



Génie logiciel

Maintenance

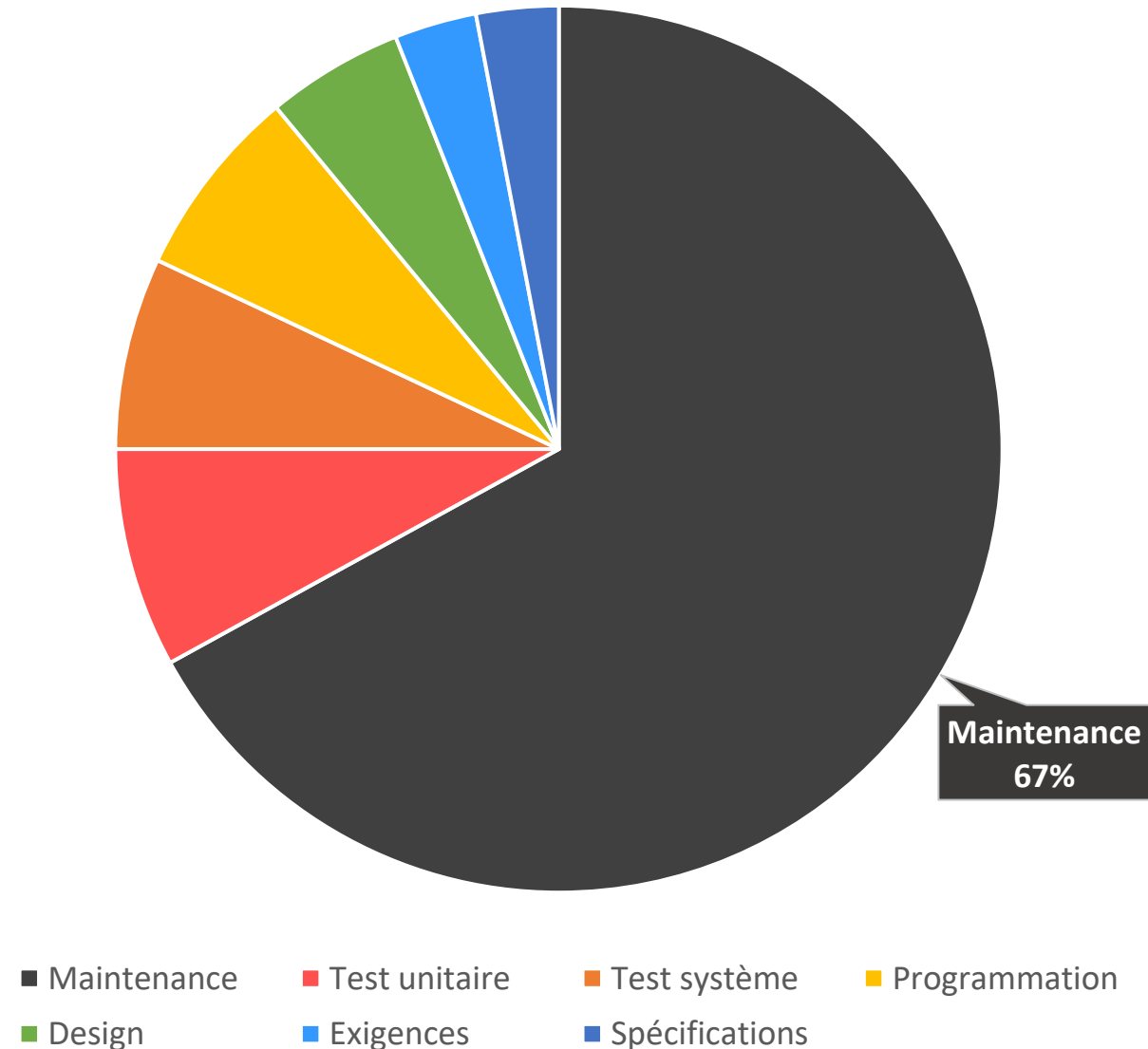
Louis-Edouard LAFONTANT



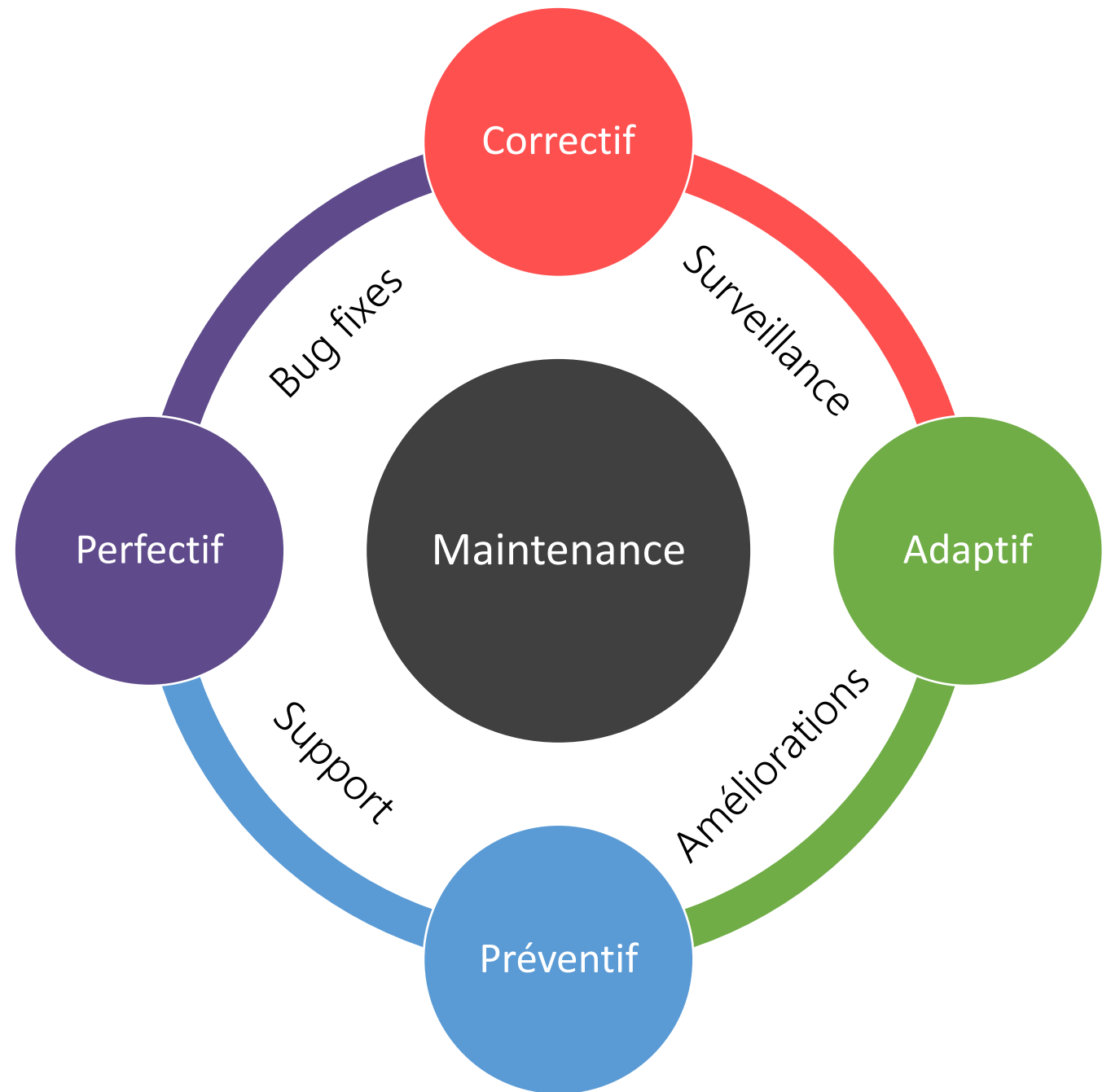
Maintenance

- Tout changement de n'importe quel artéfact du logiciel une fois déployé
 - Code source, tests, documents
- Maintenabilité doit être pensée dès le début
 - Pour réduire l'effort, le temps et les coûts
 - Ne doit pas être compromise par le processus de développement
- **Source majeure de revenu**
 - Certaines compagnies en font leur principal modèle d'affaire
- Maintenance est un des aspects les plus difficiles de la production de logiciel
 - Incorpore **tous** les autres workflows

Coûts des phases de développement logiciel



Types de changement



Type de changements

- **Perfectif (51%)** : accroître la valeur du logiciel suite à des changements demandés par le client
 - Ajouter des fonctionnalités
 - Améliorer la performance
- **Adaptatif (24%)** : préserver la valeur du logiciel en répondant aux changements de l'env.
 - Transférer vers un nouveau compilateur, OS, matériel
 - Changement légal (ex: taux d'imposition), de norme (ex: nouvelle devise)
- **Correctif (22%)** : corriger les défauts restants
 - Analyse, conception, implémentation, documentation, ...
- **Préventif (3%)** : protège le logiciel en facilitant les changements futurs par anticipation
 - Augmenter sa maintenabilité
 - Invisible à l'utilisateur (changement interne, ex: *refactoring*)

Maintenance corrective

- De quels outils dispose le programmeur de maintenance pour trouver la faute ?
 - Le **rapport de défauts** produit par un utilisateur
 - Le code source
 - Et souvent, rien d'autre (la documentation ?)
- Il doit donc avoir des talents de débogage extraordinaires
 - **La faute peut être n'importe où** dans le produit
 - La cause de la faute peut être située dans une exigence ou un document de conception aujourd'hui inexistant

Problèmes de régression

Supposons qu'on a localisé la faute.
Comment la corriger sans (ré)introduire une faute?

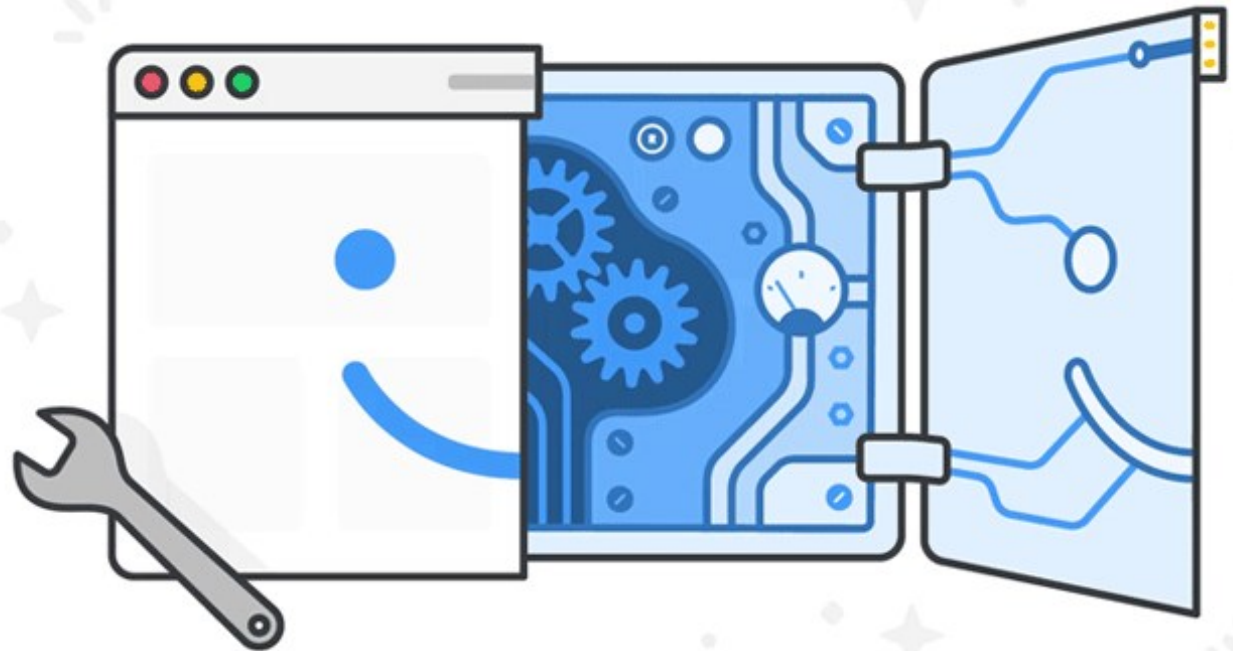
- Comment minimiser les fautes de régression ?
 - Consulter la documentation complète du produit
 - Consulter la documentation détaillée de chaque module
- Souvent, la documentation est inexistante, incomplète ou erronée
- Il doit déduire du code même toute l'information pour éviter d'introduire une faute de régression

Vérification

- Tester que les modifications fonctionnent correctement
 - En utilisant des **cas de test** spécifiquement conçus pour celles-ci
- Effectuer des **tests de régression**
 - Ré-exécuter les test en utilisant les données de test existantes
- Ajouter les nouveaux cas de test à ces données pour qu'ils fassent parties de tests de régression **futurs**
- **Documenter** tous les changements

Assurer la maintenance

Penser à la maintenance tout au long du développement



Assurer la maintenance

- Maintenance n'est pas un effort ponctuel
- Il faut la planifier tout au long du cycle de vie du logiciel
 - Workflow de conception : utiliser les techniques de dissimulation d'information et d'implémentation
 - Workflow d'implémentation : choisir des noms de variables qui facilite la compréhension
 - Documentation doit être complète, juste et refléter la version courante de tous les artefacts
- Ne pas compromettre la maintenabilité durant la maintenance

Problème de cible mobile

Solution apparente

- **Geler les exigences** une fois qu'elles ont été signées, jusqu'à la livraison du produit
- Après chaque requête de maintenance perfective, **geler les exigences** pendant (disons) 3 mois ou 1 an

En pratique

- Le client peut ordonner des changements le lendemain
- Tant qu'il paye la facture, il peut faire des demandes de changement quotidiennement
- Des **changements fréquents ont un effet nocif** sur la maintenabilité du produit
 - × **Plus il sera difficile d'intégrer de nouveaux changements**
 - × **Moins la documentation est fiable**
 - × **Moins les tests de régression sont à jour**

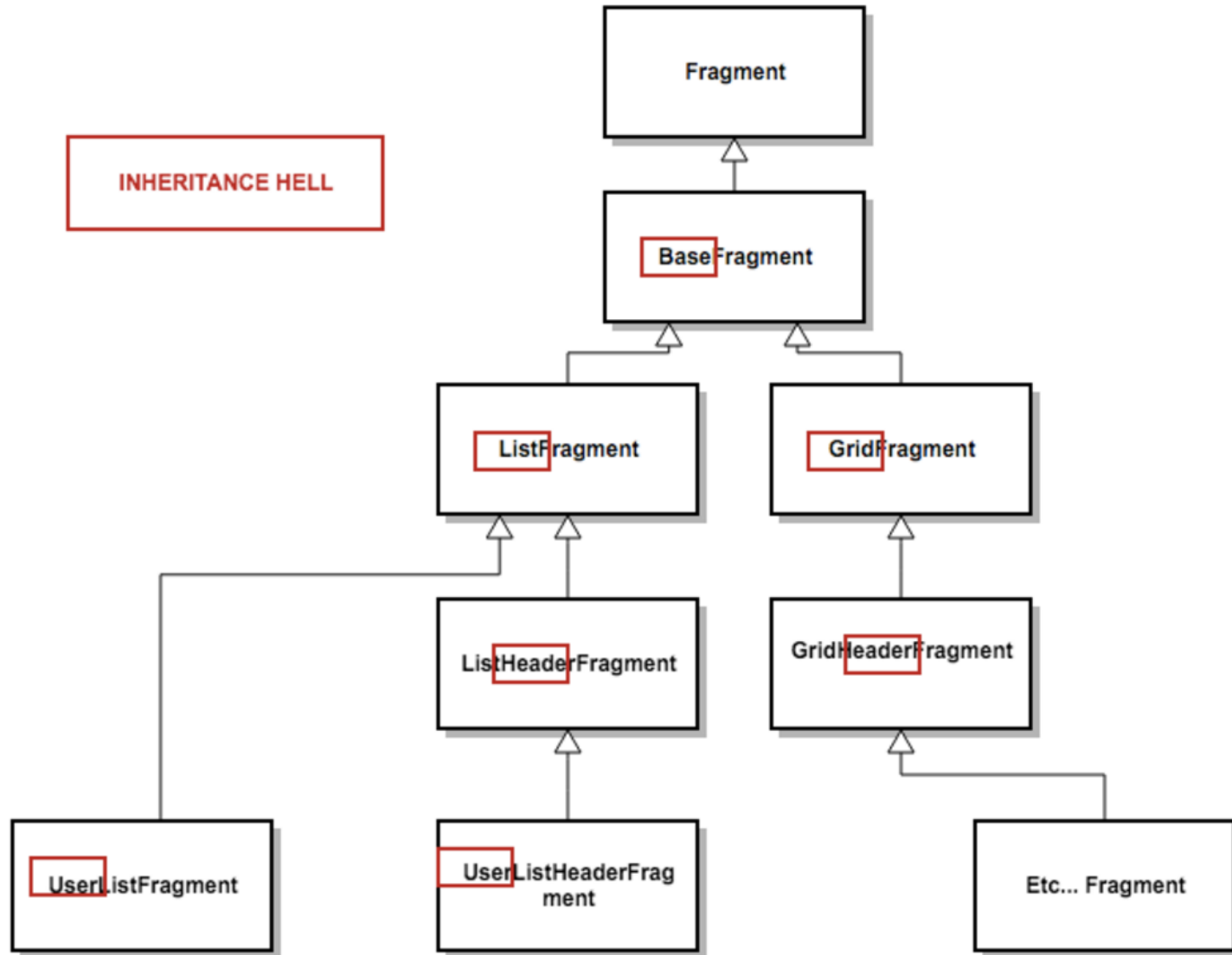
Maintenance et OO

Le paradigme OO facilite la maintenance

- ✓ Application composée **d'unités indépendantes**
- ✓ **Encapsulation et Dissimulation**
- ✓ Seul moyen de communication est via l'envoi de **messages**

...mais il l'entrave aussi

- × Trop grande **hiérarchie** d'héritage
- × Conséquences du **polymorphisme** et de liaisons dynamiques
- × **Fragilité** de l'héritage. Modifier une superclasse => **Toutes les sous-classes sont affectées**



404 error

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque non malesuada massa.

Cras in placerat felis. Duis metus orci, varius nec velit sed, aliquet laoreet est. Nulla facilisi. Pellentesque elit nisi, congue ut rutrum vel, posuere eu lectus.

