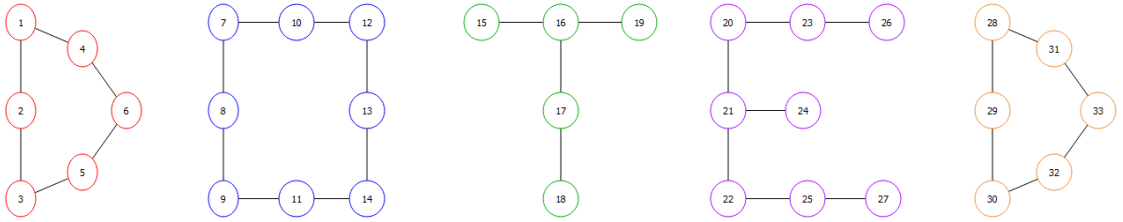
# Présentation PSTL

## Logiciel d'édition de graphes

Morvan Lassauzay et Victor Nea




UPMC SORBONNE UNIVERSITÉS

Encadrants :
Matthieu Dien et Marwan Ghanem

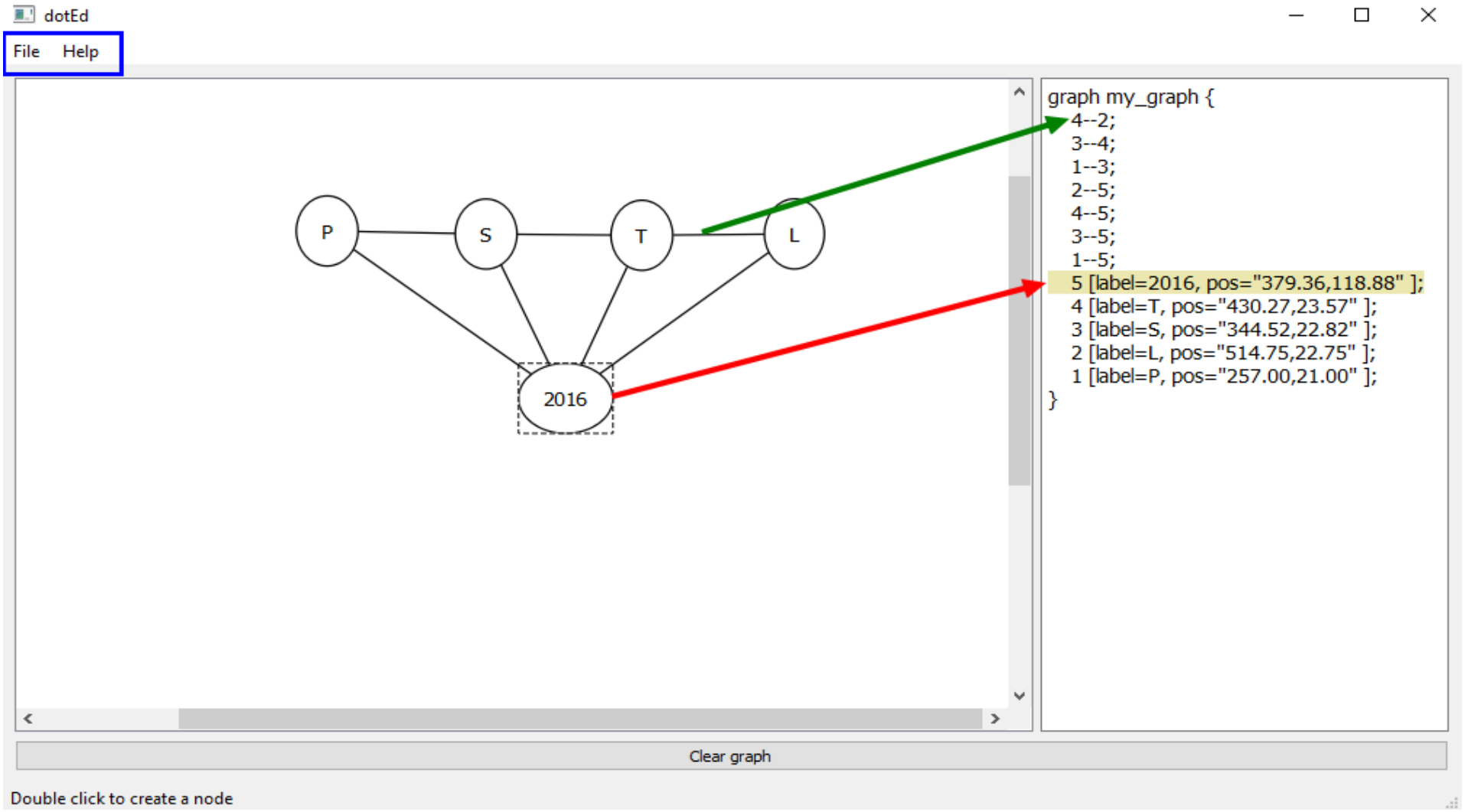
# Sommaire

- Introduction
- I. Vue d'ensemble du logiciel
- II. Fonctionnement des interfaces
- III. Maintenabilité et standardisation
- Conclusion

# Introduction

- **Besoin** : facilité l'utilisation des graphes
- **Objectifs** :
  - manipulation à la souris
  - visualisation graphique et textuelle
  - import et export de fichiers DOT
  - architecture modulaire
- **Outils choisis** :
  - Python
  - PyQt5

# I-1. Aperçu du logiciel




Capture d'écran du logiciel

# I-2. Architecture Modèle-Vue-Contrôleur

- Modèle : état du graphe
- Vue : affichage du modèle + événements
- Contrôleur : liaison vue/modèle + calcul
- 1 vue -> 1 contrôleur : indépendance des vues
- Design pattern Observer : connecter contrôleur au modèle

