TD11 : Patron « Modèle Vue Contrôleur »
UML

Pierre Gérard
pierre.gerard@univ-paris13.fr

DUT Informatique S2D
Université de Paris 13

Résumé

Ce TD illustre la composition de plusieurs patrons simples pour former une architecture plus complexe. Nous aborderons progressivement le patron MVC en le présentant pas à pas comme une composition des patrons « composite », « observer »et « strategy ».

1 Problème

Nous cherchons à produire un logiciel d'édition pour des documents structurés.

```
<section>
  <titre>TD</titre>
  <contenu>
    Les exercices ne sont pas indépendants.
    <section>
      <titre>Exercice 1</titre>
      <contenu>
        <section>
          <titre>Question 1</titre>
          <contenu>En premier lieu,...</contenu>
        </section>
        <section>
          <titre>Question 2</titre>
          <contenu>Maintenant, modélisez...</contenu>
        </section>
      </contenu>
    </section>
    <section>
      <titre>Exercice 2</titre>
      <contenu>Il est à noter que...</contenu>
    </section>
  </contenu>
</section>
```

TD

Les exercices ne sont pas indépendants.

Exercice 1

Question 1

En premier lieu,...

Question 2

Maintenant, modélisez...

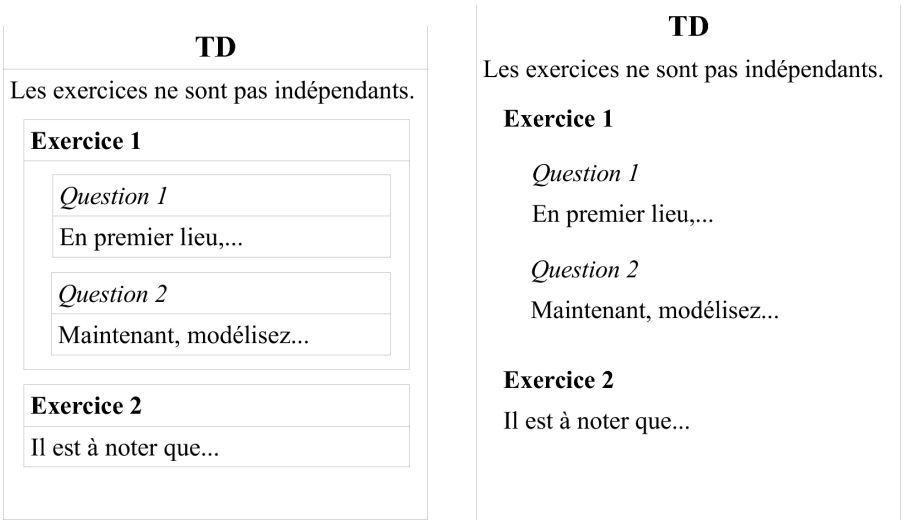
Exercice 2

Il est à noter que...

Les documents sont structurés à l'aide de balises à la façon des documents XML. A gauche on trouve le code source du document et à droite, une manière de l'interpréter pour en faire quelque chose de plus présentable. Les balises <section> permettent de structurer le document en parties et en sous parties. Les balises <titre> et <contenu> définissent des compartiments de données pour chaque section. Le compartiment « contenu »peut contenir des sous-sections. Chaque cadre dans les figures représente une vue.

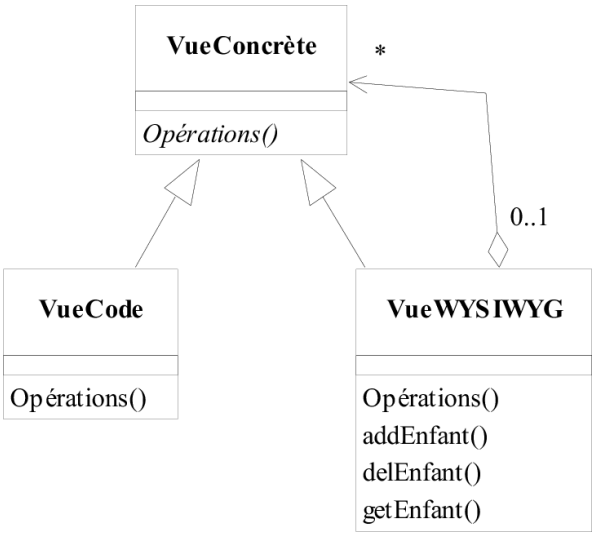
2 Patron « composite »(vue-vue)