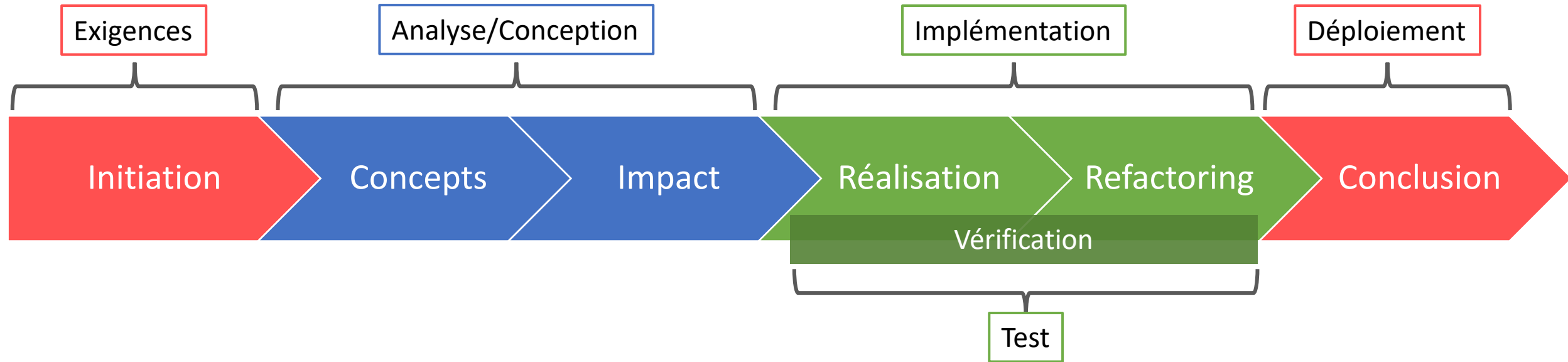
[more](#)

Flux de maintenance



Adapter le système à
de nouvelles exigences
post-déploiement

Changement post-déploiement (étapes)



Initiation

- Débute par une demande de changement
 - Exigence du changement
- Priorisés par sévérité
- Programmeur décide d'implémenter le changement
- S'auto-assigne la demande ou assignée par le gestionnaire du projet
- Une fois le bogue assigné, le programmeur doit
 - Identifier la cause
 - Trouver une solution pour le corriger
 - Trouver un moyen de contourner le problème

Logiciels de suivi de défauts

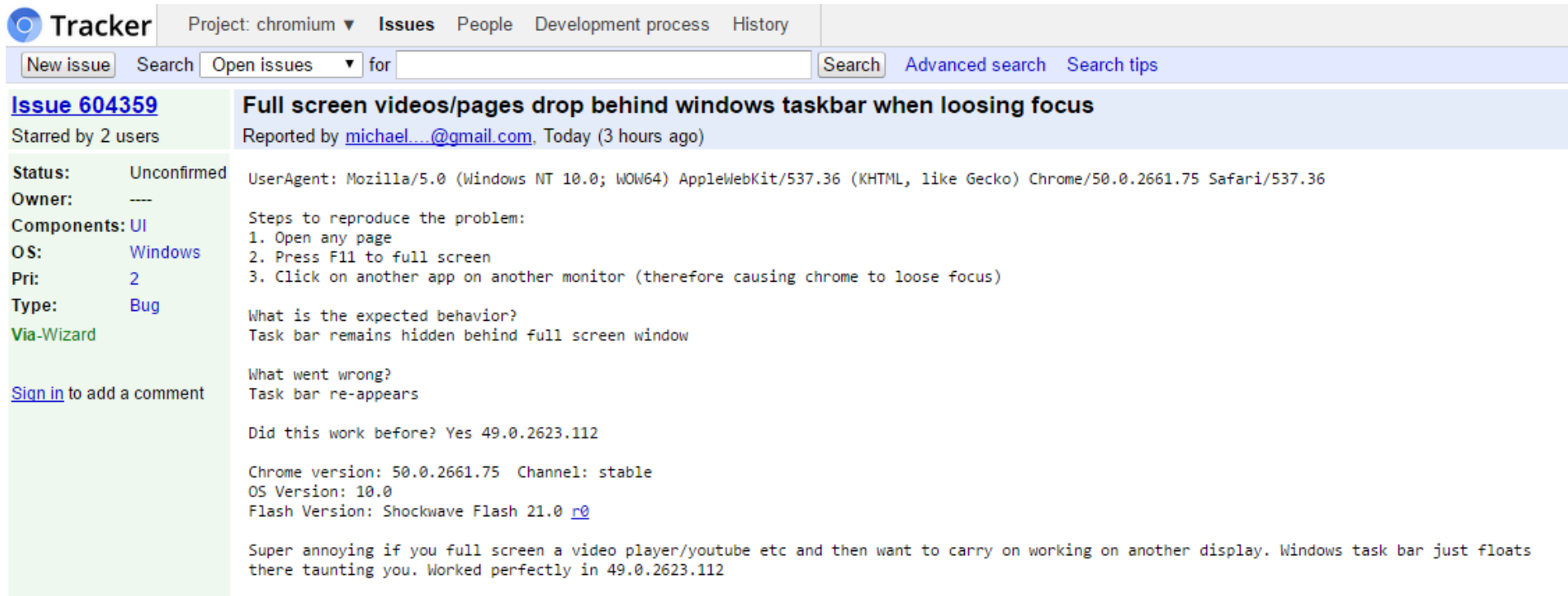
- ❖ Bugzilla
- ❖ Backlog
- ❖ BugHerd

The screenshot displays the Databug web interface. At the top, there's a navigation bar with 'Return to Dashboard', a search bar, and user information 'Working On | John Doe | Help & Feedback'. Below this is a main menu with 'OVERVIEW', 'TODO', 'MILESTONES', 'TEST RUNS & RESULTS', 'TEST CASES', 'REPORTS', and 'ADMINISTRATION'. The main content area shows a test run for 'Release 1.1: Run 1 (smoke test) > Installation' with a status of 'Failed'. A table lists test results with columns for 'Type', 'Priority', and 'Estimate'. Below the table, there are 'Steps' to verify the CSV import functionality, including instructions to open the Import dialog, select the file format (CSV), and import the test file. A 'Push Defect' dialog box is open, allowing the user to create a new bug report. The dialog includes fields for 'Title', 'Project', 'Area', and 'Priority', and a 'Comment' field. The 'Submit' button is highlighted in green. At the bottom of the screenshot, a browser window shows a Bugzilla bug report for 'Bug 39884 - cee27995-8985-4cfe-aa54-a9813cec9e0f'. The report details include 'Status: CONFIRMED', 'Product: MyOwnBadSelf', 'Component: comp2', and 'Importance: P2 - normal'. The bug report also shows a 'Reported' date of 2017-02-19 12:40 PST and a 'Modified' date of 2017-02-19 12:40 PST. A 'Save Changes' button is visible at the top right of the bug report window.



Rapport de défauts

- L'utilisateur doit disposer d'un mécanisme pour rapporter les défauts trouvés
- Doit inclure assez d'information pour permettre au programmeur de maintenance de recréer le problème pour l'investiguer



The screenshot shows a bug report on the Chromium Tracker website. The page title is "Full screen videos/pages drop behind windows taskbar when loosing focus". The report was submitted by michael...@gmail.com today (3 hours ago). The status is "Unconfirmed" and the type is "Bug". The components affected are "UI" and "Windows". The OS is "Windows" and the priority is "2". The user agent string is "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/50.0.2661.75 Safari/537.36". The steps to reproduce the problem are: 1. Open any page, 2. Press F11 to full screen, 3. Click on another app on another monitor (therefore causing chrome to loose focus). The expected behavior is that the task bar remains hidden behind the full screen window. The actual behavior is that the task bar re-appears. The report also includes information about the Chrome version (50.0.2661.75), OS version (10.0), and Flash version (Shockwave Flash 21.0 r0). The reporter notes that the issue is super annoying when full screening a video player or YouTube and then wanting to carry on working on another display, as the Windows task bar just floats there taunting you. The issue worked perfectly in version 49.0.2623.112.

Tracker Project: chromium ▾ **Issues** People Development process History

New issue Search Open issues ▾ for Search Advanced search Search tips

Issue 604359 Full screen videos/pages drop behind windows taskbar when loosing focus

Starred by 2 users Reported by [michael...@gmail.com](#), Today (3 hours ago)

Status: Unconfirmed
Owner: ---
Components: UI
OS: Windows
Pri: 2
Type: Bug
[Via-Wizard](#)

[Sign in](#) to add a comment

UserAgent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/50.0.2661.75 Safari/537.36

Steps to reproduce the problem:

1. Open any page
2. Press F11 to full screen
3. Click on another app on another monitor (therefore causing chrome to loose focus)

What is the expected behavior?
Task bar remains hidden behind full screen window

What went wrong?
Task bar re-appears

Did this work before? Yes 49.0.2623.112

Chrome version: 50.0.2661.75 Channel: stable
OS Version: 10.0
Flash Version: Shockwave Flash 21.0 [r0](#)

Super annoying if you full screen a video player/youtube etc and then want to carry on working on another display. Windows task bar just floats there taunting you. Worked perfectly in 49.0.2623.112

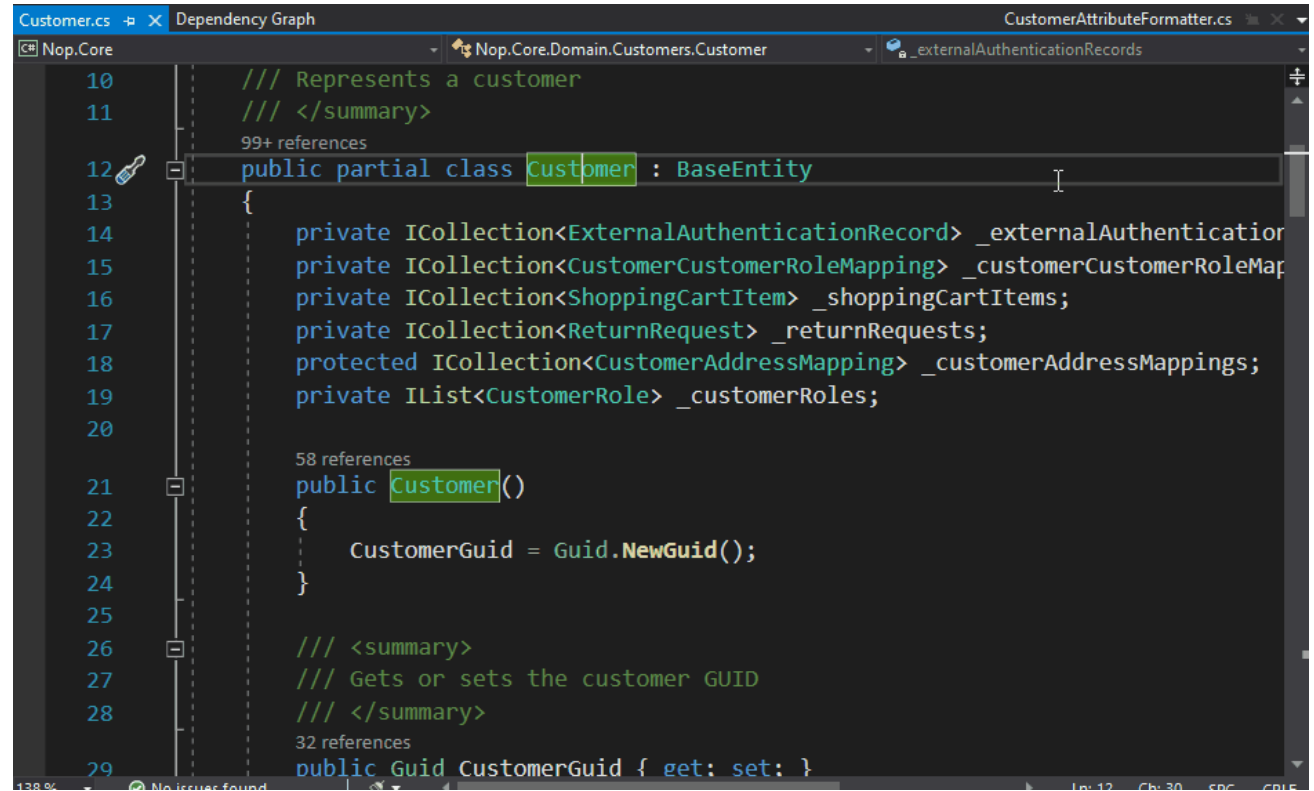
Déterminer le type de changement

- **Incrémental** : ajoute une nouvelle fonctionnalité (*perfectif*)
- **Contraction** : retire une fonctionnalité obsolète d'un logiciel (*perfectif*)
 - Permet de réduire la complexité du logiciel
- **Remplacement** : remplace une fonctionnalité existante par une autre (*adaptatif/correctif*)
 - ex: correction de bogue, amélioration de la performance
- **Réusinage (*refactoring*)** : substitue une structure par une autre sans modifier le comportement observable (*préventif*)
 - Permet de limiter la dégradation d'un logiciel due aux changements

Localisation des concepts

But: identifier l'emplacement du code qui doit subir des changements

Changements souvent formulés en termes de concepts du domaine



```
10  /// Represents a customer
11  /// </summary>
12  99+ references
13  public partial class Customer : BaseEntity
14  {
15      private ICollection<ExternalAuthenticationRecord> _externalAuthenticati
16      private ICollection<CustomerCustomerRoleMapping> _customerCustomerRoleMap
17      private ICollection<ShoppingCartItem> _shoppingCartItems;
18      private ICollection<ReturnRequest> _returnRequests;
19      protected ICollection<CustomerAddressMapping> _customerAddressMappings;
20      private IList<CustomerRole> _customerRoles;
21
22      58 references
23      public Customer()
24      {
25          CustomerGuid = Guid.NewGuid();
26      }
27
28      /// <summary>
29      /// Gets or sets the customer GUID
30      /// </summary>
31      32 references
32      public Guid CustomerGuid { get; set; }
```

Initiation

Concepts

Impact

Réalisation

Refactoring

Conclusion

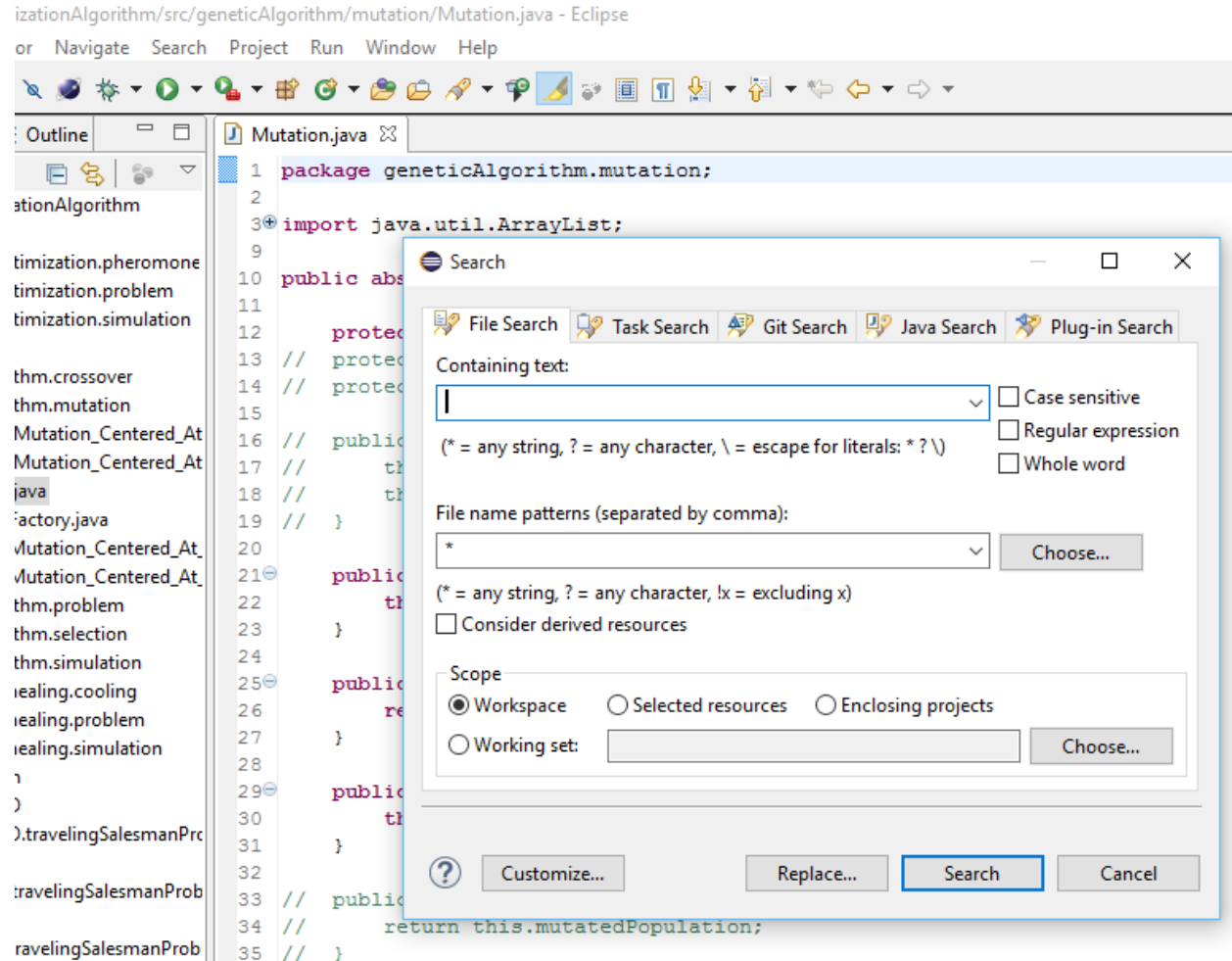
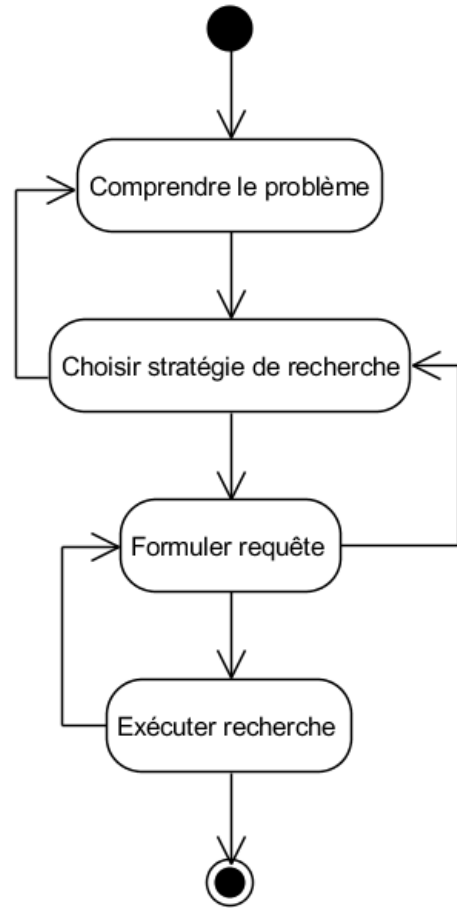
Localisation des concepts

- Concepts extraits de la requête de changement
- Repérés dans le code et deviennent le point de départ pour les changements à effectuer
- On recherche:
 - L'extrait de code à changer
 - Le module où la fonctionnalité/correction se situera
 - La partie de la documentation qui décrit ces concepts

Compréhension partielle du code

- Un très grand programme ne peut être complètement compris
- Un programmeur
 - Cherche à obtenir une compréhension minimale qui permet d'effectuer un changement
 - Ne veut explorer une partie du code que lorsque c'est nécessaire
 - Cherche à comprendre **comment les concepts sont reflétés dans le code**

Recherche textuelle dans du code inconnu



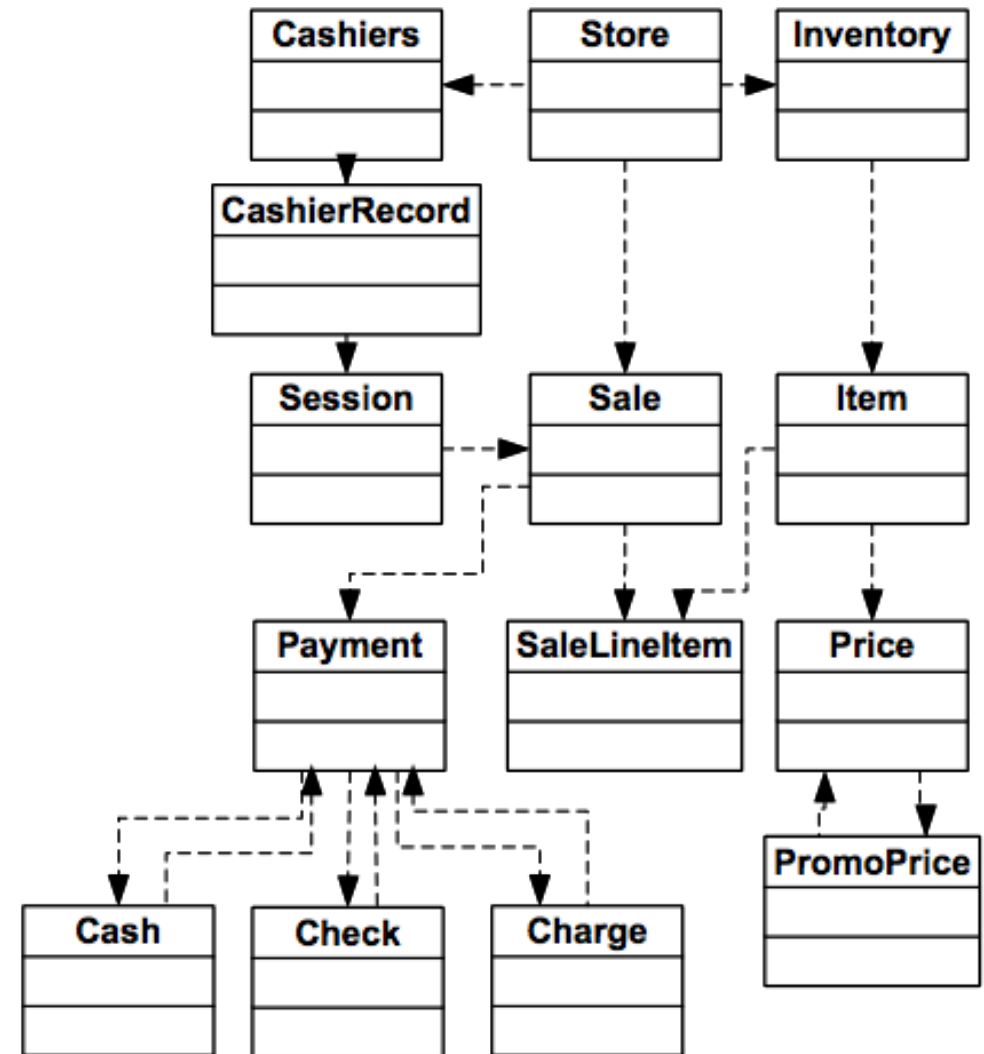
`grep "payment" *.java`

Recherche par dépendances

- Utilise la structure des classes pour guider la navigation et localiser un concept
- Chaque classe joue un **rôle** dans le système, qui correspond à sa **responsabilité**
 - Principe de responsabilité unique en programmation OO
- Ex: la classe `Item` d'un système de POS est responsable des items vendus par un commerce et leurs propriétés
 - Nouveaux items, description des items, etc.