# II-1. Vue textuelle (1)

**Cas n°1 - Edition à partir du texte :**

- Difficultés :
  - validité du texte
  - détection des modifications

- Solution :
  - 1ère étape : *parsing* pour obtenir les éléments du graphe
  - 2ème étape : vérification d'erreurs sur les attributs
  - 3ème étape : détection des modifications

# II-1. Vue textuelle (2)

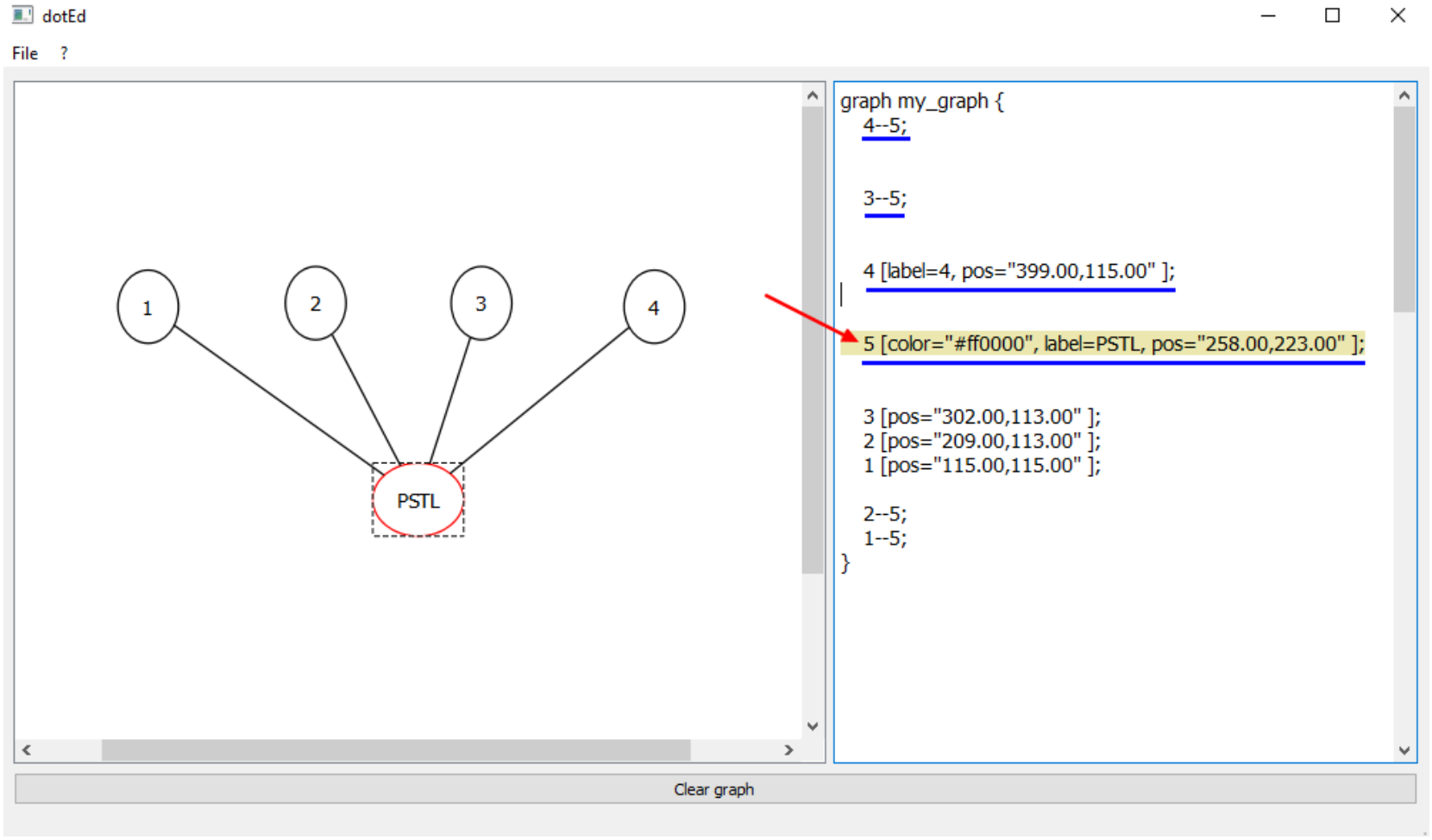
**Cas n°2 – Modification du texte sur ordre du modèle :**

- Difficulté :
  - conserver la forme => réécriture minimale du texte

- Solution en 3 étapes :