Pour visualiser le document ou seulement une section, on peut avoir recours à une vue du code, mais on veut également une représentation WYSIWYG (What You See Is What You Get) comme celle ci-dessus à droite. Pour ce faire, on considère que chaque section peut l'objet d'une vue et que la vue en question peut contenir des sous-vues représentant les sous-sections.



Dans la vue représentée à gauche, chacune des vues est décomposée en sous-vues. Dans vue représentée à droite, on a plutôt choisi de représenter chacun des exercices au moyen de leur code, sans les décomposer. Ces modes de représentation sont tous les deux possibles dans le logiciel que l'on cherche à concevoir.

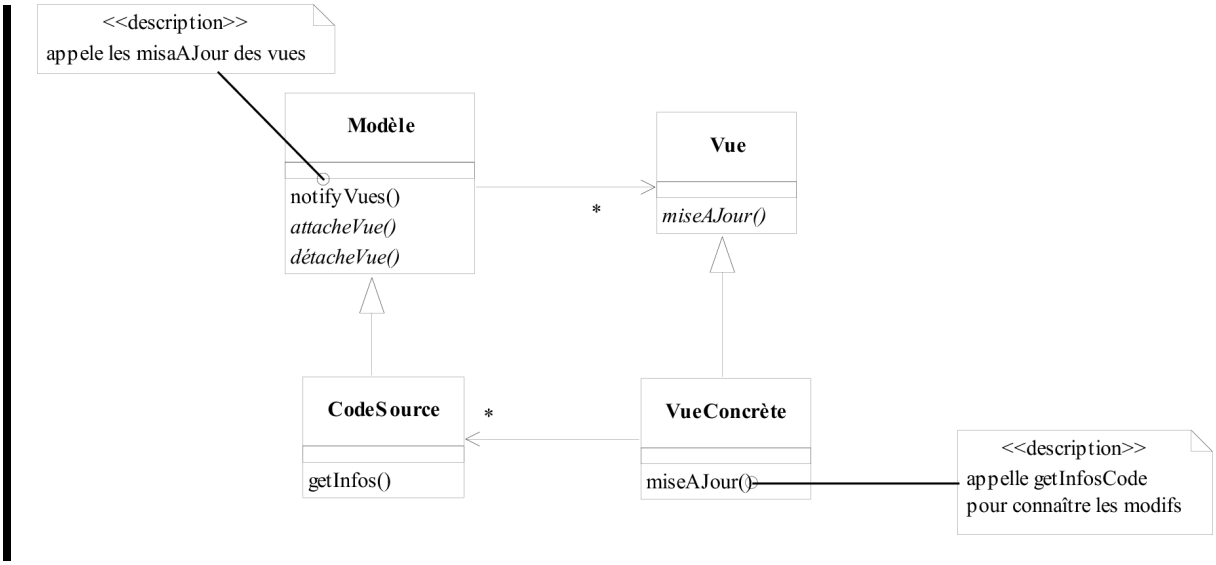
Question : Sans prendre en compte la manière dont est structuré le code source du document, en se restreignant à la seule organisation des vues entre-elles, proposez une modélisation de la structure hiérarchique des vues. Vous pourrez vous appuyer sur le patron « composite ».



3 Patron « observer »(vue-modèle)

L'application que nous voulons développer doit pouvoir gérer plusieurs vues simultanément. Par exemple, on pourra avoir en même temps une vue du code source complet et une vue WYSIWYG ou tout autre intermédiaire. Le code source du document est appelé le modèle. Dès que ce dernier est modifié, on veut que les vues se redessinent de manière adéquate.