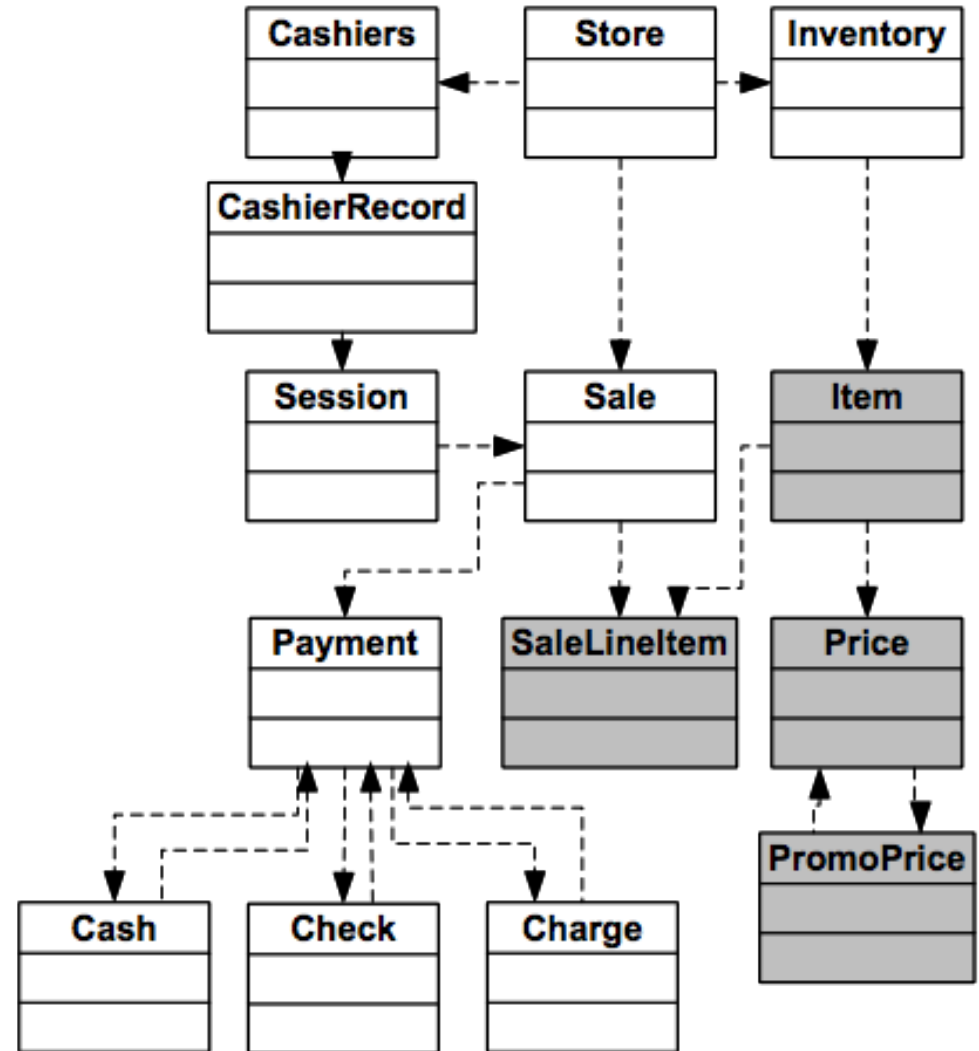
Graphe de dépendances

- L'ensemble des dépendances entre les classes forme un graphe dirigé
- Nœuds : classes
- Arcs : dépendances
 - Associations
 - Héritage (bidirectionnel dans le cas de l'héritage polymorphique)
 - Utilisations
 - Autres dépendances possibles

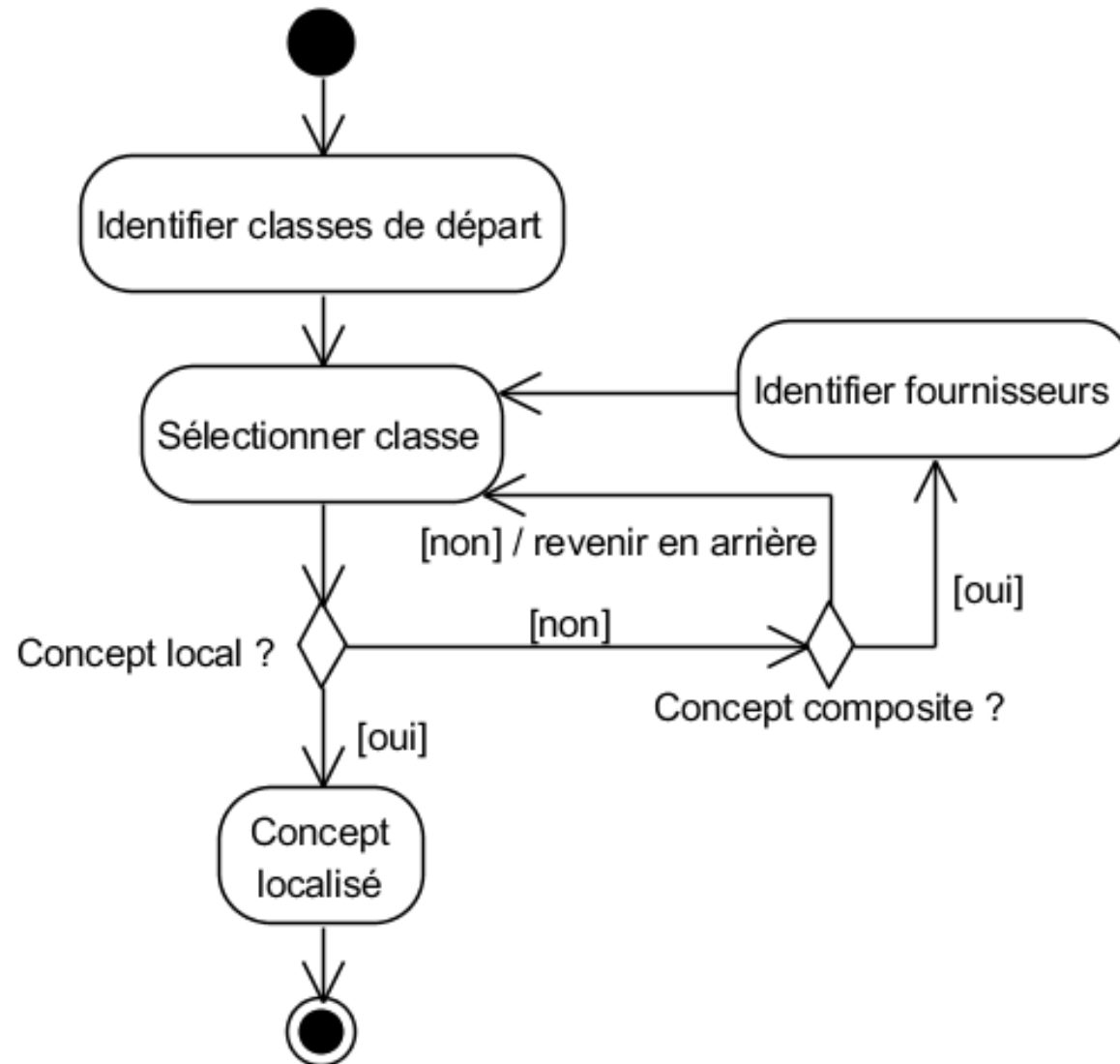


Tranche de fournisseurs

- Responsabilités **locales** : assumées par la classe seule
- Responsabilités collectives / **composites** : assumée par la tranche de fournisseurs d'une classe
 - Permet à la classe d'implémenter des fonctionnalités qu'elle ne peut assumer complètement seule



Recherche par dépendances



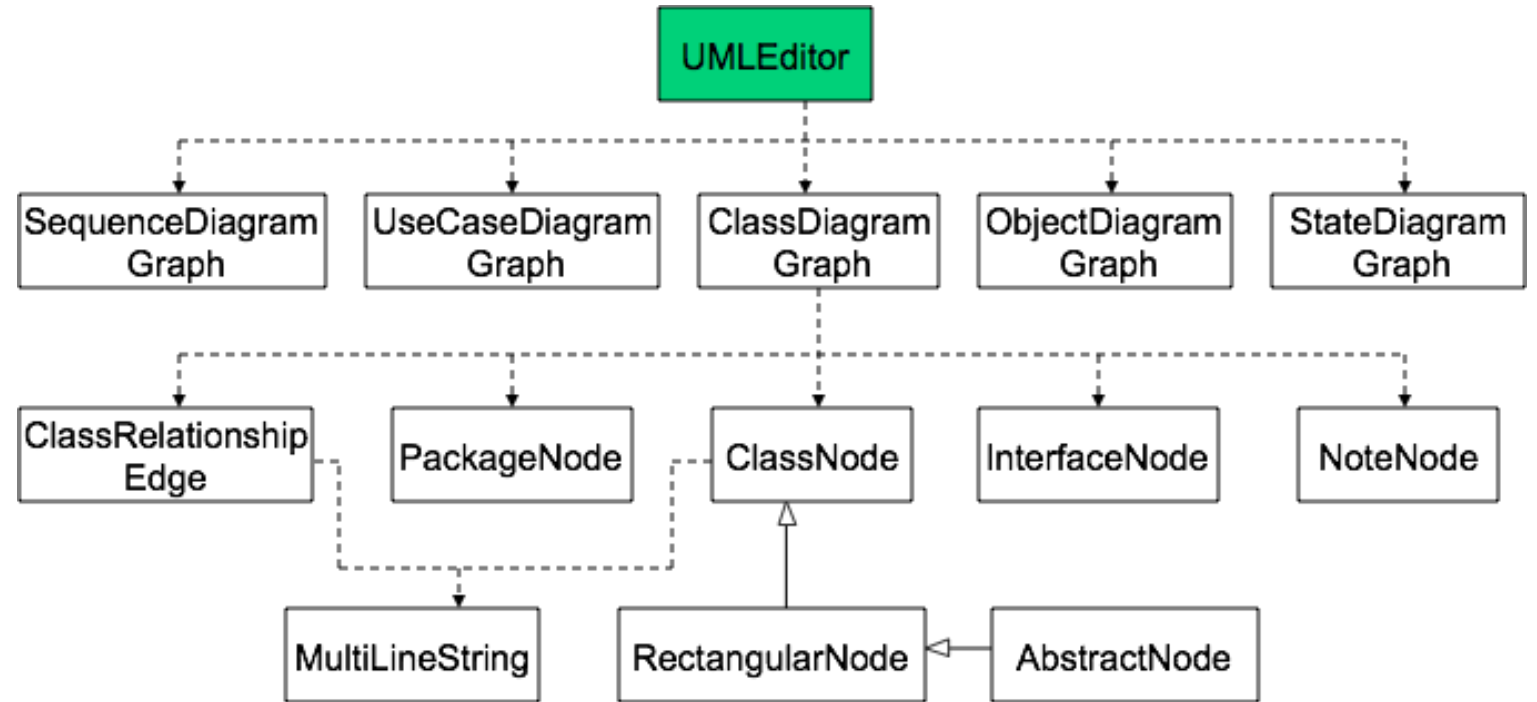
Exemple

Requête de changement :

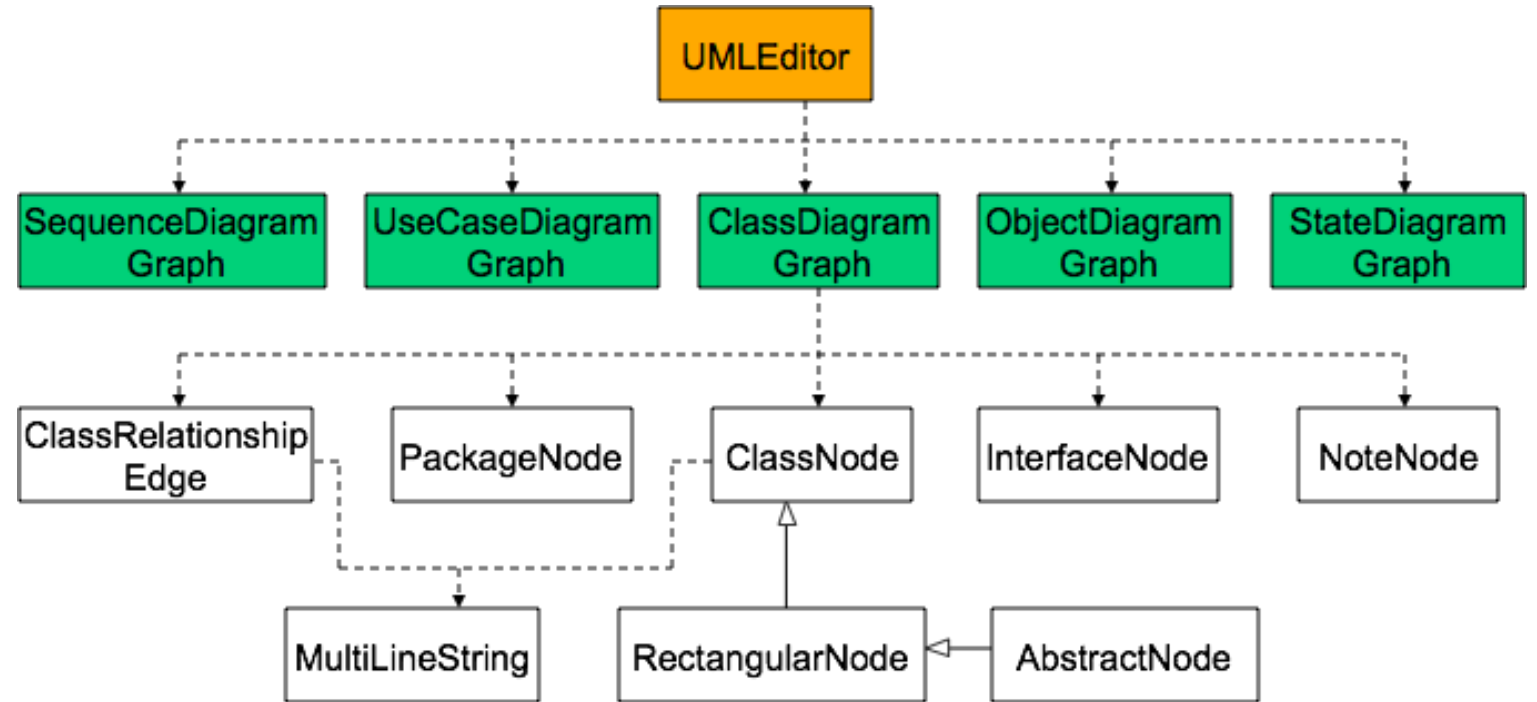
« Ajouter un auteur à chaque classe créée dans un diagramme de classes »

- Concept : « auteur »
 - Concept nouveau, n'apparaît pas dans le code
 - Correspond à une propriété d'un nœud

