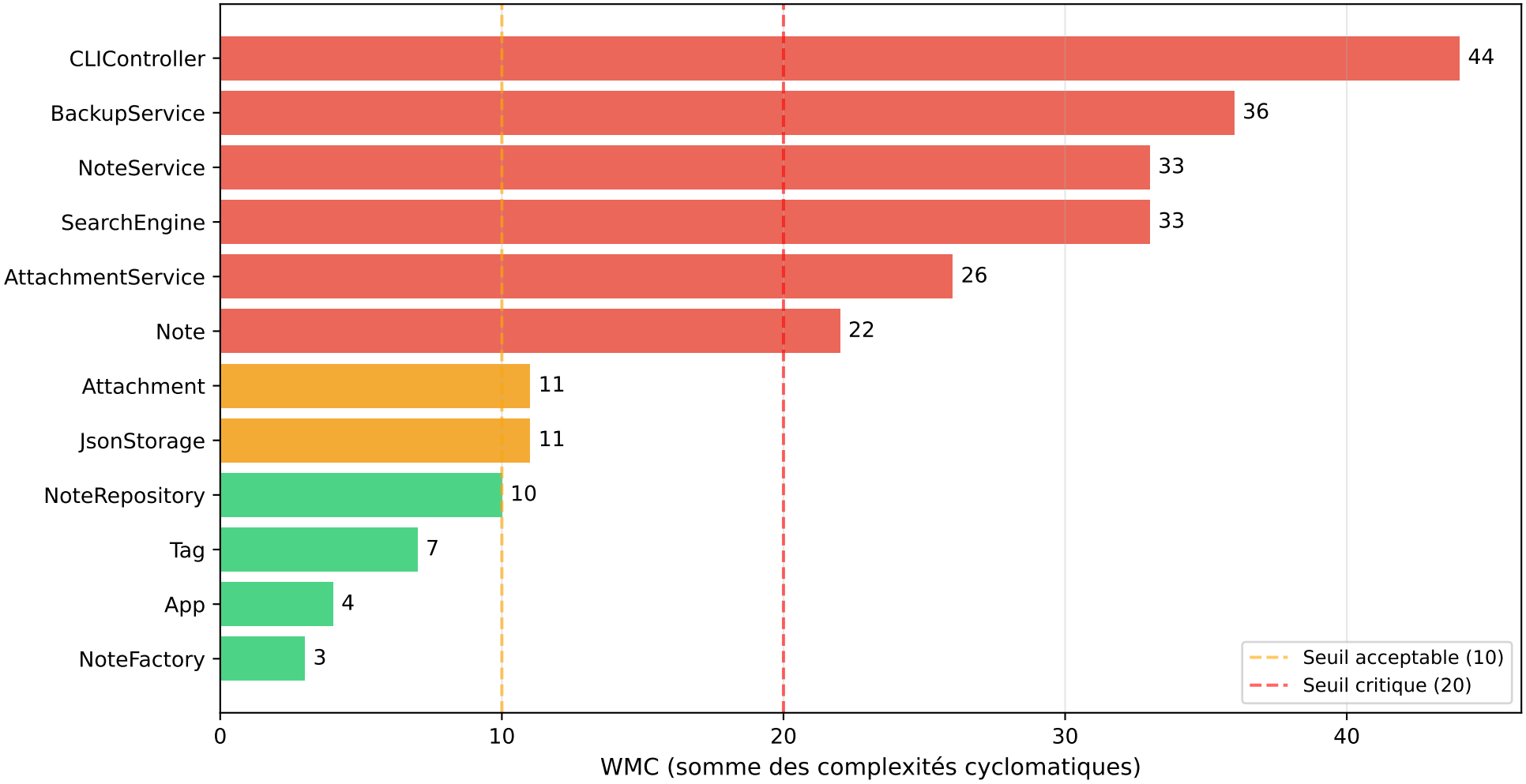
3. Densité de code par classe (LOC / méthode)



Ce graphique montre le ratio LOC/méthode pour chaque classe, trié par ordre croissant. Il permet d'identifier les classes dont les méthodes sont trop longues.

Vert = bon (< 8 LOC/méthode), Orange = acceptable (8-12), Rouge = à surveiller (> 12).
La densité moyenne du projet est de 11.9 LOC/méthode. Les classes dépassant le seuil de 10 sont : NoteService, AttachmentService, BackupService, Attachment, CLIController, SearchEngine. Elles pourraient bénéficier d'une décomposition en sous-méthodes.

