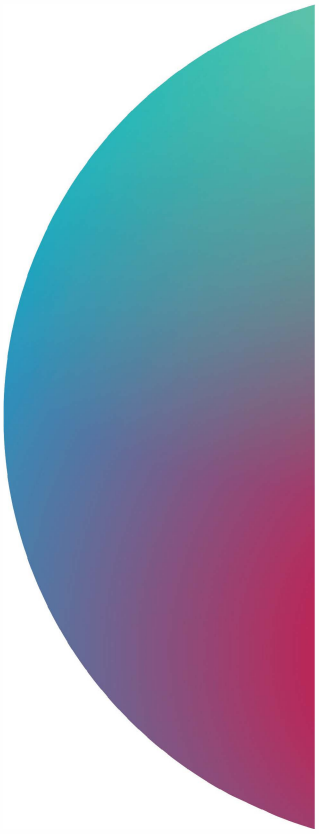
Application

Mesure



Rapport d'Évaluation

11/24/2022

Dernière mise à jour: 11/21/2022 09:13:16

Rapport d'Évaluation

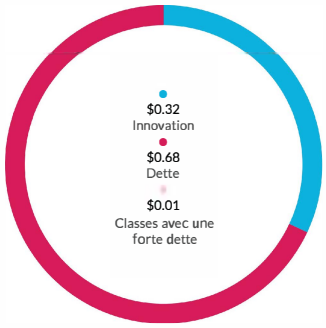
Coût de l'Innovation

Le rapport d'évaluation montre le niveau de dette technique et de complexité architecturale dans votre application. Grâce à l'analyse statique, le rapport expose comment les dépendances entre les classes affectent la modularité du code, le risque d'impacter des parties de l'application et le niveau global de dette technique. Ce niveau de dette affecte la vitesse et le coût de l'innovation et constitue une mesure clé pour l'entreprise.

Ce graphique montre comment le niveau de dette technique affecte le coût de l'innovation pour l'application. Il tente d'évaluer si l'on dépense 1 $ sur cette application, combien sera réellement consacré à l'innovation et combien sera consacré au remboursement de la dette technique à l'intérieur de l'application.

Les classes qui contribuent le plus à la dette globale, en ajoutant de la complexité et du risque dans l'ensemble de l'application, sont répertoriées à la page 2 du rapport. Le calcul post-refactorisation évalue le niveau de la dette technique en supposant que ces classes ont été refondues et que leur niveau de dette a été réduit.

Avant la refactorisation



68% - Dette technique

Le rapport calcule le coût total de possession (TCO) de l'application, comparé à une application sans dette technique.

Avant la Refactorisation

Multiplicateur du coût total de possession

x3.1

Après la Refactorisation

Multiplicateur du coût total de possession

x3.0

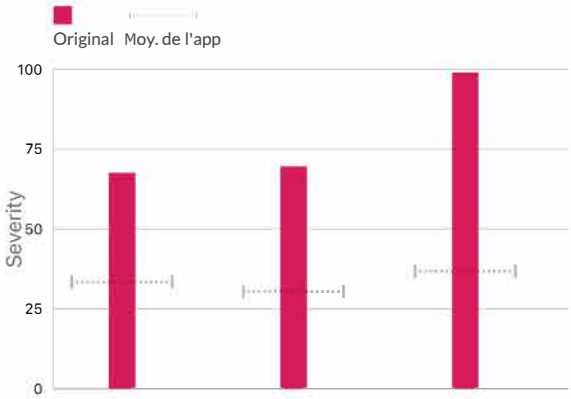
Métriques de Dette Technique

La complexité est le degré d'entrelacement des dépendances entre les classes, réduisant le niveau de modularité du code.

Le risque est corrélé à la longueur des dépendances. À quel point un changement dans l'application est-il susceptible d'affecter une partie apparemment non liée de celle-ci.

La dette technique est un calcul global prenant en compte la complexité, le risque et d'autres facteurs pour fournir une mesure unique afin de prioriser l'application pour la modernisation.

La moyenne de ces trois mesures dans le portefeuille d'applications est présentée pour aider davantage à la priorisation.



	Indice de Dette	Indice de Risque	Indice de Complexité
Pointage	67.61	69.6	99
Moy. de l'app	33	31	37
Distance par rapport à la moyenne	+102%	+128%	+169%

Rapport d'Évaluation

Cadres de travail vieillissants

Une partie importante de la dette technique est liée aux cadres vieillissants utilisés dans l'application. Ces cadres vieillissants constituent une source de risque de sécurité et indiquent un effort accru pour mettre à jour les cadres vers leurs dernières versions. L'utilisation de certains cadres vieillissants bloquera également l'application de la considération comme étant cloud-native.

Cette liste ci-dessous montre les noms des fichiers jars des cadres vieillissants utilisés dans l'application.

Plus de détails sur les différents cadres sont disponibles dans la version en ligne de ce rapport. Ces informations comprennent les noms des fichiers jars actuellement utilisés, la version actuelle et la version la plus récente du jar, ainsi que les coordonnées Maven de la version la plus récente du cadre.

Chaque cadre, en fonction de sa version actuelle et la plus récente, reçoit une étiquette de moderne, vieillissant ou inconnu, dans le cas où le jar spécifique n'a pas été trouvé dans la base de données de vFunction.