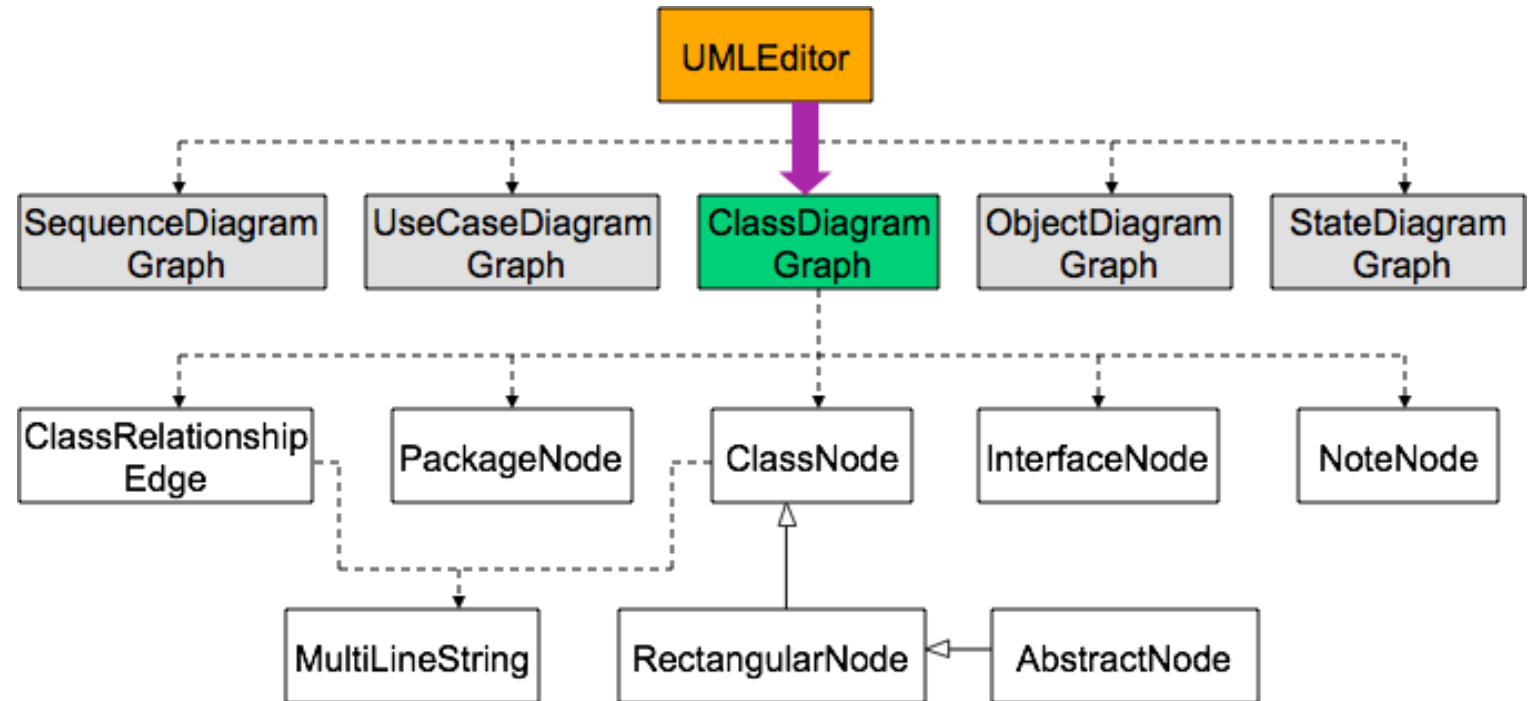
Localiser les propriétés d'une classe



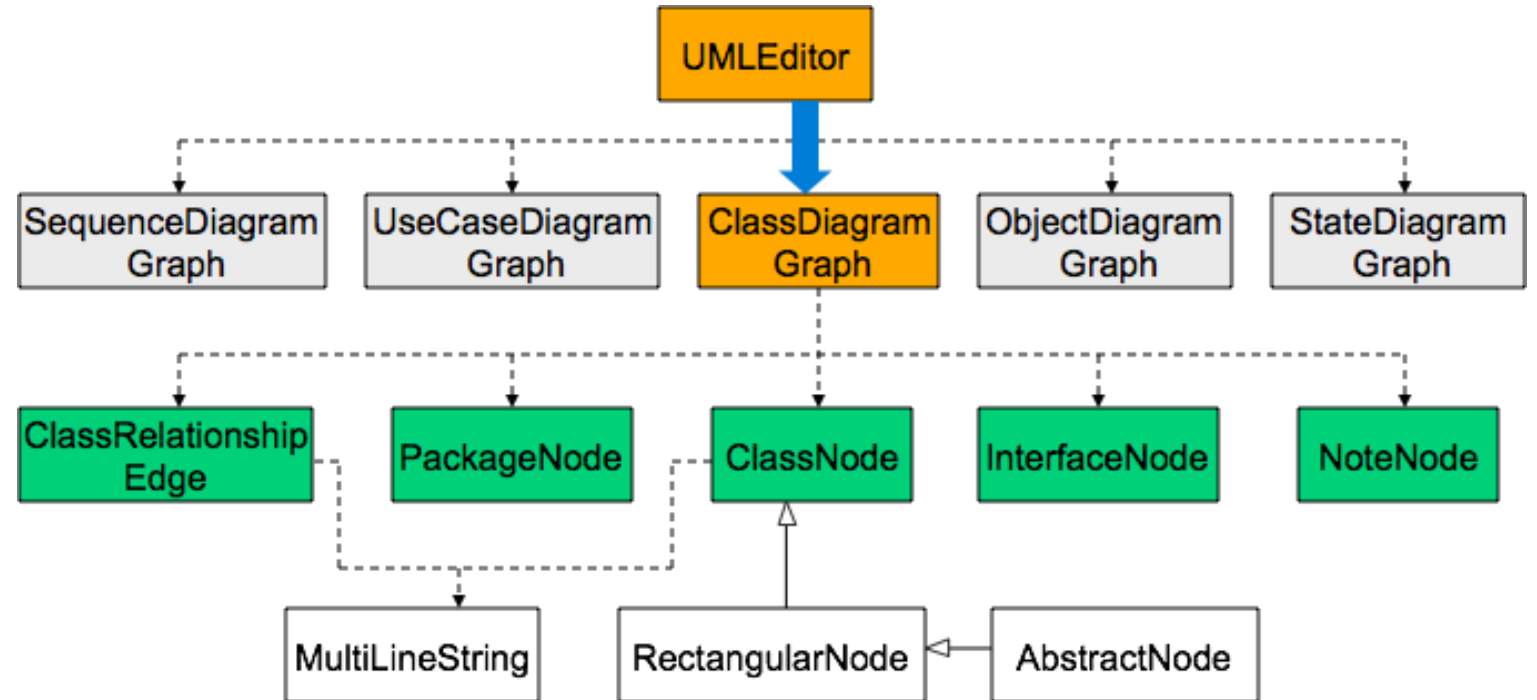
Classes à inspecter



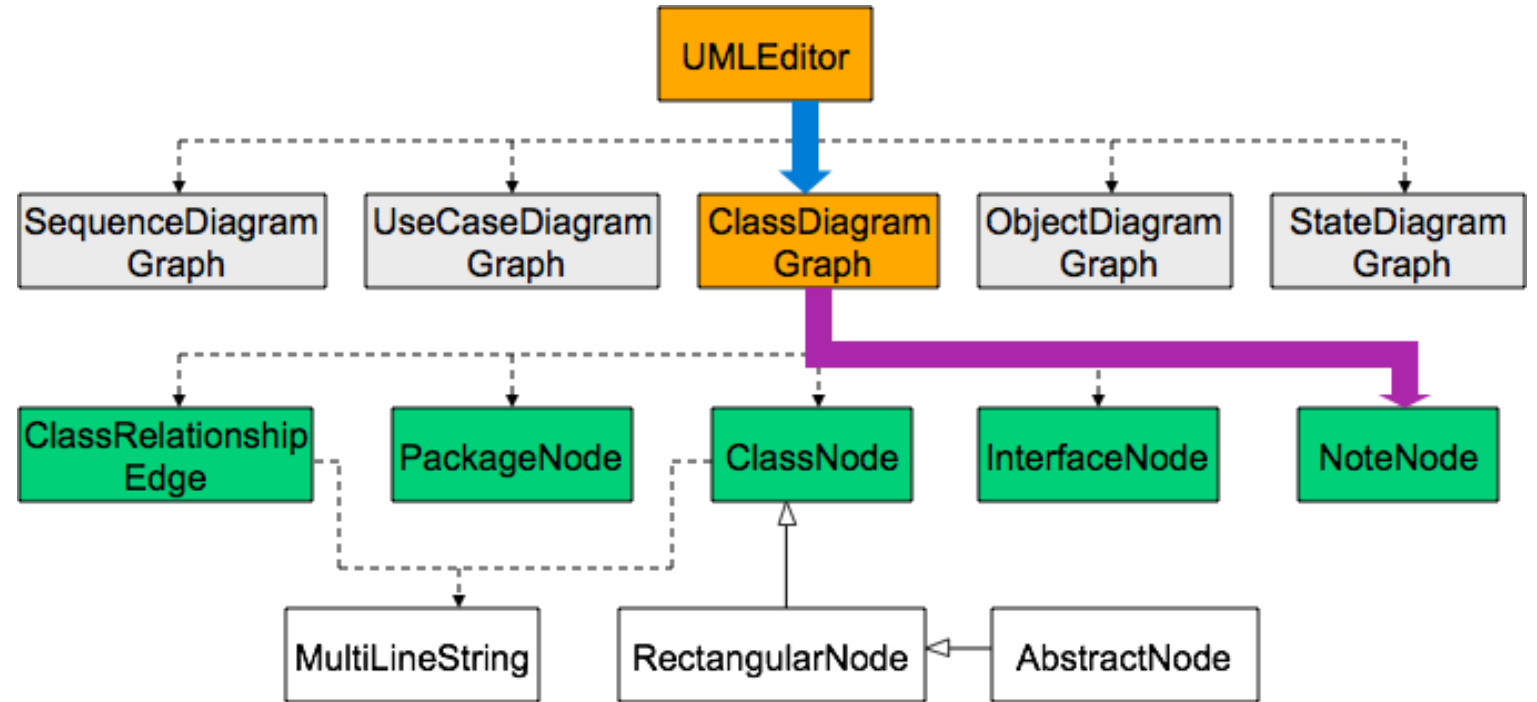
Fournisseur
le plus
probable



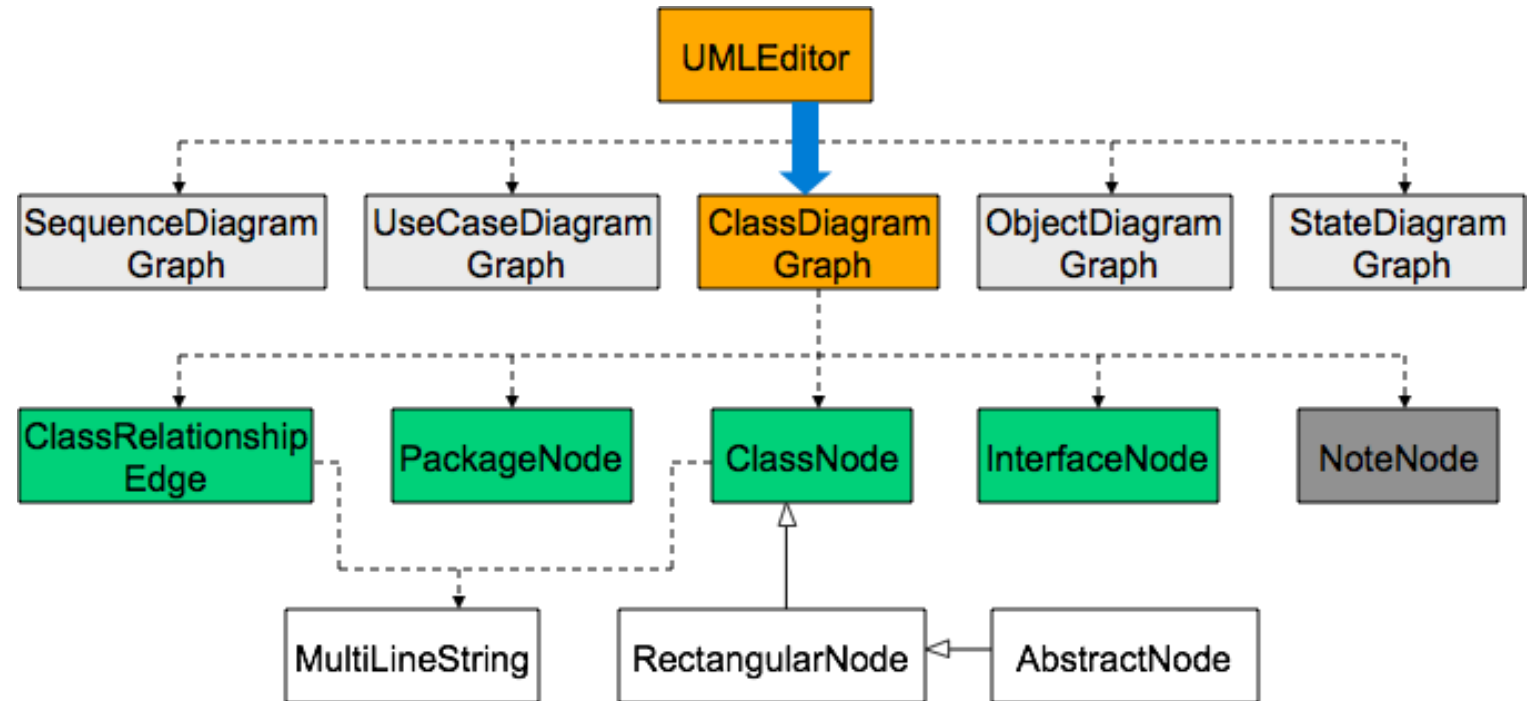
Prochaines
classes à
inspecter



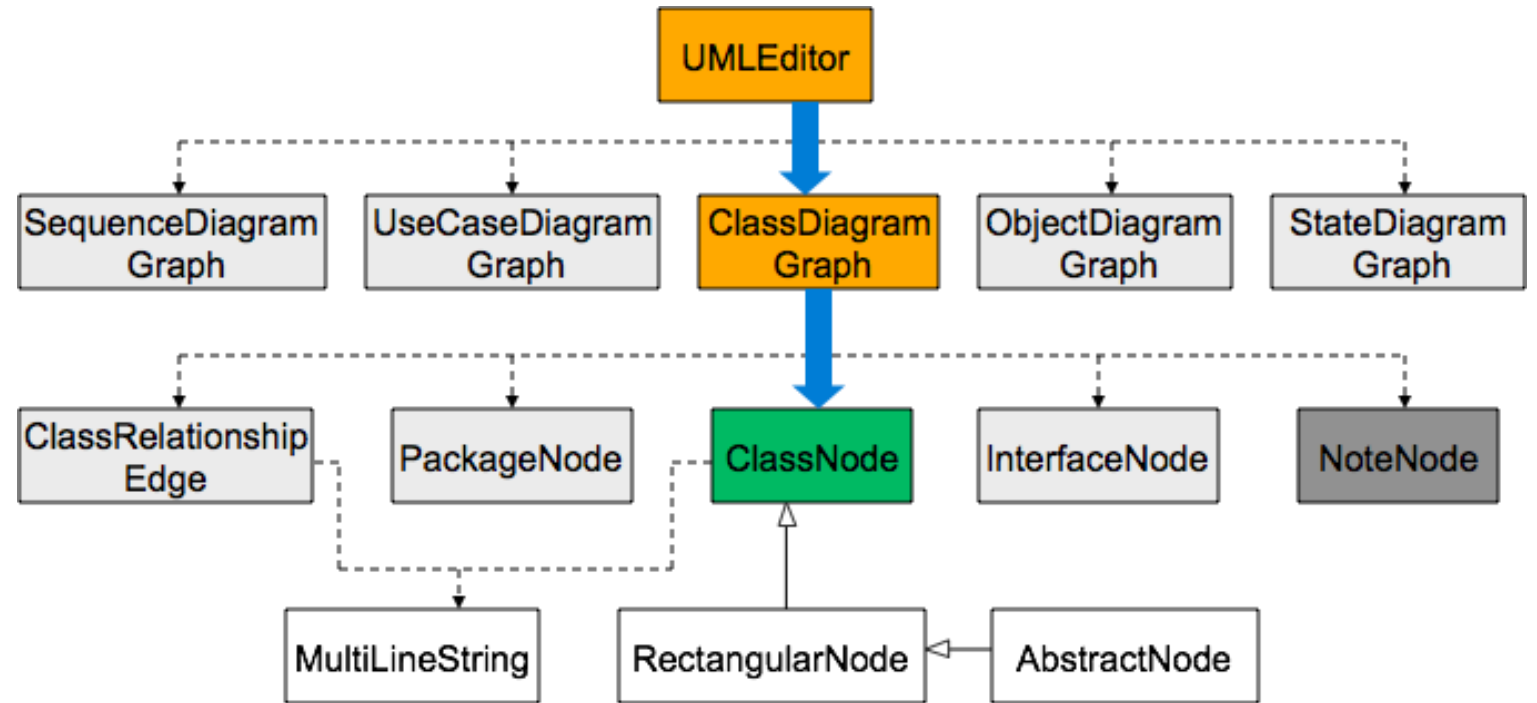
Mauvais choix...



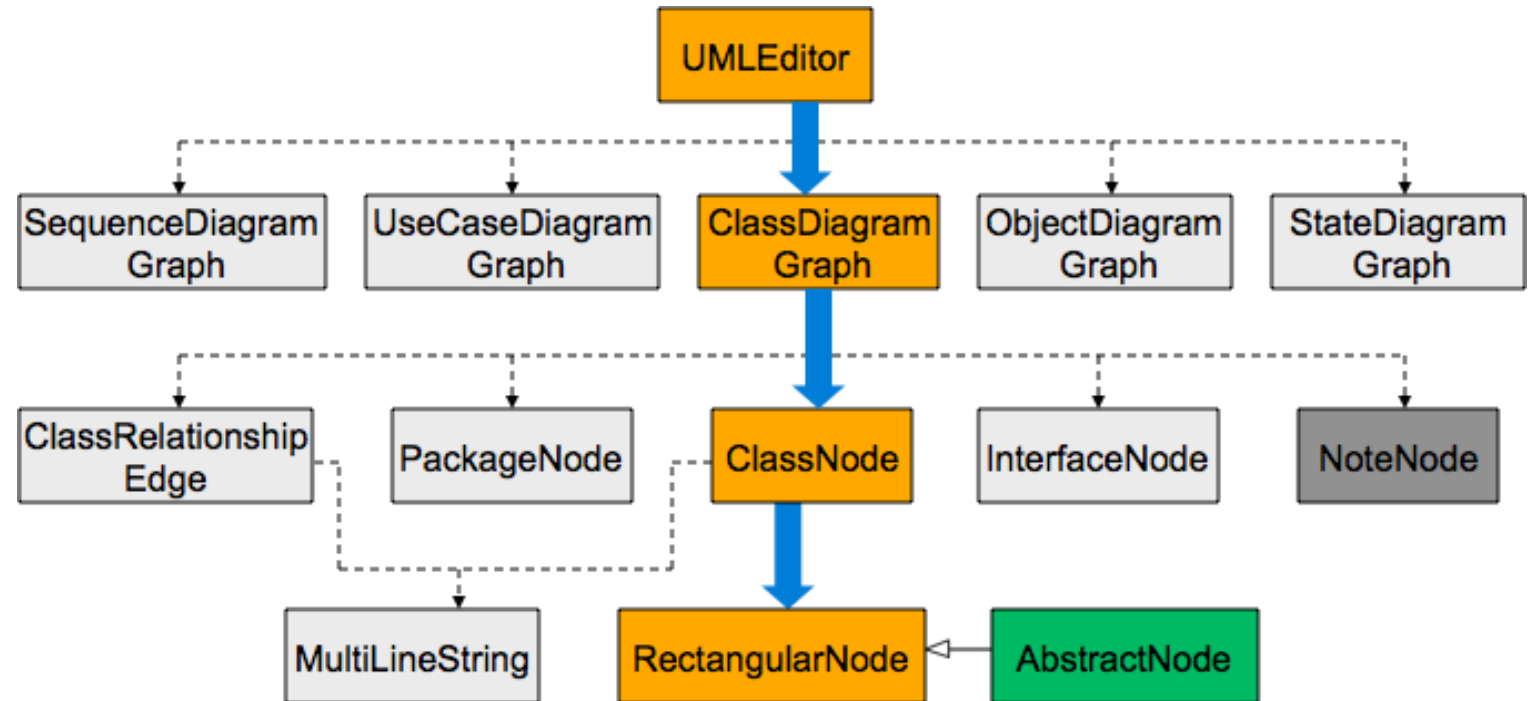
Retour en
arrière
(*backtrack*)



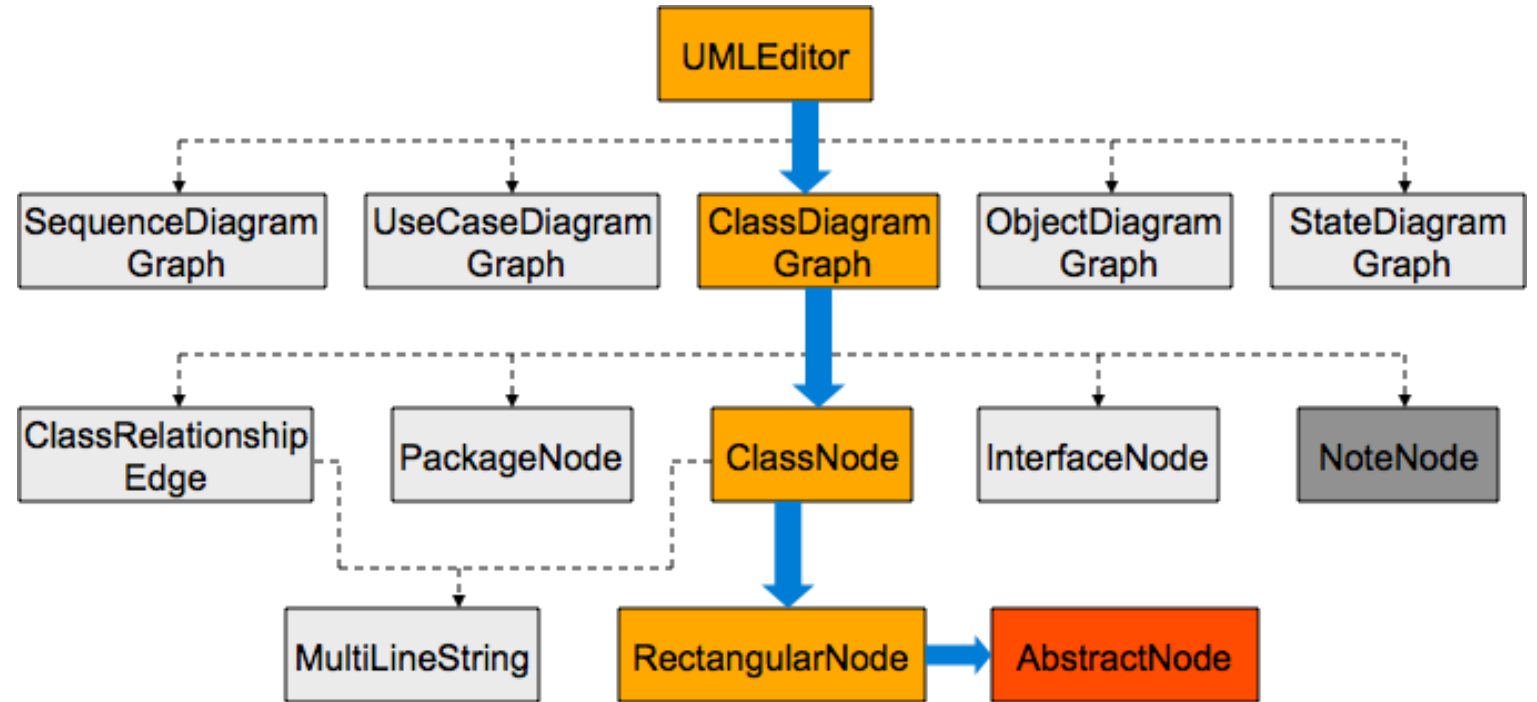
Concept
localisé



Extensions possible de la recherche



Autre emplacement trouvé



Recherche textuelle vs. par dépendance

Recherche textuelle

- Dépend des conventions de nommage
- Indépendant de la structure du programme
- Applicable à des concepts explicites seulement

Graphe de dépendances

- Utilise la structure des classes du programme
- Nécessite la compréhension des fonctionnalités locales et composites
- Applicable aux concepts implicites et explicites

Analyse de l'impact

Souvent le changement n'est pas appliqué à un seul module, mais peut affecter d'autres parties du logiciel

But : Déterminer la stratégie à utiliser et l'impact des changements à apporter

Initiation

Concepts

Impact

Réalisation

Refactoring

Conclusion

Portée de l'impact

- **Impact localisé** : affecte au plus quelques modules étroitement liés
 - Facile à réaliser, impact facile à prévoir
 - Souvent une conséquence de l'organisation du code (ex: classes faiblement couplées)
- **Impact significatif** : affecte un nombre élevé de modules
 - Plus difficile à prévoir, requiert des outils et techniques plus avancés
- **Impact massif** : affecte la majorité des modules
 - Coûteux et à haut risque \Rightarrow considérer le redéveloppement
- Un changement affecte aussi d'autres artefacts que le code source
 - Documentation, cahier des charges, etc.

