



Projet 30 : Graphe 3D

DOSSIER DE CONCEPTION

Objet du document	Dossier de conception et spécification du projet
Destinataires du document	N. Bonnel – L. Barneville
Groupe	J. Catric – E. Daubert – C. Lino – N. Magnin – I. Popa
Référence projet	n° 30
Version/ date	20 Mars 2007 – v 2.0

Sommaire

1	SPECIFICATION DU PROJET	5
1.1	CONTEXTE.....	5
1.2	OBJECTIFS.....	5
1.3	CIBLES	5
1.4	ENJEUX.....	5
1.5	LIVRABLES ATTENDUS.....	6
1.6	EXISTANT ET CONTRAINTES	6
1.7	RISQUES.....	6
2	CONCEPTION DE LA SOLUTION	7
2.1	DEFINITION DES FONCTIONNALITES	7
2.1.1	FONCTIONNALITES DE L'API DE GRAPHE 3D	7
2.1.2	FONCTIONNALITES DE L'EDITEUR DE GRAPHE 3D.....	7
2.2	ETUDE DE FAISABILITE.....	7
2.3	CHOIX TECHNIQUE	8
2.3.1	API JAVA3D	8
2.3.2	API JAXP	8
2.3.3	PARSEUR DOM/SAX	9
2.4	DEFINITION DU SCHEMA XSD	10
2.4.1	LES LIENS (« LINK »).....	10
2.4.2	LES NOEUDS (« NODE »).....	10
2.4.3	LES ATTRIBUTS (« ENUM-ATTR »).....	10
2.5	DIAGRAMME DE CLASSE UML	11
2.5.1	PACKAGE « PARSERXML ».....	11
2.5.2	PACKAGE « EXCEPTION ».....	11
2.5.3	PACKAGE « ELEMENTS »	11
2.5.4	PACKAGE « USE ».....	11
2.5.5	PACKAGE « LISTS »	12
2.5.6	PACKAGE « UNIVERSE ».....	12
2.5.7	PACKAGE « EDITORGRAPHE »	13
2.6	API GRAPHE 3D.....	13
2.6.1	SPECIFICATION DE L'INTERFACE DE VISUALISATION.....	13
2.6.2	CAS D'UTILISATIONS	16
2.7	EDITEUR DE GRAPHE 3D	20
2.7.1	SPECIFICATION DE L'INTERFACE DE L'EDITEUR.....	20
2.7.2	CAS D'UTILISATIONS	22
2.8	PLAN DE DEVELOPPEMENT	27
2.9	PLAN QUALITE	28
2.9.1	QUALITE DE LA CONCEPTION	28
2.9.2	HOMOGENEITE DU CODE	28
2.9.3	QUALITE DU CODE.....	29
2.9.4	AUTOMATISATION DES TACHES	29
2.9.5	TESTS	29
2.9.6	CVS.....	30
2.10	PLATE-FORME DE DEVELOPPEMENT.....	30
2.10.1	OUTILS DE DEVELOPPEMENT.....	30
2.10.2	CARACTERISTIQUES D'UNE MACHINE DE DEVELOPPEMENT	30

3	CAHIER DE TESTS.....	31
3.1	SCHEMA FONCTIONNEL	31
3.1.1	SCHEMA FONCTIONNEL GENERAL.....	31
3.1.2	SCHEMA FONCTIONNEL DE LA VISUALISATION.....	31
3.1.3	SCHEMA FONCTIONNEL DE L'EDITEUR.....	32
3.2	SCHEMA TECHNIQUE.....	33
3.3	TESTS.....	34
3.3.1	TESTS FONCTIONNELS.....	34
3.3.2	TESTS DE PERFORMANCES.....	34
3.3.3	TESTS TECHNIQUES	35
3.3.4	TESTS UNITAIRES.....	35
3.4	LISTE DES FICHES DE TESTS	35
3.4.1	FICHE DE TEST N°1.1 : CHARGEMENT	36
3.4.2	FICHE DE TEST N°2 : SAUVEGARDE	38
3.4.3	FICHE DE TEST N°2.1.2 : GESTION DES COLLISIONS.....	40
4	ORGANISATION.....	42
4.1	PHASE D'ANALYSE	42
4.1.1	CHANTIER 1 : ETUDE DE PROJET	42
4.1.2	CHANTIER 2 : CONCEPTION DETAILLEE	42
4.1.3	CHANTIER 3 : ORGANISATION.....	43
4.1.4	CHANTIER 4 : CAHIER DE TESTS.....	43
4.1.5	CHANTIER 5 : PRESENTATION ORALE DE L' ANALYSE.....	43
4.2	PHASE DE REALISATION	43
4.2.1	CHANTIER 1 : CODAGE DE L'INTERFACE DE VISUALISATION DE GRAPHE 3D	43
4.2.2	CHANTIER 2 : CODAGE DE L'EDITEUR DE GRAPHE 3D	43
4.2.3	CHANTIER 3 : CODAGE DES ELEMENTS COMMUNS	44
4.2.4	CHANTIER 4 : TESTS ET INTEGRATION	44
4.2.5	CHANTIER 5 : DOCUMENTATION.....	44
4.2.6	CHANTIER 6 : VALIDATION DU CLIENT	44
4.2.7	CHANTIER 7 : BILAN	44
5	ANNEXES	46
5.1	SCHEMA XML.....	46
5.2	EXEMPLE DE FICHIER XML.....	48
5.3	VUE 2D DE L'EXEMPLE DU FICHIER XML	52
5.4	DIAGRAMME UML DE L'API GRAPHE 3D.....	53
5.4.1	PACKAGE « USE », « LISTS » ET « PARSERXML ».....	54
5.4.2	PACKAGE « EXCEPTION ».....	55
5.4.3	PACKAGE « ELEMENTS ».....	56
5.4.4	PACKAGE « UNIVERSE ».....	57
5.4.5	PACKAGE « EDITORGRAPHE »	58
5.5	CHIFFRAGE PHASE D'ANALYSE.....	59
5.6	PLANNING PHASE D'ANALYSE	60
5.7	CHIFFRAGE PHASE DE REALISATION.....	61
5.8	PLANNING PHASE DE REALISATION	62
5.9	EXEMPLE D'UN TEST UNITAIRE AVEC JUNIT	63
5.10	FICHIER POUR LES FICHES DE TESTS.....	65
5.10.1	FICHIER « NONVALIDEXSD1.XML ».....	65
5.10.2	FICHIER « NONVALIDEXSD2.XML ».....	65

5.10.3	FICHER « NONVALIDESEM.XML ».....	65
5.10.4	FICHER « VALIDE.XML »	65
5.10.5	FICHER « COLLISION.XML »	66

1 Spécification du projet

1.1 CONTEXTE

Dans le cadre de notre 3^{ème} année de licence à l'UBS, nous devons sous la direction d'un élève de master 1 concevoir une API java permettant de visualiser des graphes en 3 dimensions. Ceci doit être réalisé à partir d'une représentation formelle d'un graphe statique (c'est à dire d'un fichier ayant un format spécifié).

Un graphe est un objet mathématique regroupant des sommets et des arcs. Ces sommets et ces arcs permettent de hiérarchiser diverses choses telles qu'un réseau informatique (les sommets représentent les différentes machines du réseau et les arcs, les connections entre ces machines).

Ce projet n'est pas basé sur un projet déjà existant mais nous pouvons tout de même nous appuyer sur deux logiciels existants : Walrus qui est un outil pour la visualisation interactive de graphes dans un espace à 3 dimensions, et WilmaScope qui est déjà une application Java3D qui crée des animations en 3D temps réel de graphes dynamiques.

Ce projet est du domaine de la visualisation pour la résolution de problèmes concernant la théorie des graphes.

1.2 OBJECTIFS

Le principal objectif de ce projet est de développer une API de Graphe 3D. Cette API devra offrir les possibilités suivantes qui sont de :

- pouvoir visualiser des graphes mathématiques en 3 dimensions,
- pouvoir naviguer dans ceux-ci en changeant l'angle de vue,
- pouvoir concevoir des graphes spécifiques à partir de l'API (possibilité d'héritage des composants des graphes).

Le second objectif de ce projet est la conception d'un éditeur qui devra permettre de pouvoir construire un graphe et d'enregistrer le fichier lui correspondant. Cet objectif peut être atteint avec un éditeur de texte permettant d'écrire facilement le fichier correspondant au graphe ou alors à l'aide de la vue en 3 dimensions où l'on pourrait construire directement le graphe.

Un troisième objectif qui découle des deux précédents est la définition d'un format spécifique (XML ou autre) pour représenter un graphe.

1.3 CIBLES

Ce projet est à destination de Nicolas Bonnel qui joue le rôle de client du projet. Mais ce projet ne lui est pas exclusivement réservé. Il peut en effet servir à n'importe quel programmeur souhaitant visualiser des graphes en 3 dimensions.

Sachant que notre travail ne consiste qu'à créer une API permettant de manipuler des graphes, une personne ne possédant pas de connaissances en programmation aurait des difficultés à utiliser notre travail.

1.4 ENJEUX

L'enjeu de ce projet est de concevoir une API facile d'utilisation pour un programmeur. Pour cela, il est nécessaire de bien documenter notre code source.

L'enjeu à titre éducatif de ce projet est de nous faire prendre conscience du fonctionnement du monde du travail. En effet, aujourd'hui, les entreprises faisant de la conception de logiciel travaillent de manière très ordonnée avec des méthodes spécifiques (travail en groupe, répartition des tâches,

gestion de projet, prévision d'exécution des tâches ...). Les documents que nous devons rendre permettent justement de suivre toutes ces étapes.

Les autres enjeux sont plus personnels comme la formation à de nouveaux outils (CVS, Eclipse, JAVA3D,...).

1.5 LIVRABLES ATTENDUS

Les livrables attendus par le client sont :

- un fichier jar contenant l'API Java Graphe 3D et éventuellement un fichier jar contenant l'API Java3D,
- un programme simple qui illustre l'utilisation de l'API Graphe 3D afin de fournir au client ainsi qu'aux potentiels utilisateurs un support pour comprendre le fonctionnement de l'API,
- une Javadoc la plus claire et précise possible, qui sera rédigée en français ou en anglais.

1.6 EXISTANT ET CONTRAINTES

Comme expliqué dans la partie contexte, ce projet n'est pas la poursuite d'un projet existant. Cependant, il existe d'autres projets qui permettent la visualisation de graphe en 3 dimensions. Ces projets sont principalement Walrus et WilmaScope.

Pour réaliser ce projet, nous devons travailler avec le langage JAVA (1.5) ainsi qu'avec l'API JAVA3D. Cette dernière permet de concevoir des visualisations d'objets en 3 dimensions.

Le code source doit être commenté efficacement afin d'assurer la maintenabilité de l'application et une génération optimale de la documentation Javadoc.

Il nous a été imposé d'avoir recours à CVS qui est un outil efficace afin que nous acquérions des compétences par son utilisation. CVS permet de faciliter le travail de développement en équipe, tout en permettant la collaboration de multiples intervenants sur le même projet. Chacun des participants au projet n'accède jamais qu'à une copie des fichiers tandis que les originaux demeurent sur le « référentiel ». CVS conserve les révisions successives grâce à un historique complet.

Pour le codage, nous devons utiliser l'environnement de développement Eclipse. Eclipse permet la gestion de projet, l'automatisation des tâches avec Ant, une utilisation facilitée du CVS, une génération de la documentation Javadoc aisée...

1.7 RISQUES

Ce projet comporte, comme tout projet, des risques. Ceux-ci concernent les prévisions faites sur le projet (mauvaise évaluation du temps pour chaque tâche) ce qui entraînerait un dépassement des délais. Etant donné que les délais ne peuvent être aménagés, ce ne serait pas un dépassement de ceux-ci mais une diminution du travail effectué (moins de fonctions implémentées dans l'API, abandon de certains objectifs secondaires).

Un autre risque envisageable serait que le résultat fourni ne soit pas le résultat attendu. Ce risque doit être envisagé même si normalement, il doit être étudié durant la phase de spécification.

2 Conception de la solution

2.1 DEFINITION DES FONCTIONNALITES

2.1.1 Fonctionnalités de l'API de Graphe 3D

Les différentes fonctionnalités offertes par l'API de Graphe 3D sont les suivantes :

- Le chargement d'un graphe
- L'agrandissement ou la réduction du graphe
- La rotation du graphe verticalement et horizontalement
- Le recentrage de la vue par rapport à l'élément sélectionné (nœud ou lien)
- La sélection d'élément(s) dans le graphe (nœud ou lien)
- Affichage des informations de l'élément(s) sélectionné(s)
 - Élément nœud : affichage des liens reliés au nœud et les attributs du nœud.
 - Élément lien : affichage des nœuds reliés au lien et les attributs du lien.

2.1.2 Fonctionnalités de l'éditeur de Graphe 3D

Les différentes fonctionnalités offertes par l'API de Graphe 3D pour l'éditeur sont les suivantes :

- Le chargement d'un graphe
- La création d'un nouveau graphe
- La sauvegarde d'un graphe
- La création d'élément (nœud ou lien)
- La sélection d'élément(s) (nœud ou lien)
- La suppression d'élément(s) (nœud ou lien)
- La modification des attributs de l'élément sélectionné (nœud ou lien)
- Le recentrage de la vue par rapport à l'élément sélectionné (nœud ou lien)

2.2 ETUDE DE FAISABILITE

Suite aux recherches effectuées par les différents membres de l'équipe, des solutions ont été trouvées afin de réaliser les différentes fonctionnalités proposées.

Les fichiers XML seront parsés en utilisant l'api DOM de java. La vérification de la validité d'un fichier XML sera effectuée grâce à un schéma XSD. Pour ce faire, nous utiliserons la classe GXmlParser qui prend en paramètre le fichier XML et qui essaie de créer le graphe représenté dans ce fichier.

Pour l'implémentation de la vue 3D du graphe nous utiliserons l'API Java3D, en créant un arbre des éléments du graphe avec la classe GGraphUniverse : elle contiendra la vue, héritera de VirtualUniverse, contiendra aussi les vues des nœuds avec la classe GNodeView et des liens avec la classe abstraite GLinkView dont hériteront GArrowView et GBridgeView, et enfin un objet de type Graphe qui représentera la vue logique du graphe. Cet élément GGraphUniverse fournira une méthode getCanvas() qui renverra le Canvas3D contenu dans la vue.

La sélection d'éléments est déjà implémentée dans l'API Java3D. Pour faire un zoom ou une rotation autour du graphe nous pourrons récupérer l'élément ViewPlatform associé à la vue et lui ajouter un ou plusieurs élément(s) Transform3D. Cette classe fournit les méthodes :

- setTranslation(Vecteur3f) pour le zoom

- o `rotX(double), rotY(double), rotZ(double)` pour les rotations (on n'utilisera que `rotX` et `rotY` en admettant que nous nous plaçons dans le plan (O, x, y)).

Pour implémenter l'éditeur de graphes nous utiliserons l'API Swing. Nous utiliserons une `JTabbedPane` pour les onglets et chaque élément sélectionné dans un onglet utilisera un `JPanel` inséré dans un `JScrollPane` et qui contiendra tous les attributs de l'élément sous forme de table.

Pour visualiser les liens et les nœuds reliés à l'élément courant nous utiliseront une `JList` avec un ascenseur qui sera créé en mettant la `JList` dans un `JScrollPane`.

Enfin la création d'un nouvel élément se fera via un `JButton` associé à un `JChoice` qui contiendra les types de nœuds ou de liens connus. Ceux-ci seront contenus dans des fichiers avec le schéma d'entrées suivant :

- o `(n|b|a):NomDuType:NomDeLaClasse` (une seule entrée par ligne). Où `n` signifie « node » (nœud), `b` signifie « bridge » (arête) et `a` désigne « arrow » (arc).

Les classes pourront ensuite être instanciées grâce à la méthode `forName` de `Class` :

`Class c = Class.forName(class_name)` ce qui nous permet de récupérer l'objet `Class` correspondant, puis la méthode `c.newInstance()` qui crée une nouvelle instance de cette classe (il y a aussi des méthodes qui permettent d'accéder à ses constructeurs, méthodes et attributs).

Pour la création d'un nœud, le bouton et le choix seront accompagnés d'un triplet de champs de texte formatés (nombres réels) qui contiendront les coordonnées (x, y, z) voulues dans le graphe. Pour ceci nous utiliseront 3 `JFormattedTextField` créés comme suit :

```
DecimalFormat format = (DecimalFormat) new DecimalFormat.getInstance();
format.setDecimalSeparatorAlwaysShown(true);
JFormattedTextField ftf = new JFormattedTextField(format);
```

Remarque : on pourra de même contraindre les valeurs possibles dans la table des attributs d'un élément selon le type déclaré.

2.3 CHOIX TECHNIQUE

2.3.1 Api Java3D

L'Api Java3D est une API en Java utilisant les techniques de programmation graphique en 3 dimensions. La bibliothèque de classes de Java3D est destinée à créer des scènes 3D (utilisation de formes complexes, d'éclairages, de textures, d'animations, de sons ...).

Le package Java3D est une extension du langage Java, Java3D est actuellement disponible sous Windows, Linux, Solaris, Mac OS.

La 3D est grande consommatrice de calculs, la puissance de traitement de Java3D dépend donc de la puissance du microprocesseur mais aussi de celle de la carte graphique utilisée!

La version de l'Api Java 3d qui sera utilisée sera : "java3d-1_5_0". Le package Java3d qui sera implémenté dans notre Api Graphe3D est `javax.media.j3d.*`.

Nous utiliserons l'Api Java3D pour représenter/modéliser un graphe : des sphères pour les nœuds, des lignes pour les liens...

2.3.2 Api JAXP

Java s'est vue adjoindre deux API spécialisées dans le traitement de document XML. Elles sont toutes les deux fournies dans un ensemble de packages connus sous le nom de JAXP (Java API for XML Parsing).

2.3.2.1 Langage XML

XML (eXtensible Markup Language) est un langage d'échange et de structuration d'informations.

En XML, le choix des balises est laissé libre à l'auteur du document. Ces choix sont définis dans ce que l'on appelle une DTD (Document Type Definition). Cette DTD permet de valider une construction valide pour un document donné.

2.3.2.2 Le schéma XSD

C'est le même principe qu'une DTD. Le choix du schéma XSD plutôt qu'une DTD s'est fait par la préférence du client mais aussi par la puissance fournie par le XSD. En effet, le schéma XSD est un document sous format XML. Le schéma est donc plus extensible et se lit plus facilement qu'un DTD même si celle-ci est plus concise.

De plus le schéma XSD offre plus de possibilité que la DTD telles que la gestion d'espaces de nommage et le typage des données..

2.3.2.3 Librairie JAXP

L'API JAXP est intégrée dans java 1.5.

La librairie JAXP (Java API for XML Parsing), proposée par SUN Micro system permet de traiter des documents XML. Cette librairie propose des implémentations de deux modèles : SAX et DOM.

2.3.3 Parseur DOM/SAX

2.3.3.1 Description de DOM/ SAX

DOM est l'acronyme de *Document Object Model*. Le principal rôle de DOM est de fournir une représentation mémoire d'un document XML sous la forme d'un arbre d'objets et d'en permettre la manipulation (parcours, recherche et mise à jour).

A partir de cette représentation (le modèle), DOM propose de parcourir le document mais aussi de pouvoir le modifier. Ce dernier aspect est l'un des aspects le plus intéressant de DOM. DOM est défini pour être indépendant du langage dans lequel il sera implémenté.

SAX est l'acronyme de *Simple API for XML*. Ce type de parseur utilise des événements pour piloter le traitement d'un fichier XML. Un objet doit implémenter des méthodes particulières définies dans une interface de l'API pour fournir les traitements à réaliser : selon les événements, le parseur appelle ces méthodes.

L'avantage de SAX est qu'il traite les documents élément par élément au fur et à mesure qu'ils sont rencontrés. Pour chaque élément (balise, commentaire, texte), la fonction de rappel correspondante est appelée. C'est pourquoi ce mode d'interprétation des documents XML utilise moins de mémoire, car SAX n'accumule aucune donnée dans une structure.

2.3.3.2 Aspects techniques

Les méthodes SAX et DOM adoptent chacune une stratégie très différente pour parser les documents XML, elles s'utilisent donc dans des contextes différents. DOM charge l'intégralité d'un document XML dans une structure de donnée, qui peut alors être manipulée puis reconvertie en XML. Cependant pour cela il faut que la taille de la structure représentant le document XML ne soit pas supérieure à ce que peut contenir la mémoire vive, dans notre cas le fichier XML contenant la description du graphe 3D respecte cette condition.

DOM est essentiellement utilisé pour pouvoir modifier facilement des documents XML ce qui nous sera utile pour sauvegarder les caractéristiques du graphe, qui auront été changées par l'utilisateur grâce à l'éditeur, dans un nouveau document XML.

2.3.3.3 Notre choix

DOM nous paraît la solution la mieux adaptée dans le cadre de notre projet. Nous pourrions parser le document XML contenant les caractéristiques du graphe dans le but de charger le graphe pour l'interface de visualisation mais également de modifier ces caractéristiques grâce à l'éditeur.

L'apprentissage de SAX est plus complexe que DOM et donc plus long, de plus certaines personnes de notre groupe de projet ayant déjà utilisé DOM cela nous fera gagner du temps pour comprendre et savoir utiliser DOM.

2.4 DEFINITION DU SCHEMA XSD

Un document XML est décrit par un schéma XSD qui sert à donner la forme général d'un document représentant un graphe. Ce document sera aussi bien utiliser par l'interface de visualisation de graphe en 3D qu'avec l'éditeur de graphe 3D qui pourra le modifier.

On définit un graphe par son nom et un ensemble de liens et de nœuds.

2.4.1 Les liens (« link »)

Il existe 3 types de lien :

- boucle (loop) un lien qui relie un nœud à lui même
- arc (arrow) un lien orienté
- arête (bridge) un lien non-orienté

On ne considère pas une boucle comme un type spécifique de lien car une boucle peut être un arc ou une arête sur un nœud.

Les liens sont formés d'un ensemble d'attributs « enum-attr » ainsi qu'un « type » (« arrow » ou « bridge »), une « class » qui définit la classe Java a utilisé pour définir ce genre de lien, deux nœuds correspondant aux extrémités du lien et un identifiant « name » afin de pouvoir faire plusieurs liens entre deux nœuds.