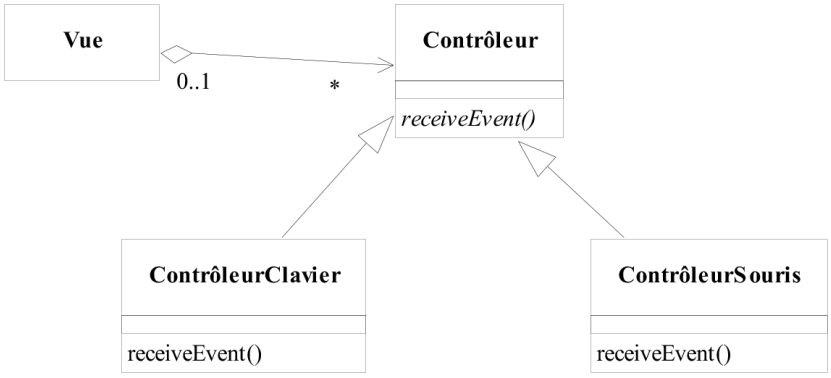
Question : Proposez une modélisation UML qui rende compte de cette fonctionnalité. Vous pourrez utiliser le patron de conception « observer »en considérant les vues comme des observateurs du modèle.



4 Patron « strategy »(vue-contrôleur)

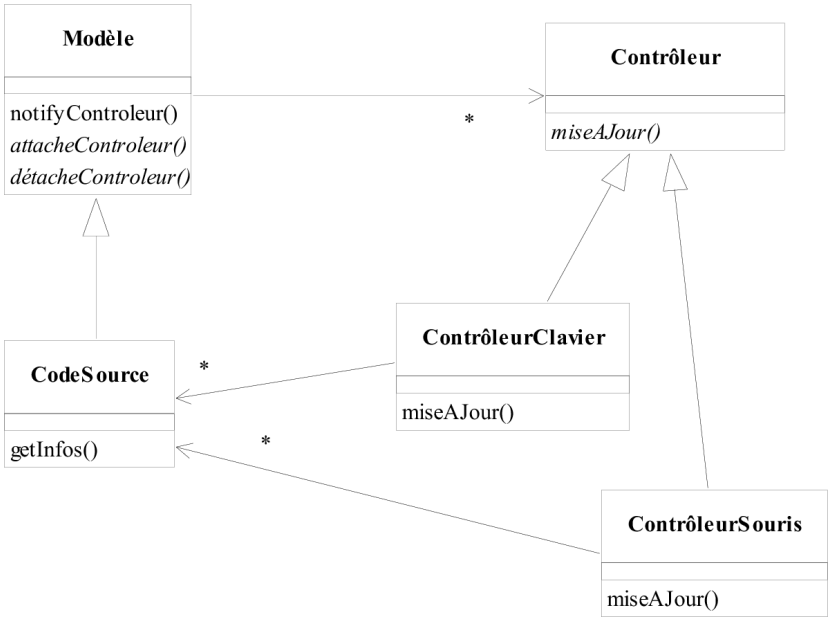
En plus des vues et du modèle du document, on souhaite disposer de moyens d'interaction avec le système pour modifier le modèle. Pour ce faire, on se dote de contrôleurs associés aux vues. Les contrôleurs réagissent aux événements extérieurs (clics sur des boutons, frappes clavier...) et modifient le modèle en réponse aux événements en question. La manière dont un contrôleur agit sur le modèle dépend de la vue à laquelle il est associé. Par exemple, une clic de souris sur une vue du code source pourra permettre de déplacer le curseur alors qu'un clic semblable sur une vue hiérarchique permettra d'accéder à certaines de ses propriétés. Ainsi, les contrôleurs sont semblables à des algorithmes qui, en fonction du contexte défini par la vue, déclencheront des opérations différentes

Question : Utilisez le patron « strategy »pour proposer une modélisation du lien entre vues et contrôleurs.