Statut des classes

Vide

Classe non-inspectée et qui n'apparaît pas dans la liste de classes à inspecter.

Changée /
Impactée

Un programmeur a inspecté la classe et a déterminé qu'elle était impactée par le changement.

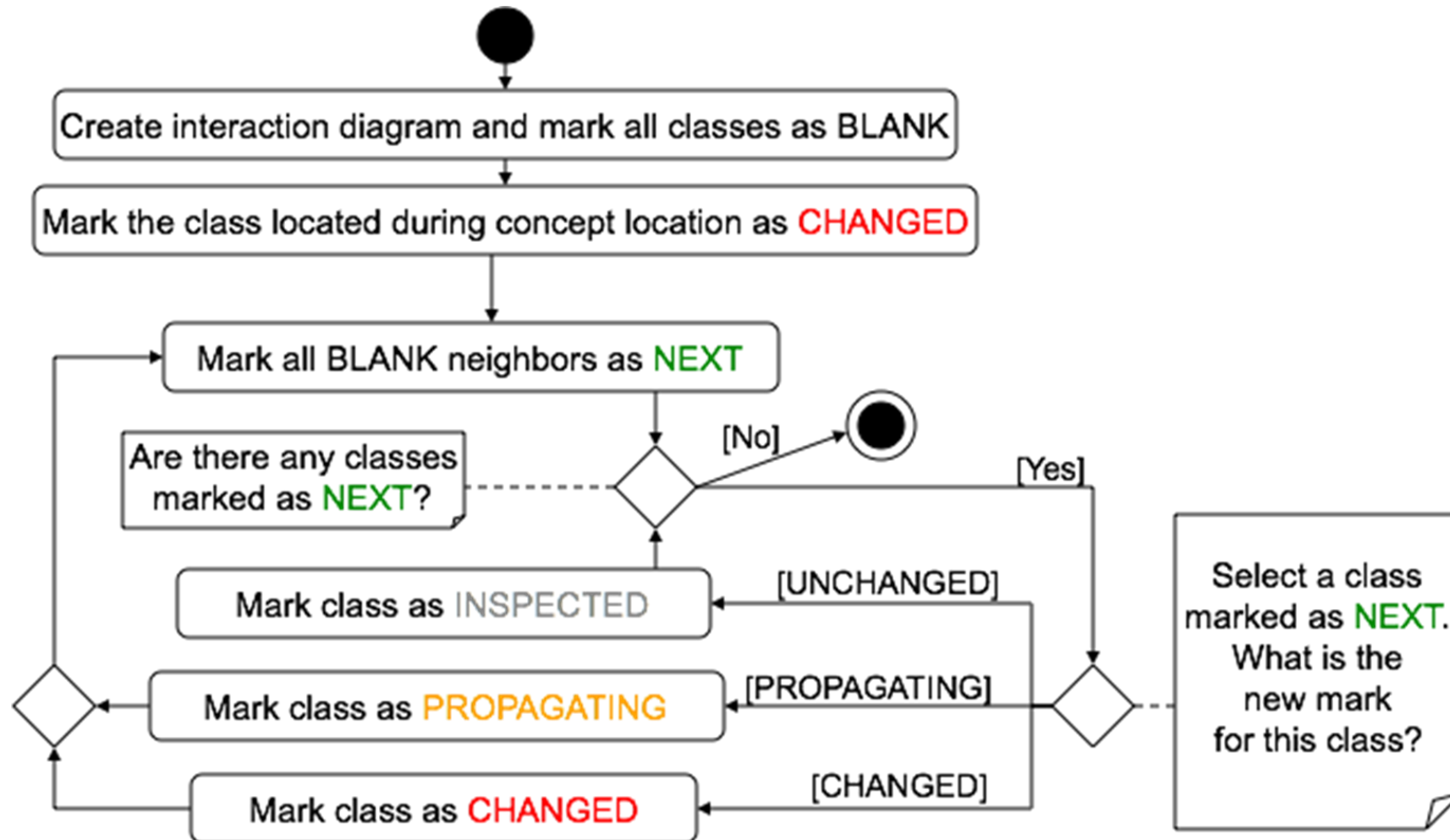
Non-impactée

Un programmeur a inspecté la classe et a déterminé qu'elle n'était pas impactée par le changement.

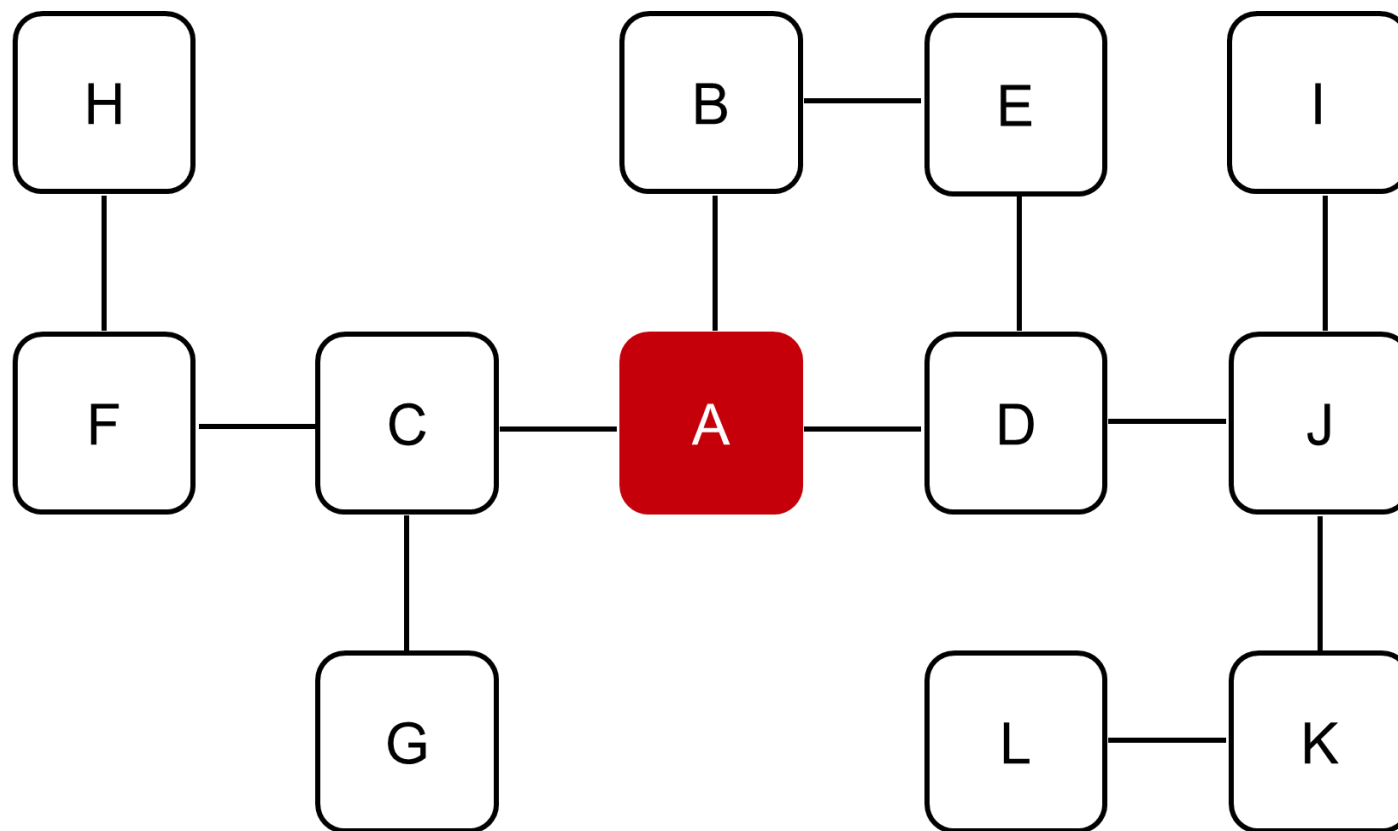
Suivante

La classe apparaît dans la liste de classes à inspecter.

Algorithme d'analyse d'impact itérative

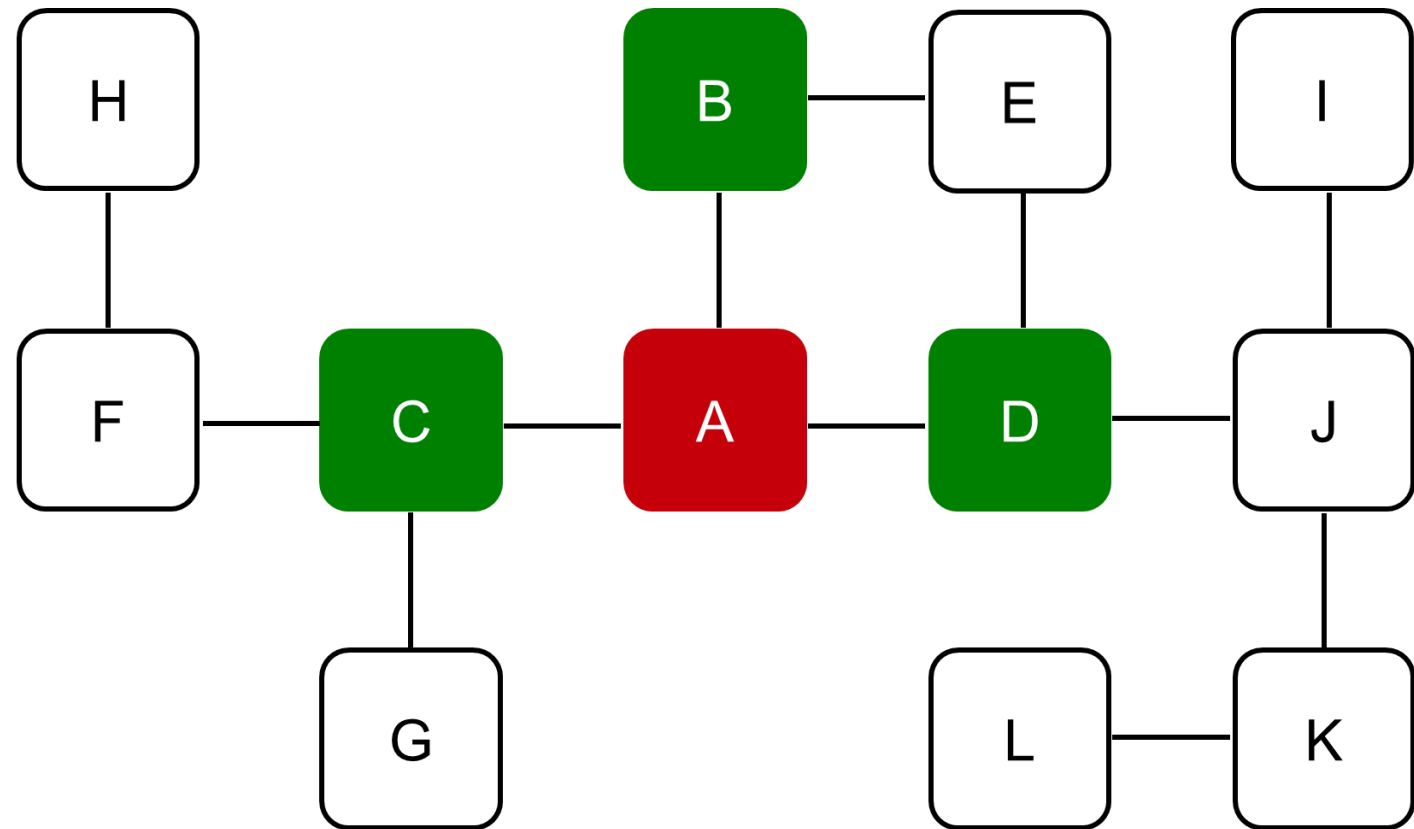


Ensemble
d'impact
initial



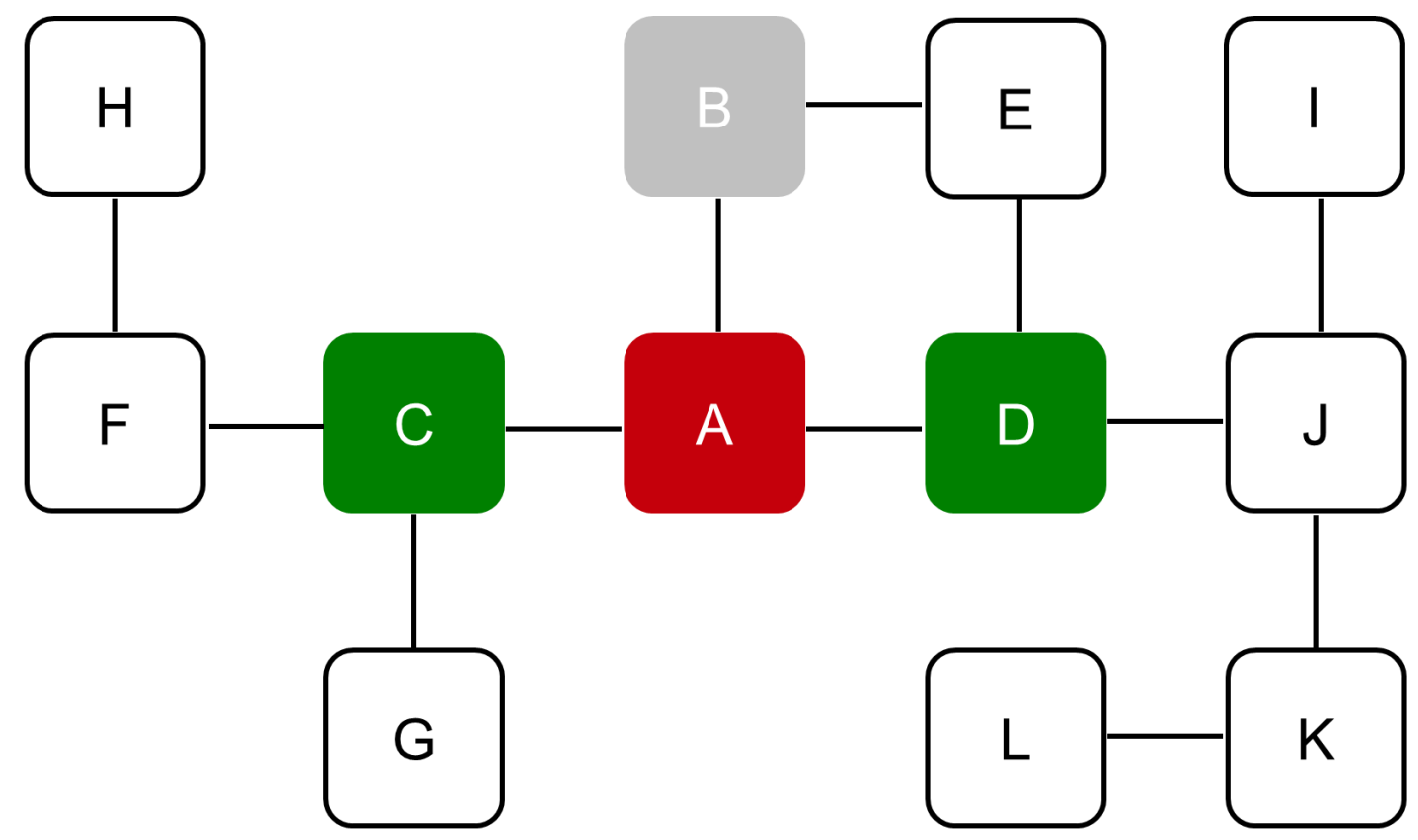
○ Inconnu ● Impacté ● À inspecter ● Non-impacté ● Propage

Inspecter les voisins



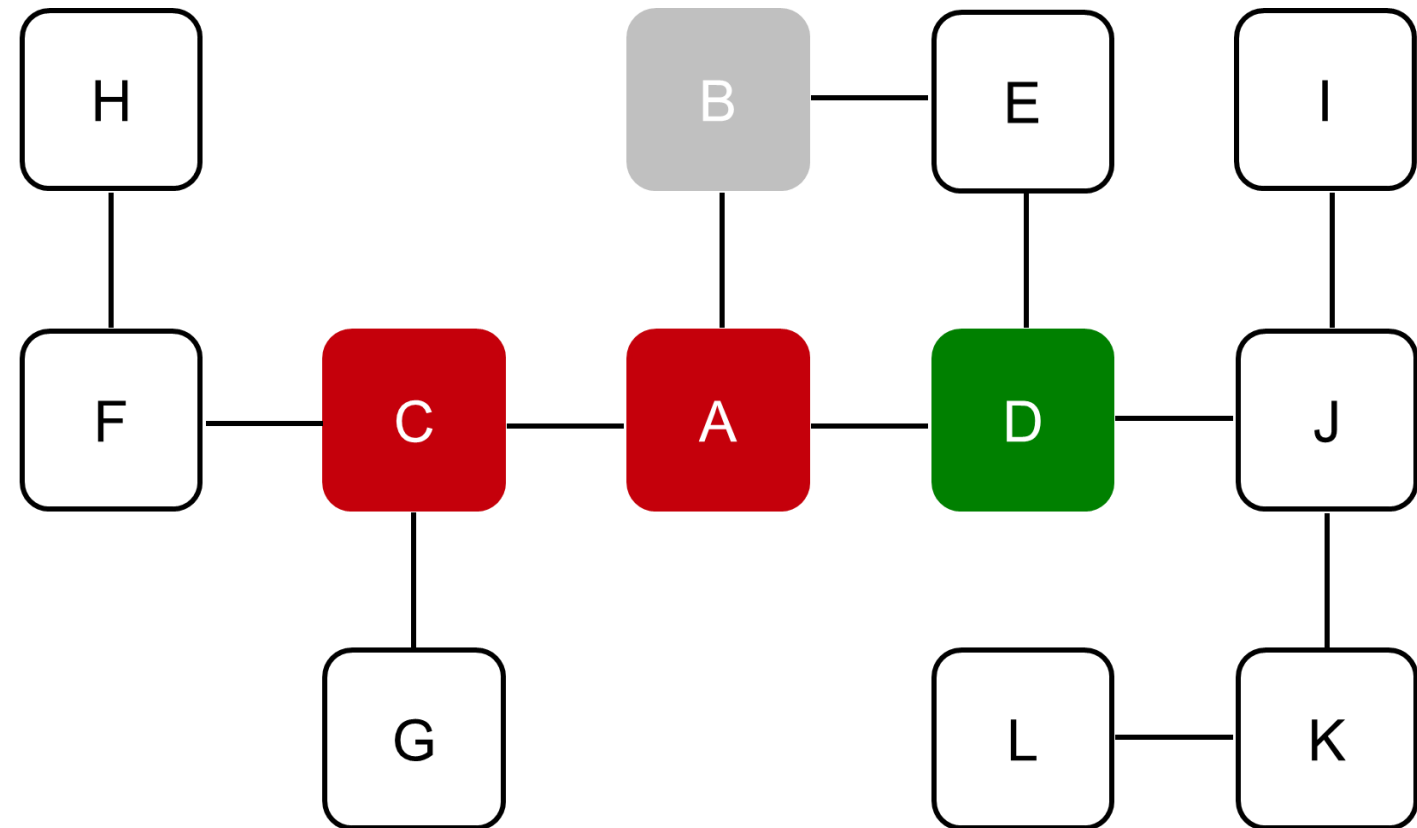
○ Inconnu ● Impacté ● À inspecter ● Non-impacté ● Propage