Si c'est un lien de type « arrow », la définition des deux nœuds a de l'importance : « link_start », « link_end » pour spécifier l'orientation du lien.

Pour une boucle, « link_start » et « link_end » sont de même valeur. C'est-à-dire que le nœud est le même.

2.4.2 Les nœuds (« node »)

Les nœuds sont eux aussi formés d'un ensemble d'attributs « enum-attr ».

Ils possèdent aussi un « name » qui sert d'identifiant et qui est unique parmi tous les nœuds. Comme pour les liens, ils possèdent un attribut « class » qui définit la classe Java a utilisé pour définir ce genre de nœud.

Les nœuds possèdent en plus un élément « coordonates » qui définit les coordonnées du nœud x, y, z.

2.4.3 Les attributs (« enum-attr »)

Les attributs varient selon la classe de lien ou de nœud (voir attribut « class »).

Un « enum-attr » contient un attribut « attr-name » qui définit l'identifiant de l'attribut dans la classe, un attribut « attr-value » qui définit la valeur de cet attribut et enfin « attr-type » qui définit le type de l'attribut.

Le type de l'attribut ne peut correspondre qu'à un type primitif de Java : « short », « byte », « int », « long », « double », « char », « String ».

Que ce soit pour les liens ou les nœuds, il faut obligatoirement que la classe définie existe et que les attributs définis par les éléments « enum-attr » existent dans cette classe.

Remarque : Le [schéma XSD](#) et un [exemple de fichier XML](#) (et une [vue en 2D](#)) sont en annexe.

2.5 DIAGRAMME DE CLASSE UML

Nous avons volontairement choisis de représenter sur le même diagramme, le diagramme de la fenêtre de visualisation et celui de l'éditeur.

En effet, l'éditeur n'est pas tellement différent de la visualisation, puisque contrairement à celle-ci, les listes des attributs et des connexions sont éditables pour l'éditeur et quelques boutons apparaissent en plus des listes (« créer nœud », « supprimer élément » « créer lien(s) », etc.

L'API Graphe 3D possède 7 packages principaux :

- parserXML
- exception
- elements
- use
- lists
- universe
- editorGraphe

Remarque : Le [diagramme de classe](#) est en annexe. Au vue de la complexité du diagramme de classe, ce dernier a été découpé en plusieurs morceaux suivant les packages afin qu'il soit lisible.

2.5.1 Package « parserXML »

Ce package regroupe les classes utilisées pour le parsing des fichiers XML. Il n'y a qu'une seule classe qui fait le parsing et qui crée un objet de type « GGraphe » défini dans le package « elements ».

2.5.2 Package « exception »

Ce package permet de gérer les exceptions dues aux parseurs, à la visualisation, et à l'éditeur. Il contient différentes classes correspondant à chaque exception qui pourrait survenir lors de l'utilisation de l'API.

La classe principale est la classe « GException » qui hérite de la classe « Exception » défini dans Java. « GException » permet d'afficher les messages d'erreurs obtenus soit dans une console (« printStackTrace » hérité de « Exception ») ou dans une boîte de dialogue (« showError »).

Toutes les autres classes héritent de « GException » et redéfinissent les deux méthodes citées ci-dessus.

2.5.3 Package « elements »

Ce package correspond à la vue logique d'un graphe mathématique. Dans ce package, est défini la classe « GGraphe » qui représente l'ensemble du graphe ainsi que « Gnode » qui représente les nœuds et « Glink » qui représente les liens. Sachant que tous les nœuds et les liens sont listés sous forme de « Hashtable » dans un objet de type « GGraphe ».

2.5.4 Package « use »

Ce package offre aux programmeurs une interface d'utilisation directe des éléments proposés dans l'API. En effet, par l'intermédiaire de cette interface, on peut directement récupérer les composants de l'API (vue en 3D, listes des attributs et des connexions gestion des interactions clavier et souris)

Ce package comporte une interface (« GPanel ») au sens Java du terme qui définit des fonctions pour récupérer les composants voulus. Deux classes implémentent cette interface :

- « GPanelEdition » qui étend « Jpanel » et qui permet de récupérer la vue 3D ainsi que les listes afin qu'elles soient éditables.
- « GPanelVisualization » qui étend aussi « Jpanel » et qui permet de récupérer la vue en 3D et les listes mais elles ne peuvent être éditées.

2.5.5 Package « lists »

Ce package correspond aux objets définissant les listes d'attributs et de connexions. Il y a deux classes dans ce package :

- « GattributesList » qui correspond à la liste des attributs. Cette classe permet de créer des onglets dans la liste des attributs ou d'en supprimer en fonction des éléments sélectionnés dans la vue 3D
- « GConnectionsList » correspond à la liste des connexions de l'élément sélectionné. Si plusieurs éléments apparaissent dans la liste des attributs, c'est l'élément courant qui est pris en compte et qui correspond à la liste des connexions.

2.5.6 Package « universe »

Ce package correspond à tous les composants utiles pour la composition de la représentation graphique du graphe mathématique.

La classe principale est « GGrapheUniverse » qui regroupe l'ensemble des représentations graphiques des « Gnode » et des « GLink ». « GGrapheUniverse » est la classe par laquelle se feront les interactions avec l'utilisateur. En effet, lorsque ce dernier cliquera dans la vue, c'est cette classe qui va gérer les événements (rotation, zoom et sélection de noeud(s) ou de lien(s)).

Concernant la gestion de la sélection et les interactions avec les listes d'attributs et de connexions, nous devons écrire un comportement spécifique. Ce comportement correspond à l'écriture d'une classe héritant de « Behavior » qui par l'intermédiaire de deux méthodes (initialize() et processStimulus) permet de définir les actions à exécuter selon l'évènement qui survient (dans notre cas, une sélection). Cette classe sera une classe interne de « GGrapheUniverse » et se nommera « SelectionBehavior ».

La représentation graphique d'un « Gnode » correspond à la classe « GNodeView » qui hérite de la classe « TransformGroup » (classe de Java3d) et associe un élément de type « Sphere » qui correspond à la forme graphique que l'on pourra voir dans la vue 3D.

L'utilisation de la classe « TransformGroup » comme super-classe permet d'interagir directement avec la vue 3D plutôt que d'utiliser la « Sphere » comme super-classe qui nous aurait obligé à rechercher l'objet « Sphere » à partir du « TransformGroup » qui le contiendrait.

En plus de posséder un objet de type « Sphere », « GNodeView » possède aussi un objet de type Transform3D qui permet d'agir directement sur la vue 3D du noeud. Cette classe possède en plus un objet de type « Appearance » qui permet d'utiliser des textures sur la visualisation de la sphère afin d'afficher une sphère plus esthétique.

La classe « GLinkView » est elle aussi une sous-classe de « TransformGroup » et possède les mêmes objets que « GNodeView » excepté pour l'objet de type « Sphere ». En effet la représentation graphique d'un lien sera un trait et non une sphère. C'est pour cela que « GLinkView » possède un objet de type « LineArray » qui permet de tracer des traits en 3D.

La classe GlinkView est une classe abstraite. En effet, un lien est obligatoirement un arc (arrow) ou une arête (bridge). La différence vient du fait que le premier est orienté et pas le second. Voilà pourquoi deux classes héritent de GlinkView :

- GArrowView qui représente un arc (lien orienté)
- GbridgeView qui représente une arête (lien non orienté).

La différence principale entre ces deux classes se fait dans l'utilisation de l'objet « LineArray » qui permet de faire, selon la définition de l'objet, une ou plusieurs lignes. On fera plusieurs lignes avec cet objet pour construire une flèche sur le trait représentant un arc.

Le dernier élément de ce package est « Gview ». Cette classe permet de manipuler la position de la caméra. C'est-à-dire de changer l'angle de vue. On peut ainsi voir le graphe sous n'importe quelle angle.

Pour définir la vue, il faut définir 6 éléments. C'est pourquoi « GView » hérite de l'un d'entre eux et possède les 5 autres. La superclasse de « GView » est la classe « BranchGroup ». C'est un objet de cette classe qui doit contenir les éléments nécessaires à la manipulation de la vue. Les 5 autres éléments sont :

- « View » qui définit la vue et permet d'initialiser ce que l'utilisateur verra dans sa fenêtre (un « Canvas3D »)
- « Canvas3D » qui correspond à ce qui sera affiché
- « PhysicalEnvironment »
- « PhysicalBody »
- « ViewPlatform »

Les 3 derniers éléments servent à positionner la vue par rapport aux référentiels x, y, z.

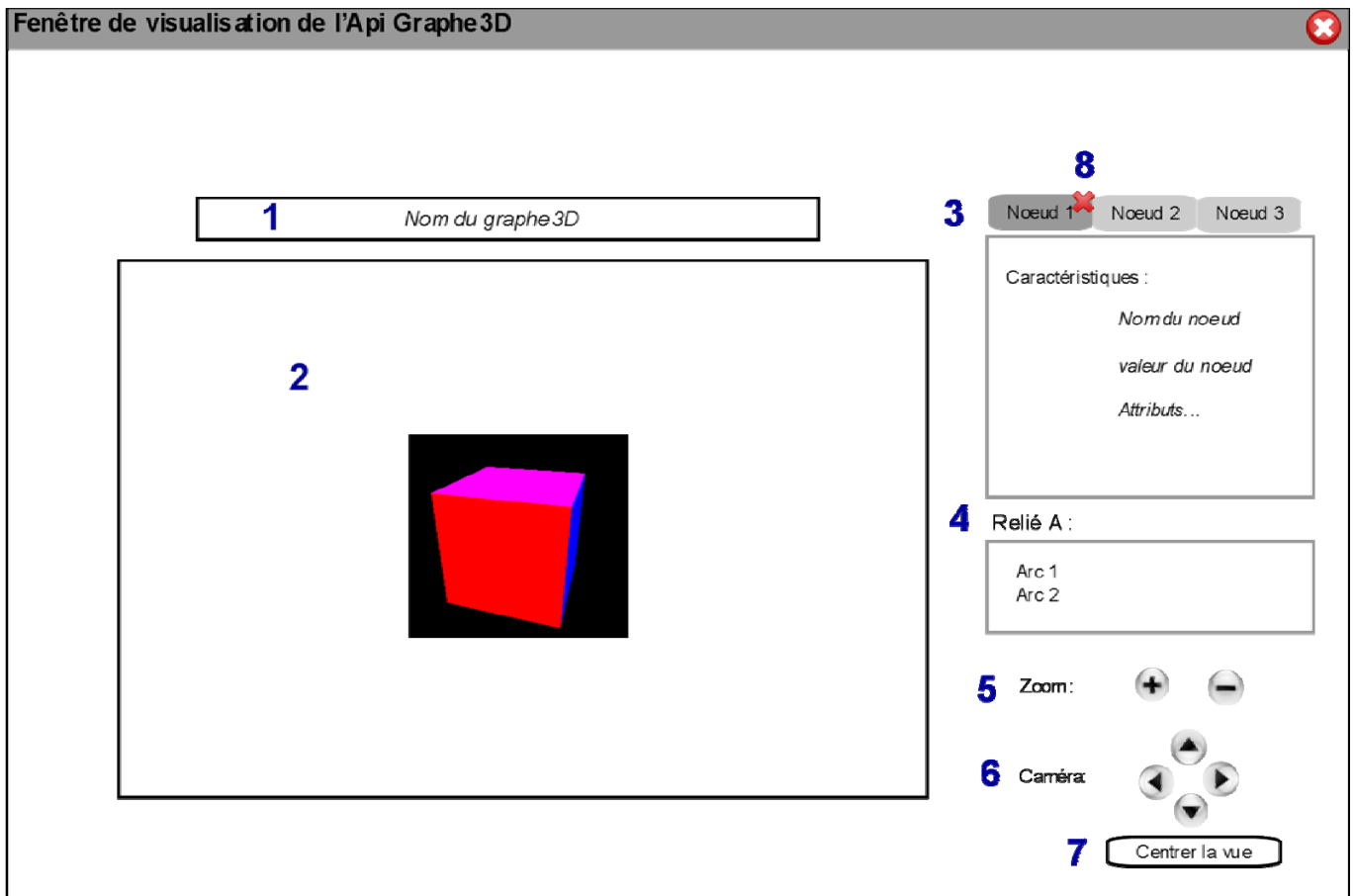
2.5.7 Package « editorGraphe »

Il y a une classe dans ce package qui permet de créer l'éditeur (« main ») ainsi qu'une classe qui représente la fenêtre de l'éditeur (« GEditor ») et qui hérite de « JFrame » (classe de Java). C'est un objet de cette classe qui sera créé dans le main et qui permettra d'avoir la fenêtre de l'éditeur. Cette classe fait appel à une des classes du package « use » qui permet de créer la vue en 3D et/ou les listes d'attributs et de connexions.

2.6 API GRAPHE 3D

2.6.1 Spécification de l'interface de visualisation

2.6.1.1 Maquette de l'interface de visualisation



Ceci est la maquette de l'interface de visualisation de l'exemple qui sera livré à la fin du projet. Dans l'avenir l'utilisateur pourra importer les éléments (encadrés par des tirets) désirés, séparément et donc créer sa propre interface de visualisation.

2.6.1.2 Légende de l'interface de visualisation

1. JTextField contenant le nom du graphe 3D.
2. JPanel contenant un Canvas3D qui permet de visualiser le graphe 3D.
3. Le(s) nom(s) de(s) objet(s) sélectionné(s) sont affichés dans des onglets (JTabbedPane). Les caractéristiques de l'objet sélectionné sont énumérées dans une JList.
4. JList affichant les éléments auxquels l'élément sélectionné dans l'onglet au dessus est relié.
5. JButton permettant de zoomer ou de dé-zoomer sur le graphe 3D.
6. JButton permettant de déplacer la vue de la caméra autour du graphe 3D.
7. JButton permettant de recentrer la vue sur le(s) élément(s) sélectionné(s).
8. Affichage d'une petite croix sur l'onglet pour le désélectionner.

2.6.1.3 Explication du fonctionnement de la maquette de visualisation

2.6.1.3.1 La scène 3D

La scène 3D dans l'interface de visualisation sera un JPanel qui contiendra un canvas3D, le canvas3D aura été importé à partir de l'api graphe3D.

Le canvas3D sera automatiquement implémenter des actions de la souris et du clavier (MouseListener...) qui permettra à l'utilisateur de sélectionner un nœud, bouger l'angle de la caméra...

Au chargement de l'interface la vue du graphe se fera dans sa totalité (tous les nœuds du graphe seront visibles dans le canvas3D).

Les nœuds du graphe seront représentés par une sphère (type : Sphere) possédant une texture permettant de mieux percevoir l'effet de 3D ainsi qu'une couleur pour différencier les sphères entre elles. Pour les liens nous utiliserons des lignes (type : LineArray), comme pour les sphères l'utilisateur aura la possibilité de choisir une couleur pour les différencier.

2.6.1.3.2 Caractéristiques de(s) l'objet(s) sélectionné(s) :

Lorsque l'utilisateur présélectionne (clic) sur un ou plusieurs objets du graphe, le nom de(s) l'objet(s) s'affiche dans des onglets. L'utilisateur pourra sélectionner les objets présélectionnés dans les onglets en cliquant sur l'onglet de son choix, cela mettra à jour la liste des caractéristiques pour le nouvel objet sélectionné. Caractéristiques d'un objet :

- Nom de l'objet
- Valeur de l'objet
- Orientation (cas où l'objet est un arc)
- Attributs

La liste des objets reliés (en relation) avec l'objet sélectionné dans un onglet sera également mise à jour lors d'une sélection d'un autre objet. L'utilisateur aura également la possibilité de cliquer sur un objet de liste des objets reliés qui mettra à jour l'onglet, la liste des caractéristiques ainsi que la liste des objets reliés.

Exemple lorsque l'utilisateur sélectionne un nœud, un onglet s'affiche avec le nom du nœud sélectionné. Ensuite les caractéristiques de l'objet s'affichent dans une liste. Dans le cas où l'utilisateur a sélectionné plusieurs objets il peut naviguer dans les caractéristiques des différents objets sélectionnés en cliquant que les onglets correspondants. Puis vu que l'utilisateur a sélectionné un nœud la liste des objets en relation avec ce nœud sera une liste de liens. La liste affichera tous les noms des liens qui seront relié à ce nœud. L'utilisateur aura la possibilité de cliquer sur le nom d'un lien, ceci effacera les onglets précédents pour afficher un nouvel onglet avec le nom de lien sélectionné.

L'ensemble (les onglets avec la liste des caractéristiques et la liste des objets reliés) sera placé dans un JPanel qui pourra directement être importé à partir de l'api graphe3D.

2.6.1.3.3 Interaction

Voici la liste des différentes interactions possible de l'interface de visualisation de graphe en 3 dimensions :

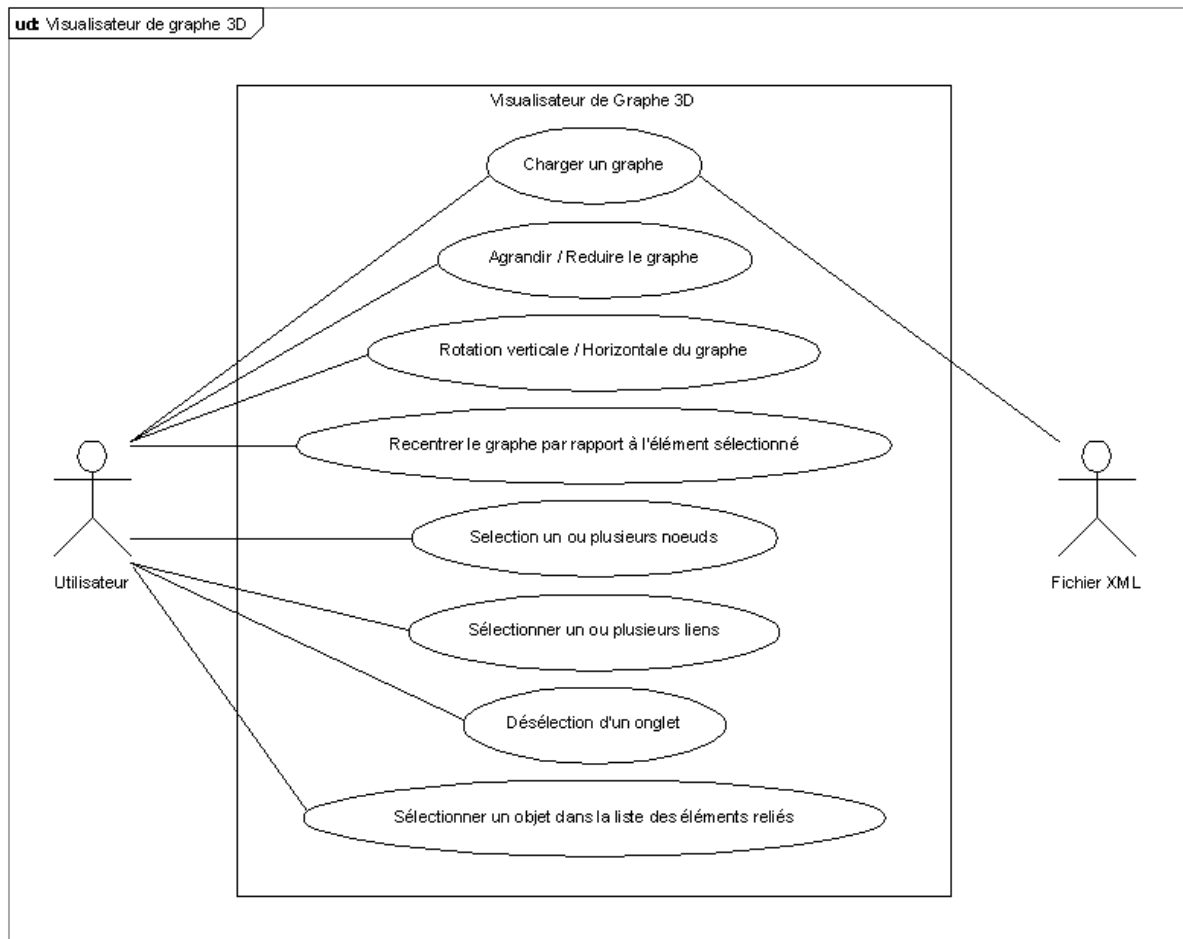
- Zoom : L'utilisateur aura le choix entre zoomer en cliquant sur les boutons prévus à cette effet (bouton + et -) où directement avec la souris ainsi qu'avec les touches flèches monter et descendre du clavier.
- Mouvements de la caméra : L'utilisateur pourra modifier l'angle de la caméra en cliquant sur les boutons prévus à cet effet, mais également avec en bougeant le graphe avec la souris où avec les touches flèches gauche de droite du clavier.
- Recentrage de la vue : Le bouton "recentrer la vue" permet de recentrer la vue sur le(s) élément(s) sélectionné(s). Si aucun élément n'est sélectionné cela ramène le graphe dans sa vue de départ (tous les objets sont visibles dans le canvas3D).

L'ensemble des boutons d'interaction sera placé dans un JPanel qui pourra directement être importé à partir de l'api graphe3D.

L'utilisateur pourra désélectionner un objet en cliquant sur la croix situé dans l'onglet courant (onglet de l'objet que l'on veut désélectionner). Cette action effacera l'onglet de l'objet ainsi que la liste de ses caractéristiques et des objets qui lui sont reliés.

