IFT2255 - Génie logiciel

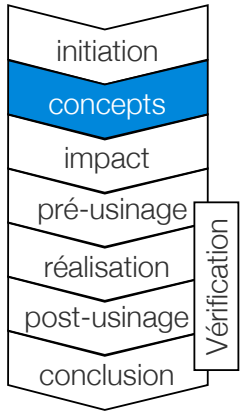
Localisation des concepts

(basé sur le matériel de Václav Rajlich)

Bruno Dufour
dufour@iro.umontreal.ca

Localisation des concepts

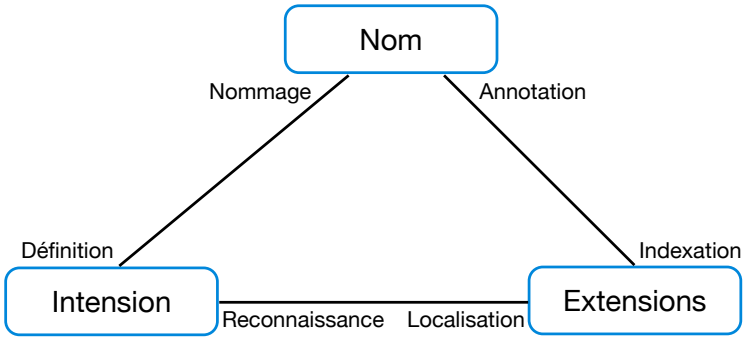
- But : identifier l'emplacement du code qui doit subir des changements
- Les changements sont souvent formulés en fonction de termes (concepts) du domaine



Compréhension partielle du code

- Un très grand programme ne peut être complèment compris
- Un programmeur
  - cherche à obtenir une compréhension minimale qui permet d'effectuer un changement
  - ne veut explorer une partie de code que lorsque c'est nécessaire
  - cherche à comprendre comment les concepts sont reflétés dans le code
- Analogie : visiter une grande ville

Concepts



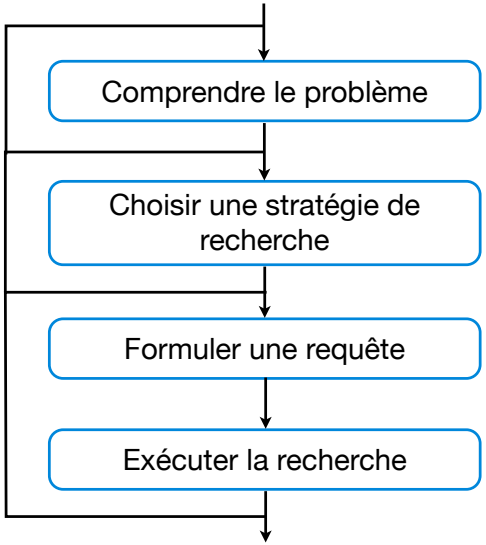
Exemple

- Nom : paiement
- Intension : “verser une somme d’argent due afin de compléter une transaction”
- Extensions :
  - classe `CreditCardPayment`
  - classe `CashPayment`

Localisation des concepts

- Les extensions de concepts correspondent à des fragments de code
  - variables, classes, méthodes, etc.
- Les programmeurs recherche ces fragments
  - Facile dans le cas de petits programmes ou de programmes familiers
  - Difficile pour de grands programmes ou des programmes inconnus

Recherche dans du code inconnu



Formuler une requête

- Extraire l’ensemble des concepts utilisés dans la requête de changement
- Éliminer les concepts visant uniquement la communication avec les programmeurs
- Retirer les concepts peu probables d’être implémentés
  - hors de la portée du programme
  - pas encore implémentés
- Trier les concepts par la facilité à les localiser

Exemple - Point de vente (*POS*)

- Requête : “Implémenter le paiement par carte de crédit”
- Concepts :
  - Implémenter
  - Paiement
  - Carte de crédit

Exemple - Point de vente (*POS*)

- Requête : “Implémenter le paiement par carte de crédit”
- Concepts :
  - ~~Implémenter~~ communication avec le programmeur
  - Paiement
  - Carte de crédit

Exemple - Point de vente (*POS*)

- Requête : “Implémenter le paiement par carte de crédit”
- Concepts :
  - ~~Implémenter~~ communication avec le programmeur
  - Paiement
  - ~~Carte de crédit~~ à implémenter