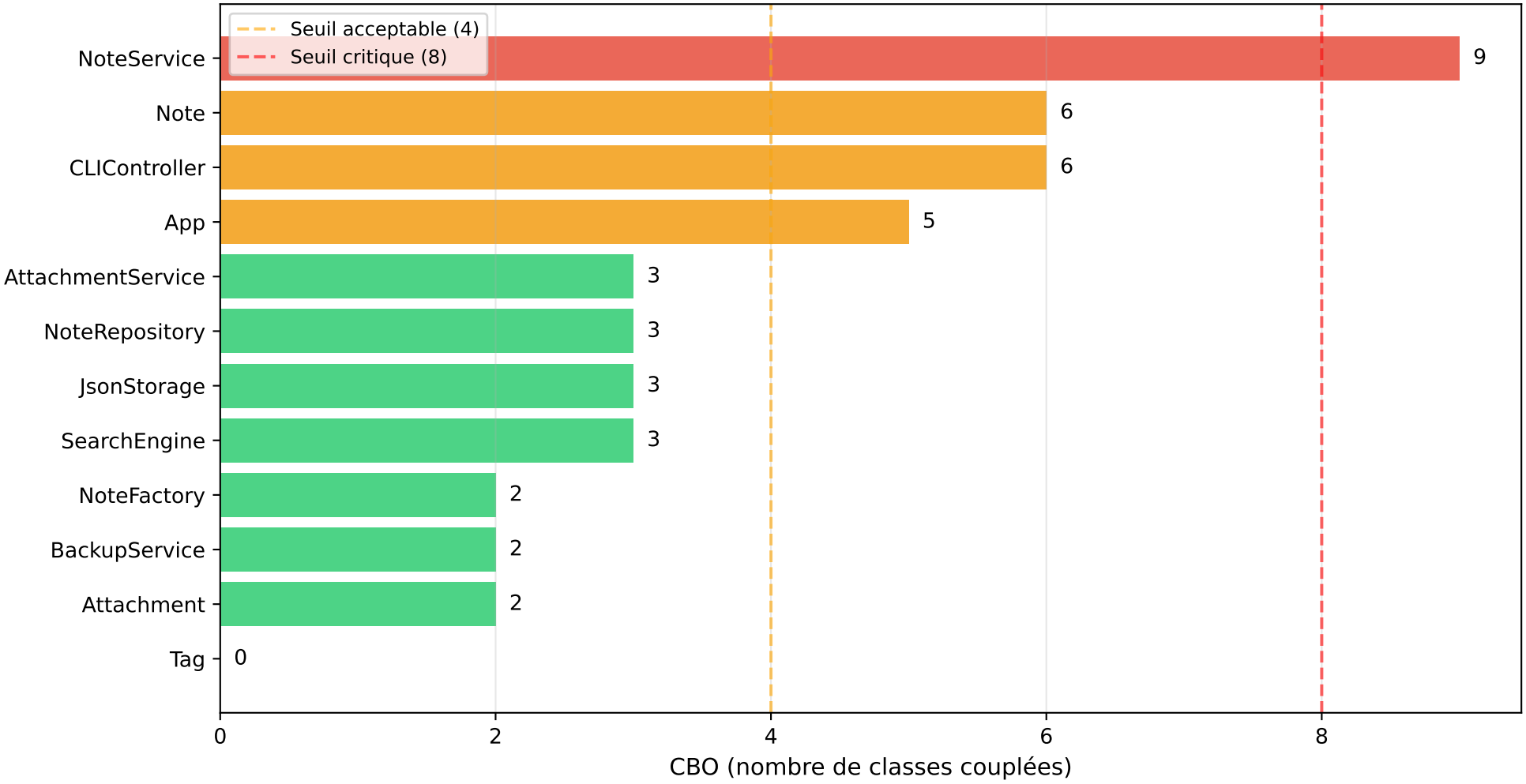
4. WMC — Weighted Methods per Class



WMC (Weighted Methods per Class) représente la somme des complexités cyclomatiques (CC) de chaque méthode. Un WMC élevé indique une classe complexe et difficile à tester.