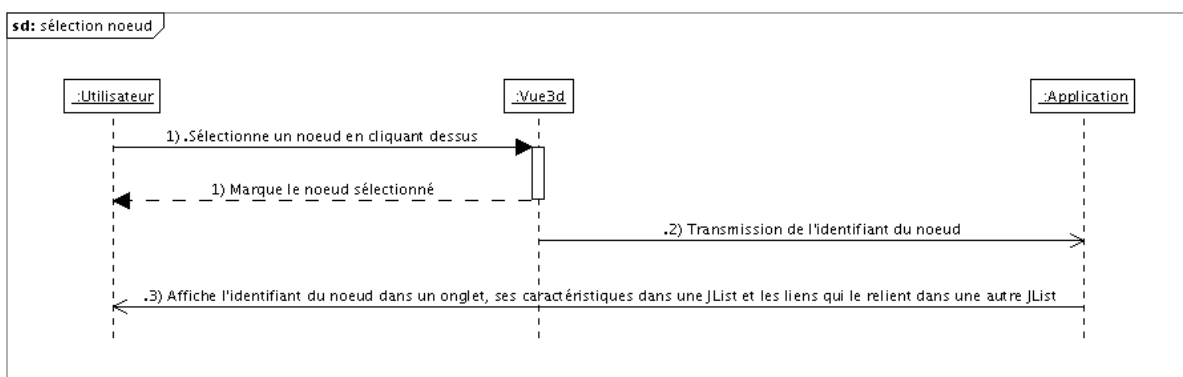
2.6.2 Cas d'utilisations

2.6.2.1 Diagramme de cas d'utilisation



2.6.2.2 Diagrammes de séquence

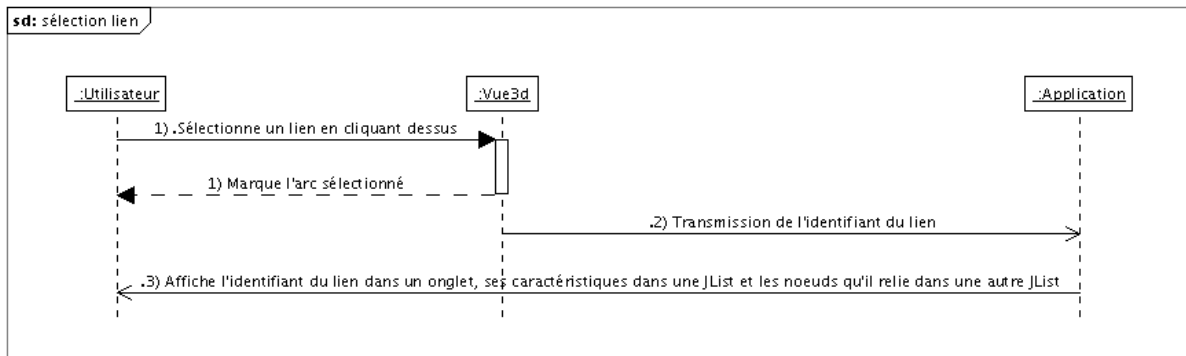
2.6.2.2.1 Sélection d'un ou plusieurs noeuds



Scénario de remplacement :

1. En maintenant la touche « ctrl » enfoncée l'utilisateur peut sélectionner plusieurs noeuds en cliquant sur plusieurs noeuds du graphe, tous les identifiants des noeuds sélectionnés seront alors transmis à l'application.

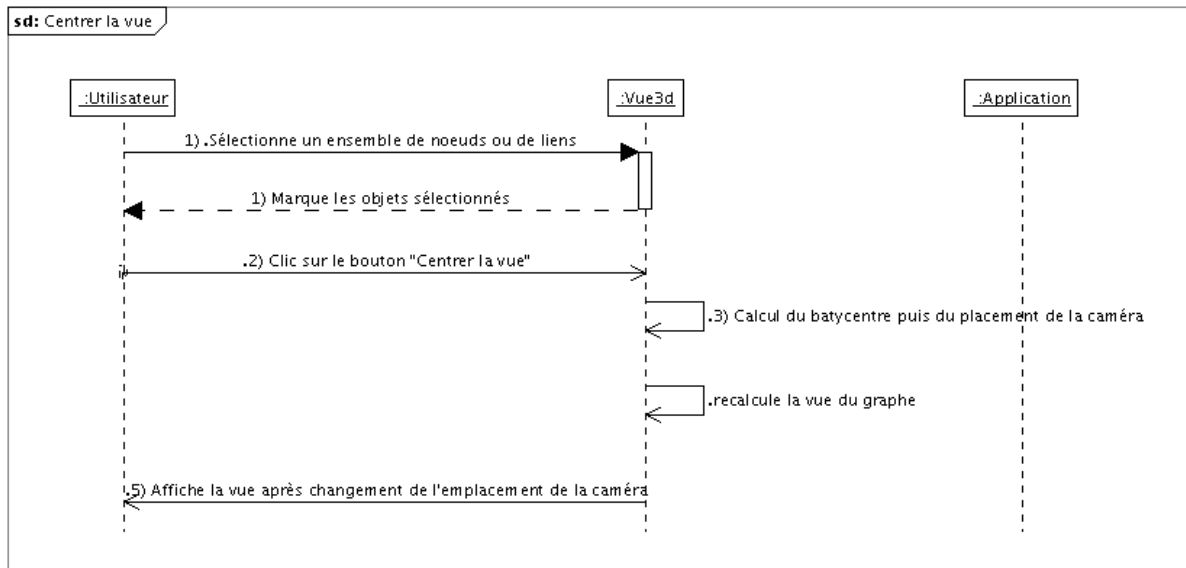
2.6.2.2.2 Sélection d'un ou plusieurs liens



Scénario de remplacement :

1. En maintenant la touche « ctrl » enfoncée l'utilisateur peut sélectionner plusieurs liens en cliquant sur plusieurs liens du graphe, tous les identifiants des liens sélectionnés seront alors transmis à l'application.

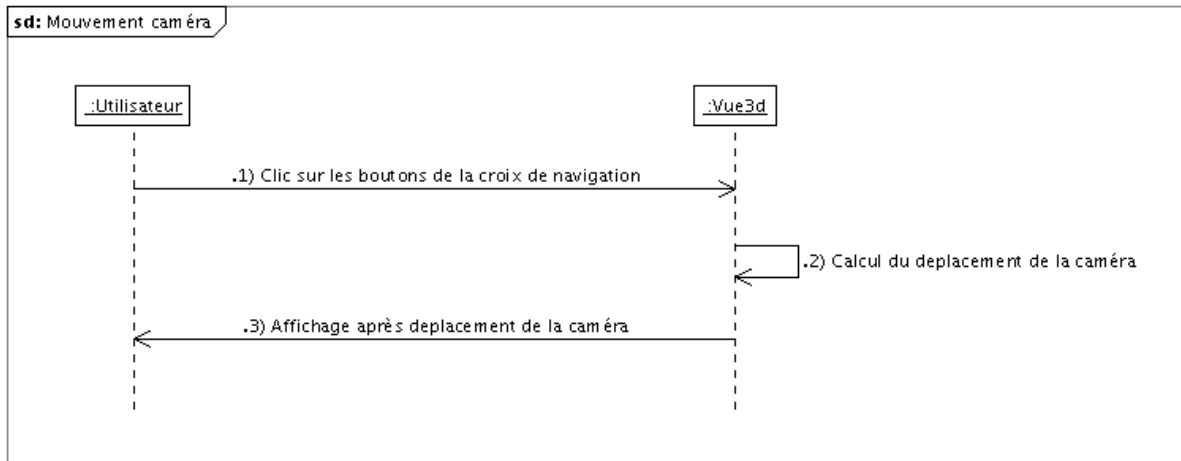
2.6.2.2.3 Recentrer la vue par rapport à l'élément sélectionné



Scénario de remplacement :

Si l'utilisateur n'a pas sélectionné d'objets et clic sur le bouton "Centrer la vue" la vue du graphe reviendra dans sa position d'origine (tous le graphe sera visible dans la fenêtre de visualisation).

2.6.2.2.4 Rotation horizontale / Verticale du graphe

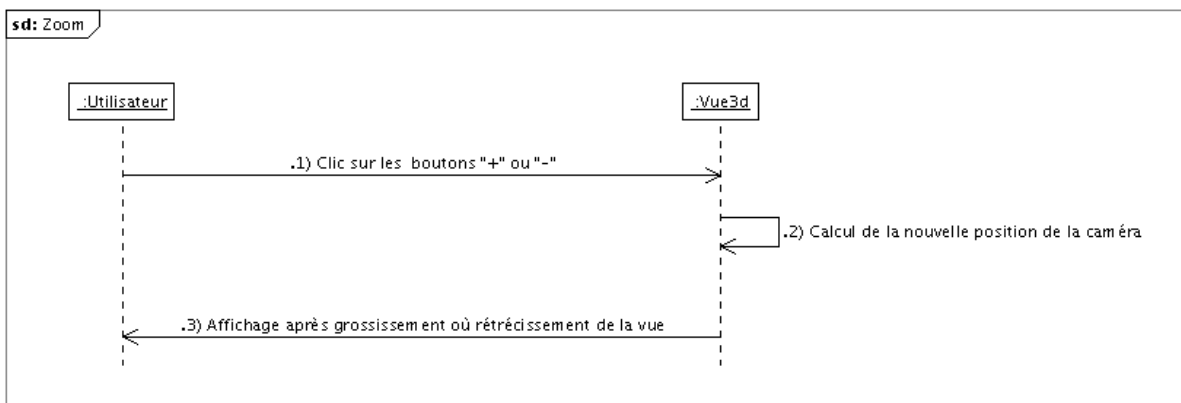


Scénario de remplacement :

Pour bouger la caméra l'utilisateur aura le choix d'utiliser les boutons de la croix de navigation mais il pourra également le faire avec la souris en effectuant un "drag" avec le bouton gauche de la souris, ainsi qu'avec les touches "flèche droite" et "flèche gauche" du clavier (la rotation de la vue avec le clavier sera donc possible que horizontalement).

Option : Déplacement de la caméra verticalement avec les touches "flèche haut" et "flèche bas" du clavier, ceci nécessitera un recodage des méthodes d'interactions du clavier avec Java3D.

2.6.2.2.5 Zoom + / - sur le graphe

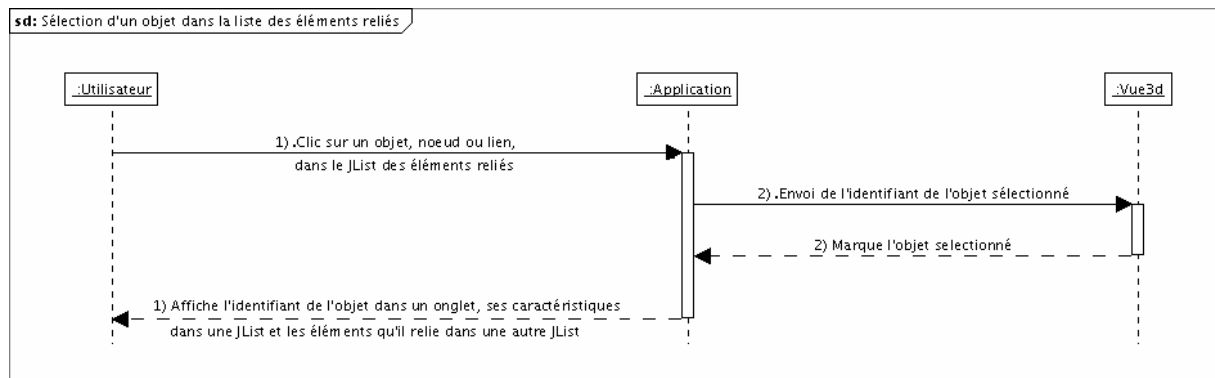


Scénario de remplacement :

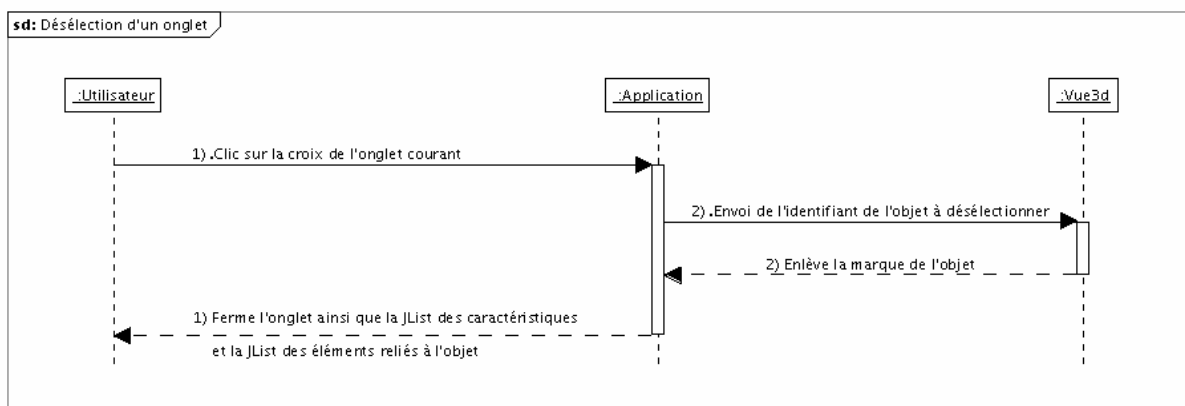
L'utilisateur pourra zoomer sur le graphe avec les boutons "+" et "-" mais également avec la souris en effectuant un drag avec le bouton droit de la souris, ainsi qu'avec le clavier en utilisant les touches "flèche haut" et "flèche bas".

Option : Possibilité de zoomer avec la molette de la souris, cette fonction nécessite un recodage de la méthode pour zoomer avec la souris.

2.6.2.2.6 Sélectionner un objet dans la liste des éléments reliés



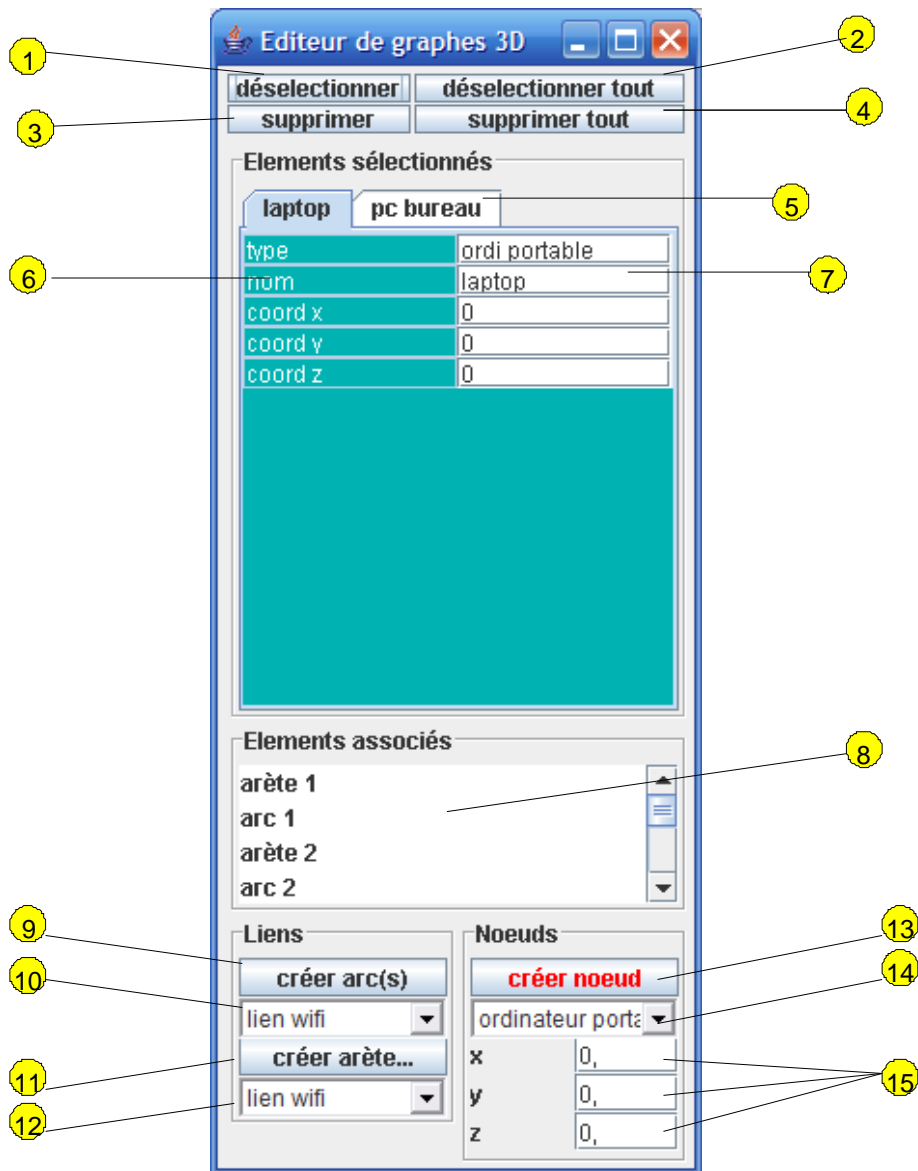
2.6.2.2.7 Désélectionner un onglet



2.7 EDITEUR DE GRAPHE 3D

2.7.1 Spécification de l'interface de l'éditeur

2.7.1.1 Maquette et légende de l'interface de l'éditeur



Légende :

1. bouton désélectionner : retire des onglets l'élément courant dans les onglets.
2. bouton désélectionner tout : retire des onglets tous les éléments sélectionnés.
3. bouton supprimer : supprime du graphe l'élément courant dans les onglets.
4. bouton supprimer tout : supprime tous les éléments sélectionnés.
5. onglet de l'élément courant : affiche l'attribut nom de l'élément.
6. nom d'un attribut de l'élément courant.

7. valeur d'un attribut de l'élément courant : elle est modifiable (la nouvelle valeur doit correspondre au type de l'attribut).
8. liste des éléments (soit de nœuds, soit de liens) reliés à l'élément courant.
9. bouton créer arc(s) : ouvre une fenêtre jaillissante qui permet de choisir parmi les arcs possibles entre les nœuds sélectionnés. Ces arcs seront du type choisi dans la liste déroulante (voir 9), et seront insérés dans les onglets.
10. liste déroulante des types connus d'arcs.
11. bouton créer arête(s) : ouvre une fenêtre jaillissante qui permet de choisir parmi les arêtes possibles entre les nœuds sélectionnés. Ces arêtes seront du type choisi dans la liste déroulante (voir 10), et seront insérés dans les onglets.
12. liste déroulante des types connus d'arêtes.
13. boutons créer nœud : permet de créer un nouveau nœud dans le graphe. Celui-ci sera du type choisi dans la liste déroulante (voir 13) et sera placé aux coordonnées spécifiées (voir 14), sauf s'il existe déjà un nœud à cet emplacement, auquel cas le nœud créé sera repositionné automatiquement.
14. liste déroulante des types connus de nœuds.
15. coordonnées spatiales (nombres réels) du nœud à créer.

2.7.1.2 Explication du fonctionnement de la maquette de l'éditeur

- Pour désélectionner un seul nœud se trouvant dans les onglets, il faut cliquer sur son onglet, puis appuyer sur le bouton « désélectionner » (voir 1 de la maquette). De même pour supprimer un élément sélectionné mais en appuyant sur le bouton « supprimer » (voir 3 de la maquette).
- Pour désélectionner tous les éléments sélectionnés, il faut appuyer sur le bouton « désélectionner tout » (voir 2 de la maquette). De même pour supprimer tous les éléments mais en appuyant sur le bouton « supprimer tout » (voir 4 de la maquette).
- Pour modifier un des attributs de l'élément courant des onglets, il suffit de modifier directement sa valeur dans sa table d'attributs (voir 6 et 7 de la maquette).
- Pour sélectionner un élément relié à l'élément courant des onglets, il faut double-cliquer sur son nom dans la liste déroulante (voir 8 de la maquette). Il sera placé dans les onglets et deviendra l'élément courant. Remarque : en appuyant sur « ctrl » lors de la sélection du nouvel élément, l'élément courant ne changera pas dans les onglets, ce qui permet la sélection multiple.
- Pour créer un ou plusieurs arc, il faut sélectionner son type puis cliquer sur le bouton « créer arc(s) » (voir 9 et 10 de la maquette). Une fenêtre apparaîtra pour vous permettre de sélectionner les arcs à mettre en place entre les différents nœuds sélectionnés.
- Pour créer une ou plusieurs arêtes, il faut sélectionner son type puis cliquer sur le bouton « créer arête(s) » (voir 11 et 12 de la maquette). Une fenêtre apparaîtra pour vous permettre de sélectionner les arêtes à mettre en place entre les différents nœuds sélectionnés.
- Pour créer un nœud, il faut sélectionner son type, ses coordonnées désirées puis cliquer sur le bouton « créer nœud » (voir 13, 14 et 15 de la maquette). Le nouveau nœud sera placé dans les onglets en tant qu'élément courant, et positionné aux coordonnées voulues si celles-ci ne collisionnent pas un autre nœud, auquel cas le nœud sera repositionné dans le graphe.

Remarque : lors de la création d'un élément, ses attributs seront initialisés avec des valeurs par défaut.