Exemple - Point de vente (*POS*)

- Requête : “Implémenter le paiement par carte de crédit”
- Concepts :
  - ~~Implémenter~~ communication avec le programmeur
  - **Paiement** → À localiser dans le code
  - ~~Carte de crédit~~ à implémenter

Concepts reliés

- L'ensemble de concepts doit parfois être étendu pour prendre en compte
  - des synonymes
    - ex: paiement → dépense
  - des abréviations
    - ex: paiement → pmt

Concepts implicites

- Certains concepts sont **explicites** dans le code
  - correspondent à des attributs, des méthodes, des classes ou autres parties du code
  - ex : document dans un logiciel de traitement de texte
- D'autres concepts sont **implicites**
  - N'apparaissent pas directement dans le code
  - ex : auteur du document dans un logiciel de traitement de texte (qui ne supporte pas la collaboration)

Reconnaître un concept

- Par lecture de code
  - Commentaires
  - Identifiants
  - Algorithme caractéristique
- Par modification
  - Modifier légèrement le code et exécuter
  - Défaire la modification après l'exécution

Stratégies de recherche

- Connaissances humaines
- Outils de traceabilité
- Recherche par exécution
  - Débogueur, profileur, analyse des traces d'exécution
- Recherche statique (code seulement)
  - Recherche de patrons ("grep")
  - Recherche par dépendances
  - Recherche d'information (IR)

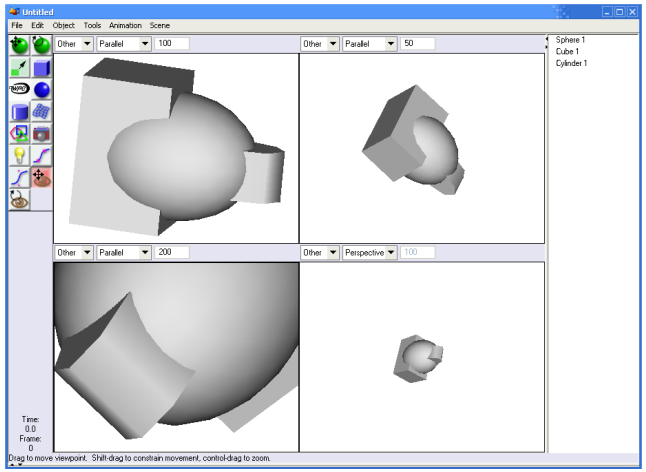
Recherche par grep

- grep = *Global / Regular Expression / Print*
  - utilitaire UNIX populaire
  - permet d'afficher les lignes d'un fichier qui correspondent à une expression régulière
- Démarche
  - Le programmeur formule une requête et inspecte les résultats
  - Si trop de résultats sont retournés, la requête est raffinée ou reformulée

Exemple : Art of Illusion

- Programme de modélisation 3D en Java
- ~600 classes, 100 KLOC
- Requête de changement : "ajouter un contrôle de zoom"
  - Version actuelle ne permet que d'entrer une nouvelle valeur de zoom dans la boîte de texte
  - Par défaut, zoom = 100%
  - Implémenter une version qui utilise les touches du clavier

Art of Illusion



Exemple de recherche

- Recherche #1 : "zoom"
  - 6 lignes retournées
  - toutes inutiles
- Recherche #2 : "scale"
  - 1544 lignes retournées
  - résultats trop volumineux pour l'inspection manuelle
- Recherche #3 : "100"
  - à partir des résultats de la recherche précédente
  - 4 lignes retournées, toutes dans la même classe
  - une des lignes est l'emplacement à modifier

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 point de vente est responsable des items vendus par un commerce et leurs propriétés
    - nouveaux items, description des items, etc.