5 Patron « observer »(contrôleur-modèle)

Il peut arriver que l'état du modèle ait un impact sur les contrôleurs. Par exemple, il se peut qu'en fonction du modèle, certaines options ne soient pas disponibles (menus, cases à cocher...). Les contrôleurs doivent donc être avertis de toutes les modifications du modèle. Utilisez le patron « observer »pour modéliser ce lien entre modèles et contrôleurs. Nous considérerons qu'il y deux types de contrôleurs : pour gérer la souris et pour gérer le clavier.

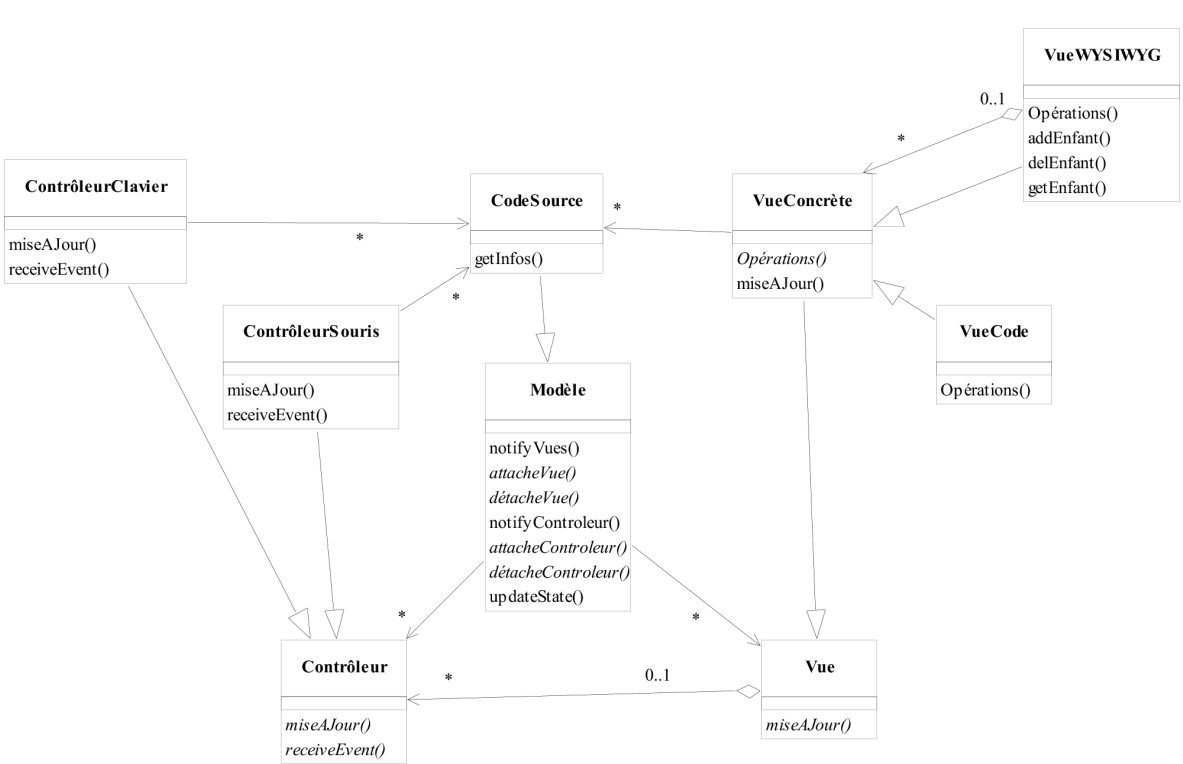


6 Fusion de l'utilisation de plusieurs patrons de conception

Dans les exercices précédents, nous avons modélisé différentes parties d'un système en nous focalisant successivement sur les relations entre :

- vues et sous-vues ;
- vues et modèle ;
- vues et contrôleurs ;
- contrôleurs et modèles.