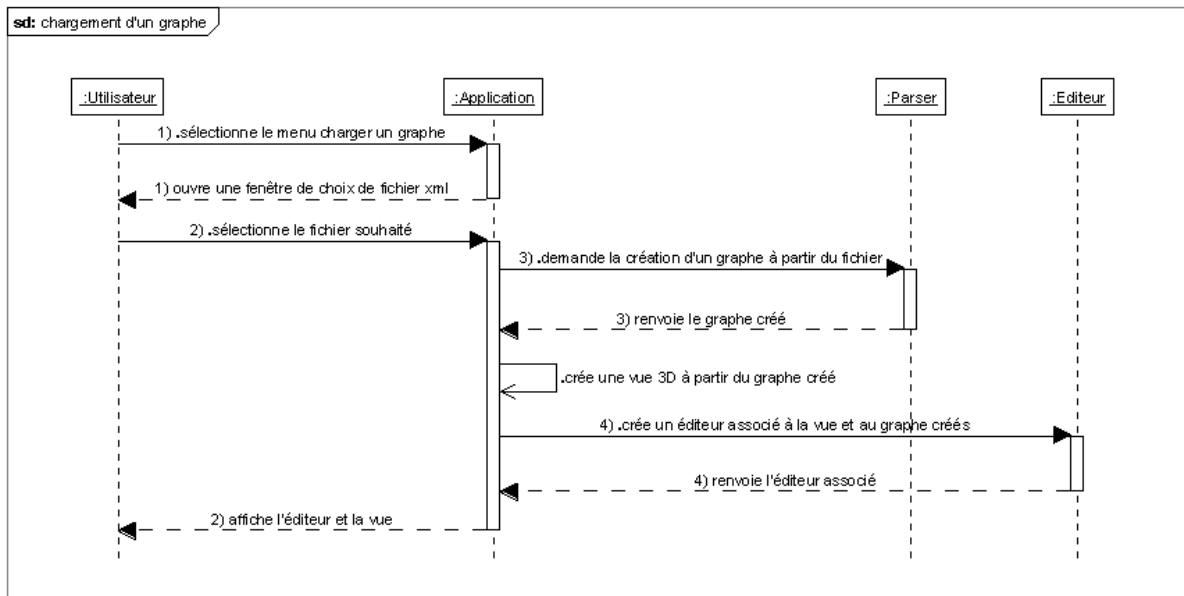
2.7.2 Cas d'utilisations

2.7.2.1 Diagramme de cas d'utilisation



2.7.2.2 Diagrammes de séquences

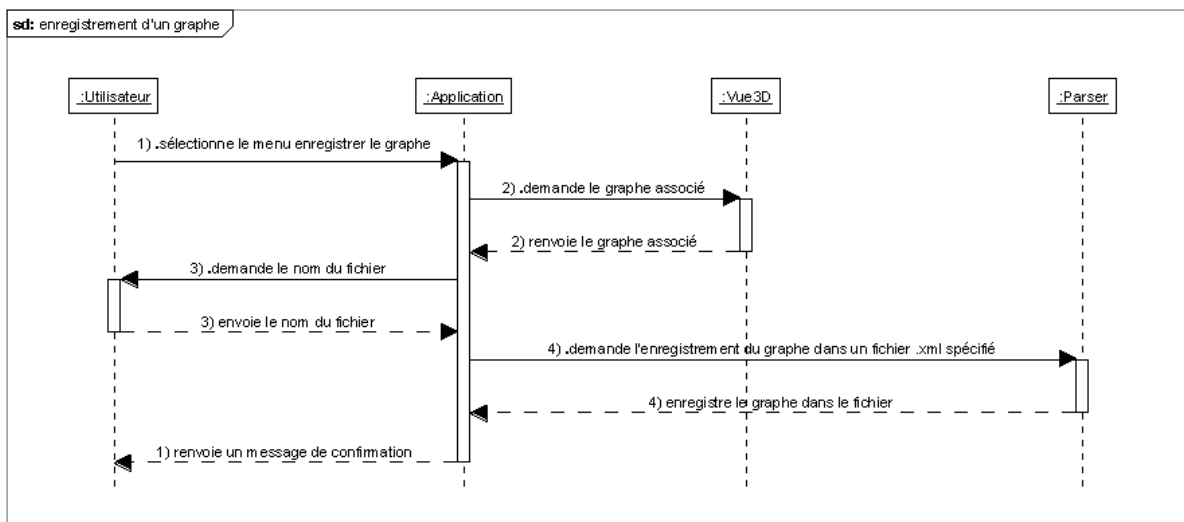
2.7.2.2.1 Charger un graphe



Scénario de remplacement :

3. Le fichier sélectionné ne correspond pas à un graphe valide. Le parseur renvoie un message d'erreur et le chargement s'arrête.

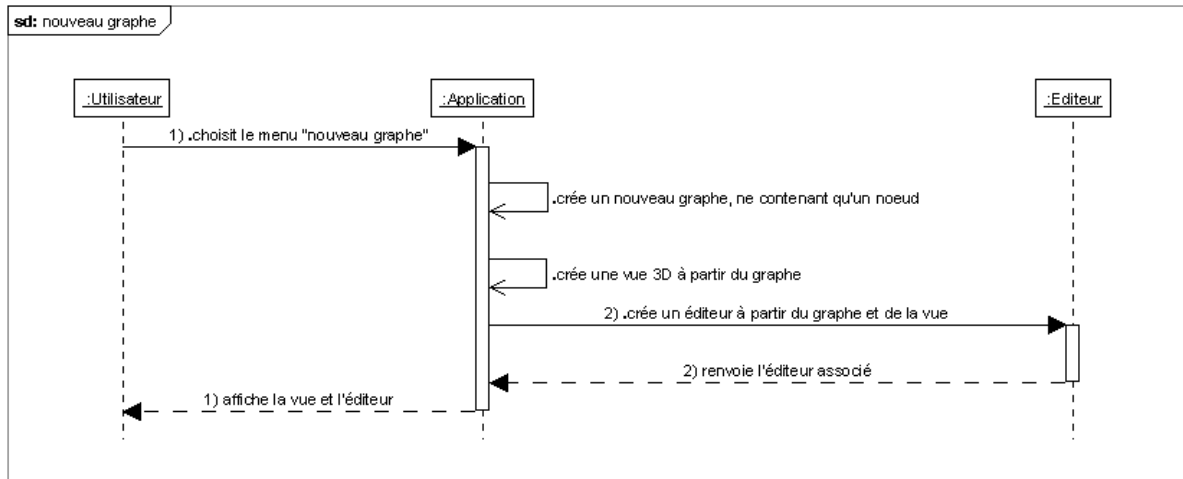
2.7.2.2.2 Sauvegarder un graphe



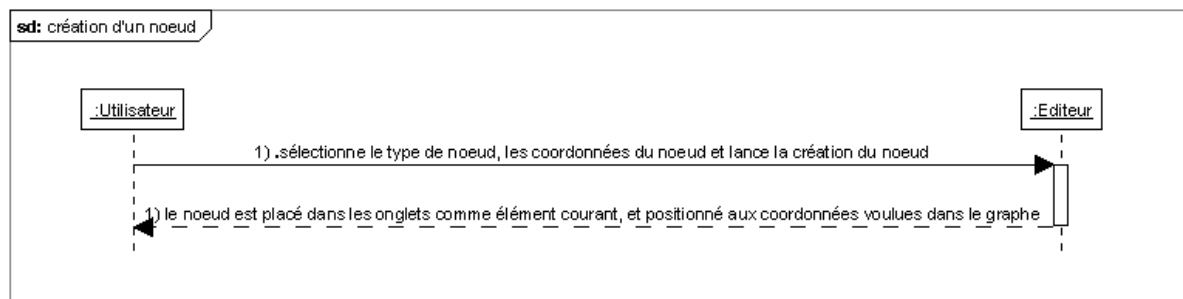
Scénario de remplacement :

4. Le nom du fichier ou l'emplacement rentrer par l'utilisateur est invalide, le parseur renvoie un message d'erreur et retour à l'étape 3.

2.7.2.2.3 Création d'un nouveau graphe



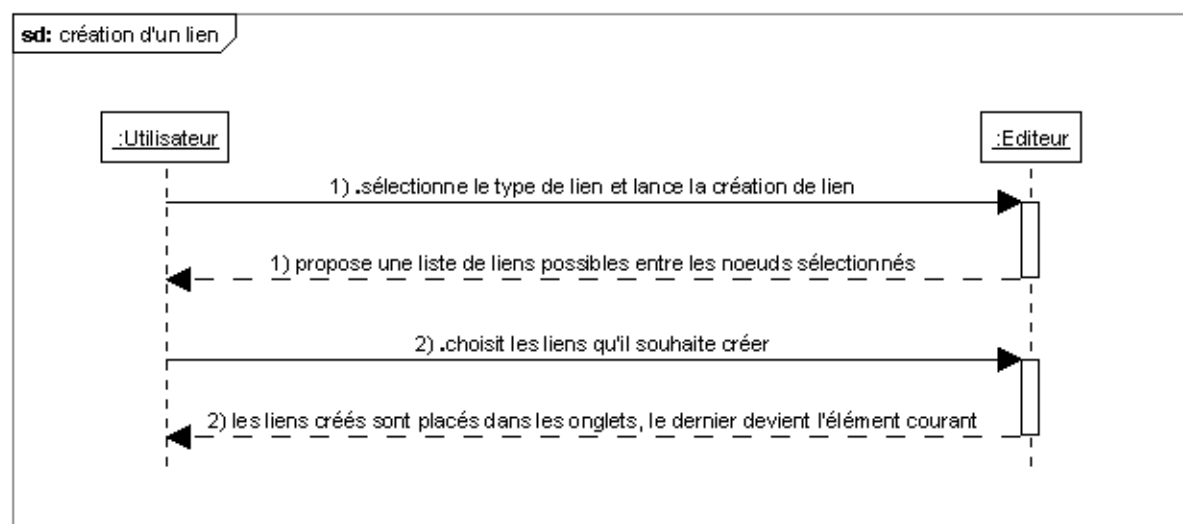
2.7.2.2.4 Création d'un noeud



Scénario de remplacement :

Les coordonnées saisies rentrent en collision avec un nœud déjà créé. La vue renvoie un message d'information « Attention, il y a une collision. Votre nœud a été repositionné automatiquement ». Le nœud est placé dans les onglets comme élément courant et positionné automatiquement dans le graphe.

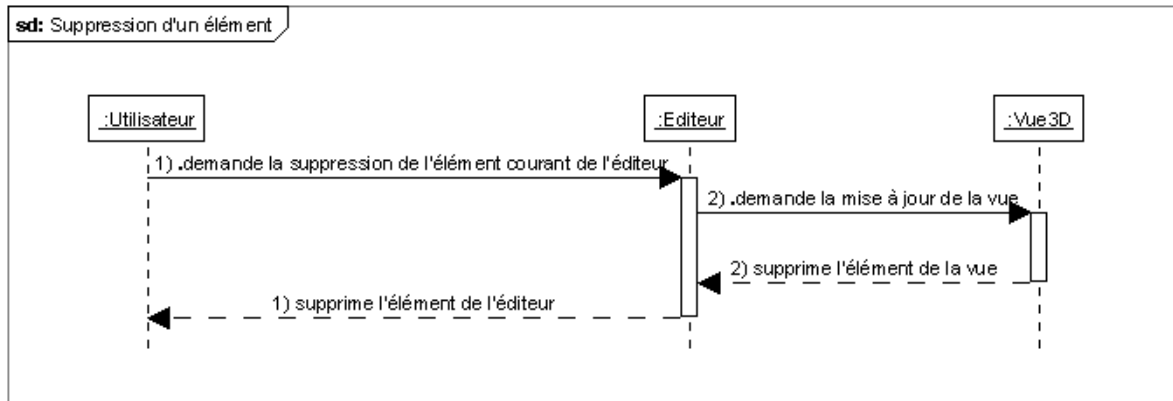
2.7.2.2.5 Création d'un lien



Scénario de remplacement :

2. Si aucun lien n'a été sélectionné, alors aucun lien n'est créé.

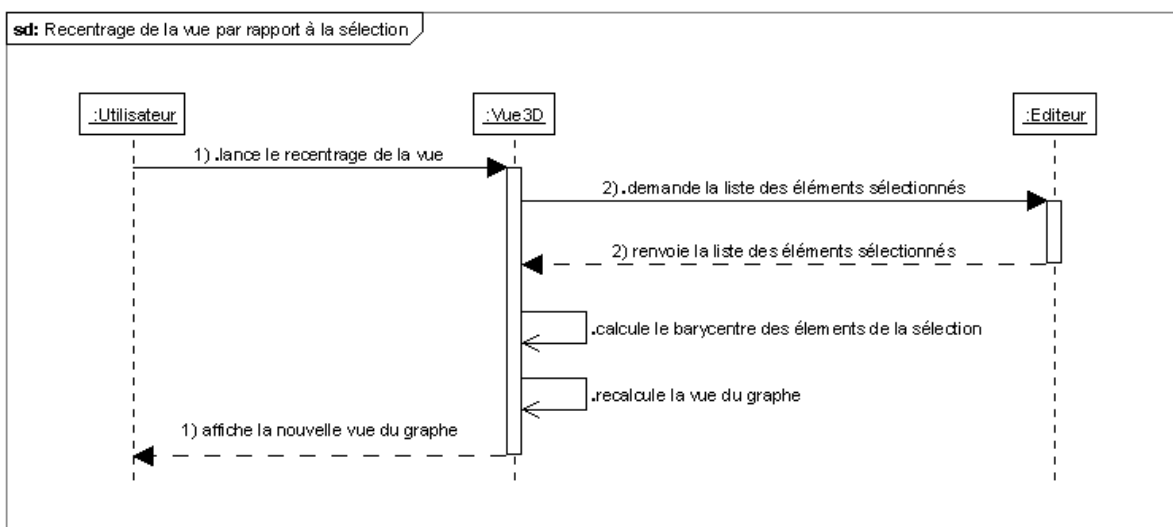
2.7.2.2.6 Suppression d'un élément



Scénario de remplacement :

2. Si l'élément à supprimer est un nœud qui possède plusieurs liens qui lui est associé. Alors c'est liens sont retirés de la vue (et de l'éditeur s'ils y sont présents) puis, le nœud est supprimé de la vue et de l'éditeur.

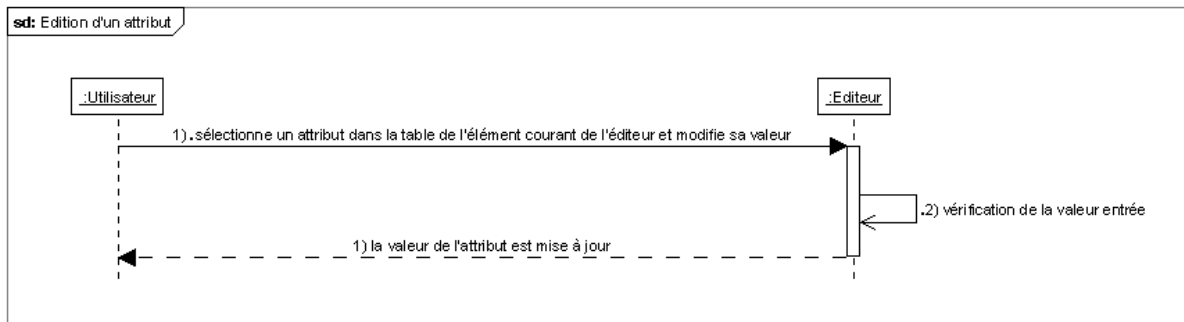
2.7.2.2.7 Recentrer la vue par rapport à la sélection



Scénario de remplacement :

2. Si aucun élément n'est sélectionné, la vue calcule le barycentre du graphe au complet, ensuite, la vue est recalculée à partir du barycentre et affiche le graphe au complet.

2.7.2.2.8 Edition d'un attribut

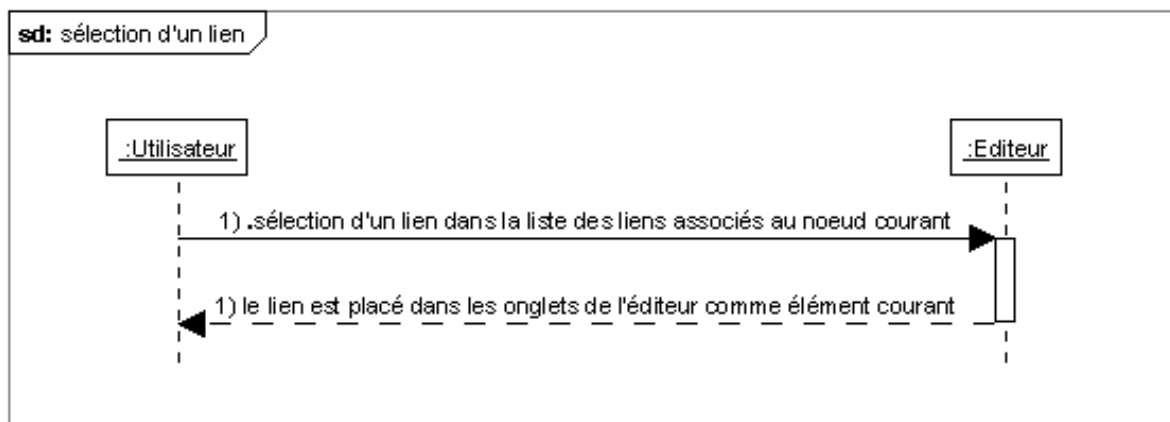


Scénario de remplacement :

2. Premier cas, si l'attribut modifier est l'identifiant de l'élément et que cette valeur est utilisée par un autre élément du même type (nœud ou lien) alors la valeur est réinitialisée.

2. Second cas, si l'attribut modifier est une coordonnée (x, y, z) alors l'éditeur demande à la vue de se mettre à jour avec les nouvelles coordonnées. Si jamais, les coordonnées nouvellement saisies entre en collision avec un autre nœud, alors la vue renvoie un message du style « Attention, il y a eu une collision. Votre nœud a été repositionné automatiquement », puis, les coordonnées sont mises à jour en fonction de celles calculées précédemment et enfin, pour terminer, la vue est recalculée et affichée.

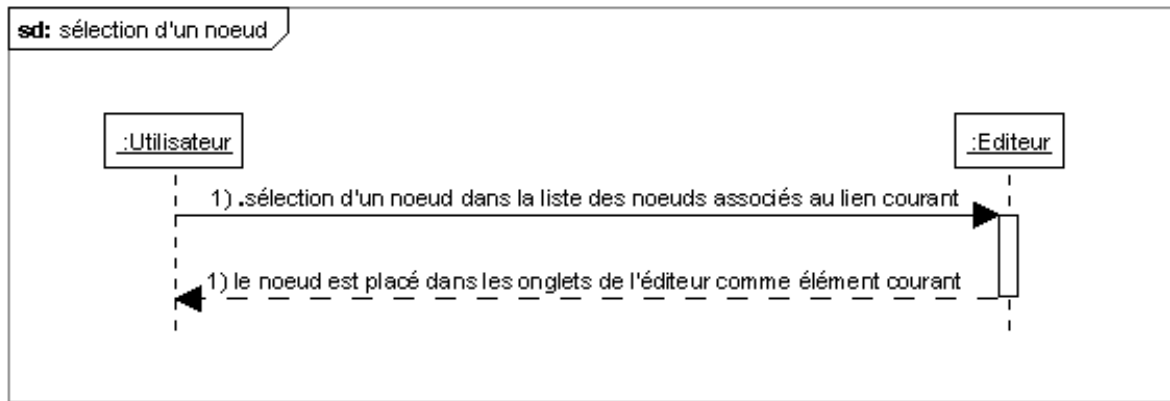
2.7.2.2.9 Sélection d'un lien



Scénario de remplacement :

1. Si le bouton « ctrl » est appuyé lors de la sélection du lien dans la liste, alors, le lien est placé dans les onglets de l'éditeur. Le nœud courant ne change pas, ce qui permet une sélection multiple.

2.7.2.2.10 Sélection d'un nœud

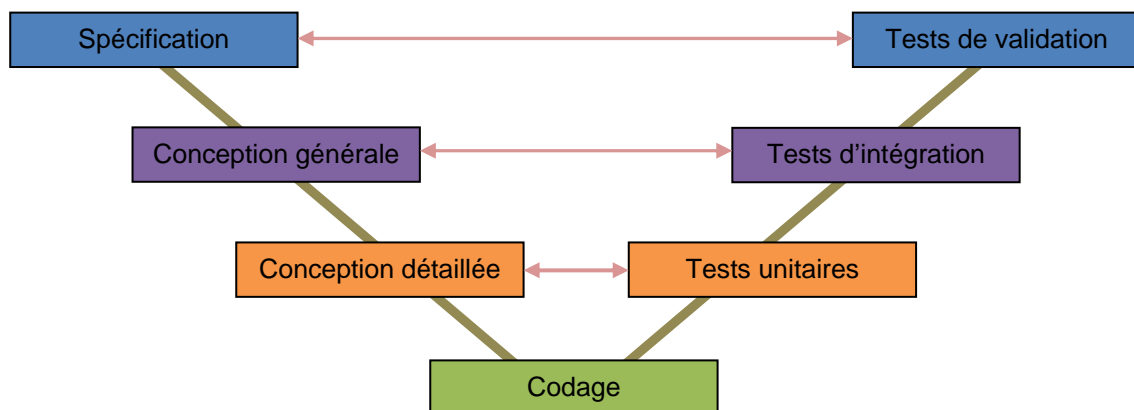


Scénario de remplacement :

1. Si le bouton « ctrl » est appuyé lors de la sélection du nœud dans la liste, alors, le nœud est placé dans les onglets de l'éditeur. Le lien courant ne change pas, ce qui permet une sélection multiple.

2.8 PLAN DE DEVELOPPEMENT

Le cycle de développement utilisé pour ce projet est un cycle en V.



Tout d'abord, ce qui nous a fait choisir ce cycle est qu'il permet de livrer en principe, sans incident, un logiciel totalement conforme au cahier des charges.

La seconde raison de ce choix est que la validation est planifiée et intégrée dans le cycle de développement. Cela permet de n'oublier aucune étape et de réduire l'importance qui est souvent donnée au codage au détriment des tests.

Ensuite, dans ce type de cycle, des points de contrôle sont posés tout au long du cycle de développement, ce qui permet de limiter les dérives, dans la mesure où une étape ne peut démarrer que si la précédente a été contrôlée et approuvée.

De plus, le cycle en V met en évidence la nécessité d'anticiper et de préparer dans les étapes descendantes les « attendus » des futures étapes montantes : ainsi les attendus des tests de validation sont définis lors des spécifications, les attendus des tests unitaires sont définis lors de la conception, etc.

Enfin, de part la structure d'un cycle en V, la portée de la validation de chaque étape s'en trouve limité, car il suffit de vérifier la conformité de l'étape courante que par rapport à la précédente du cycle en V.

2.9 PLAN QUALITE

Afin d'assurer la qualité du futur produit, nous avons mis en place une méthodologie de développement et une structure de travail adaptée, à chaque étape de la conception et de la réalisation.

2.9.1 Qualité de la conception

La phase de conception est une étape importante dans un projet. C'est également l'une des plus délicates, car pour un problème donné, il peut y avoir plusieurs solutions certaines meilleurs que d'autres, et si jamais une erreur n'est pas détecté pendant cette phase, celle-ci aura des répercussions importantes dans la suite du projet.

Nous avons choisis d'utiliser la modélisation UML afin de représenter sous une forme standardisée les différents éléments de la conception (Diagramme de classe, Cas d'utilisations,...) afin de faciliter le dialogue et la validation avec le client. Ce choix de représentation permettra aussi une reprise plus aisée du travail déjà effectuer par d'éventuelles futures équipes.

Les réunions hebdomadaires avec le client (M. Bonnel) ont permises de valider régulièrement les éléments de la phase de conception.

2.9.2 Homogénéité du code

2.9.2.1 Convention de nommage

Une convention de nommage à été mise en place afin d'améliorer la lecture (et relecture) du code, de faciliter l'échange de code entre les programmeurs. Celle-ci s'appuie sur le document « Java Code Conventions » de SUN. Nous serons amenés à suivre les règles suivantes :

- Les noms des classes et des constantes commenceront obligatoirement par une majuscule.