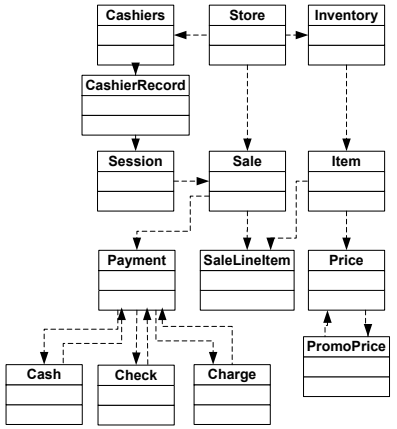
Fournisseurs et clients

- Dans certains cas, une classe ne peut pas implémenter toute la fonctionnalité
  - Elle utilise des **fournisseurs** pour gérer une partie des responsabilités
  - ex: la classe `Item` délègue la gestion des prix à la classe `Price`
- À l'inverse, une classe fournit des services à ses **clients**
  - ex: la classe `Inventory` utilise (a comme fournisseur) la classe `Item`
- Il existe une dépendance entre deux classes A et B si A est un client de B

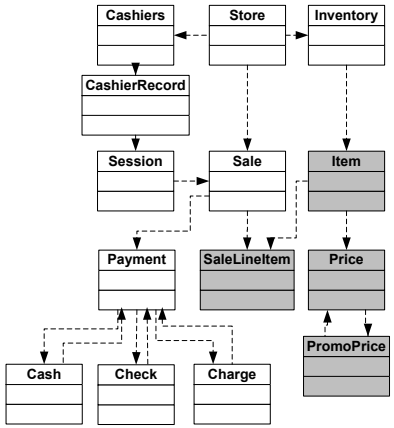
Graphe de dépendances

- L'ensemble des dépendances entre les classes forme un graphe dirigé
  - noeuds : classes
  - arcs : dépendances
    - associations
    - héritage (bidirectionnel dans le cas de l'héritage polymorphique)
    - utilisations
    - + autres dépendances possibles

Exemple



Tranche de fournisseurs



Responsabilités d'une classe

- 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 complèment seule

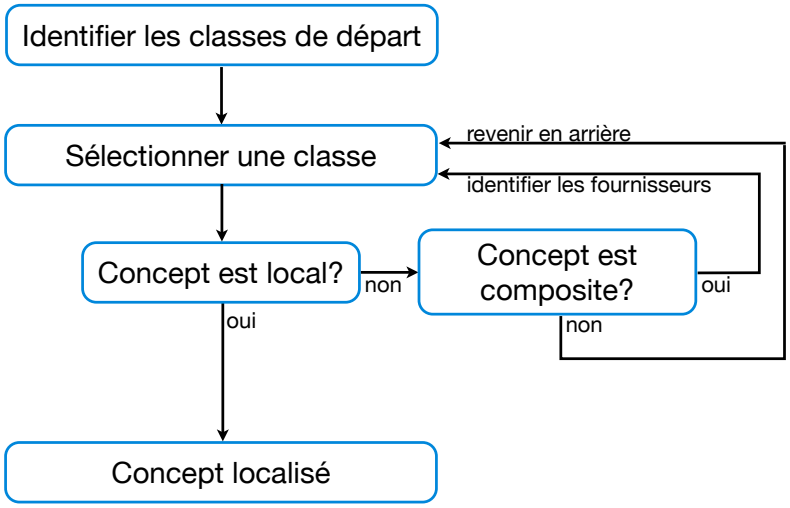
Contrats

- Les responsabilités composites sont souvent gouvernées par des **contrats**
  - limitent les interactions possibles avec les clients
  - préconditions : doivent être garanties par les clients au moment d'une requête
  - postconditions : décrivent les résultats obtenus pour une requête valide
- Exemple :
  - précondition : une liste d'items non-vide et disponibles en inventaire
  - postcondition : le total à payer pour tous les items

Description des contrats

- Les contrats peuvent être décrits à plusieurs niveaux de formalisme
  - Description formelle : langage spécialisé, logique formelle, vérifiable
  - Description en language naturelle
  - Contrat tacite
    - non-documenté
    - commun, mais peu recommandé
    - doit être redécouvert par chaque programmeur (sujet aux erreurs), difficile et coûteux