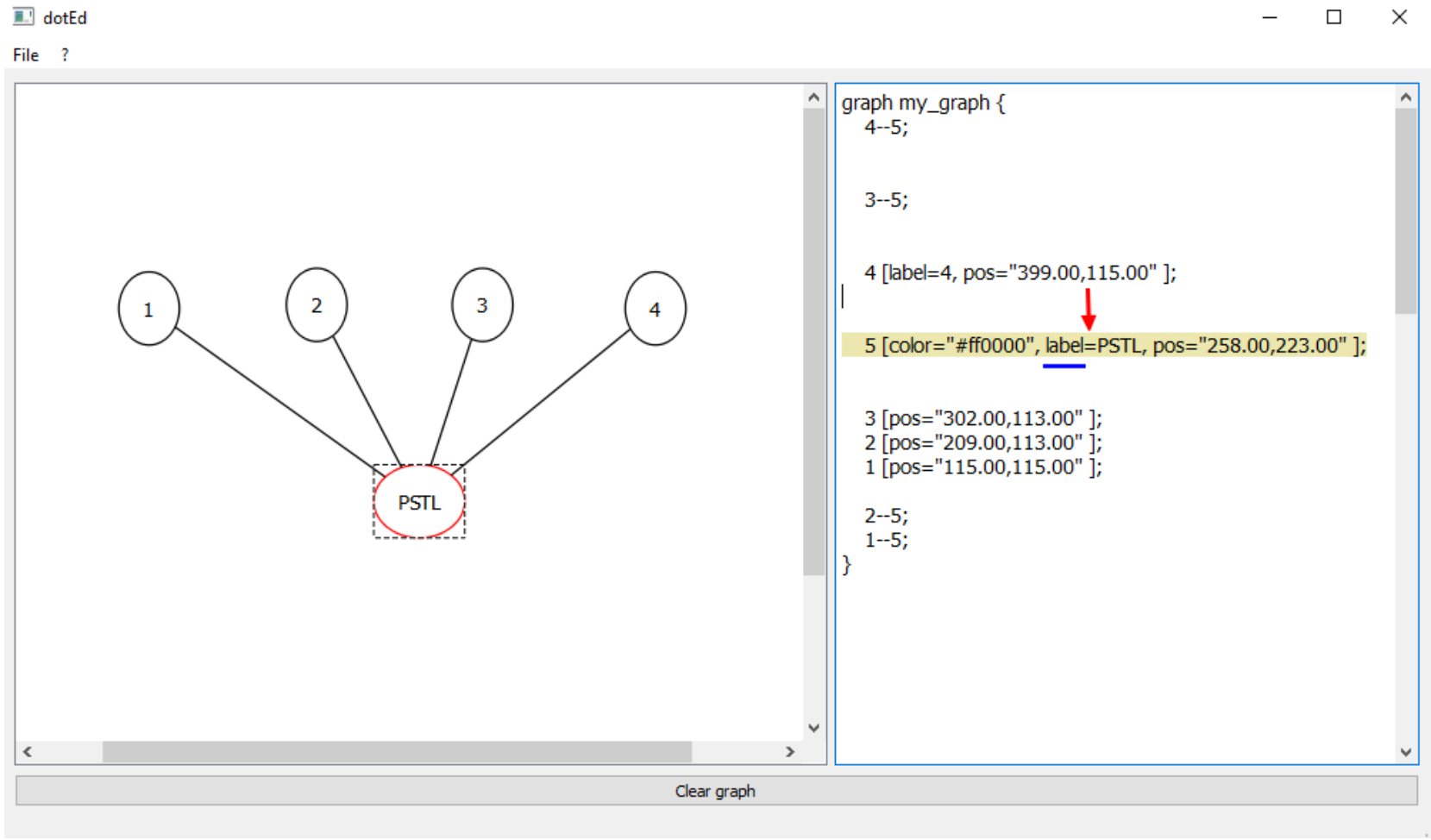
# II-1. Vue textuelle (3)

**Etape 1 : recherche du nœud**



# II-1. Vue textuelle (4)

**Etape 2 : recherche de l'attribut**



# II-1. Vue textuelle (5)

**Etape 3 : remplacement de la valeur**

```
graph my_graph {
   4--5;


   3--5;


   4 [label=4, pos="399.00,115.00" ];


   5 [color="#ff0000", label=, pos="237.20,217.40" ];


   3 [pos="302.00,113.00" ];
   2 [pos="209.00,113.00" ];
   1 [pos="115.00,115.00" ];

   2--5;
   1--5;
}
```

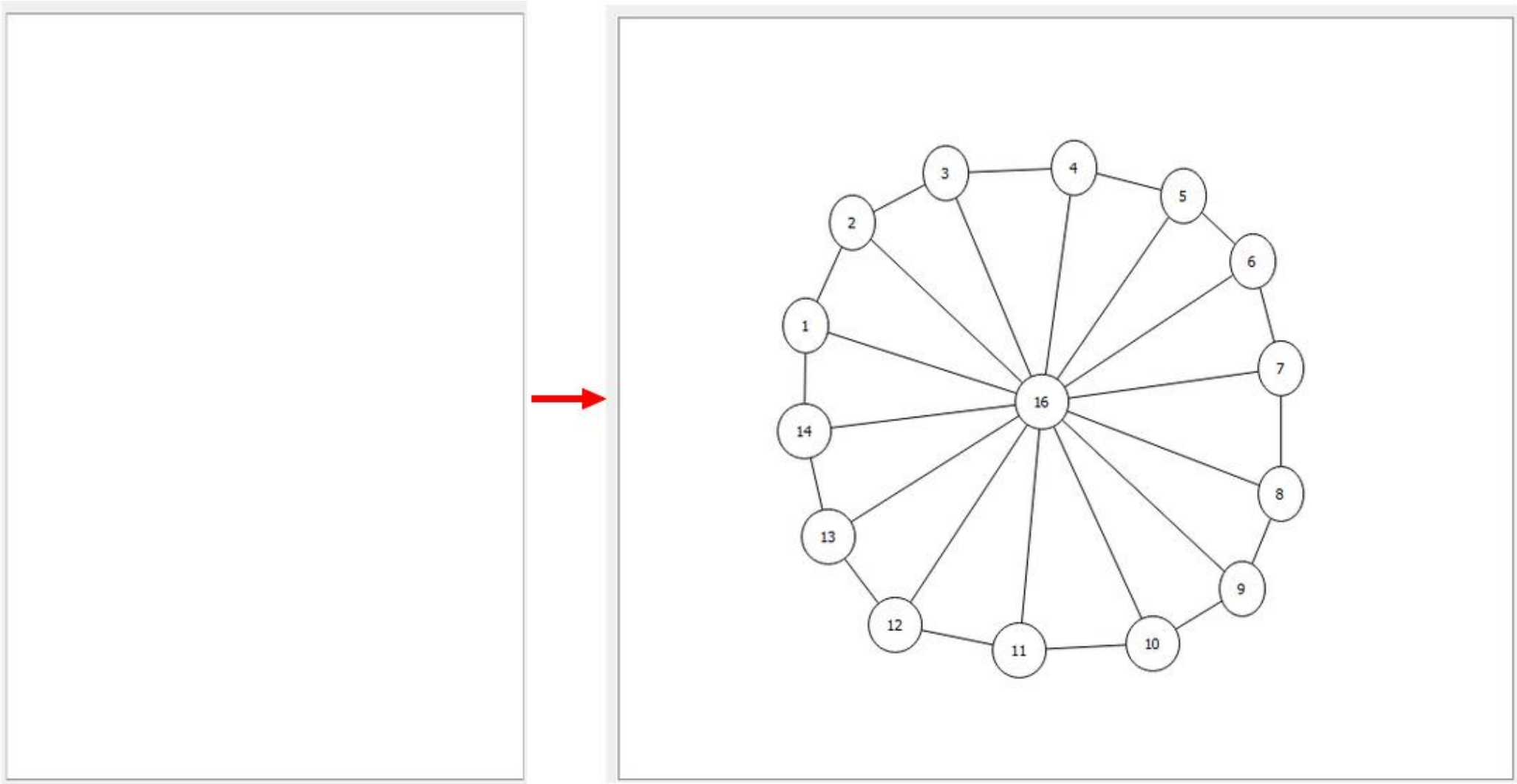
```
graph my_graph {
   4--5;


   3--5;


   4 [label=4, pos="399.00,115.00" ];


   5 [color="#ff0000", label="PSTL 2016" pos="237.20,217.40" ];


   3 [pos="302.00,113.00" ];
   2 [pos="209.00,113.00" ];
   1 [pos="115.00,115.00" ];

   2--5;
   1--5;
}
```