Exemple : NomDeMaClasse

- Le reste (paquetages, variables, méthodes) commencera toujours par une minuscule.

Exemple : nomDeMaVariable, nomDeMonPaquetage, maMethode()

- Le nom que l'on donnera a une classe, une méthode, un attribut ou une constante devra être parlant pour le programmeur.

Exemple : monNoeud.getNom() ;

- Si plusieurs noms composent le nom principal, chaque nom sera séparé par une majuscule.

Exemple : GNode monNoeud ;

- Le nom d'une constante se composera uniquement de majuscules.

Exemple : static final int ESPACE_ENTRE_LES_NOEUDS = 5 ;

- Pour le nom des méthodes, on choisira un verbe.

Exemple : void modifieNom(String nom) ;

Nous avons décidé que le nom des classes, des méthodes, des variables et des paquetages, ainsi que les commentaires et la Javadoc seront en anglais.

2.9.2.2 Mise en forme du code

Comme tout le monde travaillera sous Eclipse, on utilisera la mise en forme automatique du code selon des paramètres définis (réglage de l'unité d'indentation, commentaire Javadoc automatique, entêtes de fichiers,...). La définition de cette mise ne forme pourra être exportée dans un fichier XML afin de s'assurer que tous les développeurs utilisent la même.

2.9.3 Qualité du code

Un code de qualité assure la pérennité, la diffusion, la maintenabilité du projet. Il est important de se souvenir que la réussite d'un projet dépend en partie de la qualité du code.

Mais, il est en général difficile d'évaluer la qualité du code produit par soi-même, en particulier lorsque l'on en est l'auteur de ce code. Afin de remédier à ce problème, il existe des outils qui permettent d'effectuer des mesures sur le code produit. Suivant le résultat de l'analyse fournie par ses outils sur ce code, il est possible d'avoir un avis sur la qualité du code.

Les deux principaux outils permettant d'effectuer des mesures de qualité de code sur des fichiers sources Java sont les suivants :

- **JDepend** : Cet outil fournit des statistiques sur la qualité du code produit. Il est sensible à l'extensibilité, la réutilisabilité, et la maintenabilité des sources. Il permet d'évaluer le degré d'abstraction de chaque package, sa stabilité, sa volatilité,... En résumé un paquetage stable, abstrait et faiblement couplé est d'excellente qualité.
- **JavaNCSS** : Il permet lui aussi d'obtenir différentes métriques sur le code produit. Comme le nombre de paquetage, de classe, de méthode, d'instructions sans commentaire, le nombre de commentaires Javadoc par paquetage, classe et méthode.

Ces deux outils peuvent être intégrés dans un fichier « build.xml » qui sera utilisé avec Ant, ce qui permettra d'automatiser les tâches d'analyse et de produire facilement des rapports de qualité. Les résultats des analyses peuvent se faire sous différentes formes (Fichier XML, HTML,...).

2.9.4 Automatisation des tâches

L'automatisation des tâches permet de standardiser les processus d'un projet et de minimiser les risques d'erreurs en déléguant le maximum de tâches à un processus. Une fois qu'une tâche a été configurée et validée, elle pourra être reproduite une infinité de fois dans les mêmes conditions. Cela est nécessaire pour que tous les tests mais est aussi très utile pour d'autres aspects du développement.

Nous utiliserons Ant pour l'automatisation des tâches qui est un outil particulièrement très bien adapté pour le développement en Java. Il permet d'automatiser un très grand nombre de tâches, allant de la génération des fichiers binaires, des archives Jar, de la Javadoc, à la création de rapports (JDepend, JavaNCSS), en passant par l'exécution des tests unitaires (JUnit)

Eclipse contient une fonctionnalité permettant de compiler automatiquement un projet durant le codage. Cela permet de détecter en temps réel la présence d'erreurs de codage, de variables non utilisées ou encore les inclusions de bibliothèques nécessaires.

Les tests unitaires seront réalisés à l'aide de JUnit. Il permet de vérifier que pour chaque méthode, les post-conditions sont bien celles qui ont été prévues. Une fois tous les tests codés, il suffit de lancer l'exécution sur une classe ou un projet pour que les tests soient effectués un à un. En fin d'analyse, un rapport est produit indiquant les tests réalisés avec succès et ceux qui ont échoué. Enfin, il est intégré à Eclipse et peut donc être mis en place très simplement et il est facile d'utilisation.

2.9.5 Tests

Les tests se divisent en plusieurs catégories, selon que l'on veuille tester le fonctionnement d'une classe ou l'intégration des composants d'un logiciel. Un cahier de test regroupant l'ensemble des documents et procédures mise en place a été rédigé. On y trouve notamment les scénarios de test ainsi qu'un jeu de fiches de tests.

Afin, de s'assurer du sérieux des tests, les testeurs travailleront sur les éléments qu'ils n'auront pas codés. On pourra être amené à avoir recours à des testeurs extérieurs, dans le but d'avoir un regard nouveau, objectif sur le projet pour d'éventuellement améliorations qui pourraient avoir lieu en phase de maintenance.

Les procédures de test seront les suivantes :

- Les tests unitaires seront codés en même temps que les classes qu'ils testent. Nous utiliserons JUnit afin de simplifier et d'automatiser les procédures.
- Les scénarios de test d'intégration seront mis en place qu'après le codage des classes. Ces scénarios sont décrits dans des fiches dédiées à cet effet afin de pouvoir reproduire les tests. En cas de bug, un « rapport d'anomalie » sera rédigé afin d'identifier les erreurs à résoudre et d'en garder une trace.

2.9.6 CVS

L'utilisation d'un outil de versionnage tel que CVS permet une plus grande souplesse dans le développement. En effet, CVS permet de gérer l'évolution dans le temps d'un ensemble de fichiers modifiés par plusieurs personnes. Il permet de revenir à une version stable, si jamais des problèmes sont rencontrés suite à des modifications. Le fait que CVS puisse gérer plusieurs branches à la fois nous permet d'avoir à tout moment une version stable prête à être distribuée, de pouvoir effectuer des corrections de bugs sur une version antérieure ou encore, de pouvoir continuer à développer une autre version, incluant de nouvelles fonctionnalités ou de nouveaux algorithmes. Il est alors possible de livrer au client, à tout moment, une version stable même s'il existe de nouvelles versions plus récentes.

2.10 PLATE-FORME DE DEVELOPPEMENT

Afin de développer l'API Graphe 3D, des machines ainsi que des outils sont mis à disposition.

2.10.1 Outils de développement

Les outils utilisés pour le développement de l'API Graphe3D sont les suivants :

- **Java 1.5** : <http://java.sun.com>
Le langage Java est utilisé pour la programmation.
- **Java 3D** : <http://java.sun.com/products/java-media/3D/>
Java 3D est utilisé pour la représentation de graphe en 3 dimensions dans une fenêtre de visualisation.
- **Eclipse** : <http://www.eclipse.org>
Plateforme de développement utilisé avec CVS, Ant,...
- **CVS** : <http://ximbiot.com/cvs/cvshome/>
Système de gestion de développement en groupe...
- **JUnit** : <http://www.junit.org>
Outil utilisé pour le développement des classes de tests.
- **Ant** : <http://ant.apache.org>
Outil qui permet d'automatiser les tâches.
- **JDepend** : <http://clarkware.com/software/JDepend.html>
Outil permettant la mesure de la qualité de la structure des classes du projet...
- **JavaNCSS** : <http://www.kclee.de/clemens/java/javancss/>
Outil permettant d'analyser la qualité du code produit.
- **Poseidon For UML** : <http://www.gentleware.com>
Outil utilisé pour la création des diagrammes UML.

2.10.2 Caractéristiques d'une machine de développement

La plupart des machines actuellement disponible sur le marché permettent de faire tourner l'API Graphe 3D à condition que Java3D et Java (1.5) y soit installé préalablement. Voici les caractéristiques de base d'une machine qui sera utilisée pour la mise en œuvre de l'API Graphe 3D :

- Processeur : 1,7 GHz
- Mémoire : 512 Mo
- Carte graphique : ATI - 128 Mo
- Disque dur : 80 Go
- Système d'exploitation : Windows XP, Ubuntu.

3 Cahier de tests

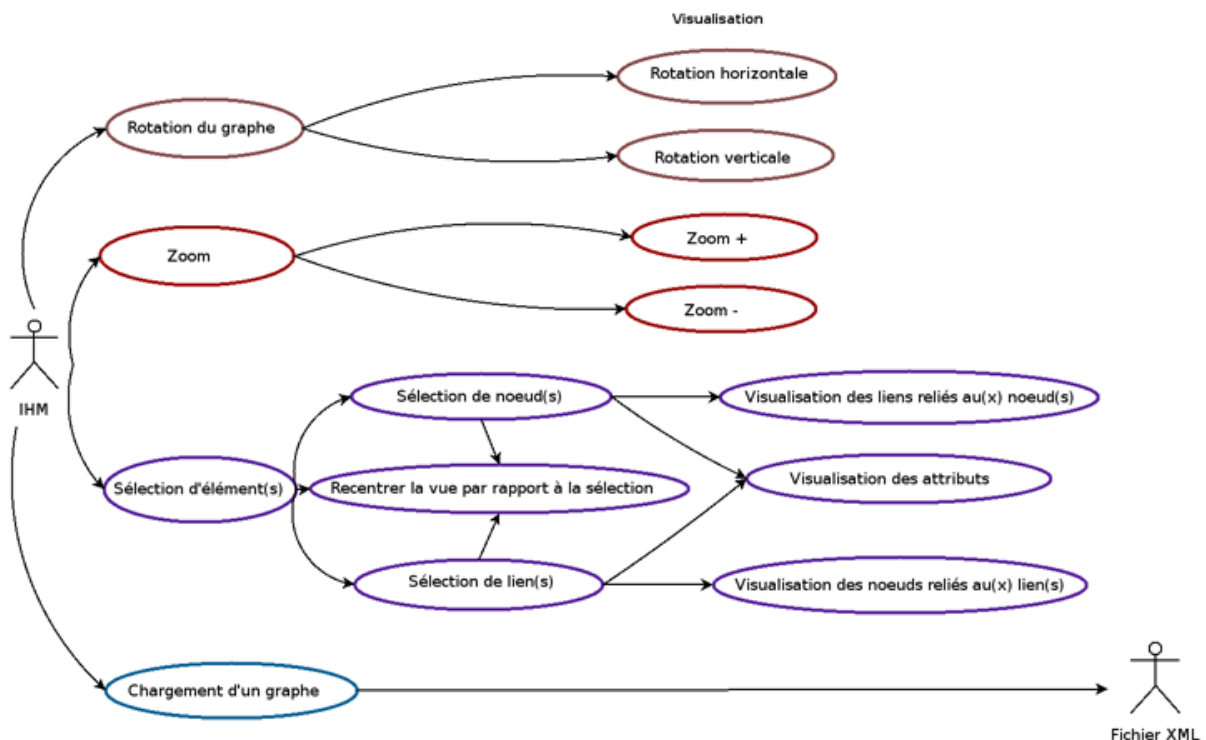
3.1 SCHEMA FONCTIONNEL

3.1.1 Schéma fonctionnel général



L'API Graphe3D permettra deux types d'utilisation. La visualisation de graphe mathématique en 3 dimensions ainsi que l'édition et la conception de graphe mathématique.

3.1.2 Schéma fonctionnel de la visualisation

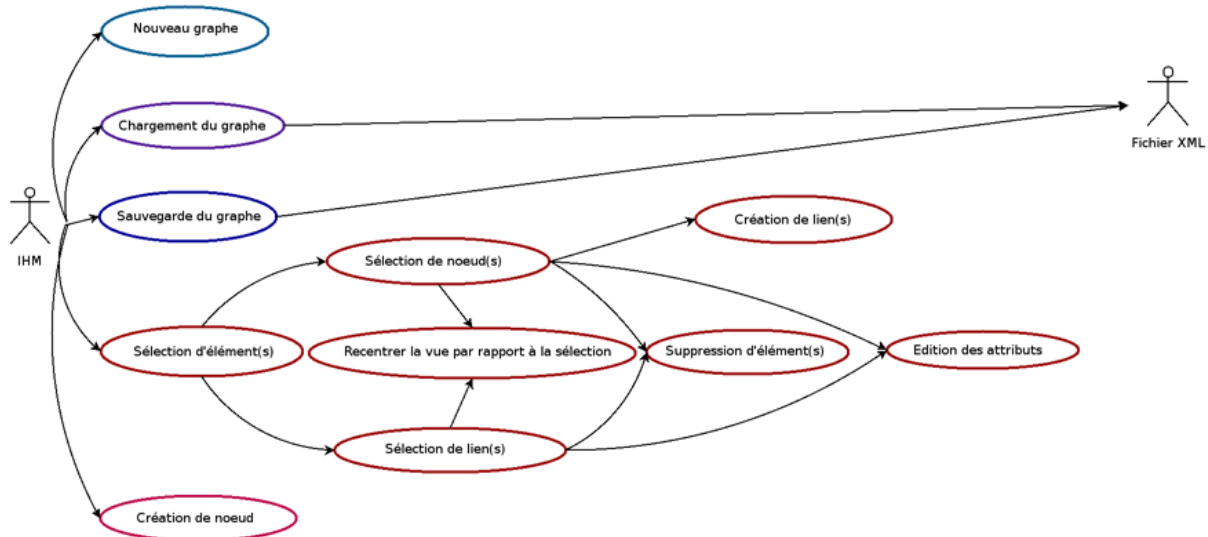


La visualisation permettra à l'utilisateur, au travers de l'interface de charger un graphe mathématique (à partir de son fichier XML) afin de voir sa représentation en 3 dimension.

Une fois un graphe chargé, l'utilisateur pourra manipuler le rendu à sa guise grâce à la possibilité de zoomer sur ce rendu (zoom + ou zoom -), d'effectuer des rotations (horizontales ou verticales) et enfin de sélectionner des éléments composant le graphe.

La sélection permettra de visualiser les attributs de l'élément sélectionnés ainsi que les autres éléments qui lui sont raccordés (exclusivement des liens pour les noeuds sélectionnés et des noeuds pour les liens sélectionnés). Elle donne aussi la possibilité de recentrer le rendu sur elle.

3.1.3 Schéma fonctionnel de l'éditeur



L'édition et la conception de graphe, quant à elle permettra de charger un graphe (toujours à partir de son fichier XML) afin de le modifier ou de concevoir un nouveau graphe.

Une fois les modifications et/ou la conception effectuée, il est possible de sauvegarder le graphe dans son fichier XML.

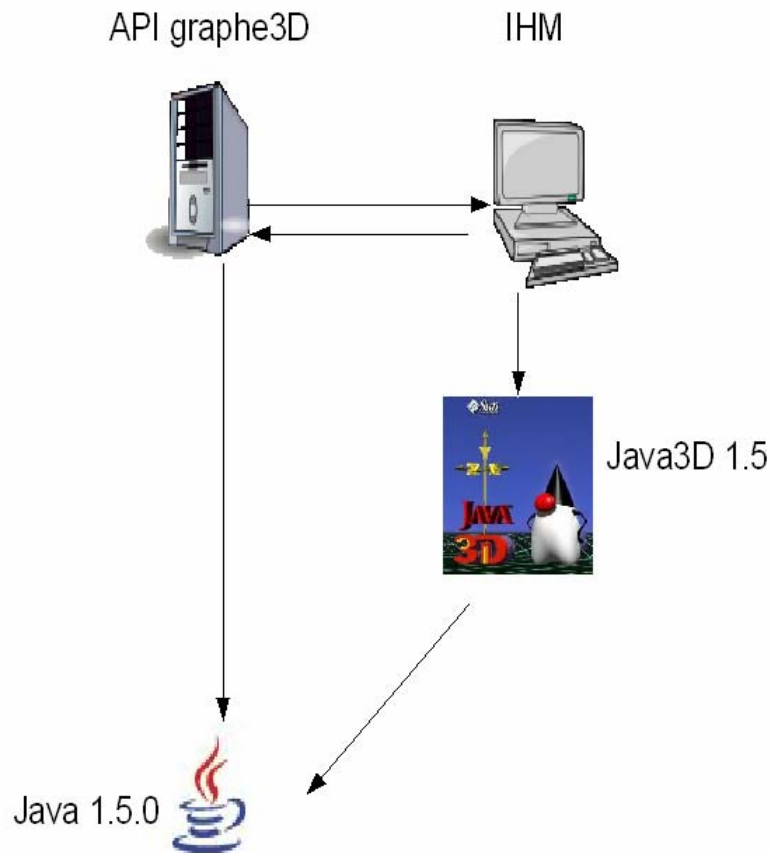
Pour pouvoir éditer on créer un graphe, il est possible de créer de nouveau noeuds.

Enfin, la modification d'éléments déjà existants se fait grâce à la sélection de ceux-ci. Une fois sélectionner, il est possible de supprimer l'élément ainsi que d'éditer ses attributs.

Si le ou les éléments sélectionnés sont des noeuds, il est possible de les connecter entre eux en créant des liens.

Comme pour la visualisation, il est possible de recentrer la vue sur les éléments sélectionnés afin de faciliter leur manipulation.

3.2 SCHEMA TECHNIQUE



L'API graphe3D s'appuie sur la technologie Java et l'interface homme machine sur Java3D qui est une API basé sur Java permettant de faire de la visualisation en 3D.

Étant donné l'utilisation de Java et ses normes qui assurent une certaine portabilité des applications, nous pouvons dire que notre API pourra être utilisé aussi bien sous Windows, Linux ou Macintosh. Il faudra cependant que sur ces machines soit disponible l'API Java3D.

Carte graphique et drivers

Il faudra aussi prendre en compte la carte graphique disponible sur les machines étant donné leur utilisation par l'API Java3D. Notamment pour les drivers installés. En effet, nous avons pu nous apercevoir que selon le driver de la carte, les performances de Java3D sont très différentes. Principalement pour Windows Vista ainsi que sous Linux.

L'idéal concernant les drivers serait d'installer le driver officiel fournir par l'éditeur et disponible sur son site internet.

Configurations testées :

Nous pourrons tester sur aux moins 4 machines de configuration différentes ainsi que sur 4 systèmes d'exploitations (Windows XP, Vista, Linux Ubuntu, Mandriva). Pour plus de précision sur les configurations qui seront testées, aller voir la partie test technique.

3.3 TESTS

3.3.1 Tests Fonctionnels

3.3.1.1 Chargement/Sauvegarde

3.3.1.1.1 Chargement

- Vérification avec le parseur que le graphe chargé est valide : le fichier XML doit respecter le schéma XSD.
- On ne doit pas pouvoir charger un fichier qui n'est pas un fichier XML (dans la fenêtre de chargement seulement les fichiers en « .xml » doivent être visibles).
- Au chargement tous les éléments doivent être visibles, la vue doit contenir tout le graphe3D.

3.3.1.1.2 Sauvegarde

- La sauvegarde du graphe doit s'effectuer dans un fichier XML valide, le chargement de ce fichier avec le parseur doit représenter le même graphe.

3.3.1.2 Vue 3D

3.3.1.2.1 Collisions

- Gestion des collisions, deux noeuds ne peuvent pas être superposés dans la vue, leurs coordonnées doivent être différentes lors d'un placement automatique après un chargement ou lors d'une modification des coordonnées par l'utilisateur.

3.3.1.2.2 Zoom

- Pas de limites pour le zoom, l'effet du zoom doit être progressif avec la gestion de rester appuyé sur le bouton du zoom.
- Vérification du zoom plus (+) respectivement du zoom moins (-).
- Il doit être possible de zoomer sur les éléments qui ont été sélectionnés (zoom sur une sélection simple où multiple).

3.3.1.2.3 Rotation (dans Java3D)

- Vérification des rotations verticales et horizontales
- Vérification du recentrage de la vue sur les élément(s) sélectionné(s)

3.3.1.2.4 Sélection

- "Feedback" dans la vue 3D, mise en valeur du ou des éléments sélectionnés.
- Mise à jour correcte de la liste des caractéristiques des éléments sélectionnés ainsi que de la liste des objets reliés.
- Possibilité d'une sélection simple ou multiple (CTRL + clic sur les éléments avec la souris).

3.3.2 Tests de performances

3.3.2.1 Édition

- Modification de l'attribut d'un élément : rafraîchissement de la vue 3D.
- Test des coordonnées pour la gestion des collisions lors :
 - de la création d'un noeud,
 - modification des coordonnées d'un noeud.
- La suppression d'un noeud plus la suppression des liens auxquels il est raccordé si nécessaire.
- La création ou suppression d'un lien.
- Suppression multiple : suppression de plusieurs éléments sélectionnés.
- Désélection des éléments dans les listes lors de leur suppression.

3.3.2.2 Java3D

- Éviter une montée en charge importante pour des graphes de taille importante.
- Pas d'erreur d'exécution de Java3D.
- Le temps processeur et mémoire ne doivent pas être trop élevés.
- Mise à jour des drivers de carte graphique parfois nécessaire pour Java3D (Linux principalement).

3.3.3 Tests techniques

- On utilise Java 1.5 / 1.6 et Java 3D 1.5.
- Nous ne pourrions tester notre application que sous Windows et Linux cependant Java respecte des normes qui nous permette de dire que notre application Java3D pourra fonctionner sous Linux / Windows / Mac.
- Nous ne pouvons certifier le fonctionnement de notre API Graphe3D pour les versions précédentes de Java 1.5.
- La configuration des ordinateurs sur lesquels l'API Graphe3D sera testée :

Type d'ordinateur	Portable	Portable
Marque	DELL	DELL
Modèle	INSPIRON 6400	INSPIRON 8600
Mémoire	2 Go	512 Mo
Disque dur	120 Go	60 Go
Processeur	Intel Centrino Duo 2 GHz	Intel Centrino 1,7 GHz
Carte graphique	ATI Mobility X1400 256 Mo	ATI Mobility 9600 128 Mo
OS	Windows Vista Mandriva 2007	Windows XP SP2 Ubuntu 7.03

Type d'ordinateur	Portable	Portable
Marque	DELL	ASUS
Modèle	INSPIRON 9400	F3T-AK018C
Mémoire	1 Go	2 Go
Disque dur		160 Go
Processeur	Intel Centrino Duo 2 GHz	AMD Turion 64 x2 TL-56
Carte graphique	ATI Mobility X1400 256 Mo	Nvidia Geforce 7600 256 Mo
OS	Window XP Media Mandriva 2007	Windows XP SP2 Ubuntu 6.10

3.3.4 tests unitaires

Vous trouverez en annexe un [exemple de test unitaire](#) avec Junit pour la classe GNode.