Non-impacté, au suivant



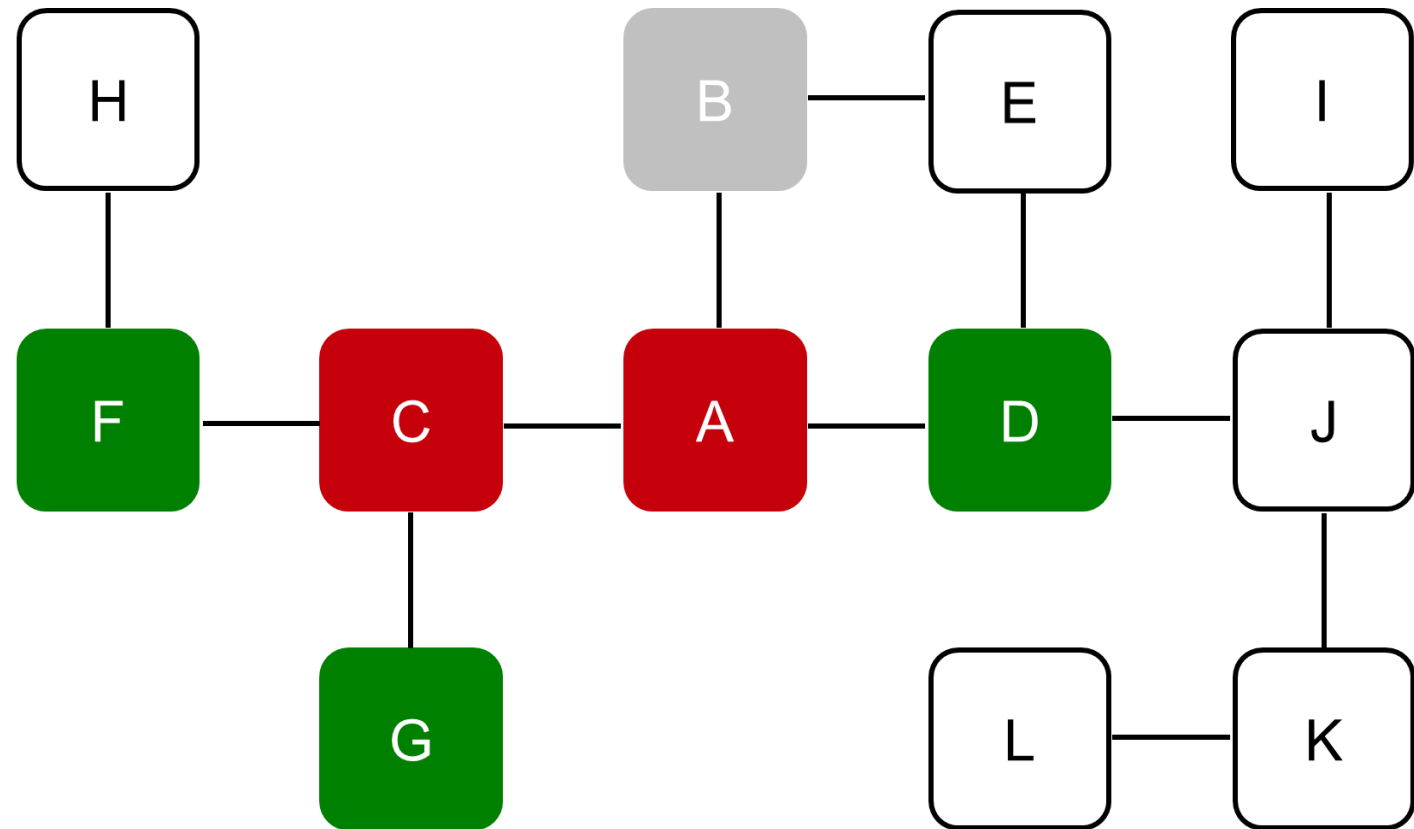
○ Inconnu ● Impacté ● À inspecter ● Non-impacté ● Propage

Voisin impacté,
ajout à
l'ensemble



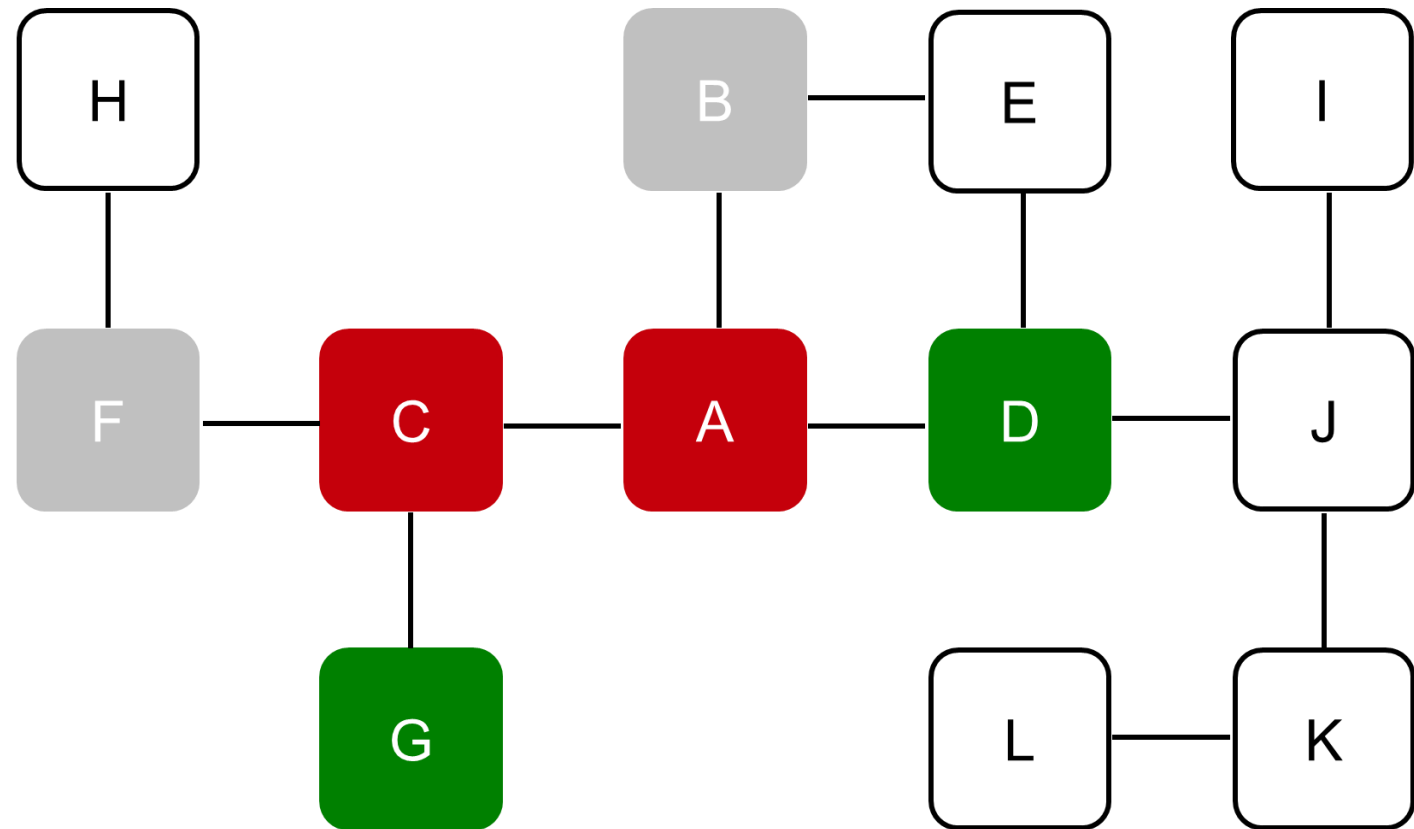
○ Inconnu ● Impacté ● À inspecter ● Non-impacté ● Propage

Inspecter les
nouveaux
voisins



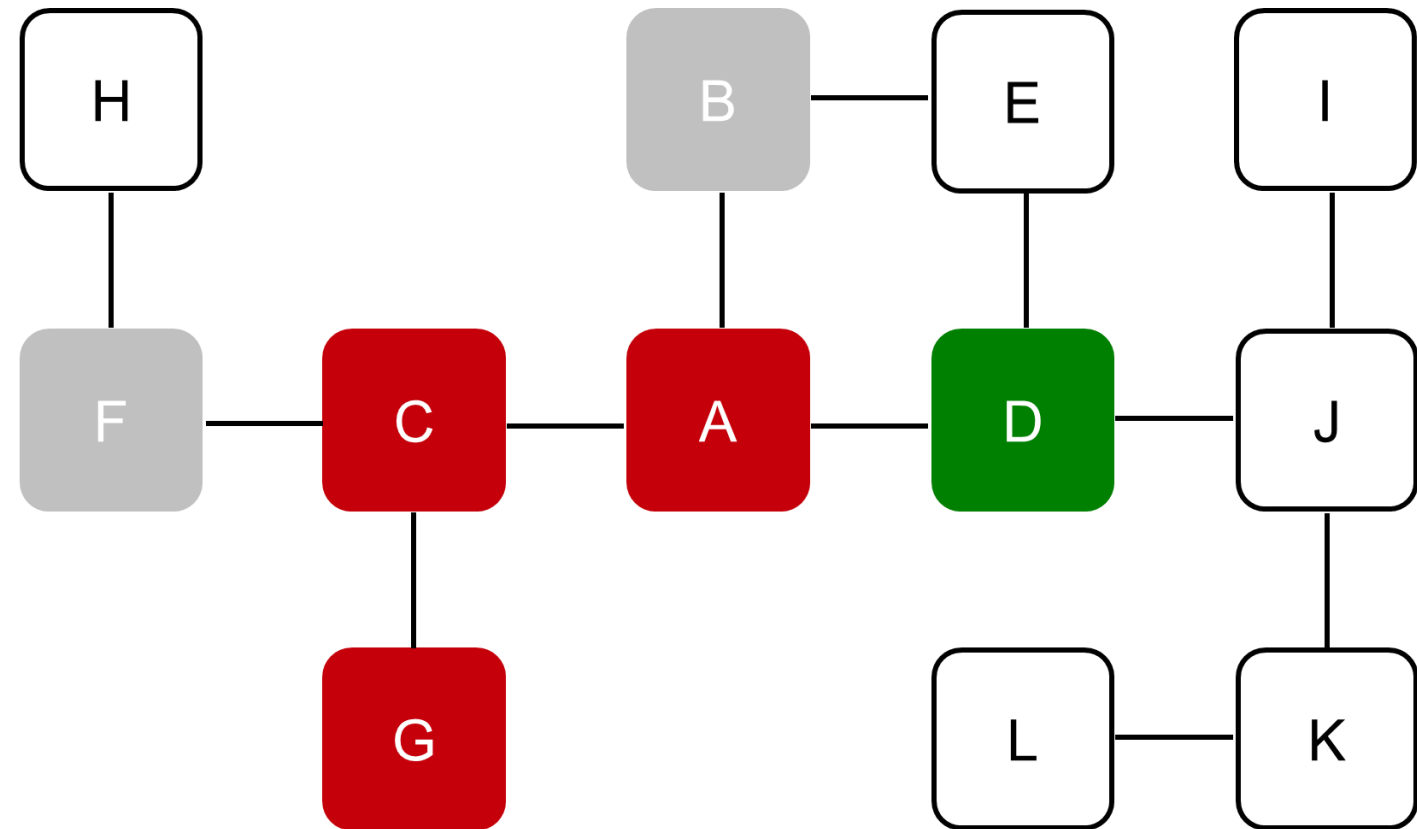
○ Inconnu ● Impacté ● À inspecter ● Non-impacté ● Propage

Non-impacté



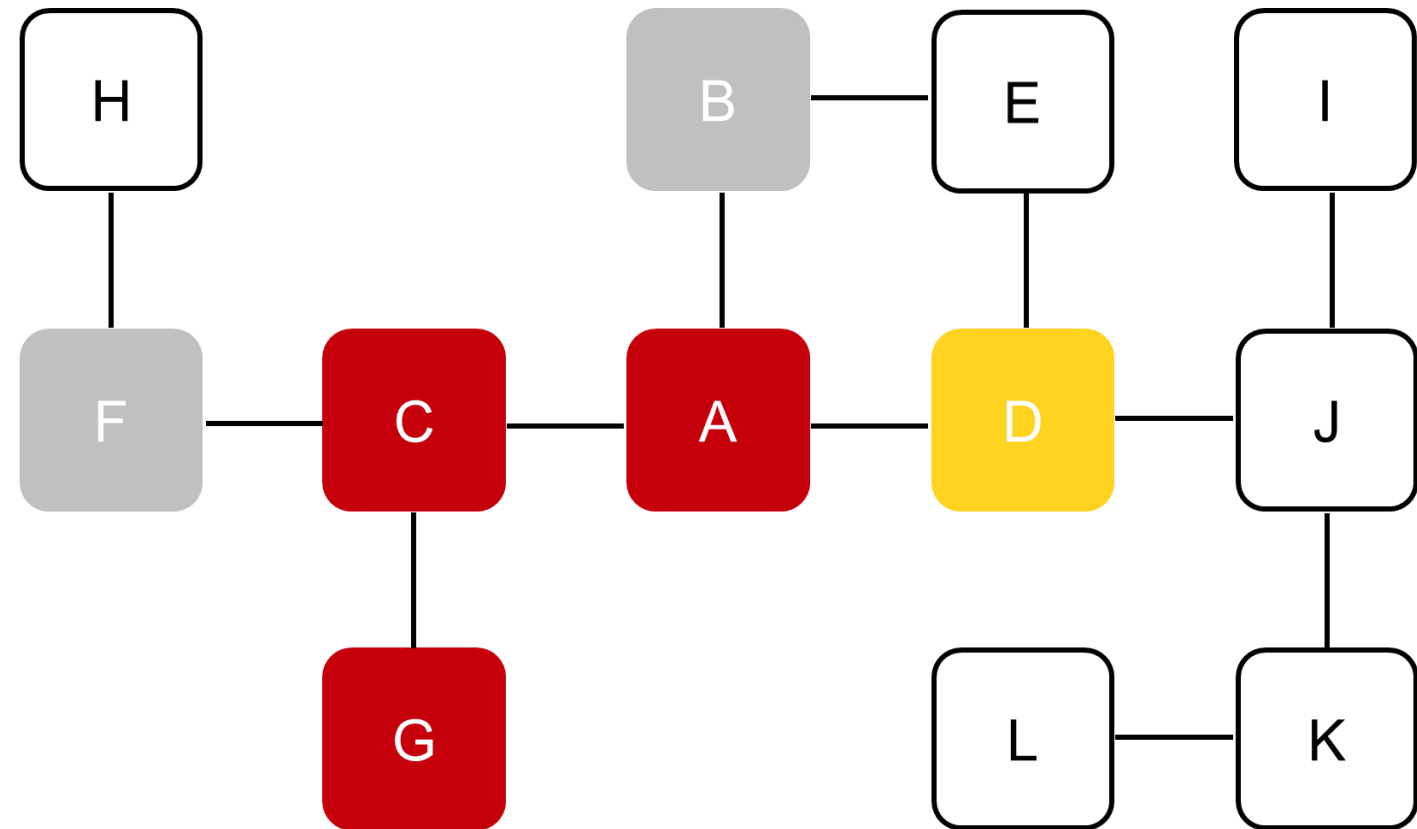
○ Inconnu ● Impacté ● À inspecter ● Non-impacté ● Propage

Impacté,
voisins déjà
traités



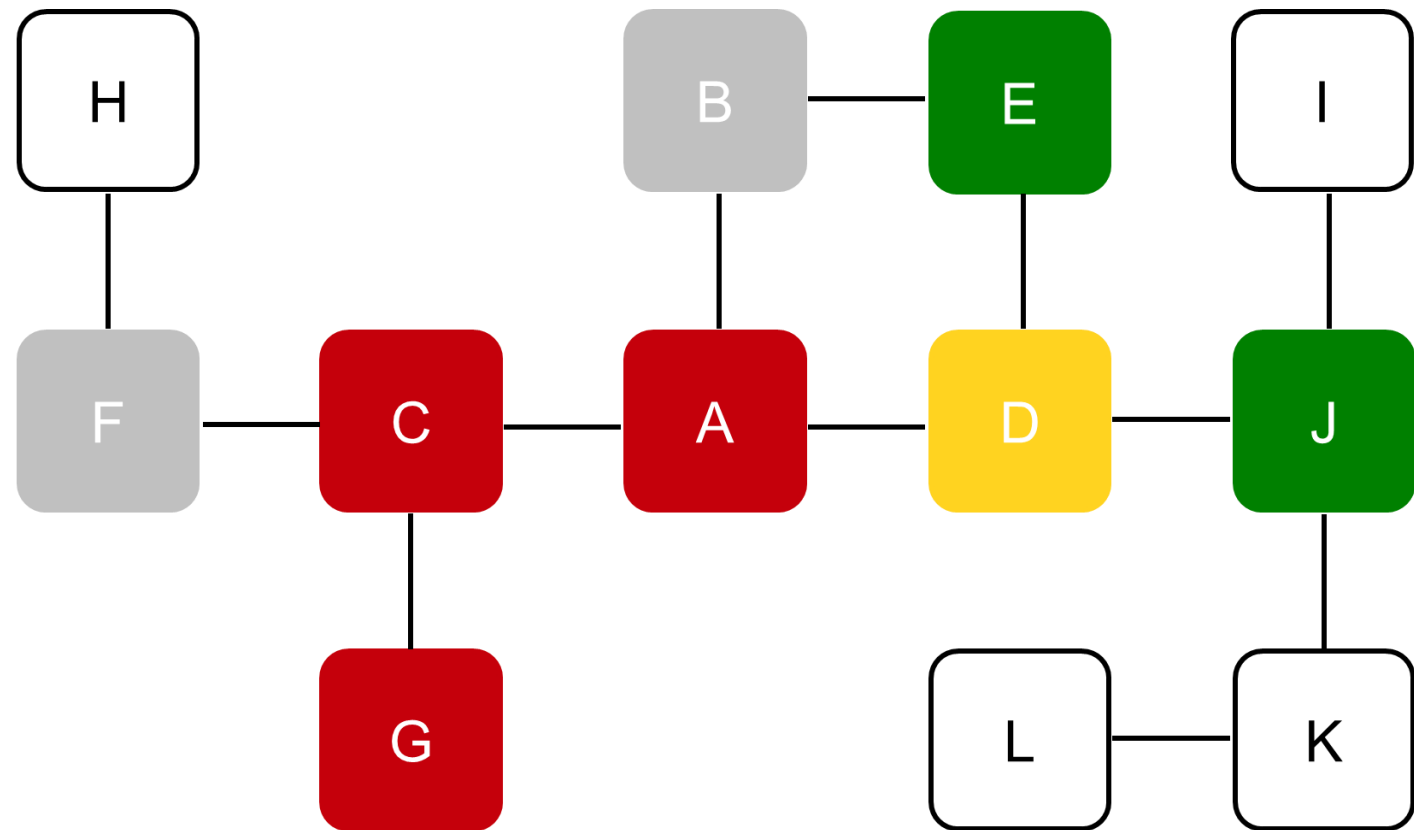
○ Inconnu ● Impacté ● À inspecter ● Non-impacté ● Propage