Question : Combinez maintenant les différents sous-modèles élaborés pour former un modèle global du système complet. Ce modèle utilise trois patrons de conception : composite, observer et strategy, à chaque fois pour obtenir des fonctionnalités différentes mais complémentaires. En outre, on prendra ici en compte le fait que lorsqu'un contrôleur reçoit un événement, il demande au modèle de se mettre à jour.



7 Patron « Modèle - Vue - Contrôleur »

Le patron de conception MVC (Modèle-Vue-Contrôleur) est une combinaison des trois patrons composite, observer et strategy. Il est très fréquemment utilisé pour construire des interfaces graphiques. Java Swing ou Struts, par exemple, sont des adaptations de MVC.

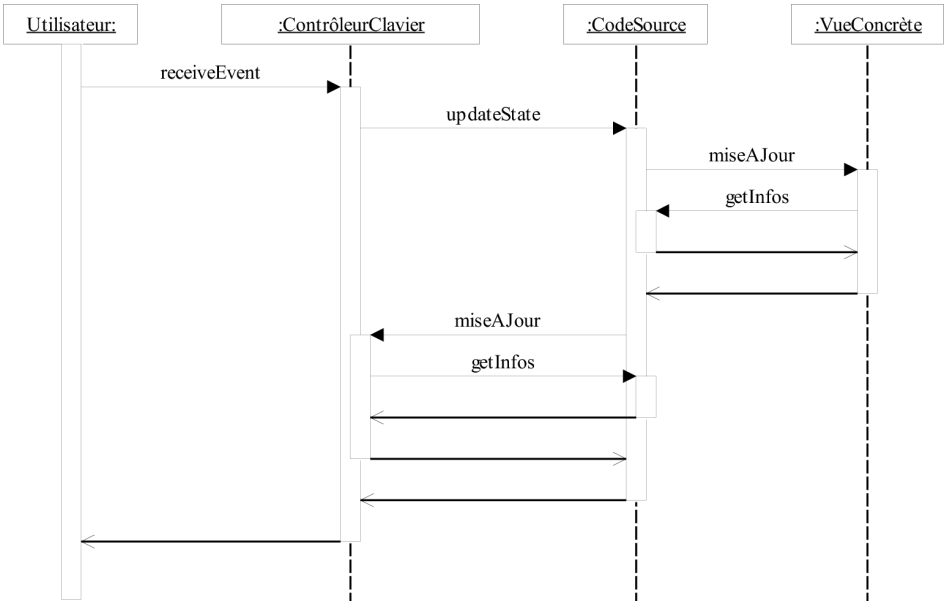
Question : En vous appuyant sur l'exemple précédent et sur la spécification de ces trois patterns, proposez une spécification de la structure du patron « Modèle-Vue-Contrôleur ».

Si vous avez fait des aménagements concernant les patrons originaux pour produire certains sous-modèles, vous veillerez à ce que les hypothèses correspondantes n'affectent pas la généricité de la structure proposée pour le patron MVC. Il est à noter que d'autres patterns peuvent se retrouver dans des modèles MVC. Par exemple, il est d'usage d'utiliser le patron « abstract factory »pour la création d'objets. Dans le modèle de structure que vous présenterez, vous ne manquerez pas d'apporter des améliorations comme la généralisation des contrôleurs et des vues en éléments dits « observables »par le modèle.

La correction ressemble beaucoup à la correction précédente, si ce n'est la généralisation de Vue et Contrôleur en Observable et la réduction du nombre de méthodes attache, détache... puisqu'il n'y a plus que des Observables à attacher, quels que soient leur type.

Considérez un objet « modèle concret », une « vue concrète »quelconque et un « contrôleur »comme des lignes de vie.

Question : Proposez un diagramme de séquence pour modéliser les interactions entre ces différents éléments, à partir du moment où un événement est intercepté par un observateur concret.



Correcetion TRES approximative. Et en plus, ça n'est pas de l'UML2. Pour bien faire, supprimer l'utilisateur, mettre un cadre autour du diagramme et faire venir l'événement du cadre.