La CC moyenne par méthode est de 1.78. Vert = bon (WMC ≤ 10), Orange = acceptable (≤ 20), Rouge = critique.
Classes critiques : Note, AttachmentService, SearchEngine, NoteService, BackupService, CLIController. Recommandation GRASP Forte Cohésion : décomposer en classes plus petites avec des responsabilités bien définies.

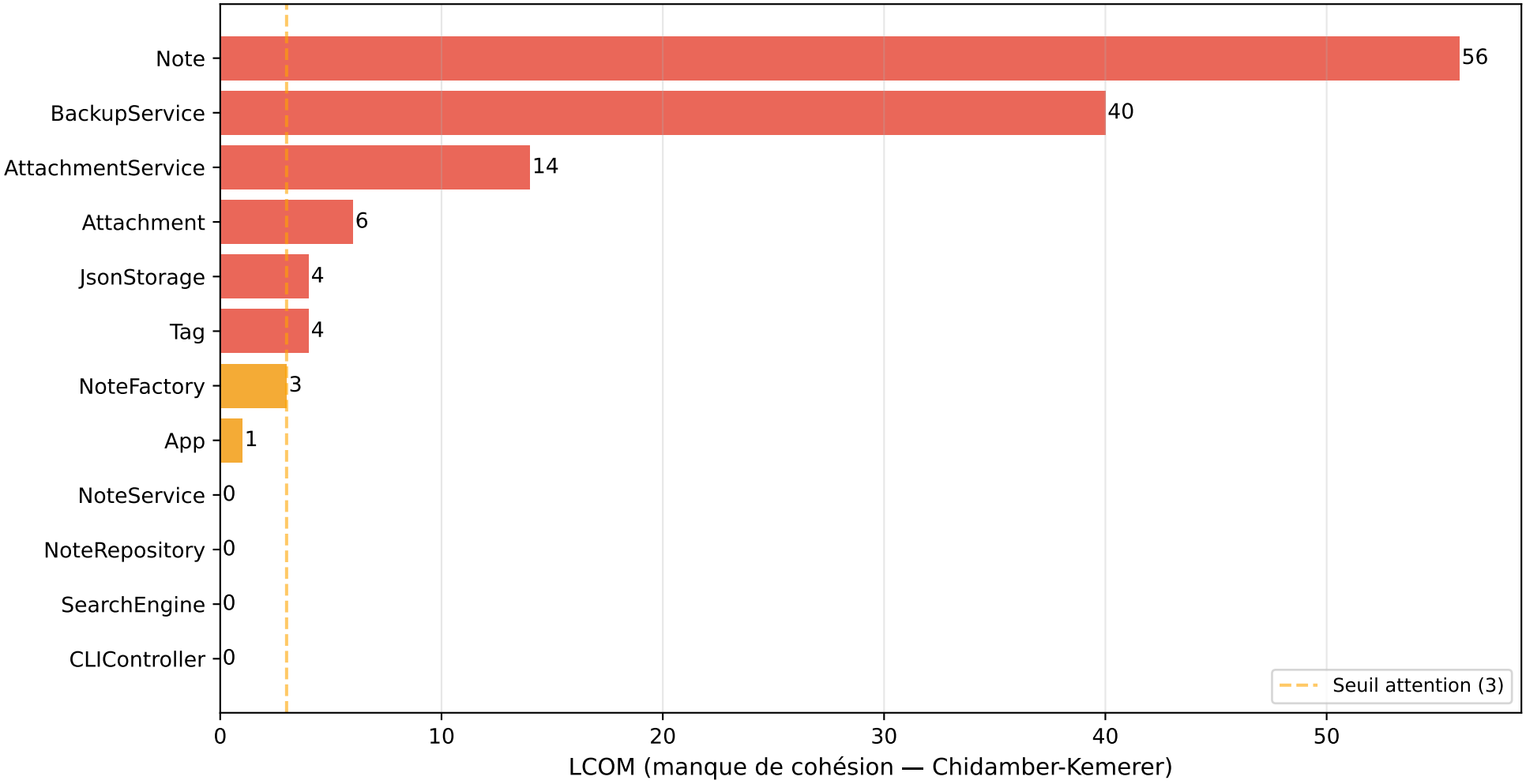
5. CBO — Coupling Between Objects



CBO (Coupling Between Objects) mesure le couplage bidirectionnel entre classes (invocations entrantes + sortantes + accès aux attributs). Un CBO élevé fragilise la maintenabilité.