Pas d'impact direct, mais la propage



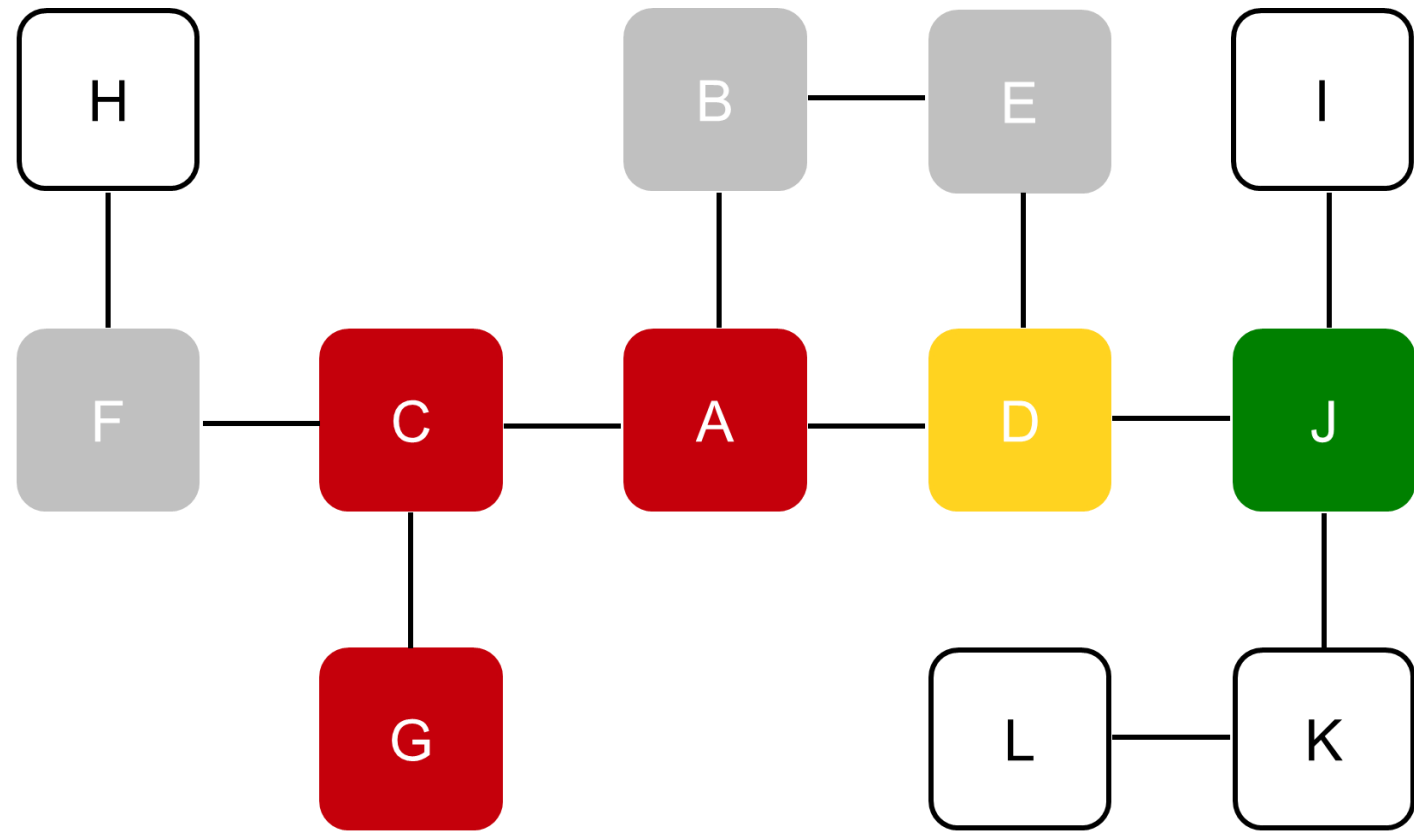
○ Inconnu ● Impacté ● À inspecter ● Non-impacté ● Propage

Nouveaux
voisins



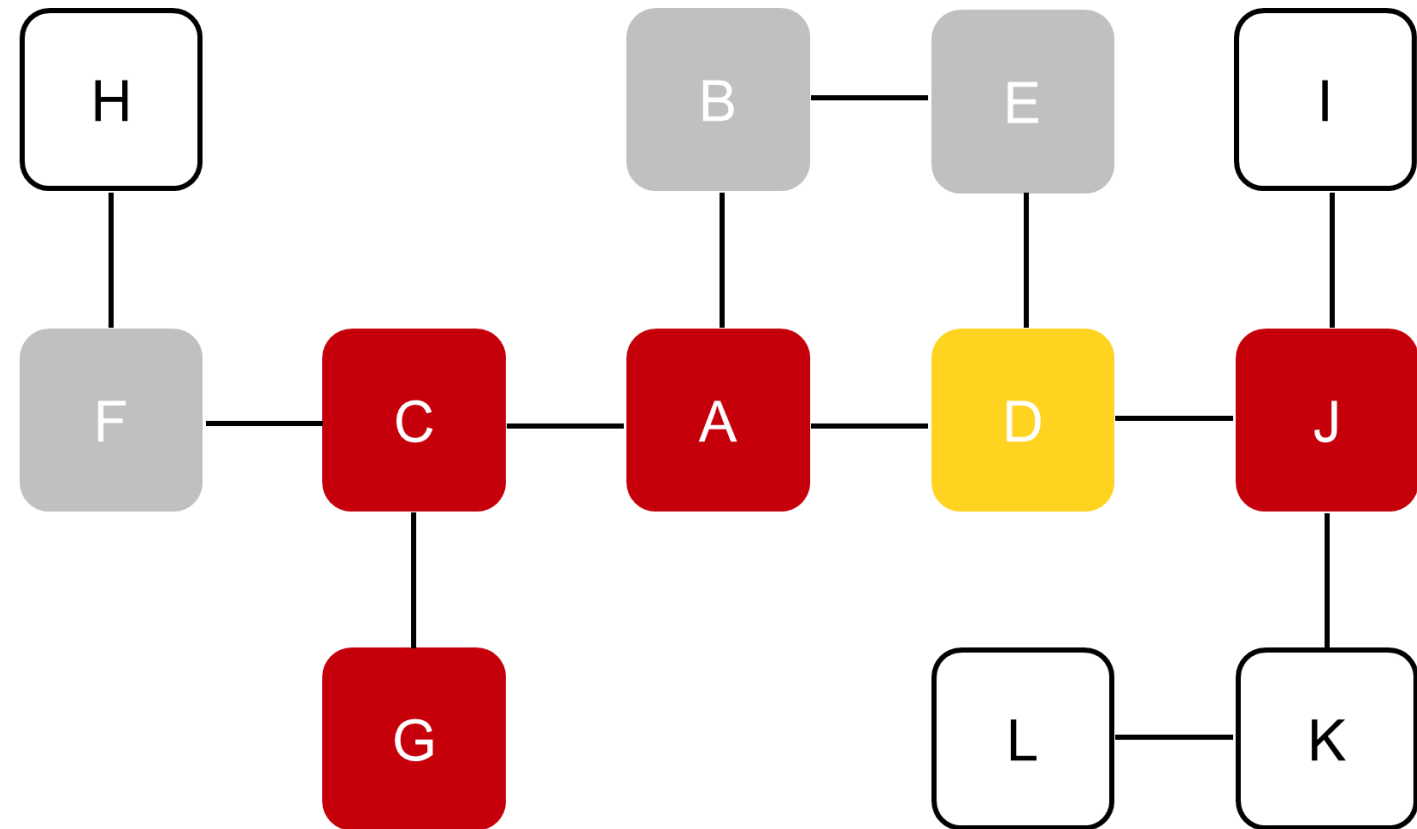
○ Inconnu ● Impacté ● À inspecter ● Non-impacté ● Propage

Non-impacté



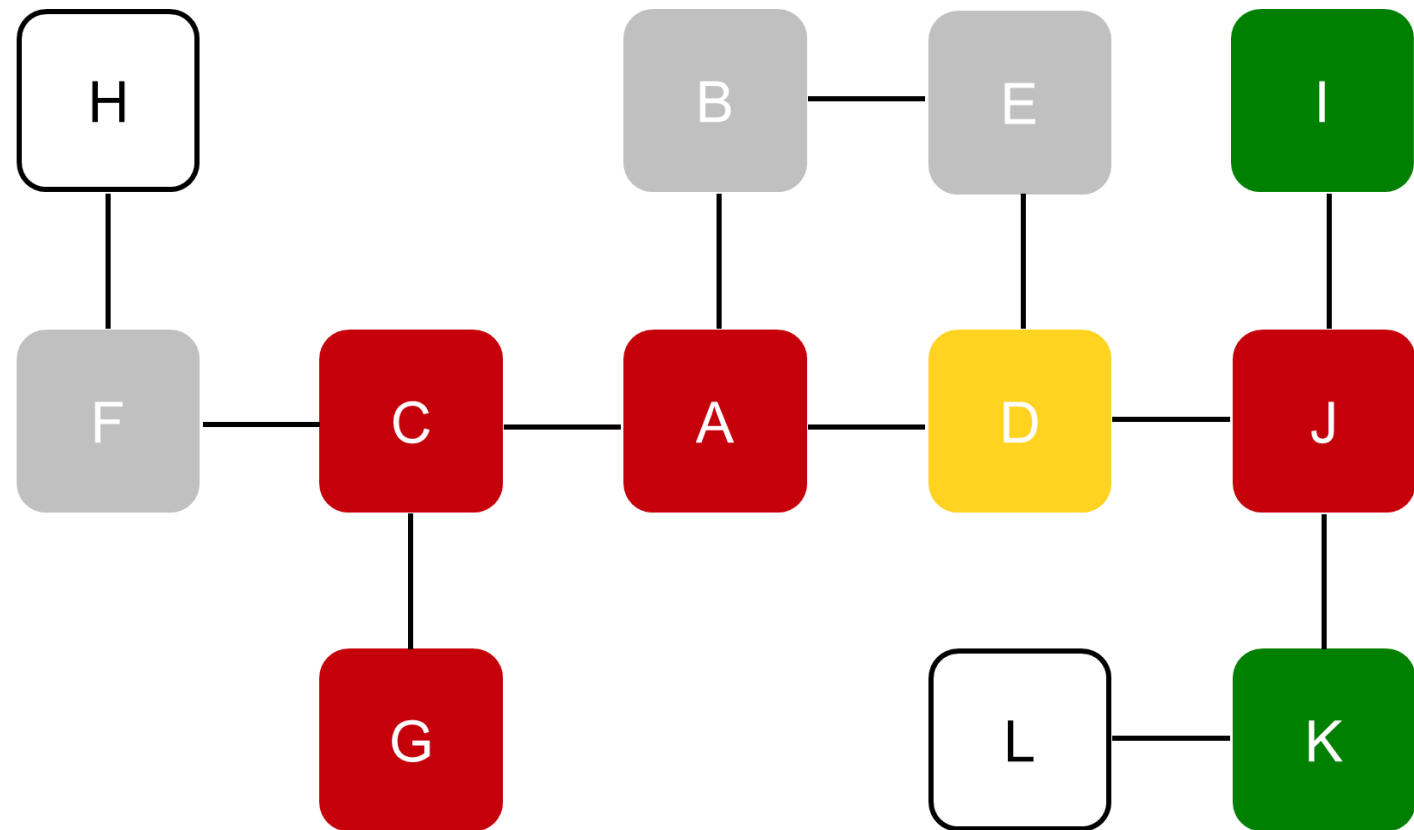
○ Inconnu ■ Impacté ■ À inspecter ■ Non-impacté ■ Propage

Impacté,
ajout à
l'ensemble



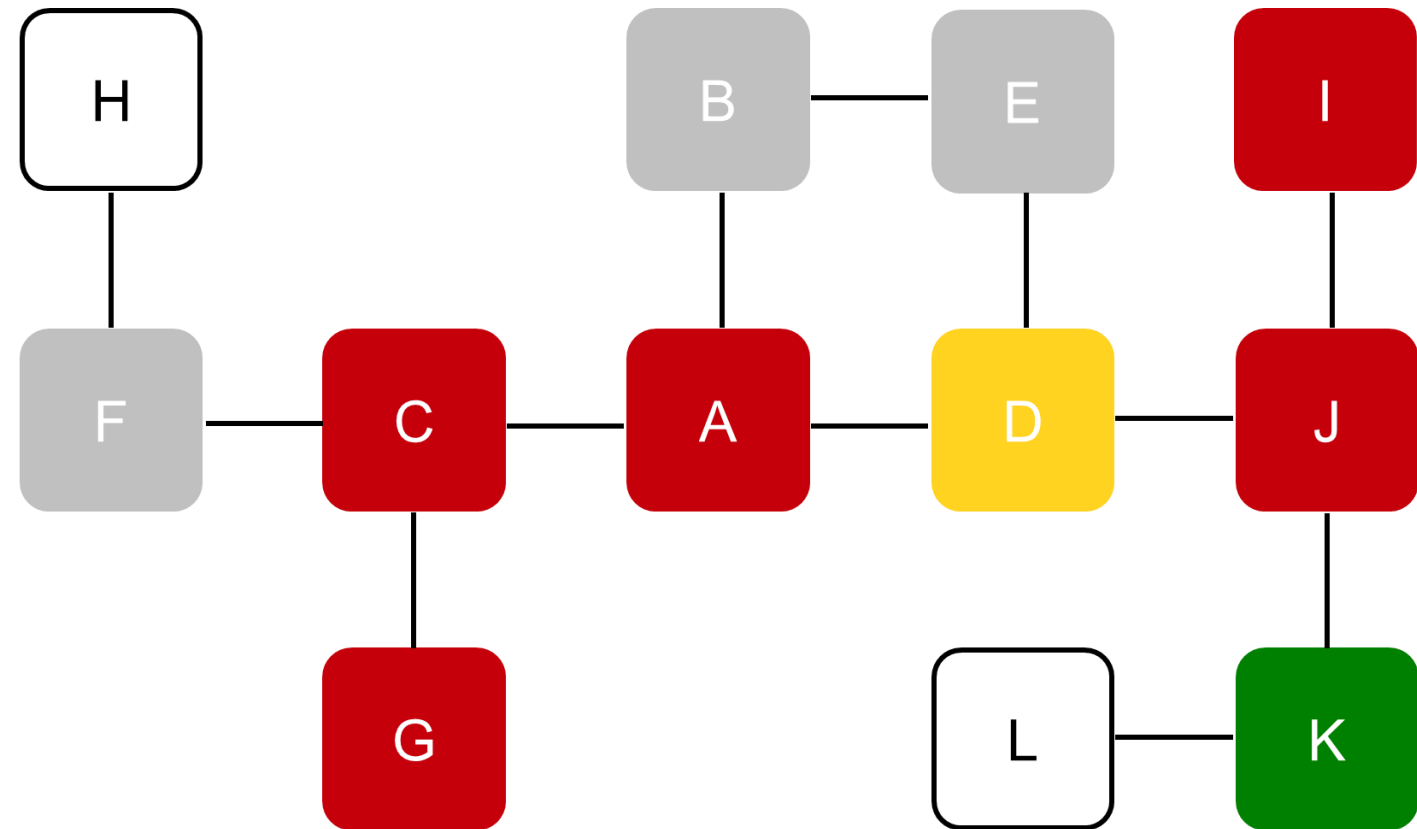
○ Inconnu ● Impacté ● À inspecter ● Non-impacté ● Propage

Nouveaux
voisins



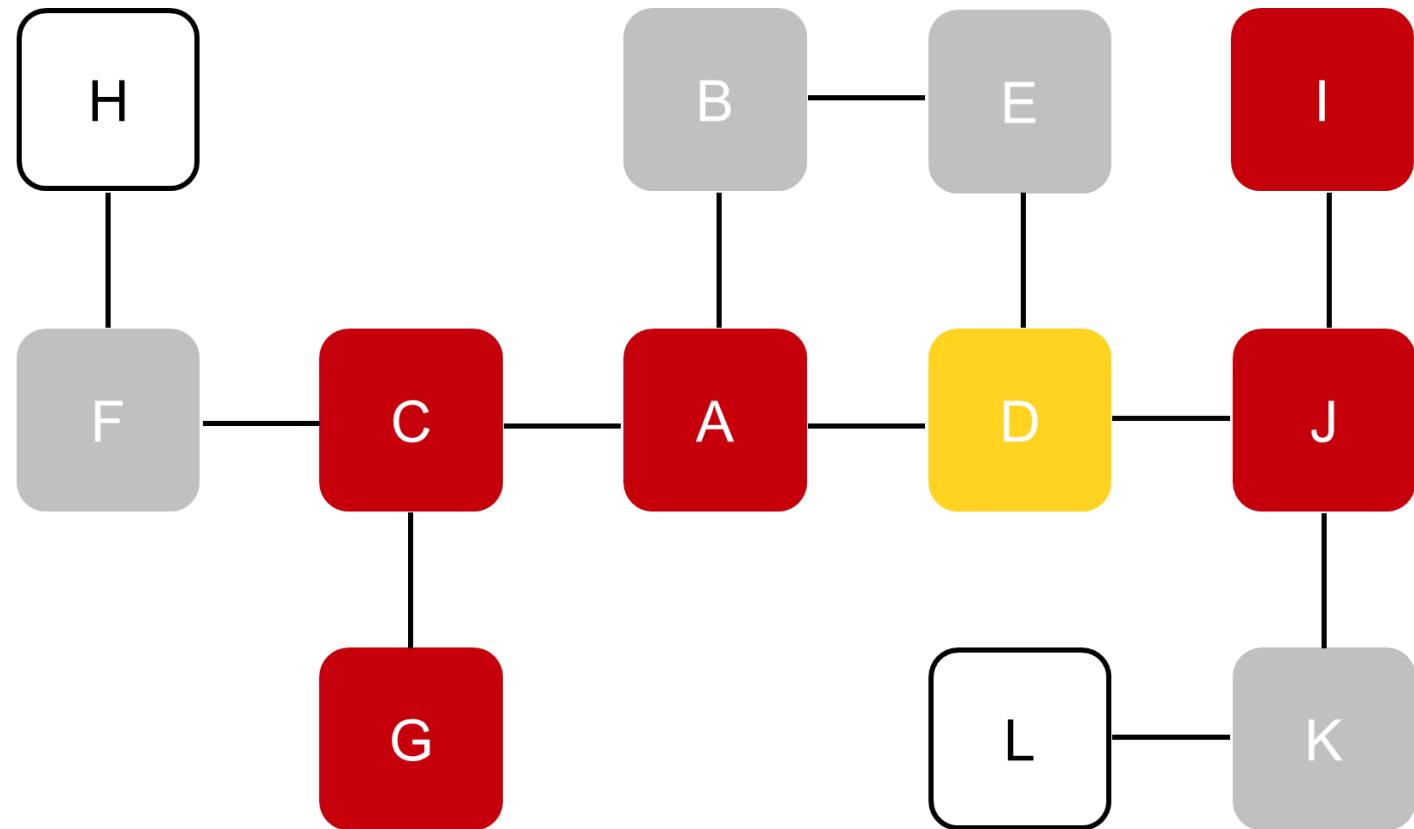
○ Inconnu ● Impacté ● À inspecter ● Non-impacté ● Propage

