LOG2410 - Conception logicielle

Examen final - Hiver 2015

Documentation : aucune documentation permise.
Calculatrice non-programmable : non permise.
Date : 2 mai 2015.
Cet examen comprend 4 questions sur 3 pages.

Mise en contexte. Le texte qui suit décrit les principales fonctionnalités du logiciel *OrthoDesign3D*.

Le logiciel *OrthoDesign3D* est une application conçue pour être utilisée en milieu hospitalier, qui peut s'installer sur un ordinateur portable, et qui permet de contrôler le processus d'acquisition de données en 3D d'un membre d'un patient ayant subi une fracture afin de permettre la conception et la fabrication d'une orthèse sur mesure. L'application *OrthoDesign3D* requiert l'installation, sur l'ordinateur, d'un dispositif d'acquisition laser capable de capturer les coordonnées des points en 3D et la couleur des surfaces (texture) du membre fracturé. Le dispositif d'acquisition laser est contrôlé directement par l'application *OrthoDesign3D* à l'aide d'une bibliothèque de programmation (API) vendue par le fabriquant du dispositif. L'interface du système *OrthoDesign3D* doit être conçue pour fonctionner en mode graphique et toutes les options du système doivent être affichées au sein d'une interface graphique comprenant des menus, des boutons et une fenêtre principale de visualisation, contrôlées par l'usager à l'aide de la souris. Le logiciel permet au/à la technicien(ne) orthésiste :

1- De créer un nouveau dossier patient ou d'ouvrir le dossier d'un patient existant afin d'ajouter de nouvelles données ou de consulter les données déjà enregistrées. Le patient est identifié par un numéro unique inscrit sur sa carte d'hôpital et par son numéro d'assurance maladie.
2- D'acquérir un nuage de points en 3D et la texture d'une partie du membre du patient à l'aide du scanner laser.
3- De sélectionner un ou plusieurs points dans un nuage de points.
4- De déplacer, tourner par rapport à un axe ou éliminer des points sélectionnés.
5- De fusionner les nuages de points de différentes parties du membre du patient afin d'obtenir un nuage de points couvrant le membre au complet.
6- De construire des surfaces lisses à partir d'un nuage de point complet.
7- De choisir un patron d'orthèse approprié pour le membre fracturé parmi un catalogue de modèles d'orthèses.
8- De mettre à l'échelle et d'ajuster les paramètres de l'orthèse afin que celle-ci soit bien adaptée aux surfaces reconstruites.
9- De valider que tous les paramètres de l'orthèse sont cohérents et ont été correctement définis.
10- D'annuler une opération (*Undo*) ou de réappliquer une opération annulée (*Redo*).

Une fois l'orthèse conçue virtuellement, le système *OrthoDesign3D* permet de visualiser celle-ci en 3D, et permet à l'orthésiste de présenter le résultat à l'orthopédiste afin de procéder à des ajustements spécifiques avant la fabrication. L'orthopédiste peut alors :

11- Effectuer une simulation afin de déterminer les forces qui seront appliquées par l'orthèse lorsqu'elle sera installée sur le membre.
12- Visualiser la répartition et l'intensité des forces à certains points clés sur le membre.
13- Ajuster certains paramètres de forme de l'orthèse afin de modifier la répartition des forces.