# II-2. Vue graphique (1)

- **Liste des actions possibles :**
  - créer/éditer/déplacer/supprimer des nœuds/arêtes
  - sélection multiple
  - zoomer/dézoomer

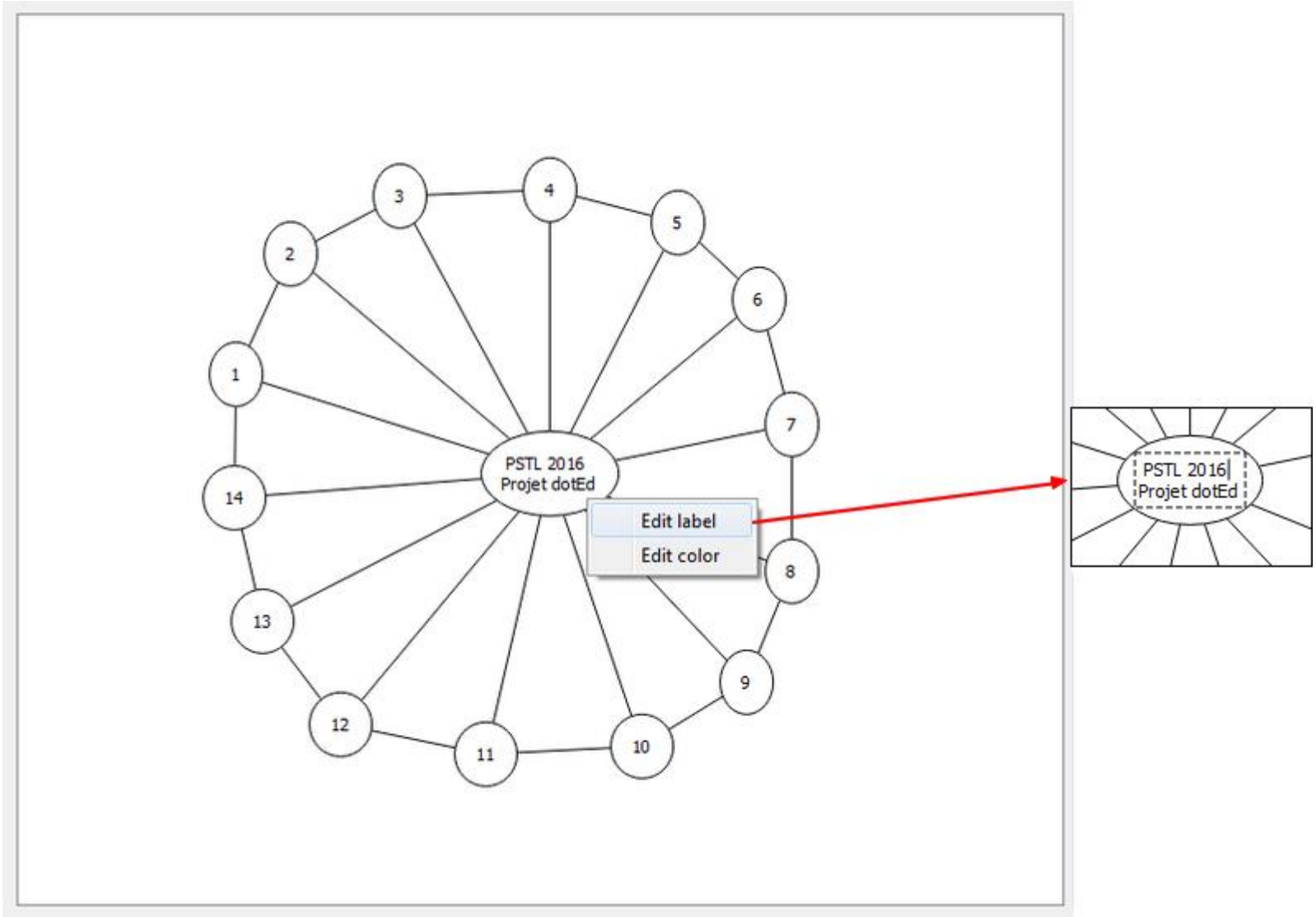
# II-2. Vue graphique (2)