CBO moyen du projet : 3.7. Vert = bon (≤ 4), Orange = acceptable (≤ 8), Rouge = critique.
Classes à fort couplage : NoteService. Recommandations : GRASP Faible Couplage, principe DIP (Dependency Inversion), ou patron Façade pour réduire les dépendances directes.

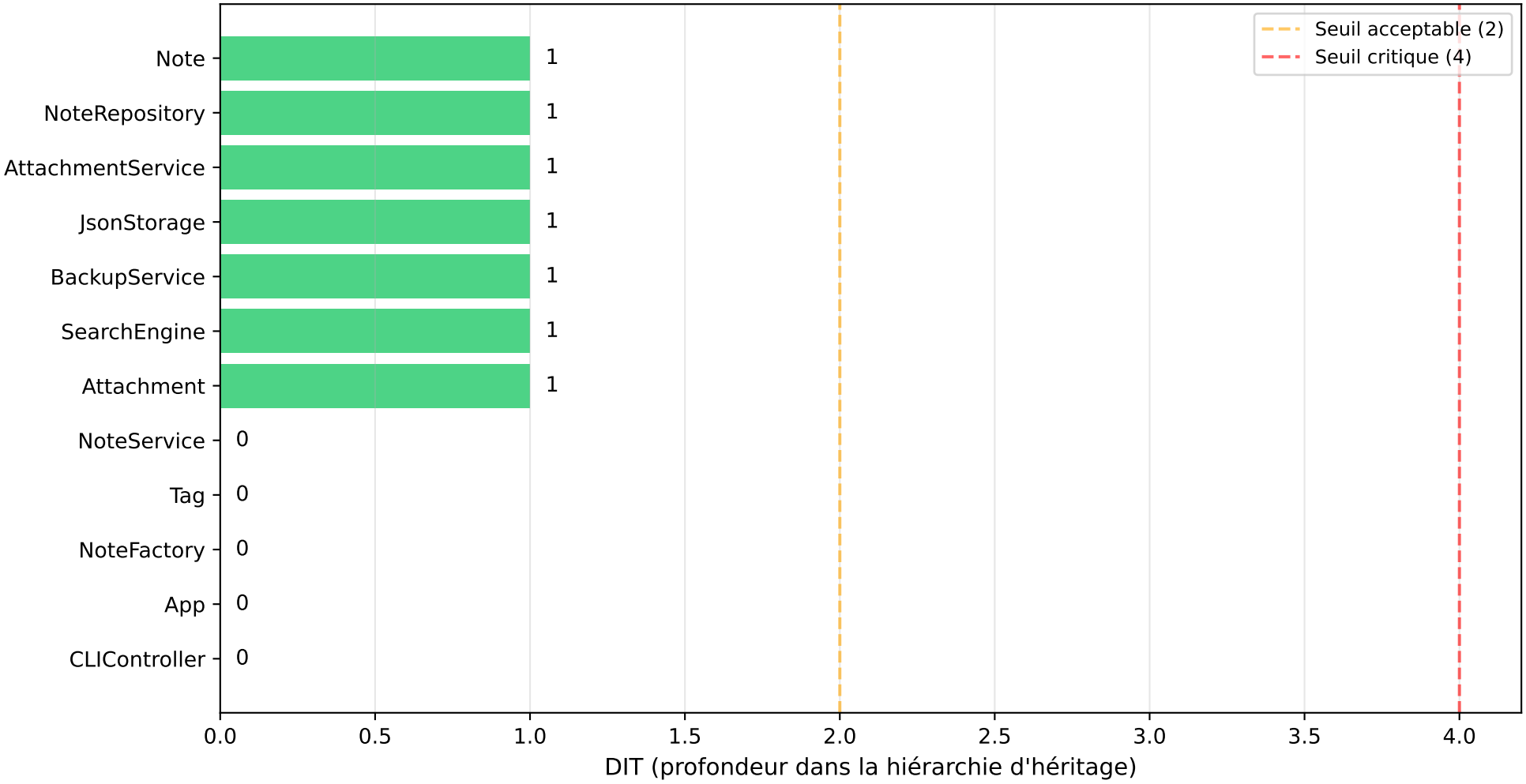
6. LCOM — Lack of Cohesion of Methods



LCOM (Lack of Cohesion of Methods) mesure le nombre de paires de méthodes sans attributs communs, moins le nombre de paires partageant des attributs (formule CK, min=0). Un LCOM élevé révèle une faible cohésion syntaxique.