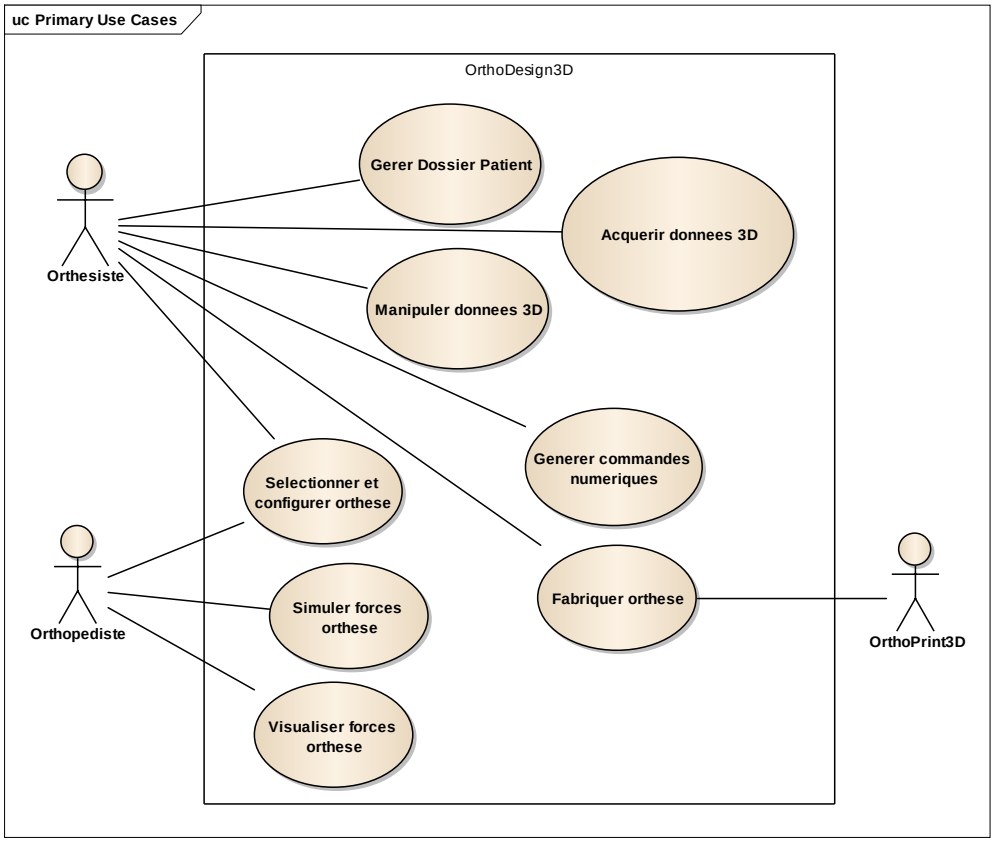
Une fois les ajustements des forces effectués, le système *OrthoDesign3D* permet à l'orthésiste de préparer un fichier de commande numérique approprié pour une imprimante 3D afin de procéder à la fabrication des pièces de l'orthèse. L'orthésiste peut alors :

14- Sélectionner certaines ou toutes les pièces de l'orthèse.