Votre application utilise

19%

Cadre moderne

0%

Cadre inconnu

81%

Cadre vieillissant

Nombre de classes

3669

Version minimale de compilation

1.8

Cadres de travail vieillissants

Cadre moderne

On a découvert qu'un cadre utilisé était d'une version récente, allant jusqu'à la dernière version mineure disponible.

Cadre inconnu

Le cadre utilisé n'a pas été identifié par la plateforme vFunction. Il est possible qu'il s'agisse d'un cadre propriétaire.

Cadre vieillissant

On a découvert qu'un cadre utilisé était plus ancien que la dernière version disponible d'au moins une version mineure, et devrait être mis à jour.

Classes les plus endettées

Nom
service.OrganizationService
.service.UserServiceImpl
service.DocumentService
service.OrganizationServiceImpl
service.EntityOrgCapabilitiesServiceImpl
.service.EntityServiceImpl
service.RoomService
.resource.process.OrganizationProcessor
.resource.process.RoomProcessor
service.RoomServiceImpl

Cadres vieillissants (affichage de 10 sur 139)

Nom	Version
activation	1.1.1-hudson-1
ActiveMQ Broker in Tomcat - Core	5.14.5
ActiveMQ Broker in Tomcat - Core	5.14.5
AspectJ Runtime	1.9.9.1
AspectJ Weaver	1.9.9.1
AWS SDK for Java - Core	1.12.336
AWS SDK for Java Mobile - Core ...	2.6.19
AWS Java SDK for AWS KMS	1.12.336
AWS SDK for Java Mobile - AWS KMS	2.6.19
axis	1.4

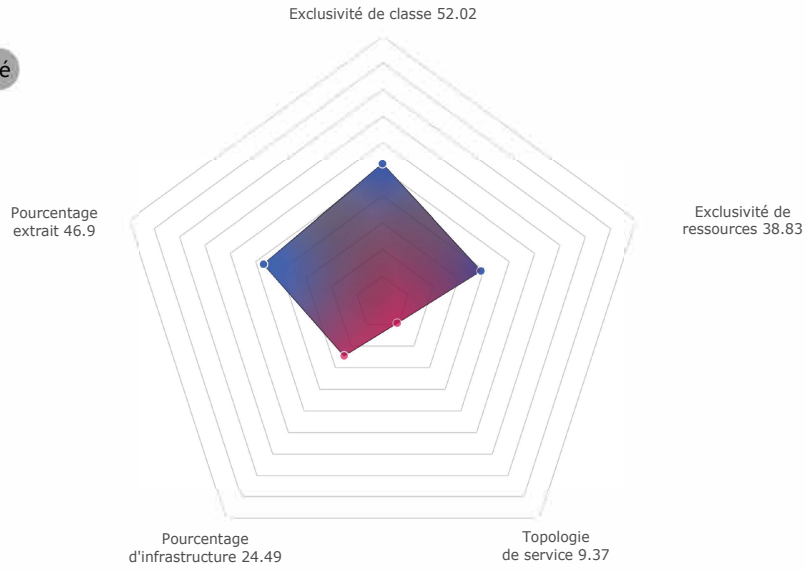
Effort de refactoring

Score d'évaluation 4 Effort élevé

Le graphique radar montre les principaux facteurs de modernisation de l'application. Plus la zone est couverte par le graphique, plus l'application est adaptée à la modernisation.

Bien que ces paramètres n'indiquent pas directement la complexité de l'application elle-même, ils indiquent le niveau d'effort nécessaire pour ré-architecturer l'application.

Cette évaluation est basée sur une mesure spécifique et dépend de l'analyse qui a été effectuée sur elle.



Le score d'effort de refactoring vFunction est classé de 1 à 5. Plus le score est élevé, plus l'effort requis est important.

1	2	3	4	5
Faible effort	Effort medium	Effort moyen	Effort élevé	Effort très élevé
0%-19%	20%-39%	40%-59%	60%-79%	80%-100%

Pourcentage extrait : Med
Le pourcentage de classes qui ne sont pas nécessaires dans le monolithe après l'extraction des services. Pour augmenter ce pourcentage, ajoutez des points d'entrée plus proches de la racine de l'application.

Exclusivité de ressources : Med
Le pourcentage de ressources exclusives aux services. Pour l'augmenter, examinez les ressources non exclusives dans les services identifiés.

Pourcentage commun : Med
Le niveau de classes communes trouvées. Pour l'augmenter, examinez les classes communes, marquez les classes de logique métier comme non communes et marquez les jars d'infrastructure.

Exclusivité de classe : Med
Le pourcentage de classes exclusives aux services. Pour l'augmenter, examinez les classes non exclusives des services.

Topologie de service : Faible
Le nombre d'appels de service à service nécessaires pour appeler n'importe quel service. Pour l'augmenter, réduisez le nombre de services partagés.

Sortie d'Analyse

Services	27
Points d'entrée	82
Classes	147
Classes communes	63
Exclusivité des ressources	82%
Exclusivité de classe	87%
Pourcentage extrait	79%

Services identifiés (27)

ActiveDirectoryReEnableJob
ActiveDirectorySyncJob
AsyncQueueExecutor
CleanJob
com.confidela.sealdoc.job_O
ConversionReportJob
CSVCreator
DeleteEmbeddedNotificationsJo
DocumentServiceImpl
DocuSignJob (+1)
Entity

...