**Etape 1 : création d'un graphe**



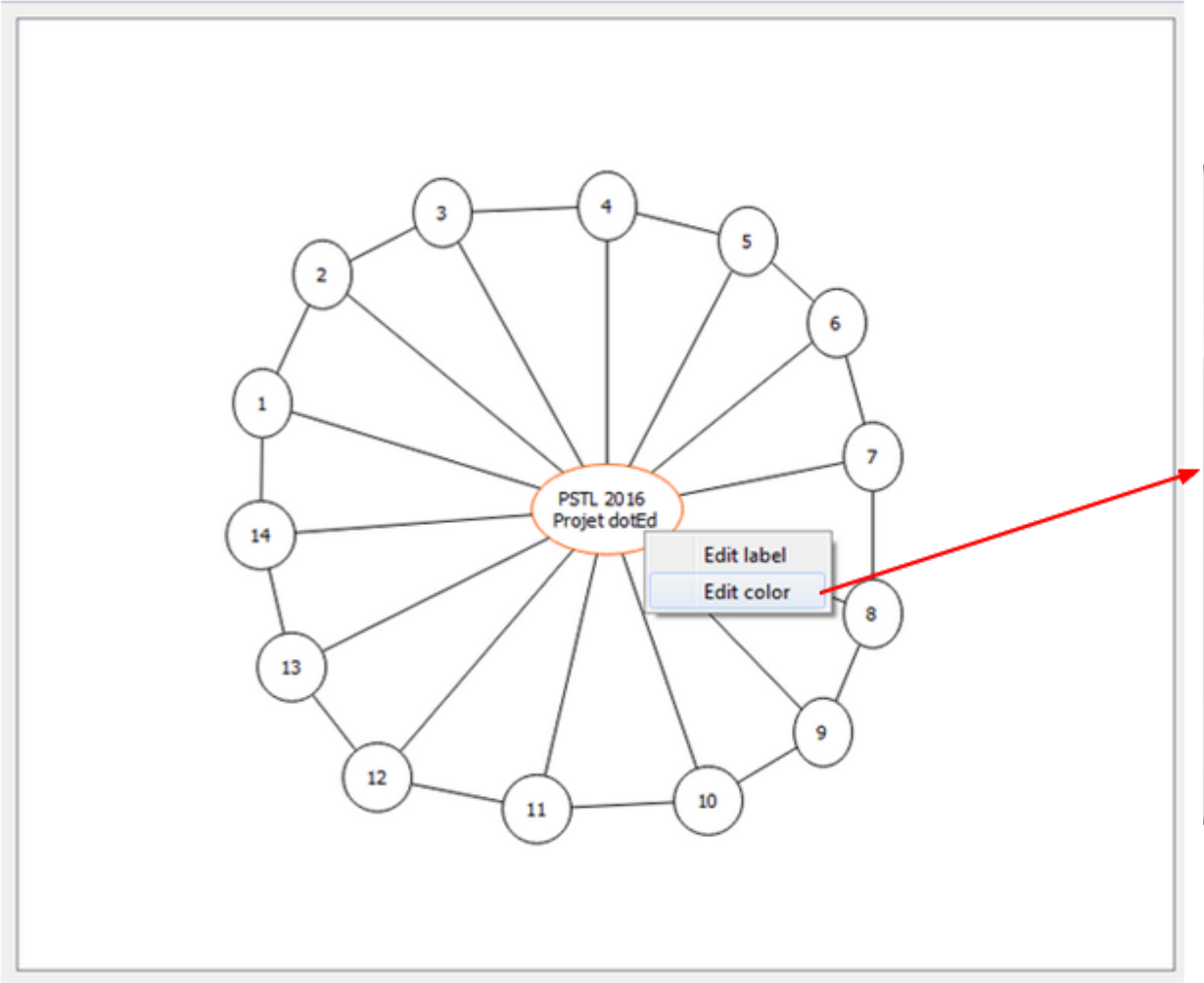
# II-2. Vue graphique (3)

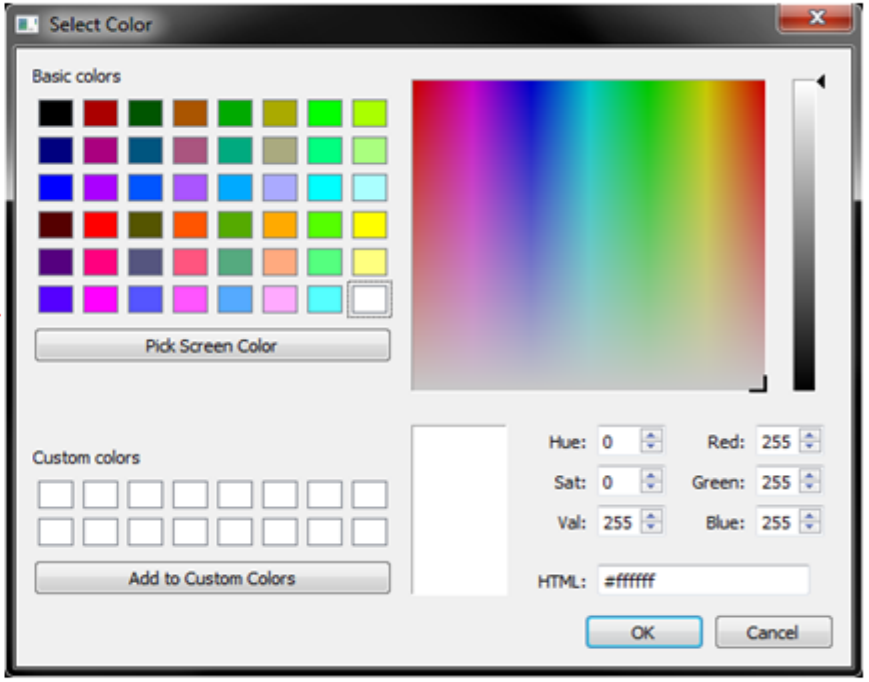
**Etape 2 : édition de l'attribut label d'un nœud**