LCOM moyen : 10.7. Vert = cohésif (LCOM = 0), Orange = attention (≤ 3), Rouge = critique.
Note (§1.5 du cours) : la cohésion syntaxique (LCOM) diffère de la cohésion sémantique — un LCOM=0 n'implique pas nécessairement une bonne conception.
Classes à faible cohésion : Tag, JsonStorage, Attachment, AttachmentService, BackupService, Note. Recommandation SOLID SRP : chaque classe devrait avoir une seule responsabilité. Envisager de séparer les responsabilités distinctes.

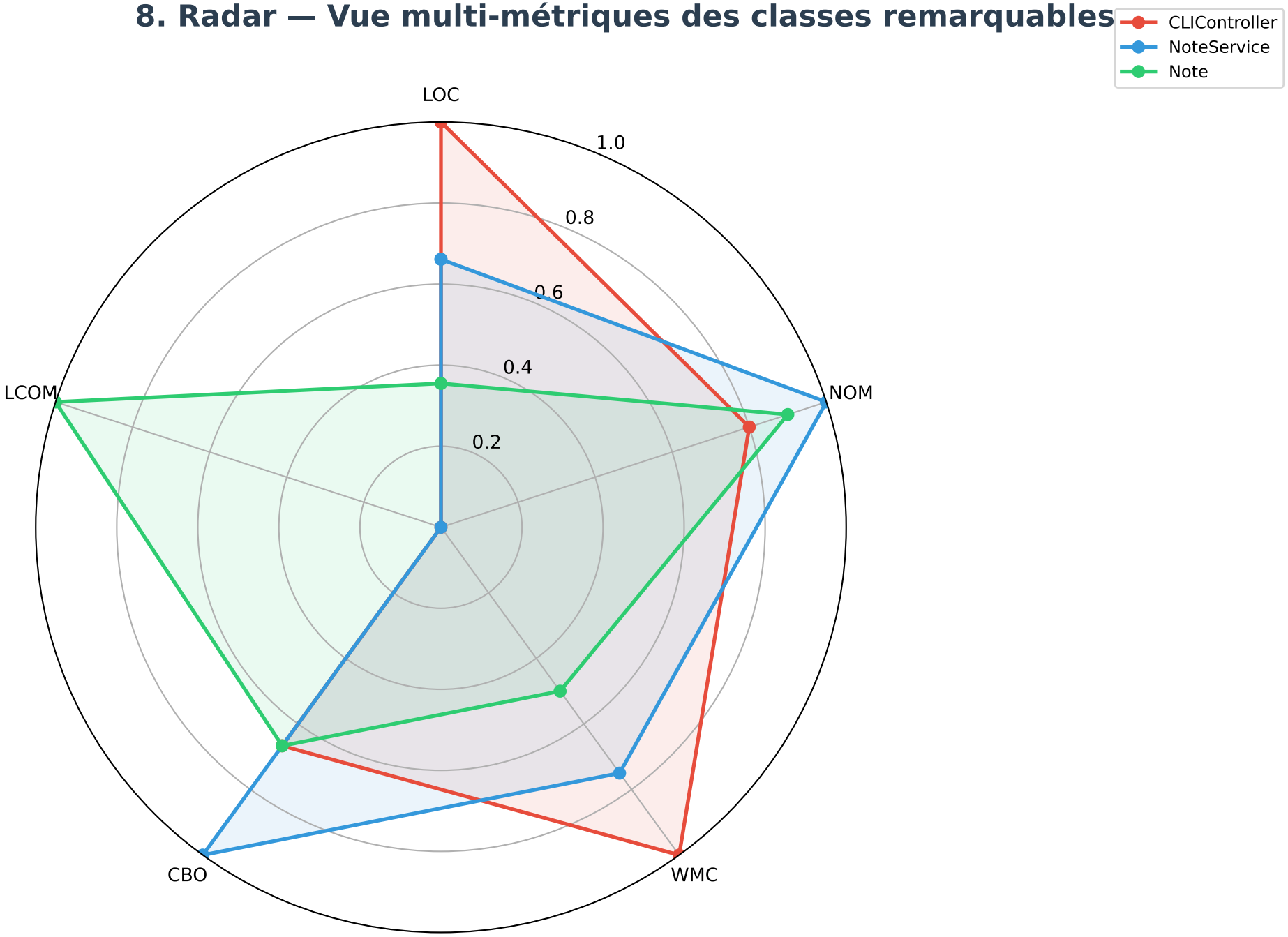
7. DIT — Depth of Inheritance Tree



DIT (Depth of Inheritance Tree) mesure la profondeur d'une classe dans la hiérarchie d'héritage. Un DIT élevé complique la compréhension et la maintenance.

DIT moyen : 0.6. Vert = bon (≤ 2), Orange = acceptable (≤ 4), Rouge = critique.
Toutes les classes présentent une hiérarchie d'héritage raisonnable. Le projet favorise la composition plutôt que l'héritage — conforme à l'heuristique §3.3.

8. Radar — Vue multi-métriques des classes remarquables



Ce graphique radar superpose les métriques LOC, NOM, WMC, CBO et LCOM (normalisées entre 0 et 1) pour les classes les plus remarquables du projet. Il permet d'identifier rapidement les classes qui dominent sur plusieurs axes.

Classes représentées : CLIController, NoteService, Note.
Une classe occupant beaucoup de surface radar concentre plusieurs risques qualité.

9. Tableau récapitulatif des métriques

Classe	NOM	NOA	LOC	WMC	DIT	CBO	LCOM	LOC/M	CC/M
Attachment	5	9	86	11	1	2	6	17.2	2.2
CLIController	16	1	310	44	0	6	0	19.4	2.75
App	3	2	27	4	0	5	1	9.0	1.33
NoteFactory	3	0	18	3	0	2	3	6.0	1.0
BackupService	17	6	246	36	1	2	40	14.5	2.12
SearchEngine	11	6	297	33	1	3	0	27.0	3.0
JsonStorage	6	1	60	11	1	3	4	10.0	1.83