3.4 LISTE DES FICHES DE TESTS


Les fichiers utilisés dans les fiches de test se trouvent en annexe.


3.4.1 Fiche de test n°1.1 : Chargement

Nom : _____ prénom : _____ date : __ / __ / ____

Pré requis : avoir lancé l'application.

Données d'entrée : chargement de fichiers contenant des graphes

	NOM(S) DU OU DES FICHIERS	TEST REUSSI
<i>Fichiers ne respectant pas le schéma XSD</i>	nonValideXsd1.xml	<input type="checkbox"/> Oui <input type="checkbox"/> Non
Résultat attendu : <i>l'application doit afficher le message d'erreur suivant :</i> <div><div>Erreur au chargement de XXX.xml</div><div> Ce fichier ne respecte pas le schéma XSD défini pour l'application</div><div>OK</div></div>		
Remarques :		

	NOM(S) DU OU DES FICHIERS	TEST REUSSI
Fichiers ne respectant pas le schéma XSD	nonValideXsd2.xml	<input type="checkbox"/> Oui <input type="checkbox"/> Non
Résultat attendu : <i>l'application doit afficher le message d'erreur suivant :</i> <div><div>Erreur au chargement de XXX.xml</div><div> Ce fichier ne respecte pas le schéma XSD défini pour l'application</div><div>OK</div></div>		
Remarques :		

	NOM(S) DU OU DES FICHIERS	TEST REUSSI
<i>Fichiers non valide sémantiquement</i>	nonValideSem.xml	<input type="checkbox"/> Oui <input type="checkbox"/> Non
Résultat attendu : <i>l'application doit afficher le message d'erreur suivant :</i> <div data-bbox="301 506 1059 748" data-label="Image"></div>		
Remarques :		


	NOM(S) DU OU DES FICHIERS	TEST REUSSI
<i>Fichier valide</i>	valide.xml	<input type="checkbox"/> Oui <input type="checkbox"/> Non
Résultat attendu : <i>l'application doit afficher une vue correspondante au graphe.</i>		
Remarques :		


3.4.2 Fiche de test n°2 : sauvegarde

Nom : _____ prénom : _____ date : __ / __ / ____

Pré requis : avoir lancé l'application et chargé le fichier « valide.xml ».

Données d'entrée : sauvegarde du graphe dans un fichier. Le nom du fichier est celui qui doit être passé en paramètre à l'application pour la sauvegarde.

	NOM(S) DU OU DES FICHIERS	TEST REUSSI
Nom de fichier non suffixé .xml	nonvalide.doc	<input type="checkbox"/> Oui <input type="checkbox"/> Non
Résultat attendu : <i>l'application doit afficher le message d'erreur suivant :</i> <div><div>Erreur au chargement de XXX.xml</div><div> Ce fichier ne respecte pas le schéma XSD défini pour l'application</div><div>OK</div></div>		
Remarques :		

	NOM(S) DU OU DES FICHIERS	TEST REUSSI
Nom de fichiers incorrect pour le système de fichiers	non:valide?.xml	<input type="checkbox"/> Oui <input type="checkbox"/> Non
Résultat attendu : <i>l'application doit afficher le message d'erreur suivant :</i> <div><div>Erreur au chargement de XXX.xml</div><div> Ce fichier ne respecte pas le schéma XSD défini pour l'application</div><div>OK</div></div>		
Remarques :		

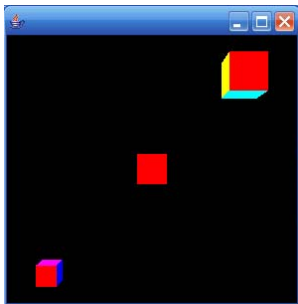
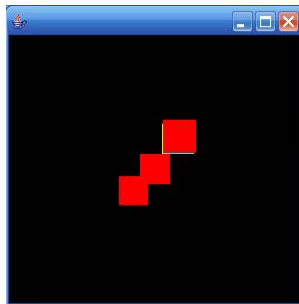
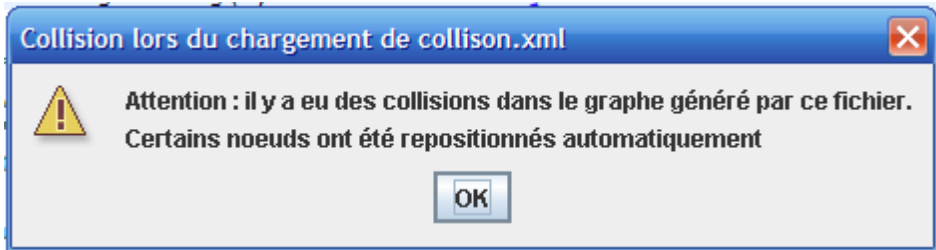
	NOM(S) DU OU DES FICHIERS	TEST REUSSI
Nom du fichier correct	sauvegarde.xml	
Résultat attendu : <i>l'application ne doit pas afficher de message d'erreur lors de la sauvegarde.</i> <i>Vous devez alors vérifier qu'en rechargeant le fichier « sauvegarde.xml » :</i> <ul style="list-style-type: none"> ▪ aucun message d'erreur n'apparaît ▪ votre graphe n'a pas changé 		<input type="checkbox"/> Oui <input type="checkbox"/> Non
Remarques : 		


3.4.3 Fiche de test n°2.1.2 : gestion des collisions

Nom : _____ prénom : _____ date : __ / __ / ____

Pré requis : avoir lancé l'application

Données d'entrée : chargement d'un fichier contenant un graphe dont le nom est spécifié, puis pour le second cas modification des coordonnées d'un noeud.

	NOM(S) DU OU DES FICHIERS	TEST REUSSI
Chargement d'un fichier avec collisions	collision.xml	<div><input type="checkbox"/> Oui</div> <div><input type="checkbox"/> Non</div>
<p>Résultat attendu :</p> <p><i>l'application doit afficher une vue avec le premier noeud aux coordonnées de départ et les autres noeuds doivent être repositionnés automatiquement.</i></p> <p>La vue doit donc être du type :</p> <div></div> <p>et pas de ce type :</p> <div></div> <p>De plus un message de signalisation doit apparaître :</p> <div></div>		
Remarques :		

	NOM(S) DU OU DES FICHIERS	TEST REUSSI
Modification des coordonnées d'un noeud	collision.xml	<input type="checkbox"/> Oui <input type="checkbox"/> Non
<p>Résultat attendu : <i>après avoir sélectionné dans la vue puis changé les coordonnées du noeud appelé « node 2 » (coordonnées mis à X=0, Y=0, Z=0), l'application doit repositionner le noeud dans le graphe et afficher un message de signalisation comme suit :</i></p> <div><p>Collision lors du déplacement d'un noeud</p><p> Attention : il y a eu une collision dans le graphe. Le noeud a été repositionné à ses coordonnées précédentes</p><p>OK</p></div>		
Remarques :		

4 Organisation

4.1 PHASE D'ANALYSE

Pour la réalisation de la phase d'analyse, un crédit de 12 jours par étudiants est disponible. Le chiffrage détaillé des tâches des différents chantiers est en annexe. Les 5 grands chantiers qui constituent cette phase d'analyse sont les suivants :

4.1.1 Chantier 1 : Etude de projet

Les différentes tâches de l'étude du projet sont :

- Analyse de la documentation
- Analyse de l'existant
- Formation aux outils (Eclipse, CVS,...)
- Rendez-vous avec le tuteur
- Rédaction validation des spécifications du projet

Ce chantier consiste dans un premier temps à la rédaction de la fiche de projet, qui constitue une première analyse du projet et de ses points importants, afin de valider la bonne compréhension du sujet. Il définit le contexte du projet, les objectifs, les cibles, les enjeux, les livrables attendus, ainsi que les contraintes et les dérives possibles du projet.

Il est réalisé par l'ensemble des membres du groupe afin de s'assurer que tous les étudiants aient bien compris le sujet du projet.

4.1.2 Chantier 2 : Conception détaillée

Les différentes tâches de la conception détaillée sont :

- **Définition des fonctionnalités (Editeur et API)** : la liste des fonctionnalités sera définie en accord avec le tuteur par proposition du groupe.
- **Conception, réalisation, validation du diagramme de classe de l'API Graphe 3D** : cette étape sera réalisée par les étudiants de L3, soit par M. Magnin et M. Daubert. Le chef de projet pourra apporter des suggestions et il validera le diagramme avant présentation au tuteur.
- **Conception, réalisation, validation du diagramme de classe de l'éditeur de Graphe 3D** : cette étape sera réalisée par les étudiants de L3, soit M. Daubert et M. Lino. Le chef de projet pourra apporter des suggestions et il validera le diagramme avant présentation au tuteur.
- **Spécification, validation de l'IHM de l'éditeur de Graphe 3D** : cette tâche sera accomplie par M. Lino. Le chef de projet pourra apporter des suggestions.
- **Spécification, validation de l'IHM de visualisation de Graphe 3D** : cette tâche sera accomplie par M. Magnin. Le chef de projet pourra apporter des suggestions.
- **Définition du format de stockage de graphe3D** : il s'agit de définir le schéma XML pour la représentation d'un graphe 3D et de donner un exemple de graphe 3D en XML. Cette tâche sera effectuée par l'ensemble du groupe dans un premier temps.
- **Cas d'utilisations, diagrammes de séquences et scénarios de remplacement (Editeur et API)** : Il s'agit ici d'établir le diagramme des cas d'utilisations, de définir les scénarios de base et de remplacement et des diagrammes de séquences. M^{lle} Popa sera en charge de cette tâche.
- **Choix techniques** : Il s'agit de définir le choix des « outils » utilisés dans le projet. (XML, Dom, Swing...)

Ce chantier permet d'analyser le projet en profondeur, de fixer la liste des fonctionnalités, de rédiger les spécifications (diagramme de classe, cas d'utilisation, diagramme de séquence, ihm).

4.1.3 Chantier 3 : Organisation

Les différentes tâches de l'organisation sont :

- **Organisation de la phase d'analyse et de réalisation** : le chef de projet va définir les différentes étapes qui sont à réaliser pour mener à terme le projet. Il va aussi attribuer des rôles ainsi que les différentes tâches effectuées aux différents membres de son équipe.
- **Gestion des sources et mise en place de l'environnement de développement** : cette tâche consistera tout d'abord à la mise en place du dépôt du CVS afin de pouvoir partager les documents, puis en l'installation des différents outils de développement (Eclipse, JDK, Java3D,...) sur les différents comptes (ordinateurs) des étudiants.

Le chef de projet est responsable de ce chantier et des différentes tâches qui s'y trouvent.

4.1.4 Chantier 4 : Cahier de tests

Les différentes tâches du cahier de tests sont :

- Schéma technique
- Schéma fonctionnel
- Tests unitaires
- Tests d'intégration
- Définition et rédaction des jeux de tests

Suite au cours de « tests et intégrations » enseigné par Frank Mosser, les étudiants de L3 seront assistés par ce dernier pour la conception du dossier de tests.

4.1.5 Chantier 5 : Présentation orale de l'analyse

Ce chantier consiste en la préparation de la présentation orale de la phase d'analyse qui aura lieu le 29 mars. Un support de présentation sera créé au cours de cette tâche et des répétitions auront lieu.

Le chef de projet ainsi que tous les étudiants de L3 participeront à cette tâche.

4.2 PHASE DE REALISATION

Pour la phase de réalisation, un crédit de 13 jours par étudiants est disponible. Le chiffrage détaillé des tâches des différents chantiers est en annexe. Le chef de projet dans cette phase fera essentiellement de la surveillance et s'assurera du bon déroulement des chantiers et des tâches qui les composent. Les 7 grands chantiers qui constituent cette phase d'analyse sont les suivants :

4.2.1 Chantier 1 : Codage de l'interface de visualisation de Graphe 3D

- **Réalisation de l'interface de visualisation** : Cette tâche sera à la charge de M. Daubert et de M. Magnin. Il s'agit de réaliser la fenêtre de visualisation de graphe 3D et les fonctionnalités qui lui sont associées.
- **Réalisation et exécution des tests unitaires** : Il s'agit ici de coder les tests unitaires qui auront été définis dans le cahier de tests en utilisant JUnit et de lancer l'exécution des tests unitaires codés. Cette tâche sera réalisée également par M^{lle} Popa.

4.2.2 Chantier 2 : Codage de l'éditeur de Graphe 3D

- **Réalisation de l'éditeur de graphe 3D** : Cette tâche sera à la charge de M. Lino. Il s'agit ici de réaliser l'éditeur de graphe 3D et des fonctionnalités qui lui sont associées.
- **Réalisation et exécution des tests unitaires** : Il s'agit ici de coder les tests unitaires qui auront été définis dans le cahier de tests en utilisant JUnit et de lancer l'exécution des tests unitaires codés. Cette tâche sera réalisée M^{lle} Popa.

4.2.3 Chantier 3 : Codage des éléments communs

- **Codage du package du parseur XML** : Il s'agit ici de coder le package. Soit les fonctionnalités, de création, d'ouverture, d'écriture et de parsing d'un fichier XML. Cette tâche sera réalisée par M. Daubert.
- **Codage du package élément** : Il s'agit de réaliser ici le codage du package éléments qui permet de représenter la structure du graphe 3D sous forme d'objets. La réalisation de cette tâche incombe à M. Daubert.
- **Réalisation et exécution des tests unitaires** : Il s'agit ici de coder les tests unitaires qui auront été définis dans le cahier de tests en utilisant JUnit et de lancer l'exécution des tests unitaires codés. Cette tâche sera réalisée par M^{lle} Popa.

4.2.4 Chantier 4 : Tests et intégration

- **Réalisation des tests fonctionnels** : Il s'agit de réaliser les tests fonctionnels qui ont été décrits dans le cahier de tests pour l'éditeur de graphe 3D ainsi que ceux pour l'API de graphe 3D. Cette tâche sera accomplie par tous les membres de l'équipe.
- **Réalisation d'un programme illustrant l'utilisation de l'API de Graphe 3D (interface de visualisation)** : Il s'agit de réaliser un programme qui illustre l'utilisation de l'interface de visualisation de Graphe 3D à partir de l'API. Cette tâche sera réalisée par M. Magnin.
- **Réalisation d'un programme illustrant l'utilisation de l'API de Graphe 3D (éditeur)** : Il s'agit de « réaliser » un programme qui illustre l'utilisation de l'éditeur de Graphe 3D à partir de l'API. Cette tâche est à la charge de M. Lino.

Nb : C'est deux dernières tâches font office de tâche de validation d'intégration. Vu que ces tâches vont montrer que d'une certaine manière, tous les éléments codés séparément, s'assemblent parfaitement bien ensemble, les uns avec les autres.

4.2.5 Chantier 5 : Documentation

- **Réalisation d'un manuel utilisateur – API Graphe 3D** : Il s'agit ici de créer un manuel expliquant à l'utilisateur comment utiliser la fenêtre de visualisation de Graphe 3D. Cette tâche sera effectuée par M^{lle} Popa.
- **Réalisation d'un manuel utilisateur – Editeur de Graphe 3D** : Il s'agit ici de créer un manuel expliquant à l'utilisateur comment utiliser l'éditeur de Graphe 3D. Cette tâche sera à la charge de M^{lle} Popa.

4.2.6 Chantier 6 : Validation du client

- **Validation du produit par le client** : Il s'agit ici de valider que le produit répond aux attentes que le client avait fixées. Cette tâche sera accomplie par l'ensemble du groupe.
- **Livraison** : Il s'agit ici de livrer l'API de graphe 3D ainsi que son éditeur, avec la documentation et de faire une démonstration au client et un bilan des résultats obtenus. Cette tâche sera effectuée par l'ensemble des membres du groupe.
- **Maintenance** : ça consiste à faire de la maintenance évolutive, si le client a été satisfait lors de la livraison, dans le cas contraire, il s'agit de faire de la maintenance corrective. Cette tâche sera réalisée par l'ensemble des membres du groupe.

4.2.7 Chantier 7 : Bilan

- **Rédaction du bilan du projet** : Comme le nom de la tâche l'indique, il s'agit ici de réaliser la rédaction du bilan du projet. Cette tâche sera accomplie par l'ensemble des membres du groupe.

- **Préparation de la présentation orale du bilan du projet** : Il s'agit ici de créer un PowerPoint et de faire des simulations de présentation orale. Cette tâche sera effectuée par l'ensemble des étudiants de L3 ainsi que le chef de projet.

5 Annexes

5.1 SCHEMA XML

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <!-- definition de la balise racine du fichier XML -->
  <xsd:element name="graph" type="graphType" />

  <!-- definition du contenu entre les balises racines (graph) -->
  <xsd:complexType name="graphType">
    <xsd:sequence>
      <xsd:element name="node" type="nodeType"
        minOccurs="1" maxOccurs="unbounded" />
      <xsd:element name="link" type="linkType"
        minOccurs="0" maxOccurs="unbounded" />
    </xsd:sequence>
    <xsd:attribute name="id" type="xsd:string" use="optional" />
  </xsd:complexType>

  <!-- definition du contenu entre les balises node -->
  <xsd:complexType name="nodeType">
    <xsd:sequence>
      <xsd:element name="enum-attr" type="attrType"
        minOccurs="0" maxOccurs="unbounded" />
      <xsd:element name="coordonates" type="coordonatesType"
        minOccurs="1" maxOccurs="1" />
    </xsd:sequence>
    <xsd:attribute name="class" type="xsd:string" use="optional" />
    <xsd:attribute name="name" type="xsd:string" use="required" />
  </xsd:complexType>

  <!-- definition du contenu entre les balises link -->
  <xsd:complexType name="linkType">
    <xsd:sequence>
      <xsd:element name="enum-attr" type="attrType"
        minOccurs="0" maxOccurs="unbounded" />
    </xsd:sequence>

    <!-- L'attribut "type" ne peut prendre que les valeurs -->
    <!-- specifiees ci-dessous : -->
    <!-- une boucle est un arc ou une arrete sur un seul noeud -->
    <!-- Par la suite les attributs link_start et link_end -->
    <!-- prendront le meme nom de noeud -->
    <!-- un arc (arrow) est un lien oriente -->
    <!-- une arrete (bridge) est un lien sans orientation -->
    <xsd:attribute name="type" use="required">
      <xsd:simpleType>
        <xsd:restriction base="xsd:NMTOKEN">
          <xsd:enumeration value="arrow" />
          <xsd:enumeration value="bridge" />
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>

  </xsd:complexType>

</xsd:schema>
```

```
<xsd:attribute name="class" type="xsd:string" use="optional" />

<!-- l'attribut link_start specifie, dans le cas d'un arrow,-->
<!-- le noeud source du lien-->
<xsd:attribute name="link_start" type="xsd:string"
               use="required" />

<!-- l'attribut link_end specifie, dans le cas d'un arrow,-->
<!-- le noeud destination du lien -->
<xsd:attribute name="link_end" type="xsd:string"
               use="required" />

<!-- l'identifiant de l'objet link -->
<xsd:attribute name="name" type="xsd:string" use="required" />
</xsd:complexType>

<!-- definition du contenu entre les balises enum-attr -->
<xsd:complexType name="attrType">
  <xsd:attribute name="attr-name" type="xsd:string"
                 use="required" />
  <xsd:attribute name="attr-type" use="required">
    <xsd:simpleType>
      <xsd:restriction base="xsd:NMTOKEN">
        <xsd:enumeration value="short" />
        <xsd:enumeration value="byte" />
        <xsd:enumeration value="int" />
        <xsd:enumeration value="long" />
        <xsd:enumeration value="float" />
        <xsd:enumeration value="double" />
        <xsd:enumeration value="char" />
        <xsd:enumeration value="String" />
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:attribute>
  <xsd:attribute name="attr-value" type="xsd:string"
                 use="required" />
</xsd:complexType>

<!-- definition du contenu entre les balises coordonates -->
<xsd:complexType name="coordonatesType">
  <xsd:attribute name="x" type="xsd:string" use="required" />
  <xsd:attribute name="y" type="xsd:string" use="required" />
  <xsd:attribute name="z" type="xsd:string" use="required" />
</xsd:complexType>
</xsd:schema>
```

5.2 EXEMPLE DE FICHIER XML