# II-2. Vue graphique (4)

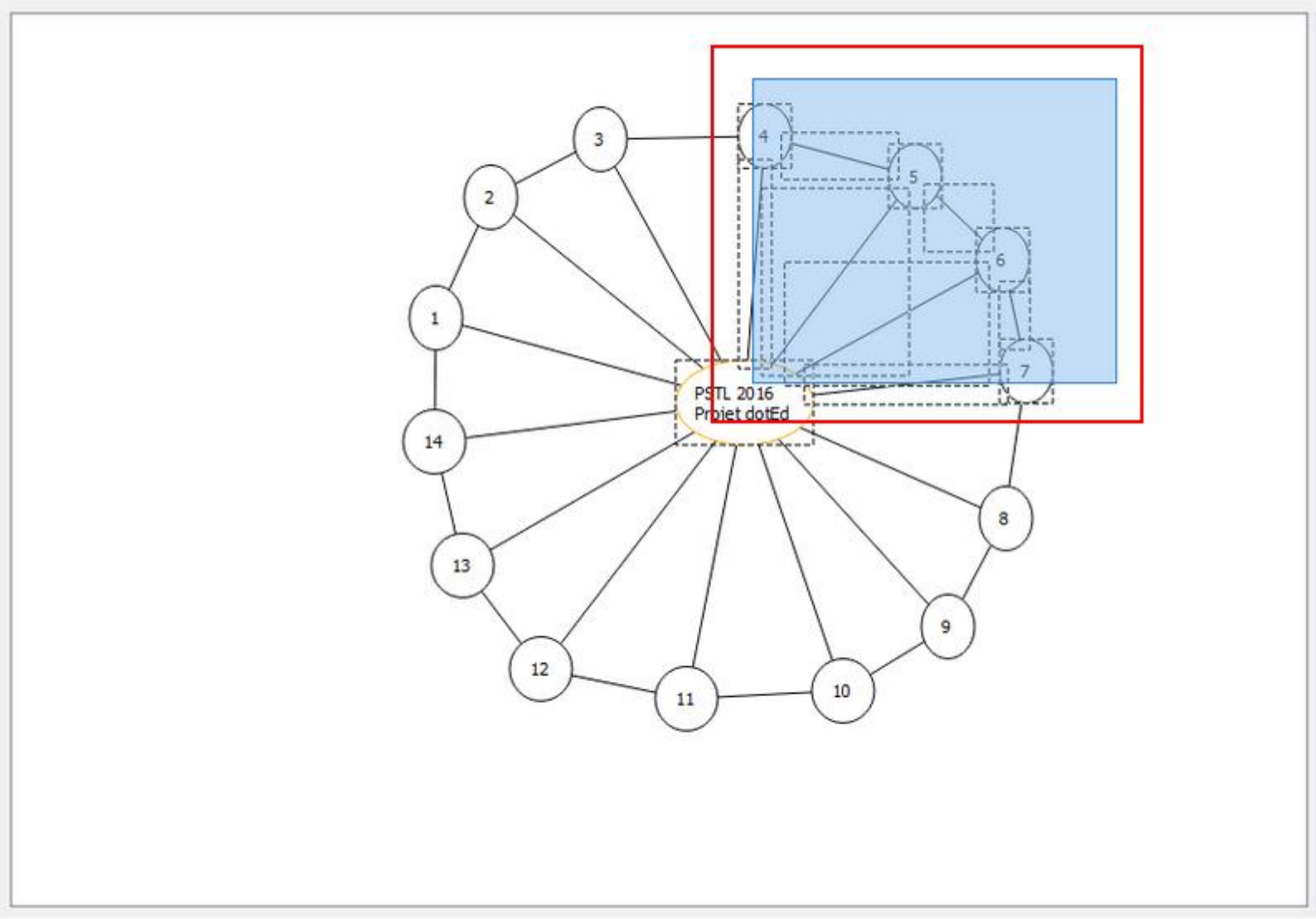
**Etape 3 : édition de l'attribut color d'un nœud (extension)**