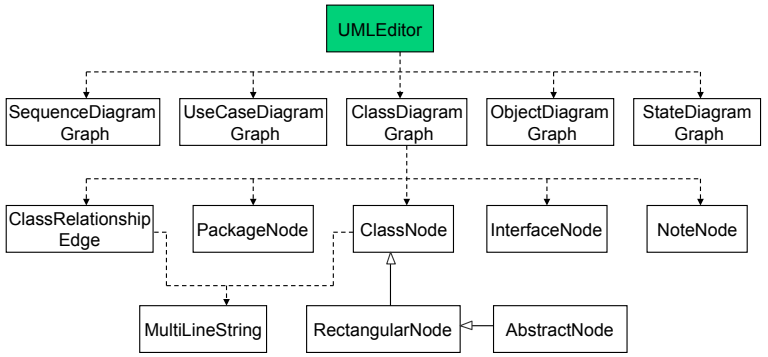
Recherche par dépendances



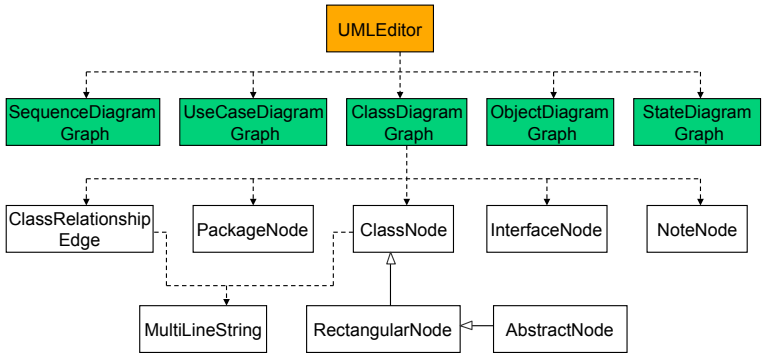
Exemple : Violet

- Requête de changement : “Ajouter un auteur à chaque noeud créée dans un diagramme de classes”
  - Concept : “auteur”
    - Concept implicite, n’apparaît pas dans le code
    - Correspond à une propriété d’un noeud