```
<?xml version="1.0" encoding="UTF-8"?>

<graph id="reseau informatique"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="file:schema.xsd">
  <node name="server" class="node">
    <enum-attr attr-name="IP" attr-type="String"
      attr-value="192.168.1.1" />
    <coordonates x="0" y="0" z="0" />
  </node>
  <node name="PC1" class="node">
    <enum-attr attr-name="IP" attr-type="String"
      attr-value="192.168.1.101" />
    <coordonates x="0" y="0" z="1" />
  </node>
  <node name="PC2" class="node">
    <enum-attr attr-name="IP" attr-type="String"
      attr-value="192.168.1.101" />
    <coordonates x="0" y="1" z="0" />
  </node>
  <node name="PC3" class="node">
    <enum-attr attr-name="IP" attr-type="String"
      attr-value="192.168.1.101" />
    <coordonates x="1" y="0" z="0" />
  </node>
  <node name="PC4" class="node">
    <enum-attr attr-name="IP" attr-type="String"
      attr-value="192.168.1.101" />
    <coordonates x="2" y="4" z="1" />
  </node>
  <node name="PC5" class="node">
    <enum-attr attr-name="IP" attr-type="String"
      attr-value="192.168.1.101" />
    <coordonates x="0" y="6" z="0" />
  </node>
  <node name="PC6" class="node">
    <enum-attr attr-name="IP" attr-type="String"
      attr-value="192.168.1.101" />
    <coordonates x="3" y="9" z="1" />
  </node>
  <node name="PC7" class="node">
    <enum-attr attr-name="IP" attr-type="String"
      attr-value="192.168.1.101" />
    <coordonates x="7" y="9" z="5" />
  </node>
  <node name="PC8" class="node">
    <enum-attr attr-name="IP" attr-type="String"
      attr-value="192.168.1.101" />
    <coordonates x="1" y="2" z="3" />
  </node>
  <node name="PC9" class="node">
    <enum-attr attr-name="IP" attr-type="String"
      attr-value="192.168.1.101" />
    <coordonates x="0" y="0" z="10" />
  </node>
</graph>
```

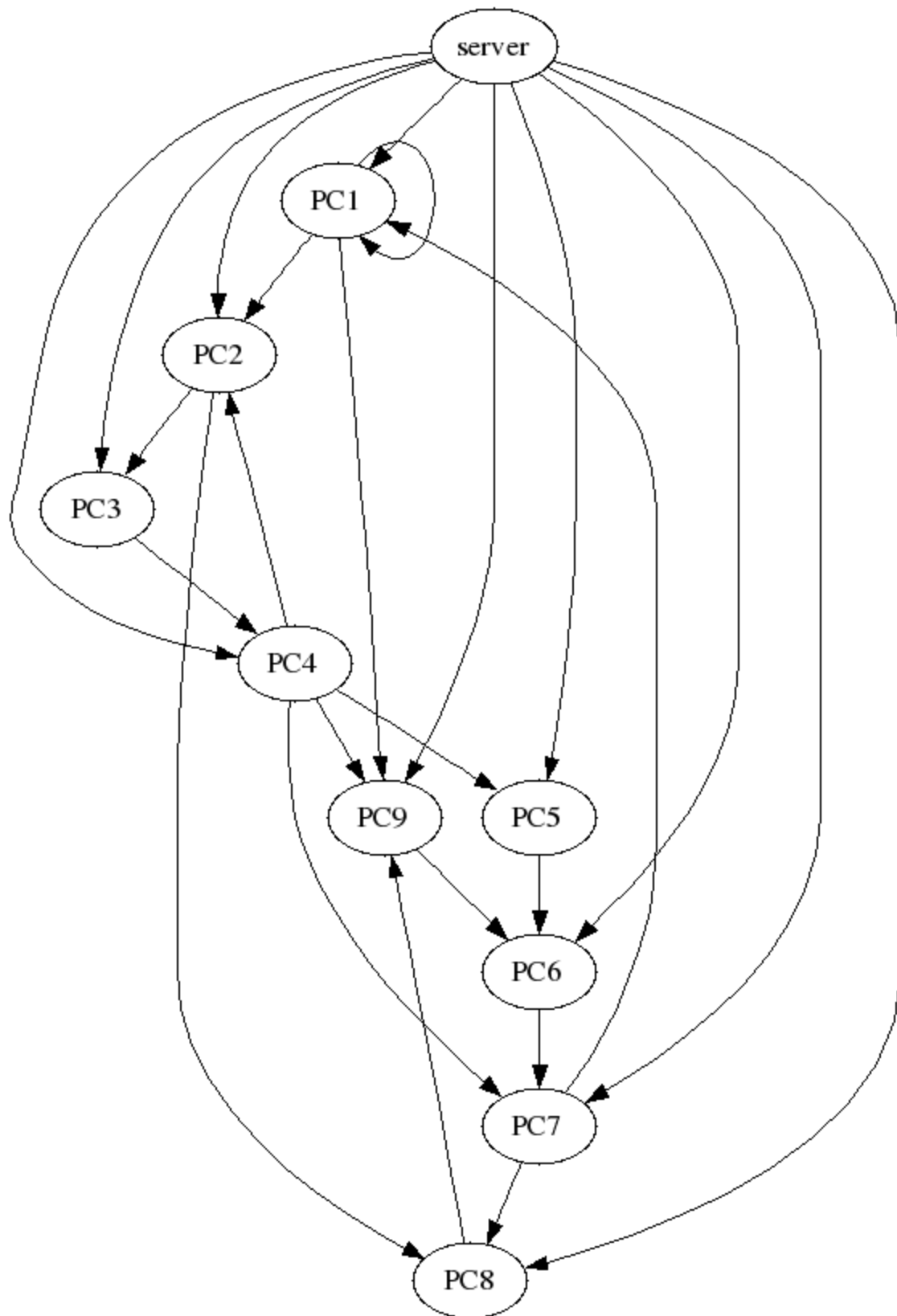


```
<link name="link1" type="arrow" class="link" link_start="server"
                                link_end="PC1">
    <enum-attr attr-name="connectionType" attr-type="String"
              attr-value="wifi" />
</link>
<link name="link2" type="arrow" class="link" link_start="server"
                                link_end="PC2">
    <enum-attr attr-name="connectionType" attr-type="String"
              attr-value="ethernet" />
</link>
<link name="link3" type="arrow" class="link" link_start="server"
                                link_end="PC3">
    <enum-attr attr-name="connectionType" attr-type="String"
              attr-value="cable coaxial" />
</link>
<link name="link4" type="arrow" class="link" link_start="server"
                                link_end="PC4">
    <enum-attr attr-name="connectionType" attr-type="String"
              attr-value="wifi" />
</link>
<link name="link5" type="arrow" class="link" link_start="server"
                                link_end="PC5">
    <enum-attr attr-name="connectionType" attr-type="String"
              attr-value="ethernet" />
</link>
<link name="link6" type="arrow" class="link" link_start="server"
                                link_end="PC6">
    <enum-attr attr-name="connectionType" attr-type="String"
              attr-value="ethernet" />
</link>
<link name="link7" type="arrow" class="link" link_start="server"
                                link_end="PC7">
    <enum-attr attr-name="connectionType" attr-type="String"
              attr-value="ethernet" />
</link>
<link name="link8" type="arrow" class="link" link_start="server"
                                link_end="PC8">
    <enum-attr attr-name="connectionType" attr-type="String"
              attr-value="wifi" />
</link>
<link name="link9" type="arrow" class="link" link_start="server"
                                link_end="PC9">
    <enum-attr attr-name="connectionType" attr-type="String"
              attr-value="ethernet" />
</link>
<link name="link9" type="arrow" class="link" link_start="PC1"
                                link_end="PC2">
    <enum-attr attr-name="connectionType" attr-type="String"
              attr-value="wifi" />
</link>
<link name="link10" type="arrow" class="link" link_start="PC2"
                                link_end="PC3">
    <enum-attr attr-name="connectionType" attr-type="String"
              attr-value="wifi" />
</link>
<link name="link11" type="arrow" class="link" link_start="PC3"
                                link_end="PC4">
    <enum-attr attr-name="connectionType" attr-type="String"
              attr-value="wifi" />
</link>
```

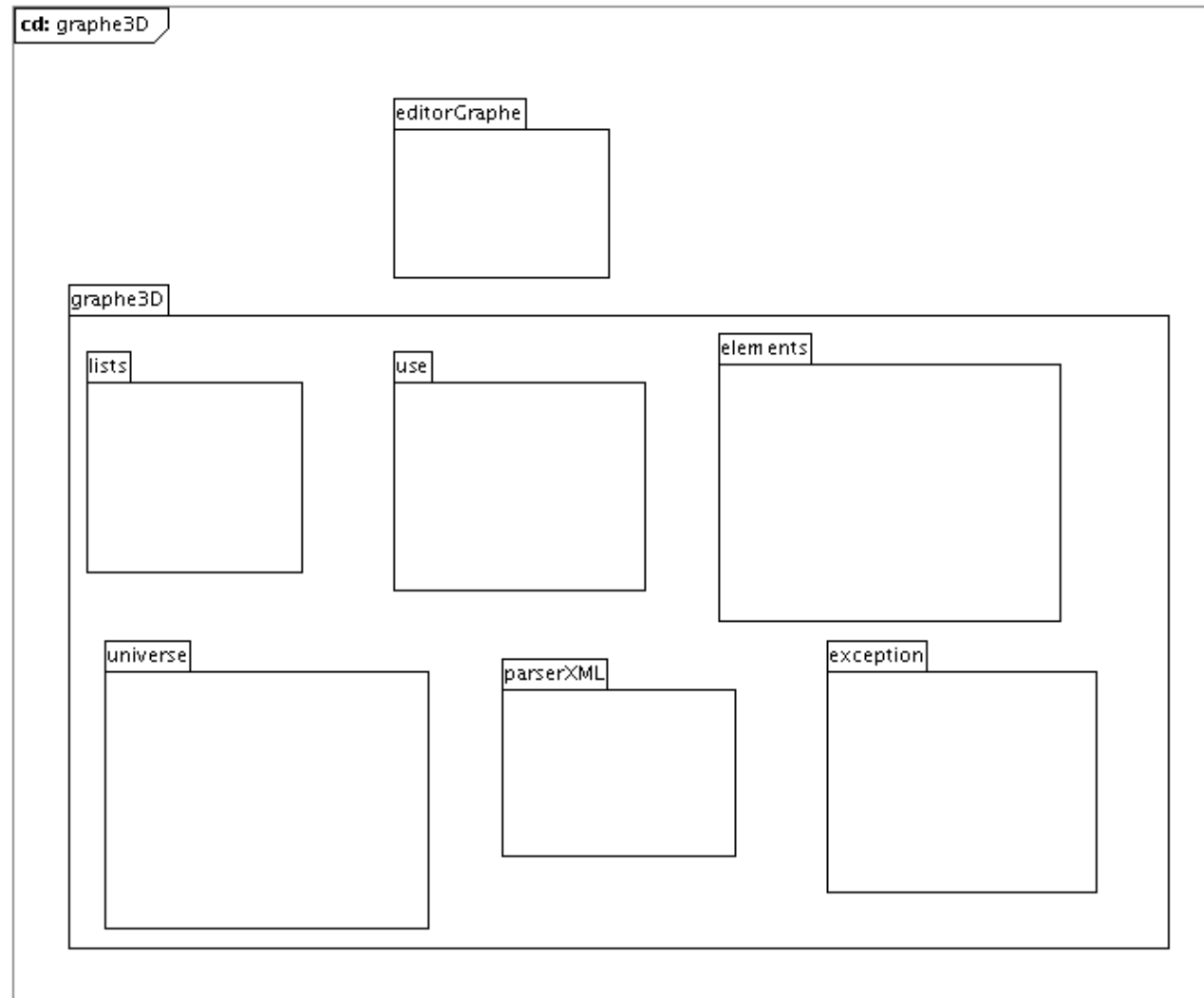
```
<link name="link12" type="arrow" class="link" link_start="PC4"
                                         link_end="PC5">
    <enum-attr attr-name="connectionType" attr-type="String"
              attr-value="wifi" />
</link>
<link name="link13" type="arrow" class="link" link_start="PC5"
                                         link_end="PC6">
    <enum-attr attr-name="connectionType" attr-type="String"
              attr-value="wifi" />
</link>
<link name="link14" type="arrow" class="link" link_start="PC6"
                                         link_end="PC7">
    <enum-attr attr-name="connectionType" attr-type="String"
              attr-value="wifi" />
</link>
<link name="link15" type="arrow" class="link" link_start="PC7"
                                         link_end="PC8">
    <enum-attr attr-name="connectionType" attr-type="String"
              attr-value="wifi" />
</link>
<link name="link16" type="arrow" class="link" link_start="PC8"
                                         link_end="PC9">
    <enum-attr attr-name="connectionType" attr-type="String"
              attr-value="wifi" />
</link>
<link name="link17" type="arrow" class="link" link_start="PC1"
                                         link_end="PC1">
    <enum-attr attr-name="connectionType" attr-type="String"
              attr-value="wifi" />
</link>
<link name="link18" type="arrow" class="link" link_start="PC1"
                                         link_end="PC9">
    <enum-attr attr-name="connectionType" attr-type="String"
              attr-value="wifi" />
</link>
<link name="link19" type="arrow" class="link" link_start="PC2"
                                         link_end="PC8">
    <enum-attr attr-name="connectionType" attr-type="String"
              attr-value="wifi" />
</link>
<link name="link20" type="arrow" class="link" link_start="PC4"
                                         link_end="PC2">
    <enum-attr attr-name="connectionType" attr-type="String"
              attr-value="wifi" />
</link>
<link name="link21" type="arrow" class="link" link_start="PC4"
                                         link_end="PC7">
    <enum-attr attr-name="connectionType" attr-type="String"
              attr-value="wifi" />
</link>
<link name="link22" type="arrow" class="link" link_start="PC4"
                                         link_end="PC9">
    <enum-attr attr-name="connectionType" attr-type="String"
              attr-value="wifi" />
</link>
<link name="link23" type="arrow" class="link" link_start="PC7"
                                         link_end="PC1">
    <enum-attr attr-name="connectionType" attr-type="String"
              attr-value="wifi" />
</link>
```