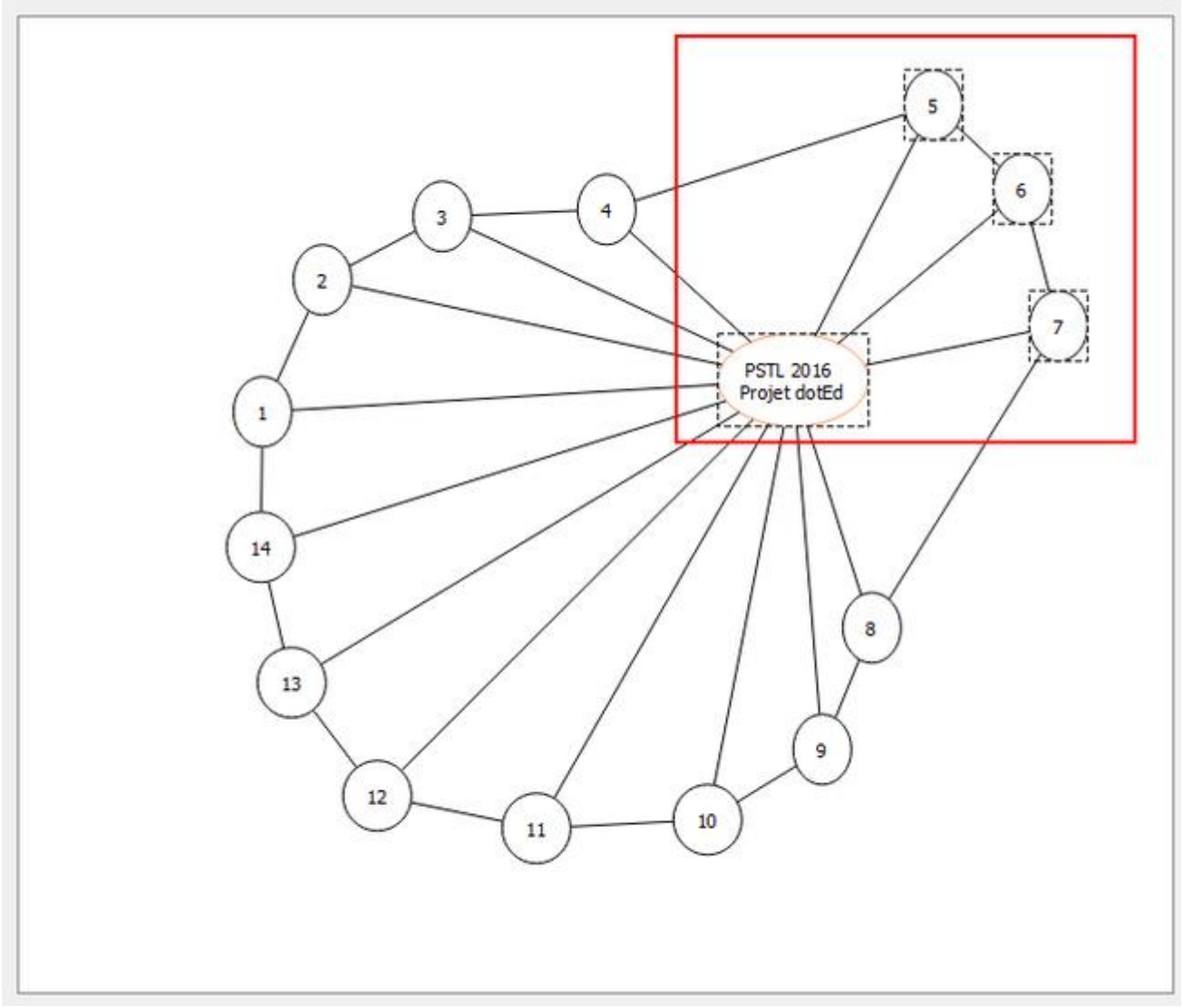
# II-2. Vue graphique (5)

**Etape 4 : sélection multiple**



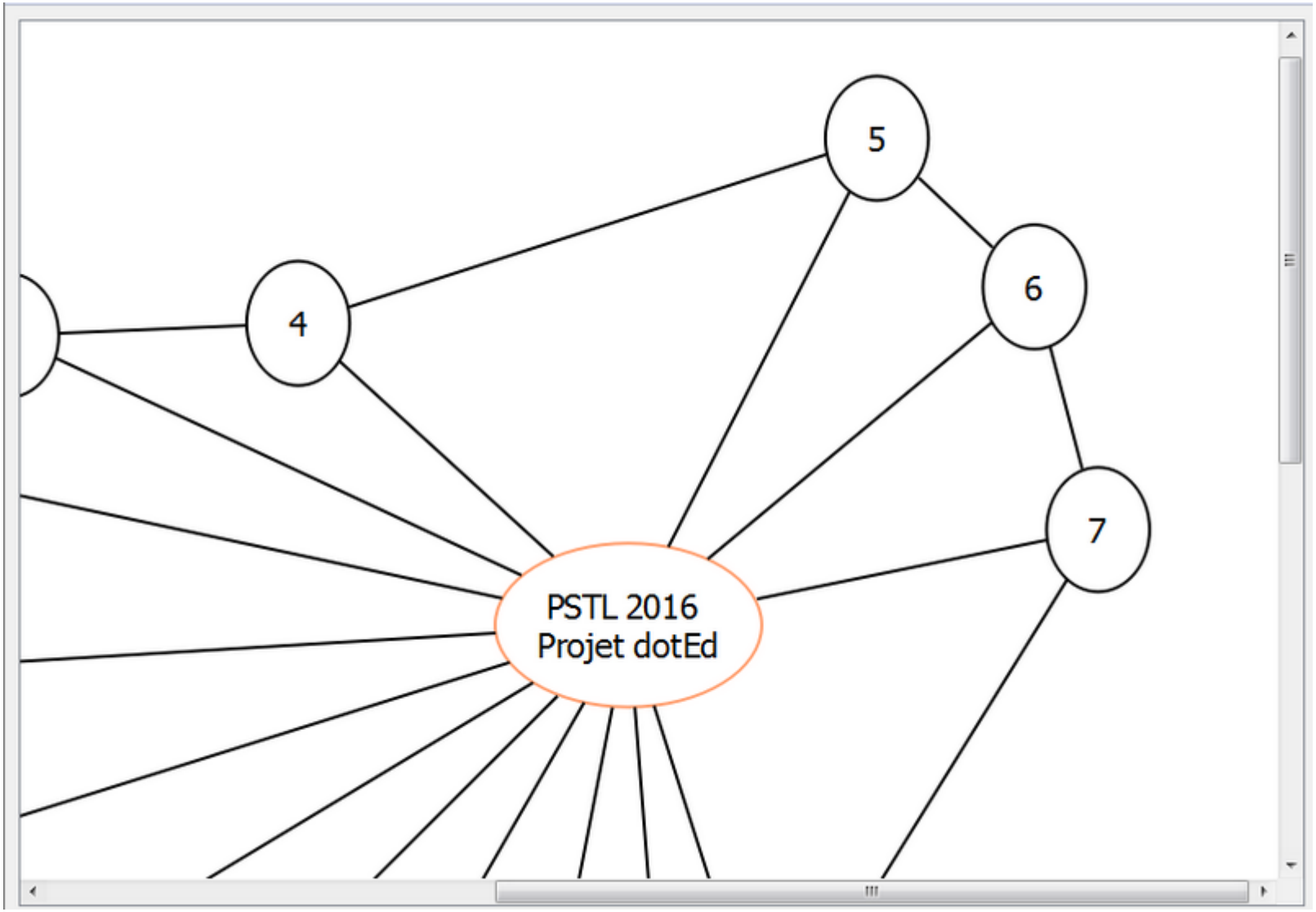
# II-2. Vue graphique (6)