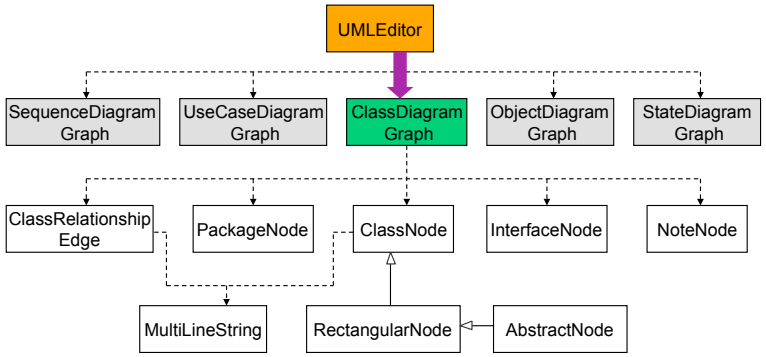
Localiser les propriétés d’un noeud



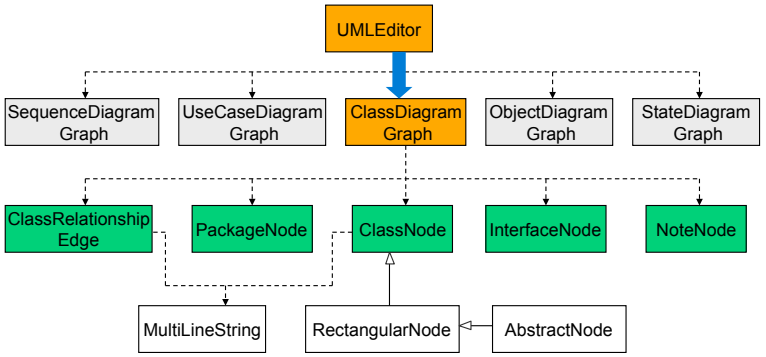
Classes à inspecter



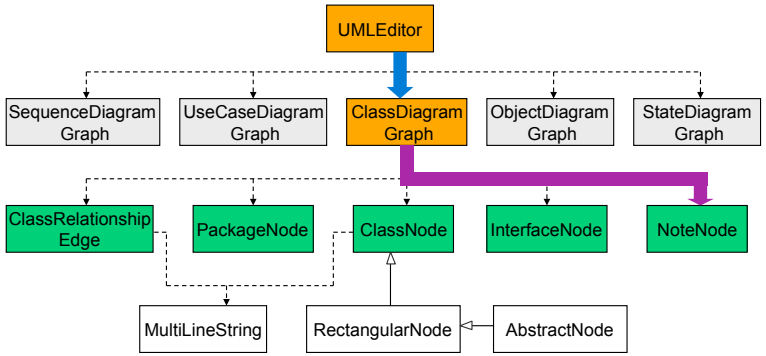
Fournisseur le plus probable



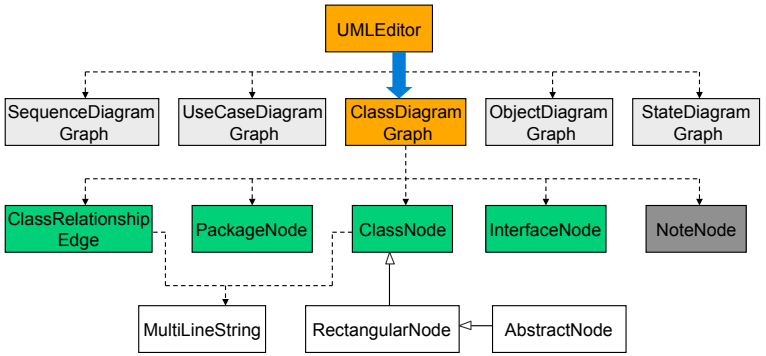
Prochaines classes à inspecter



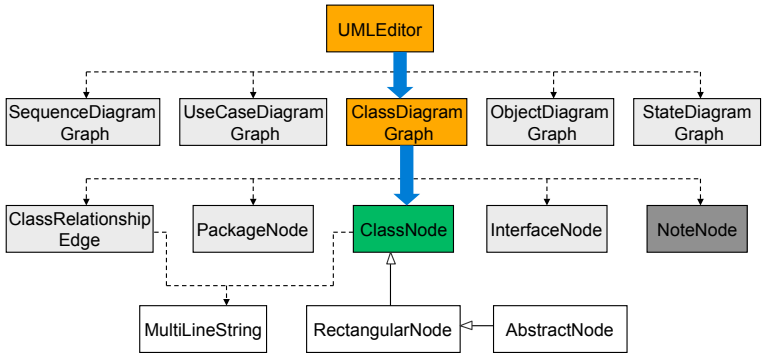
Mauvais choix...



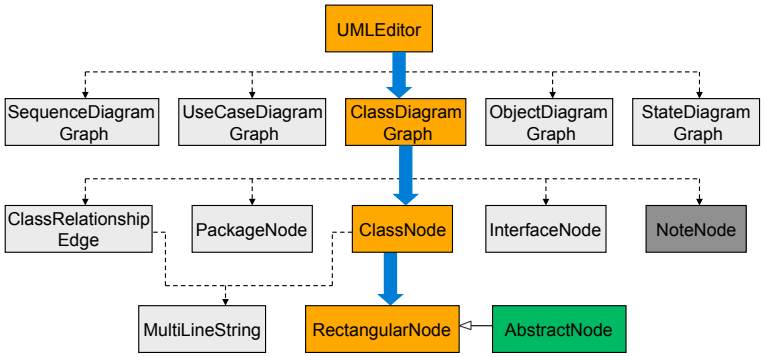
Retour en arrière (*backtrack*)



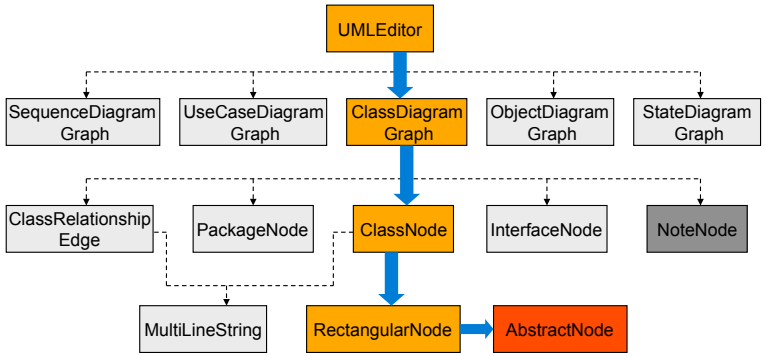
Concept localisé



Extensions possible de la recherche



Autre emplacement trouvé



Comparaison

- grep
  - dépend des conventions de nommage
  - indépendant de la structure du programme
  - applicable à des concepts explicites seulement
- dépendances
  - utilise la structure des classes du programme
  - nécessite la compréhension des fonctionnalités locales et composites
  - applicable aux concepts implicites et explicites