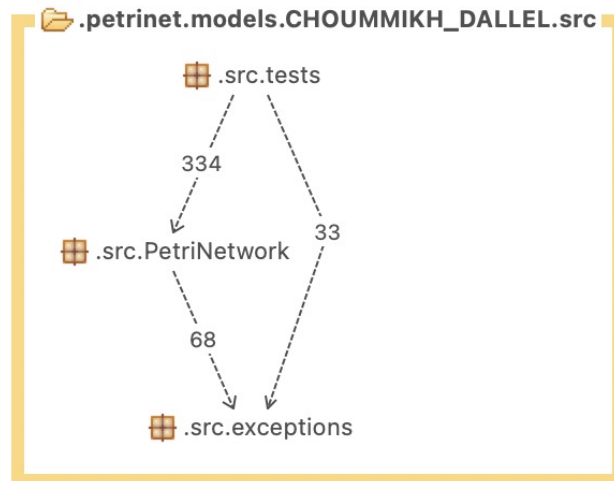
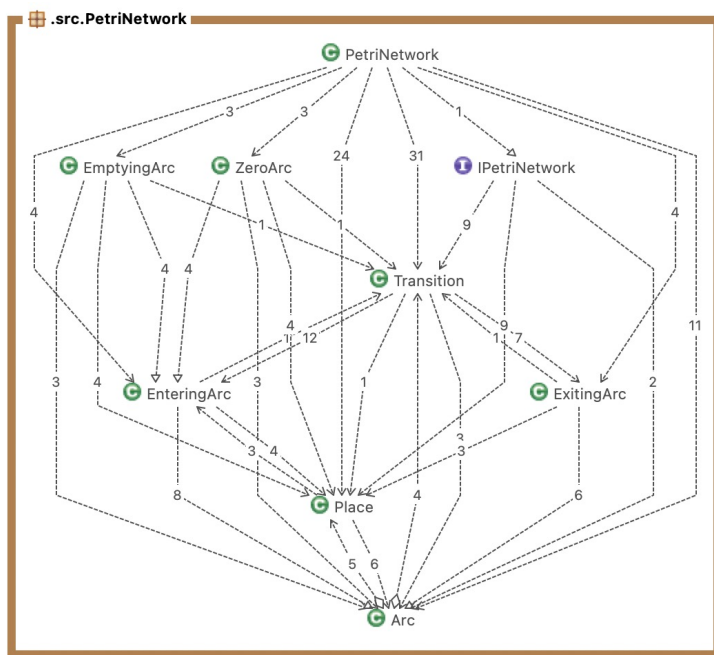


Analyse structurelle du code par STAN:



Dépendances entre les paquets depuis l'onglet Composition

L'analyse structurelle effectuée par le Plugin STAN permet de visualiser les dépendances structurelles entre les classes et les paquets de notre modèle du code d'après la fenêtre « Composition ». On observe un nombre plus au moins logique et raisonnable de dépendances surtout un grand nombre de dépendances du paquetage tests sur le paquetage PetriNetwork qui regroupe nos classes. On observe aussi qu'il n'y a pas de couplage entre les paquetages ce qui rend la suppression de certains d'entre eux facile (Pas d'influence sur le reste des paquetages si on supprime le paquetage `src.tests`).



| Category/Metric | Value |
|---------------------|--------|
| Count | |
| Units | 9 |
| Classes / Class | 0 |
| Methods / Class | 9.78 |
| Fields / Class | 1.11 |
| ELOC | 558 |
| ELOC / Unit | 62 |
| Complexity | |
| CC | 1.52 |
| Fat | 33 |
| ACD - Unit | 59.72% |
| Robert C. Martin | |
| D | -0.56 |
| A | 0.11 |
| I | 0.33 |
| Ca | 12 |
| Ce | 6 |
| Chidamber & Kemerer | |
| WMC | 14.89 |
| DIT | 1.56 |
| NOC | 0.44 |
| CBO | 4.89 |
| RFC | 16.78 |
| LCOM | 14.11 |

Dépendances entre les classes du modèle depuis l'onglet Composition et métriques associées au paquetage PetriNetwork



En sélectionnant le paquetage `PetriNetwork`, on arrive à visualiser les dépendances entre les classes de notre modèle. On repère quelques couplages entre les classes comme entre les deux classes `Transition` et `EnteringArc`. En effet, la classe `Transition` fait appel à des instances de la classe `EnteringArc` dans 12 méthodes alors que la classe `EnteringArc` ne fait appel à la classe `Transition` que dans le constructeur (Voir les deux figures ci-dessous). Le CBO, qui représente le nombre de classes auxquelles une classe est couplée, est d'une moyenne de 4.89 ce qui reste bon en terme de qualité structurelle. On note aussi que la complexité cyclomatique ($\approx Nb \text{ arêtes} - Nb \text{ noeuds} + 2$) de ce graphe est de $1.52 < 30$ ce qui indique que notre code n'est pas complexe.

Au niveau des métriques obtenues par l'analyse STAN, il n'y a pas de violations des 4 métriques proposées par STAN : métriques de taille, métriques de complexité, métriques de R.Martin et celles de Chidamber et Kemerer. Ceci permet de conclure que notre code est bien structuré et qu'il respecte globalement les métriques regroupées par CISQ (fiabilité, sécurité, maintenance, taille et efficacité).

On remarque que la métrique D (distance) de Robert C.Martin (Permet de mesurer le degré de balance entre la stabilité et l'abstraction du code) apparait en couleur jaune (-0.56) mais reste dans les normes puisque sa valeur absolue (0.56) est inférieure à 1.

| | | |
|---|------------|---|
|  .Transition.addEntringArc(EnteringArc) | has param |  .EnteringArc |
|  .Transition.enteringArcList | references |  .EnteringArc |
|  .Transition.exist(boolean, Place) | references |  .EnteringArc |
|  .Transition.fire() | calls |  .EnteringArc.execute() |
|  .Transition.fire() | references |  .EnteringArc |
|  .Transition.getEnteringArcList() | references |  .EnteringArc |
|  .Transition.isFirable() | calls |  .EnteringArc.isActive() |
|  .Transition.isFirable() | references |  .EnteringArc |
|  .Transition.removeEntringArc(EnteringArc) | has param |  .EnteringArc |
|  .Transition.toString() | calls |  .EnteringArc.isZero() |
|  .Transition.toString() | calls |  .EnteringArc.isEmptying() |
|  .Transition.toString() | references |  .EnteringArc |

Méthodes dans lesquelles la classe Transition dépend de la classe EnteringArc

| Source | ^ | --> | Target |
|---|---|-----------|---|
|  .EnteringArc.(int, Place, Transition) | | has param |  .Transition |

Méthodes dans laquelle la classe EnteringArc dépend de la classe Transition.