**Etape 5 : déplacement de nœuds/arêtes**



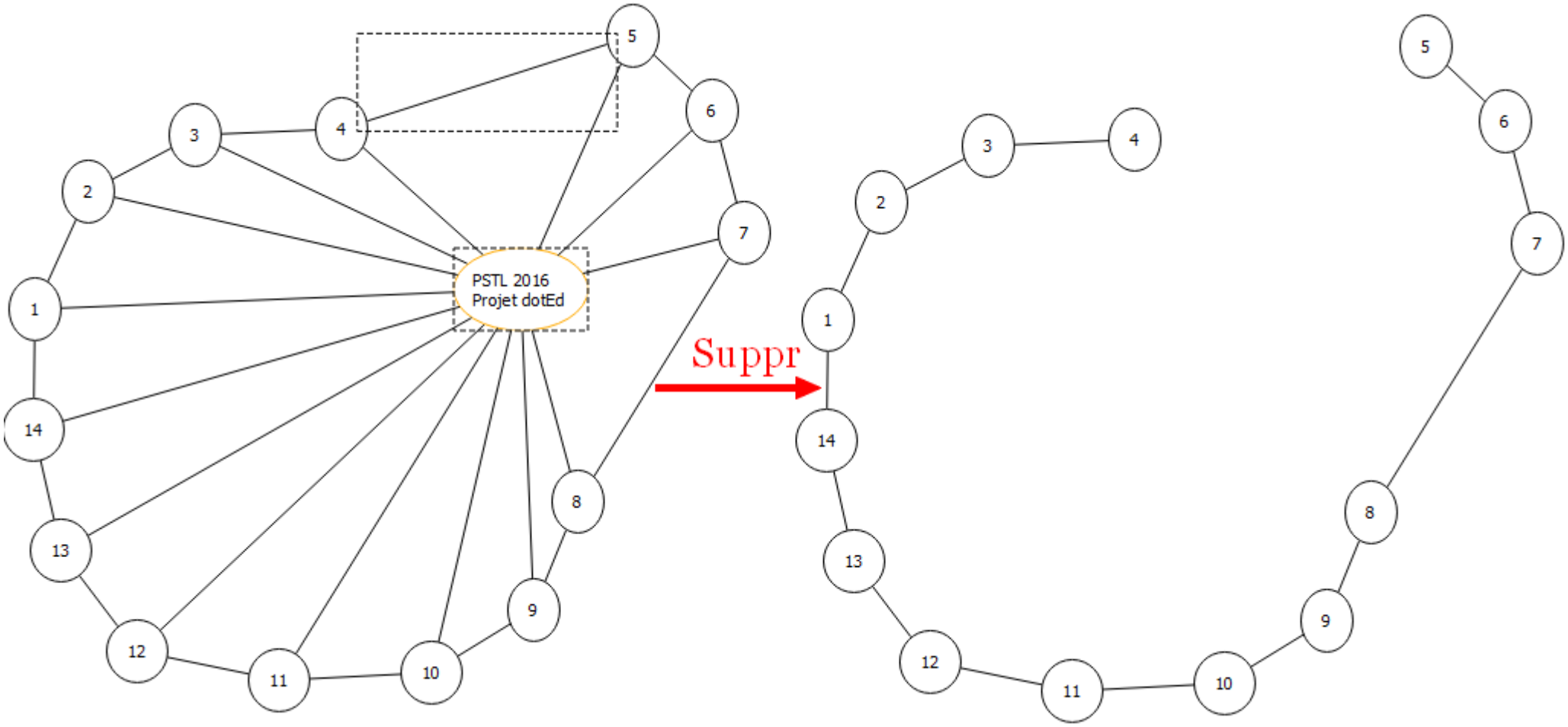
# II-2. Vue graphique (7)

**Etape 6 : zoom**



# II-2. Vue graphique (8)

**Etape 7 : suppression de nœuds/arêtes**



# III-1. Extension d'un attribut DOT

- **Démarche à suivre :**

1. sélectionner un attribut DOT : http://www.graphviz.org/doc/info/attrs.html
2. identifier les valeurs possibles
3. implémentation du code

→ ~50-300 lignes de code

# III-2. Standardisation

- Packaging et utilisation : fichier « setup.py »
- PEP : conventions et bonnes pratiques pour coder en Python
- Licence choisie : GNU GPL
- Git (projet + wiki) : https://github.com/vnea/dotEd

# Conclusion

- ~1300 lignes de codes, 1000 de commentaires
- Seule solution open source fonctionnelle ?
- Supporte l’essentiel du format
- Facilité d’utilisation
- Aller plus loin :
  - plus d’attributs
  - éléments particuliers du langage
  - ...