Impacté,
ajout à
l'ensemble



○ Inconnu ● Impacté ● À inspecter ● Non-impacté ● Propage

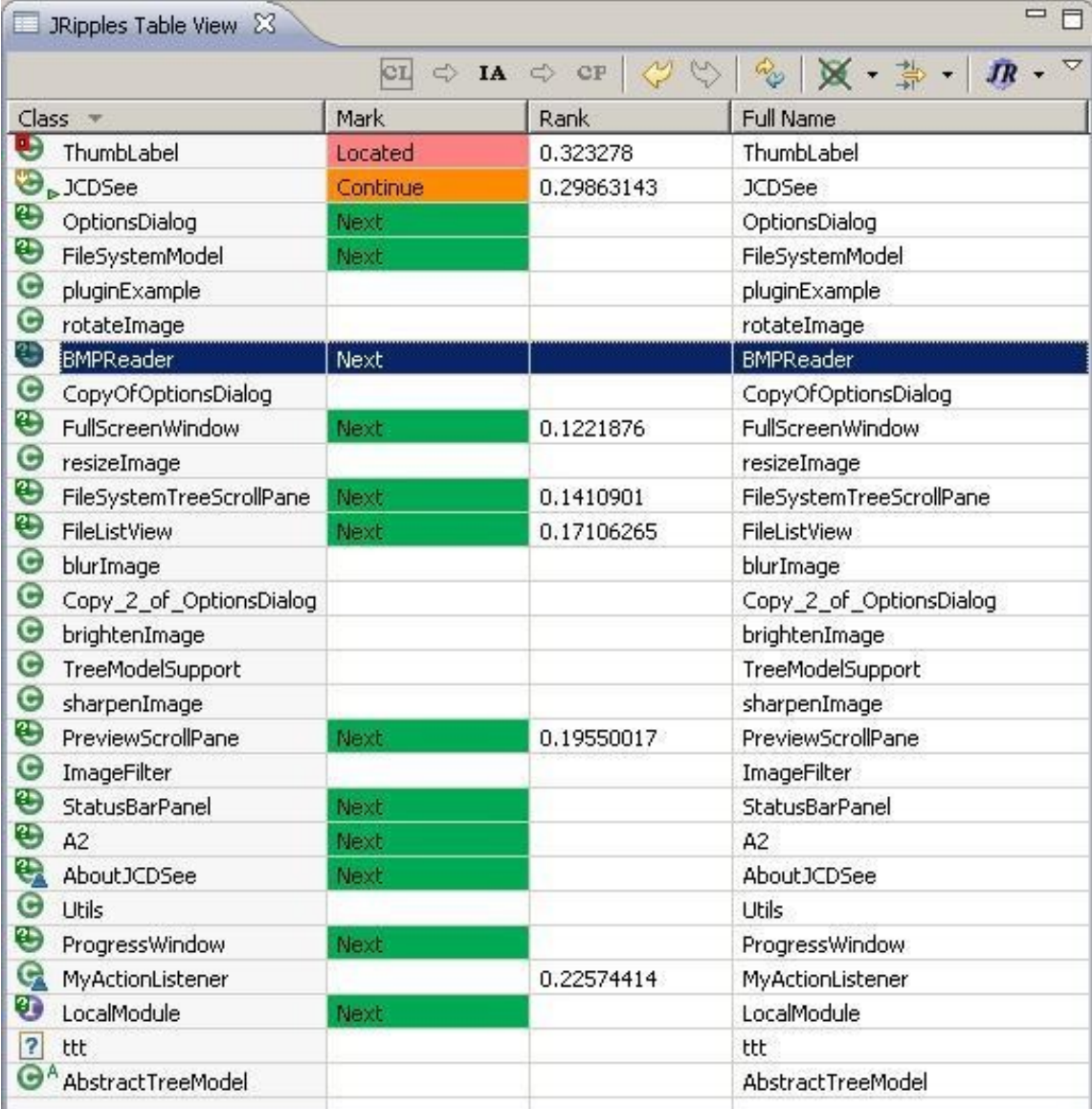
Non-
impacté,
analyse de
l'impacte
achevée



○ Inconnu ● Impacté ● À inspecter ● Non-impacté ● Propage

Outils pour l'analyse d'impact

- Certains outils supportent le processus d'exploration manuelle
 - Ex: JRipples, XRay
- Certains outils peuvent prédire l'impact à partir du code source
 - Se basent sur plusieurs indicateurs (peut inclure l'historique du projet)
- Certains outils peuvent décomposer un changement complexe pour analyser l'impact individuel de chaque changement élémentaire
 - Ex: Chianti



The screenshot shows the JRipples Table View window. The table lists various classes and their impact analysis results. The columns are Class, Mark, Rank, and Full Name. The rows are sorted by Rank in descending order.

Class	Mark	Rank	Full Name
ThumbLabel	Located	0.323278	ThumbLabel
JCDSee	Continue	0.29863143	JCDSee
OptionsDialog	Next		OptionsDialog
FileSystemModel	Next		FileSystemModel
pluginExample			pluginExample
rotateImage			rotateImage
BMPReader	Next		BMPReader
CopyOfOptionsDialog			CopyOfOptionsDialog
FullScreenWindow	Next	0.1221876	FullScreenWindow
resizeImage			resizeImage
FileSystemTreeScrollPane	Next	0.1410901	FileSystemTreeScrollPane
FileListView	Next	0.17106265	FileListView
blurImage			blurImage
Copy_2_of_OptionsDialog			Copy_2_of_OptionsDialog
brightenImage			brightenImage
TreeModelSupport			TreeModelSupport
sharpenImage			sharpenImage
PreviewScrollPane	Next	0.19550017	PreviewScrollPane
ImageFilter			ImageFilter
StatusBarPanel	Next		StatusBarPanel
A2	Next		A2
AboutJCDSee	Next		AboutJCDSee
Utils			Utils
ProgressWindow	Next		ProgressWindow
MyActionListener		0.22574414	MyActionListener
LocalModule	Next		LocalModule
ttt			ttt
AbstractTreeModel			AbstractTreeModel

Réalisation

Modification du code existant ou création de nouvelle fonctionnalité

Insérer nouveau code dans l'ancien code

Traverser les classes dépendantes et les mettre à jour



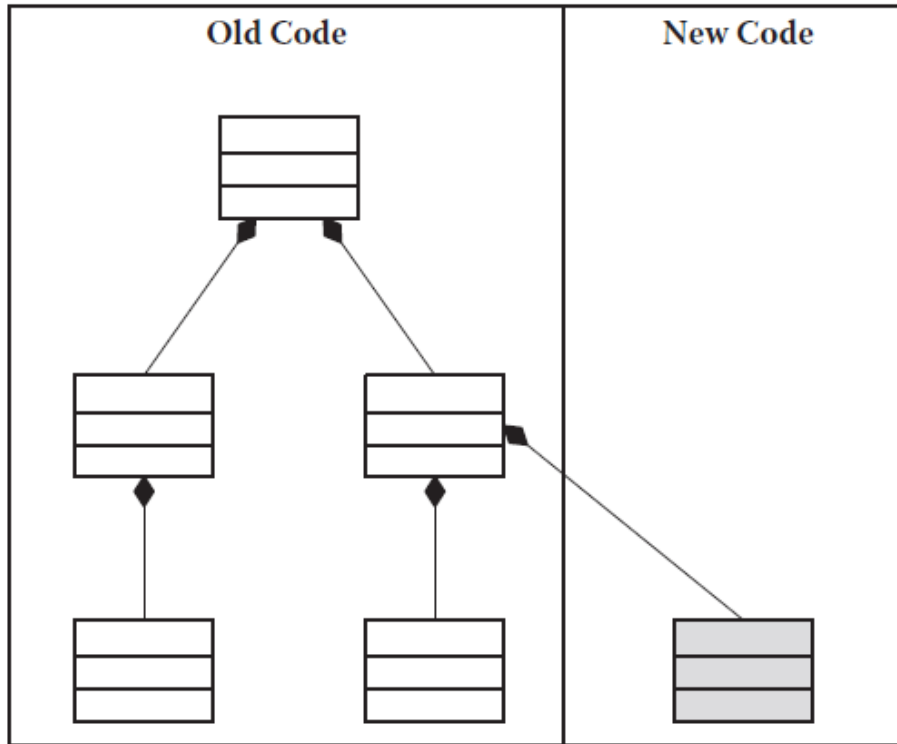
Stratégie adoptée

- « **Quick fix** » : changement d'urgence
 - Utilisé lorsque le logiciel n'est plus activement maintenu (service)
- **Changement durable** : toutes les autres situations
 - Le coût d'un changement rapide est plus élevé à long terme que le bénéfice qu'on en retire
 - Il est préférable de bien exécuter un changement, de façon professionnelle et réfléchie

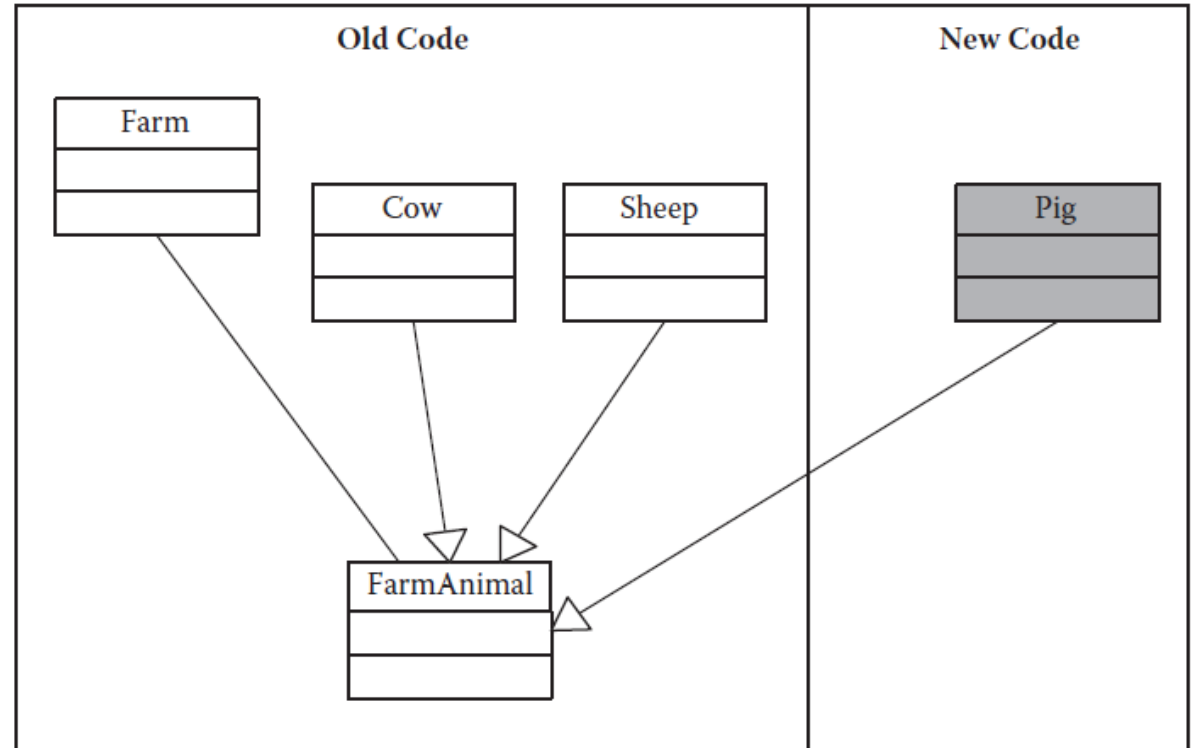
Ajout de fonctionnalité

- Ajout de nouveau code qui interagit avec le code existant
- L'impact de l'ajout ou de modifications se répercute sur les classes voisines
 - Propagation des changements
 - Effet domino
- **Petit changement** : directement dans le code existant
 - Ex: augmenter la taille d'un tableau
- **Changement important** : ajout de nouvelles classes

Exemple



Ajout de fonctionnalité locale



Ajout d'une nouvelle classe

Retrait de fonctionnalité obsolète

- Peut aussi propager le changement
- Toutes les références à la fonctionnalité retirée doivent aussi être retirées
 - Les changements secondaires se propagent à d'autres classes

Propagation des changements

- L'analyse de l'impact prédit les classes qui seront impactées
- La propagation de changements modifie le code des classes impactées
- L'analyse d'impact peut surestimer ou sous-estimer les changements à effectuer
 - Conséquence de l'invisibilité du logiciel
 - Rend la planification difficile

Réusinage (Refactoring)

Modifie la structure du code sans affecter la fonctionnalité

```
3 references
25 export class TypeScriptVersionPicker {
26     6 references
27     private _currentVersion: TypeScriptVersion;
28     1 reference
29     public constructor(-
30     ) {-
31     }
32     }
33     }
34     }
35     }
36     }
37     }
38     }
39     }
40     }
41     }
42     1 reference
43     public async show(firstRun?: boolean): Promise<{ oldVersion?: TypeScriptVersion, newVersion?:
44     const pickOptions: MyQuickPickItem[] = [];
45     pickOptions.push({
46         label: localize('learnMore', 'Learn More'),
47         description: '',
48         id: MessageAction.learnMore
49     });
50     }
51     const shippedVersion = this.versionProvider.defaultVersion;
52     }
53     pickOptions.push({
54     label: (!this.useWorkspaceTsdkSetting
```

Initiation

Concepts

Impact

Réalisation

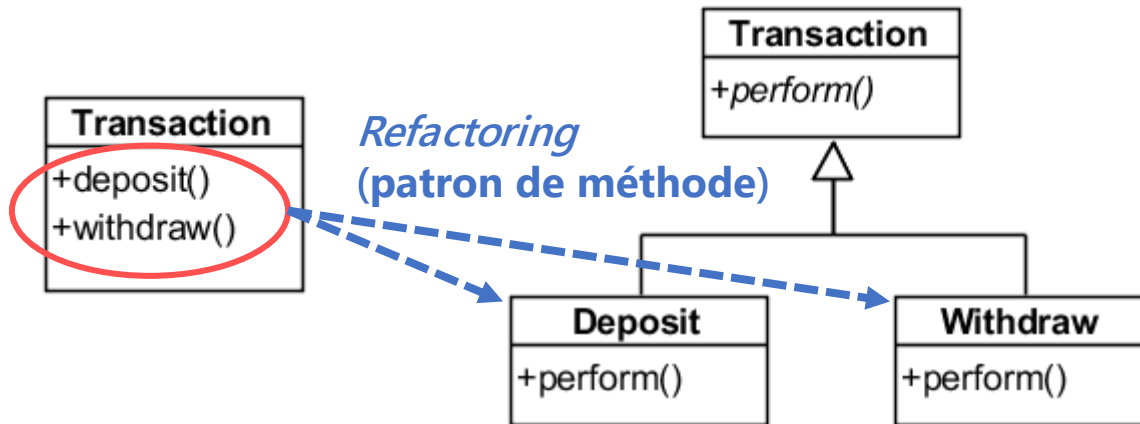
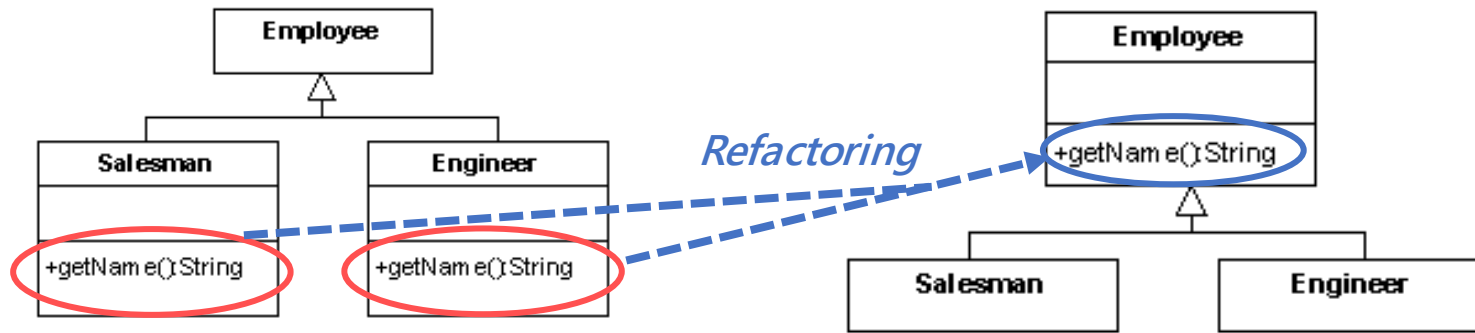
Refactoring

Conclusion

Refactoring

- **Modifie la structure du code sans affecter la fonctionnalité**
 - Activité importante lors de l'évolution
- Certains changements sont apportés afin de **minimiser l'impact** des changements
- Consiste en une séquence de **transformations** qui **préservent le comportement**

Catalogue de patrons de refactoring



```
if (isSpecialDeal()):  
    total = price * 0.95  
    send()  
else:  
    total = price * 0.98  
    send()
```

Refactoring

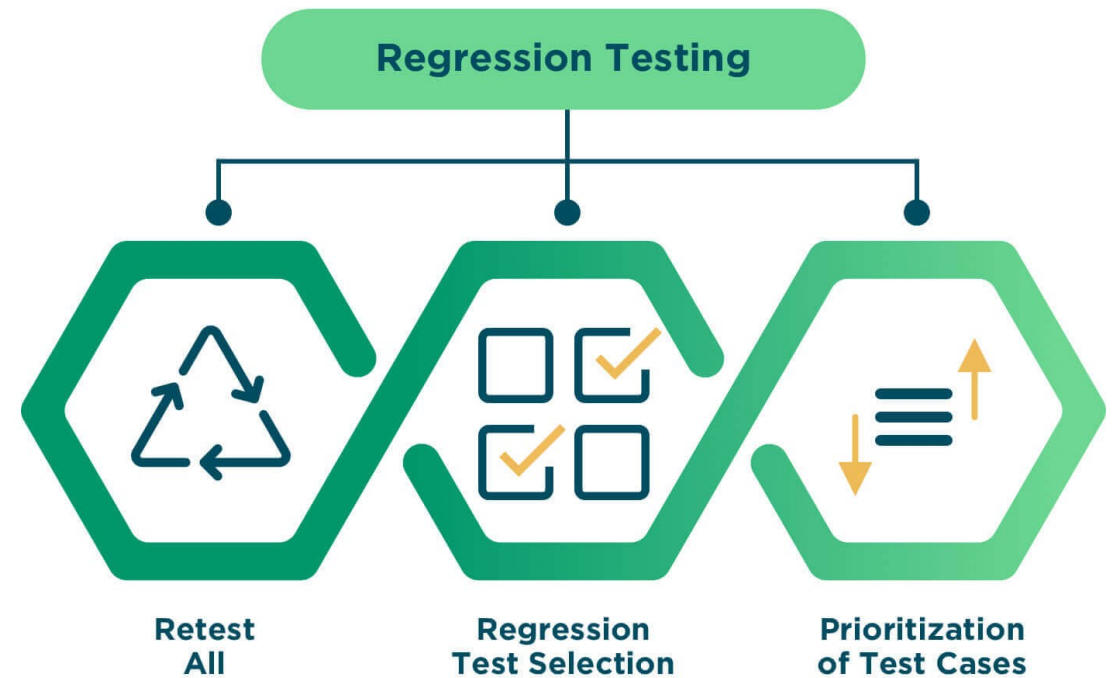
```
if (isSpecialDeal()):  
    total = price * 0.95  
else:  
    total = price * 0.98  
send()
```


Vérification

Garantir que le changement est correct

Créer de nouveaux tests pour
le nouveau code

Re-tester le code existant



Test de régression

- Après un changement, le programmeur re-teste le code
 - Rétablit la confiance que les fonctionnalités antérieures fonctionnent toujours
 - Changement peut avoir introduit par inadvertance des problèmes dans des parties non-modifiées
- Cas de test antérieurs font partie du test de régression
 - La suite de tests augmente
 - Souvent effectués durant la nuit
- Préserver toutes les suites de test après le déploiement !

Conclusion

- **Soumettre** le code complété et vérifié dans le système de contrôle de révisions
 - Créer une **nouvelle révision** ou un **tag**
- Changer le statut de la demande de changement à **complété**
 - Écrire un **prologue** qui commente les changements effectués
- **Redéploiement** et livraison ?