```
<link name="link24" type="arrow" class="link" link_start="PC9"
      link_end="PC6">
  <enum-attr attr-name="connectionType" attr-type="String"
    attr-value="wifi" />
</link>
</graph>
```

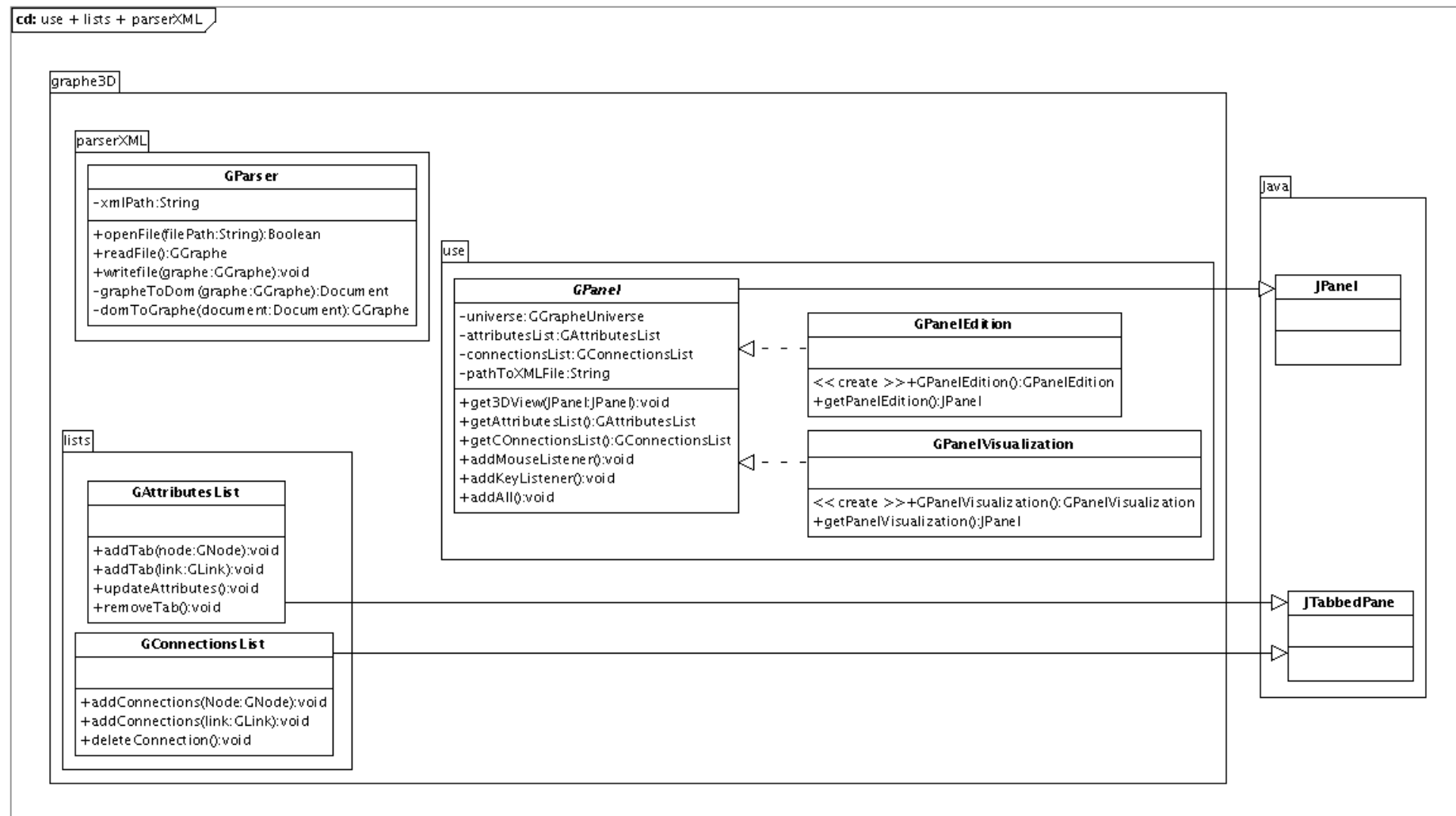
5.3 VUE 2D DE L'EXEMPLE DU FICHIER XML



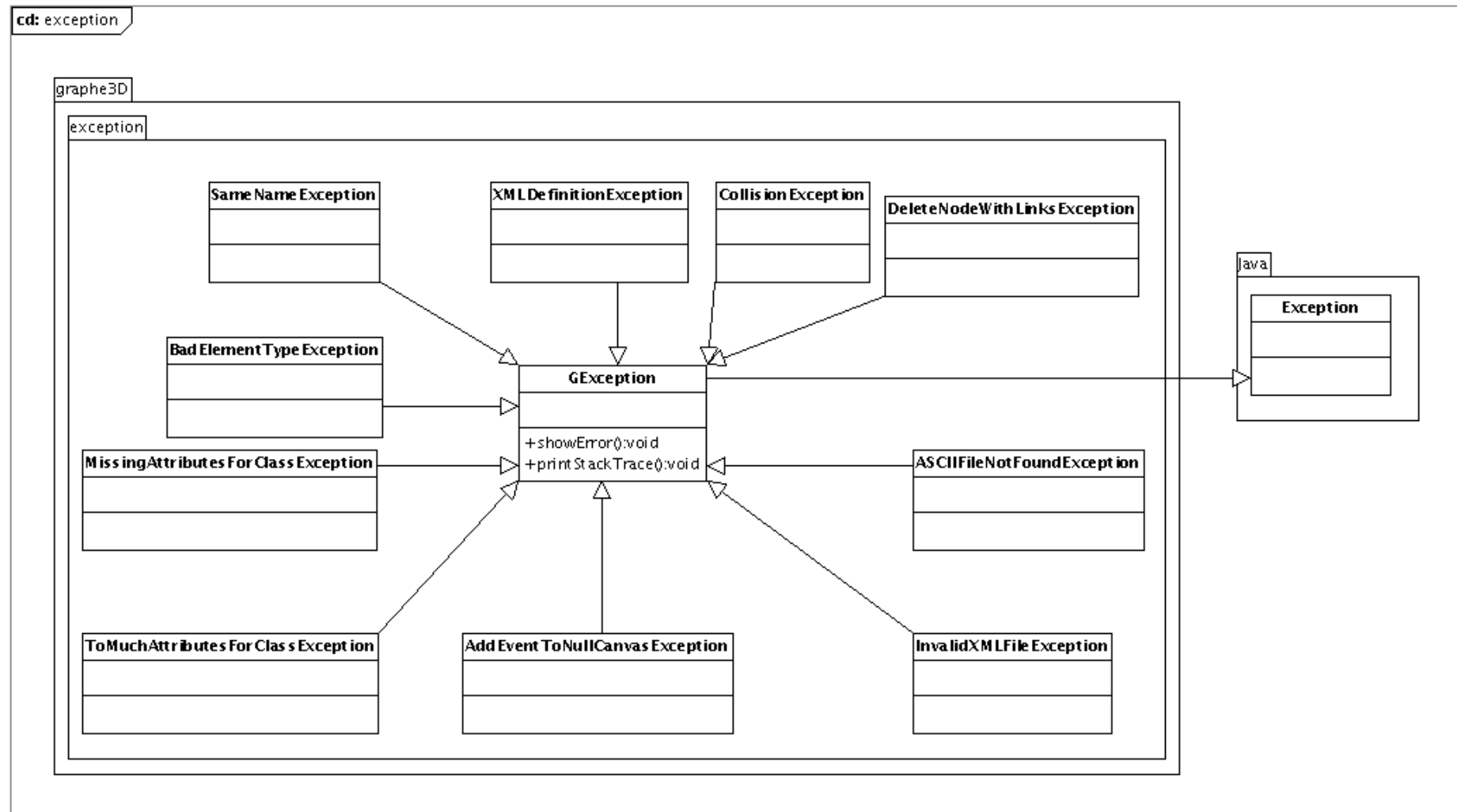
5.4 DIAGRAMME UML DE L'API GRAPHE 3D



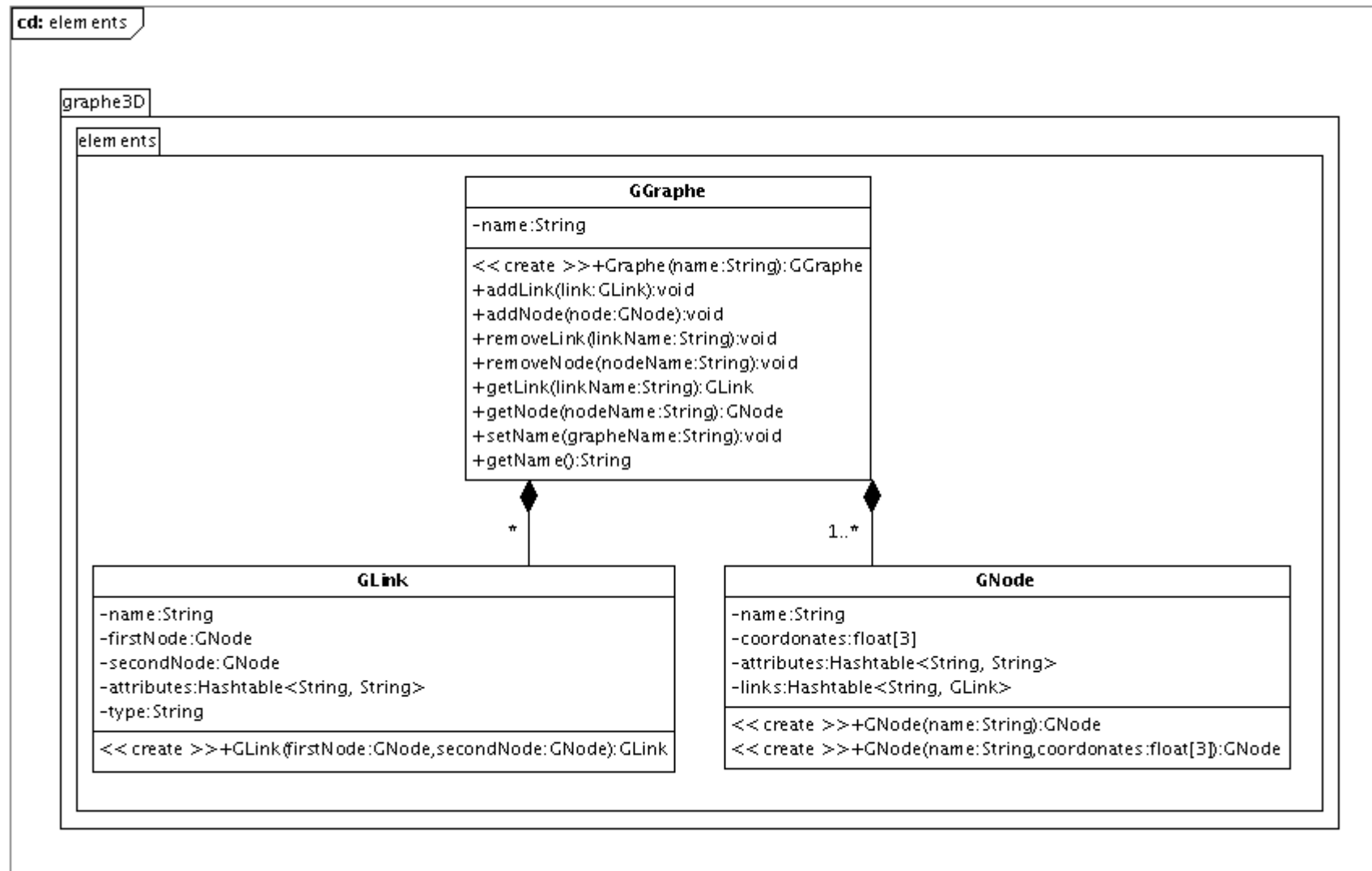
5.4.1 Package « use », « lists » et « parserXML »



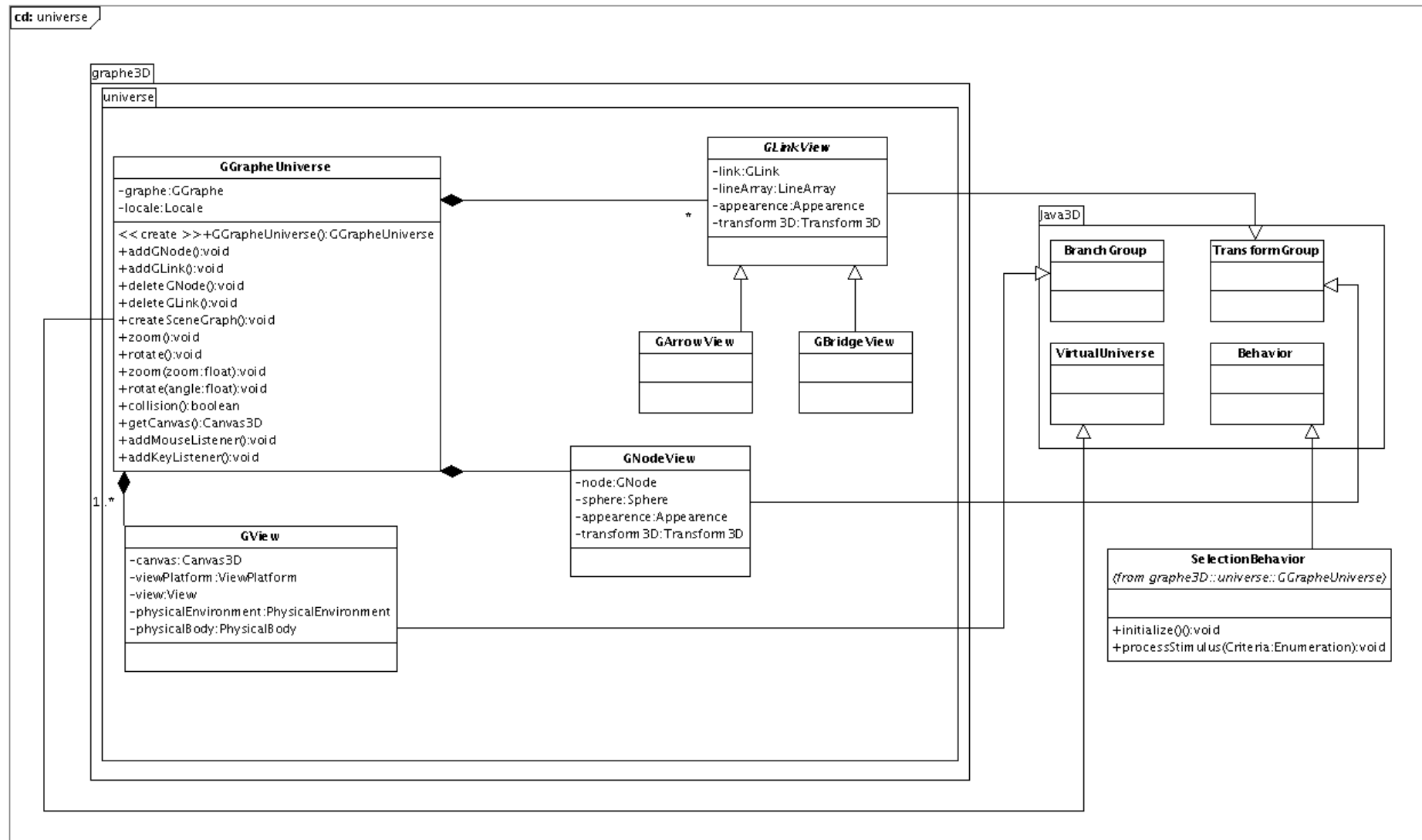
5.4.2 Package « exception »



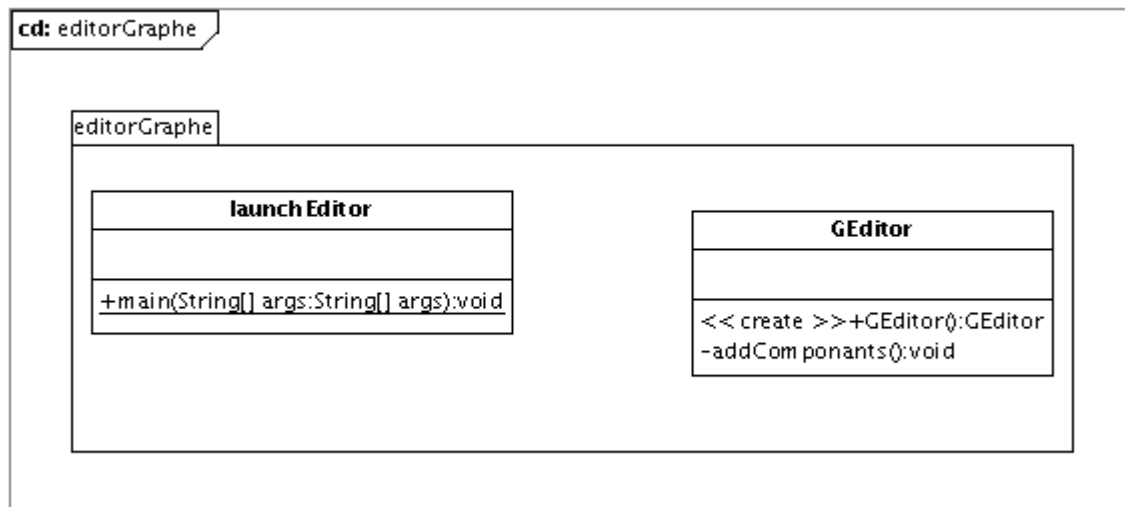
5.4.3 Package « elements »



5.4.4 Package « universe »



5.4.5 Package « editorGraphe »



5.5 CHIFFRAGE PHASE D'ANALYSE

12-mars-07	unité : jour homme					
Projet 30 : Graphe3D	PREVISION					
	TOT	ET1	ET2	ET3	ET4	CDP
ETUDE DU PROJET						
Analyse de la documentation	2,5	0,5	0,5	0,5	0,5	0,5
Analyse de l'existant	1,3	0,3	0,3	0,3	0,3	0,3
Formation aux outils	4,5	1,0	1,0	1,0	1,0	0,5
RDVs avec le tuteur	5,0	1,0	1,0	1,0	1,0	1,0
Rédaction validation spécifications du projet	5,0	1,0	1,0	1,0	1,0	1,0
sous-total chantier 1	18,3	3,8	3,8	3,8	3,8	3,3
CONCEPTION DETAILLEE						
Choix techniques	1,5	0,3	0,3	0,3	0,3	0,3
Définition du format de stockage de graphe (schema XML et un exemple de graphe)	2,8	0,6	0,6	0,6	0,6	0,4
Définition des fonctionnalités	2,8	0,6	0,6	0,6	0,6	0,4
Spécification et validation de l'IHM du visualisateur de Graphe 3D	2,3	0,0	0,0	1,5	0,5	0,3
Spécification et validation de l'IHM de l'éditeur de Graphe 3D	2,3	0,0	1,5	0,0	0,5	0,3
Conception, rédaction et validation du diagramme de classe de l'API Graphe 3D	3,8	1,5	0,0	1,5	0,0	0,8
Conception, rédaction et validation du diagramme de classe de l'éditeur de Graphe 3D	3,8	1,5	1,5	0,0	0,0	0,8
Cas d'utilisation, diagramme de séquence et scénario de remplacement	3,7	0,3	0,3	0,3	2,5	0,3
sous-total chantier 2	23,0	4,8	4,8	4,8	5,0	3,6
CAHIER DE TESTS						
Schema fonctionnel	0,9	0,2	0,2	0,2	0,2	0,1
Schema technique	0,9	0,2	0,2	0,2	0,2	0,1
Tests unitaires	2,2	0,5	0,5	0,5	0,5	0,2
Tests d'intégration	2,2	0,5	0,5	0,5	0,5	0,2
Définition et rédaction des jeux de tests	4,5	1,0	1,0	1,0	1,0	0,5
sous-total chantier 3	10,7	2,4	2,4	2,4	2,4	1,1
ORGANISATION						
Organisation de la phase d'Analyse	1,5	0,0	0,0	0,0	0,0	1,5
Organisation de la phase Réalisation	1,5	0,0	0,0	0,0	0,0	1,5
Gestion des sources et mise en place de l'environnement de développement	1,5	0,0	0,0	0,0	0,0	1,5
sous-total chantier 4	4,5	0,0	0,0	0,0	0,0	4,5
PRESENTATION ORALE DE L'ANALYSE						
Conception et présentation	2,5	0,5	0,5	0,5	0,5	0,5
sous-total chantier 5	2,5	0,5	0,5	0,5	0,5	0,5
TOTAL	59,0	11,5	11,5	11,5	11,7	13,0

ET1 : Daubert Erwan ET3 : Magnin Nicolas CDP : Catric Jérôme
ET2 : Lino Christophe ET4 : Popa Iuliana

5.6 PLANNING PHASE D'ANALYSE

18-mars-07		S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13
Projet 30 : Graphe3D	du	1/1	8/1	15/1	22/1	29/1	5/2	12/2	19/2	26/2	5/3	12/3	19/3	26/3
	au	7/1	14/1	21/1	28/1	4/2	11/2	18/2	25/2	4/3	11/3	18/3	25/3	1/4
ETUDE DU PROJET														
<i>Analyse de la documentation</i>														
<i>Analyse de l'existant</i>														
<i>Formation aux outils</i>														
<i>RDVs avec le tuteur</i>														
<i>Rédaction validation spécifications du projet</i>				1										
sous-total chantier 1														
CONCEPTION DETAILLEE														
<i>Choix techniques</i>														
<i>Définition du format de stockage de graphe (schema XML et un exemple de graphe)</i>														
<i>Définition des fonctionnalités</i>														
<i>Spécification et validation de l'IHM du visualisateur de Graphe 3D</i>														
<i>Spécification et validation de l'IHM de l'éditeur de Graphe 3D</i>														
<i>Conception, rédaction et validation du diagramme de classe de l'API Graphe 3D</i>														
<i>Conception, rédaction et validation du diagramme de classe de l'éditeur de Graphe 3D</i>														
<i>Cas d'utilisation, diagramme de séquence et scénario de remplacement</i>														
sous-total chantier 2														
CAHIER DE TESTS														
<i>Schema fonctionnel</i>														
<i>Schema technique</i>														
<i>Tests unitaires</i>														
<i>Tests d'intégration</i>														
<i>Définition et rédaction des jeux de tests</i>														
sous-total chantier 3														
ORGANISATION														
<i>Organisation de la phase d'Analyse</i>														
<i>Organisation de la phase Réalisation</i>														
<i>Gestion des sources et mise en place de l'environnement de développement</i>														
sous-total chantier 4														
PRESENTATION ORALE DE L'ANALYSE														
<i>Conception et présentation</i>														2
sous-total chantier 5														

Légende

- 1 – Lundi 15 Janvier 17H : Livraison de la fiche de présentation de projet
2 – Jeudi 29 Mars 14h10 : Présentation orale de l'analyse

5.7 CHIFFRAGE PHASE DE REALISATION

19-mars-07	jour homme					
Projet 30 : Graphe3D	PREVISION					
	TOT	ET1	ET2	ET3	ET4	CDP
Codage API Graphe 3D						
Réalisation de l'interface homme-machine	13,2	5,0		8,0		0,2
Réalisation et exécution des tests unitaires	2,5				2,0	0,5
sous-total chantier 1	15,7	5,0	0,0	8,0	2,0	0,7
Codage de l'éditeur de Graphe 3D						
Réalisation de l'interface homme-machine	8,2		8,0			0,2
Réalisation et exécution des tests unitaires	2,5				2,0	0,5
sous-total chantier 2	10,7	0,0	8,0	0,0	2,0	0,7
Codages des éléments communs						
Codage du package du parserXML (création, ouverture, enregistrement, parsing XML)	2,7	2,5				0,2
Codage du package elements (représentation du graphe sous forme d'objets)	1,7	1,5				0,2
Réalisation et exécution des tests unitaires	3,0				2,0	1,0
sous-total chantier 3	7,4	4,0	0,0	0,0	2,0	1,4
Tests et intégration						
Réalisation des tests fonctionnels (api et editeur)	2,5	0,5	0,5	0,5	0,5	0,5
Réalisation d'un programme illustrant l'utilisation de l'API Graphe 3D (interface de visualisation)	1,1			1,0		0,1
Réalisation d'un programme illustrant l'utilisation de l'API Graphe 3D (editeur)	1,1		1,0			0,1
sous-total chantier 4	4,7	0,5	1,5	1,5	0,5	0,7
Documentation						
Réalisation d'un manuel utilisateur : API Graphe 3D	1,7				1,5	0,2
Réalisation d'un manuel utilisateur : Editeur de Graphe 3D	1,7				1,5	0,2
sous-total chantier 5	3,4	0,0	0,0	0,0	3,0	0,4
Validation du client						
Validation du produit par le client	1,0	0,2	0,2	0,2	0,2	0,2
Livraison	2,5	0,5	0,5	0,5	0,5	0,5
Maintenance	7,0	1,5	1,5	1,5	1,5	1,0
sous-total chantier 6	10,5	2,2	2,2	2,2	2,2	3,2
Bilan						
Rédaction du bilan du projet	4,0	0,5	0,5	0,5	0,5	2,0
Préparation de la présentation orale du bilan du projet	1,5	0,3	0,3	0,3	0,3	0,3
sous-total chantier 7	5,5	0,8	0,8	0,8	0,8	2,3
TOTAL	57,9	12,5	12,5	12,5	12,5	9,4

ET1 : Daubert Erwan ET3 : Magnin Nicolas ET4 : Popa Iuliana
ET2 : Lino Christophe CDP : Catric Jérôme

5.8 PLANNING PHASE DE REALISATION

18-mars-07		S13	S14	S15	S16	S17	S18	S19	S20	S21	S22
Projet 30 : Graphe3D	du	26/3	2/4	9/4	16/4	23/4	30/4	7/5	14/5	21/5	28/5
	au	1/4	8/4	15/4	22/4	29/4	6/5	13/5	20/5	27/5	3/6
Codage API Graphe 3D											
Réalisation de l'interface homme-machine											
Réalisation et exécution des tests unitaires											
sous-total chantier 1											
Codage de l'éditeur de Graphe 3D											
Réalisation de l'interface homme-machine											
Réalisation et exécution des tests unitaires											
sous-total chantier 2											
Codages des éléments communs											
Codage du package du parserXML (création, ouverture, enregistrement, parsing XML)											
Codage du package elements (représentation du graphe sous forme d'objets)											
Réalisation et exécution des tests unitaires											
sous-total chantier 3											
Tests et intégration											
Réalisation des tests fonctionnels (api et editeur)											
Réalisation d'un programme illustrant l'utilisation de l'API Graphe 3D (interface de visualisation)											
Réalisation d'un programme illustrant l'utilisation de l'API Graphe 3D (editeur)											
sous-total chantier 4											
Documentation											
Réalisation d'un manuel utilisateur : API Graphe 3D											
Réalisation d'un manuel utilisateur : Editeur de Graphe 3D											
sous-total chantier 5											
Validation du client											
Validation du produit par le client							1			2	
Livraison							1			2	
Maintenance											
sous-total chantier 6											
Bilan											
Rédaction du bilan du projet											
Préparation de la présentation orale du bilan du projet										3	
sous-total chantier 7											

Légende

- 1 – Lundi 30 Avril : Livraison et réception (avant maintenance)
- 2 – Livraison et réception (après maintenance)
- 3 – Jeudi 24 Mai : Présentation orale des résultats

5.9 EXEMPLE D'UN TEST UNITAIRE AVEC JUNIT

```
package graphe3D.elements;

import junit.framework.TestCase;
import java.util.Hashtable;

/**
 * Classe de test pour la classe {@link GNode}.
 */
public class GNodeTest extends TestCase {

    /**
     * Test du premier constructeur de la classe.
     * GNode avec un nom et coordonnées par défaut.
     */
    public final void testGNode1() {

        // Création des éléments à tester :
        GNode nodel = new GNode("node one");

        // Vérification du nom du nœud
        assertNotNull(nodel.getName());
        assertEquals(nodel.getName(), "node one");

        // Vérification des coordonnées
        assertEquals(nodel.getCoordonates()[0], new Float(0));
        assertEquals(nodel.getCoordonates()[1], new Float(0));
        assertEquals(nodel.getCoordonates()[2], new Float(0));

        // Verification des attributs
        assertNull(nodel.getAttributes());
    }

    /**
     * Test du second constructeur de la classe.
     * GNode avec un nom et des coordonnées.
     */
    public final void testGNode2() {

        // Création des éléments à tester :
        GNode nodel = new GNode("node two",2,4,5);

        // Vérification du nom du noeud
        assertNotNull(nodel.getName());
        assertEquals(nodel.getName(), "node two");

        // Vérification des coordonnées
        assertEquals(nodel.getCoordonates()[0], new Float(2));
        assertEquals(nodel.getCoordonates()[1], new Float(4));
        assertEquals(nodel.getCoordonates()[2], new Float(5));

        // Verification des attributs
        assertNull(nodel.getAttributes());
    }
}
```

```
/**
 * Test de la méthode permettant de changer le nom d'un nœud.
 * On vérifie qu'un changement effectué par l'appel du modificateur
 * est correctement enregistré.
 */
public final void testSetName(){

    // Création des éléments à tester :
    GNode node3 = new GNode("node three");

    // Modification du nom du noeud
    node3.setName("Node 3");

    // Verification du nom du noeud
    assertNotNull(node3.getName());
    assertEquals(node3.getName(), "Node 3");
}

/**
 * Test de la méthode permettant de changer les coordonnées d'un
 * nœud.
 * On vérifie qu'un changement effectué par l'appel du modificateur
 * est correctement enregistré.
 */
public final void testSetCoordonates(){

    // Création des éléments à tester :
    GNode node3 = new GNode("my node");

    // Modification des coordonnées d'un noeud
    node3.setCoordonates(new float[]{3,2,5});

    // Verification des coordonnees
    assertEquals(node3.getCoordonates()[0], new Float(3));
    assertEquals(node3.getCoordonates()[1], new Float(2));
    assertEquals(node3.getCoordonates()[2], new Float(5));
}

/**
 * Test de la méthode permettant d'ajouter des attributs à un nœud.
 * On vérifie qu'un changement effectué par l'appel du modificateur
 * est correctement enregistré.
 */
public final void testSetAttributes(){

    // Création des éléments à tester :
    GNode node3 = new GNode("my node");
    Hashtable attribut1 = new Hashtable<String, String>();
    attribut1.put("IP", "192.168.1.1");
    attribut1.put("Type", "Ethernet");

    // Modification de la liste des attributs
    node3.setAttributes(attribut1);

    // Vérification des attributs
    assertNotNull(node3.getAttributes());
    assertEquals(node3.getAttributes(), attribut1);
}
}
```


5.10 FICHIER POUR LES FICHES DE TESTS

5.10.1 Fichier « nonValideXsd1.xml »

```
<graphe>
  <node coordx=1 coordz=0/>
  <baliseinconnue>
  <node coordx=0 coordy=0 coordz=0 attributinconnu=val/>
</graphe>
```

5.10.2 Fichier « nonValideXsd2.xml »

```
<graphe>

</graphe>
```

5.10.3 Fichier « nonValideSem.xml »

```
<graphe>
  <arrow id='lien1' type=typeinconnu nodestart='node1'
                                         nodeend='node1' />
  <node id='node1' coordx=0 coordy=0 coordz=0>
    <attribut-optionel type=typenonautorise val=0/>
  </node>
  <node id='node1' coordx=1 coordy=1 coordz=1/>
  <arrow id='lien1' type=typeinconnu nodestart='node1'
                                         nodeend='node1' />
</graphe>
```

5.10.4 Fichier « valide.xml »

```
<graphe>
  <node name="node1" class="node">
    <enum-attr attr-name="IP" attr-type="String"
              attr-value="192.168.1.101" />
    <coordonates x="5" y="5" z="1" />
  </node>
  <node name="node2" class="node">
    <coordonates x="0" y="0" z="1" />
  </node>
  <link name="link23" type="arrow" class="link" link_start="node1"
                                             link_end="node2">
</graphe>
```

5.10.5 Fichier « collision.xml »

```
<graphe>
  <node name="PC1" class="node">
    <coordonates x="0" y="0" z="1" />
  </node>

  <node name="PC2" class="node">
    <coordonates x="0" y="0" z="1" />
  </node>

  <node name="PC3" class="node">
    <coordonates x="0" y="0" z="1" />
  </node>
</graphe>
```