Tag	6	1	28	7	0	0	4	4.7	1.17
NoteService	20	6	205	33	0	9	0	10.2	1.65
AttachmentService	13	3	188	26	1	3	14	14.5	2.0
NoteRepository	9	1	43	10	1	3	0	4.8	1.11
Note	18	6	110	22	1	6	56	6.1	1.22

Tableau complet des métriques pour les 12 classes du projet.

Statistiques globales :
- Total LOC : 1618 | Total méthodes : 127 | Total attributs : 42
- Ratio moyen LOC/méthode : 12.7 | WMC moyen : 20.0 | CBO moyen : 3.7 | LCOM moyen : 10.7

Conclusion et recommandations

Vue d'ensemble

Le projet NoteManager est composé de 12 classes, totalisant 1618 LOC dans 127 méthodes. Densité moyenne : 11.9 LOC/méthode, WMC moyen : 20.0, CBO moyen : 3.7, LCOM moyen : 10.7, DIT moyen : 0.6.

Points forts

- Bonne densité LOC (< 8 LOC/m) : NoteFactory, Tag, NoteRepository, Note.
- DIT moyen de 0.6 — le projet favorise la composition plutôt que l'héritage (§3.3).
- CBO moyen de 3.7 — couplage maîtrisé (Faible Couplage GRASP).

Points d'attention

- WMC élevé : CLIController, BackupService, SearchEngine, NoteService, AttachmentService, Note. → Décomposer (GRASP Forte Cohésion).
- CBO élevé : NoteService. → Réduire dépendances (DIP, Façade, Faible Couplage).
- LCOM élevé : Attachment, BackupService, JsonStorage, Tag, AttachmentService, Note. → Séparer responsabilités (SOLID SRP).
- Méthodes longues (> 12 LOC/m) : Attachment, CLIController, BackupService, SearchEngine, AttachmentService. → Décomposer en sous-méthodes.

Classe la plus volumineuse : CLIController (310 LOC, WMC=44, CBO=6).

Pipeline de validation

Métriques générées par le pipeline CI/CD GitHub Actions (ts2famix → Pharo/Moose → Python), validées par vérification croisée sur 4 niveaux (structure, invariants, calcul indépendant, comparaison avec tolérances WMC±2, CBO±2, LCOM±3).