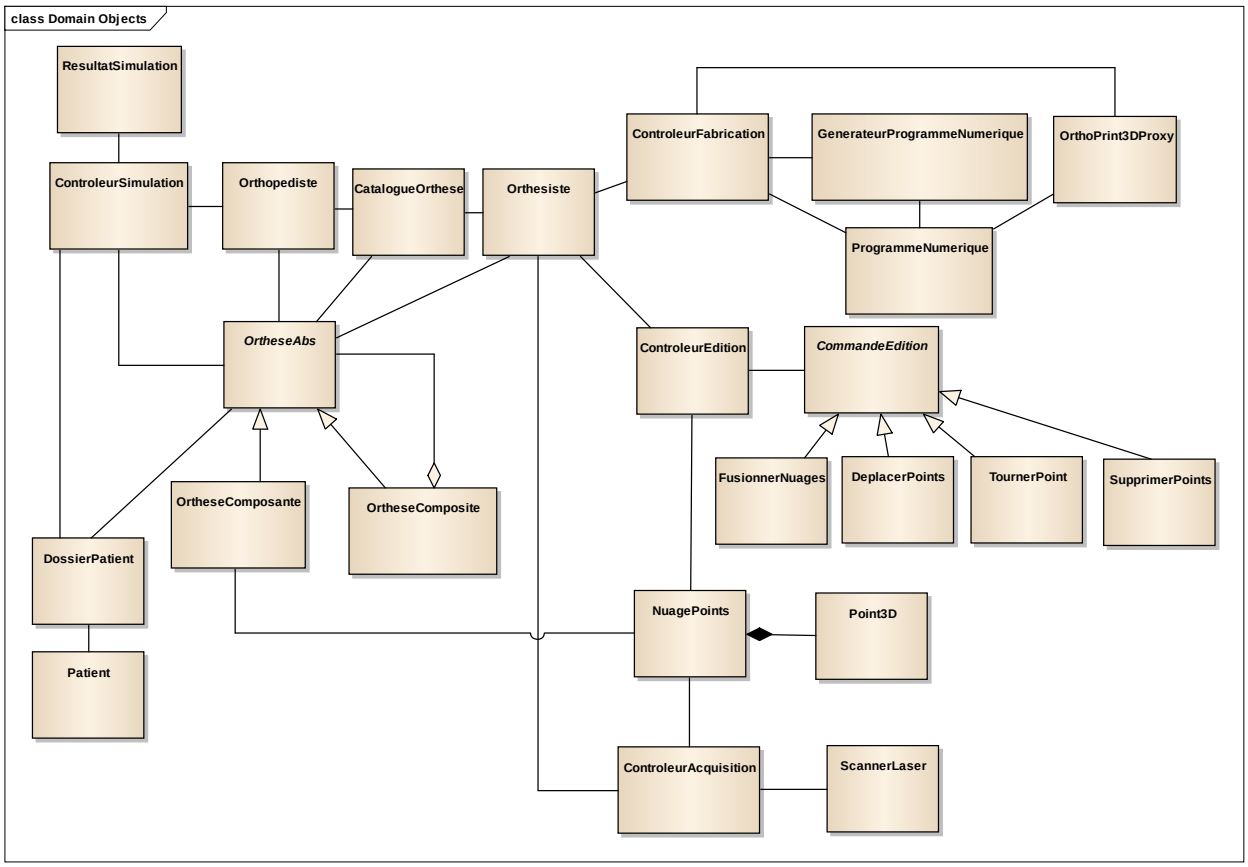
15- Lancer la génération du fichier de commande numérique pour les pièces de l'orthèse sélectionnées.
16- Soumettre le fichier de commande au serveur *OrthoPrint3D,* qui contrôle l'imprimante 3D afin de procéder à l'impression 3D des pièces sélectionnées.

Question 1 - Processus de conception et cas d'utilisation (18 points)

a) Tracez le diagramme UML de haut niveau (diagramme de contexte) des cas d'utilisation du logiciel *OrthoDesign3D* (7 points).



b) Proposez un diagramme de concepts pour le logiciel *OrthoDesign3D* (7 points).



c) Définissez en vos propres mots ce qu'on entend par partie prenante. Identifiez au moins deux parties prenantes au système *OrthoDesign3D*. Justifiez votre réponse. (2 points)
Une partie prenante est un intervenant dans le développement du système qui a un intérêt à ce que le système soit développé et fonctionnel, mais qui ne joue pas nécessairement un rôle ou n'utilise pas nécessairement le système directement. Le patient et les gestionnaires du système de santé sont des exemples de parties prenantes.

d) Dans son livre "*UML 2 et les design patterns*", Larman décrit en détail les patrons GRASP. Parmi ceux-ci, les patrons *Faible couplage* et *Cohésion élevée* jouent tous les deux un rôle important afin d'assurer la qualité globale d'une conception logicielle. Décrivez chacun de ces patrons et donnez une conséquence positive pour chacun (2 points).
Le patron Faible couplage suggère qu'une classe ne devrait pas être lié à un trop grand nombre d'autres classes du système. En réduisant le couplage entre les classes, on favorise la réutilisation. Le patron cohésion élevée suggère qu'une classe ne devrait être impliquée que dans un petit nombre de responsabilités toutes liées les unes aux autres. En maintenant une cohésion des classes, on facilite la compréhension des classes et du système.

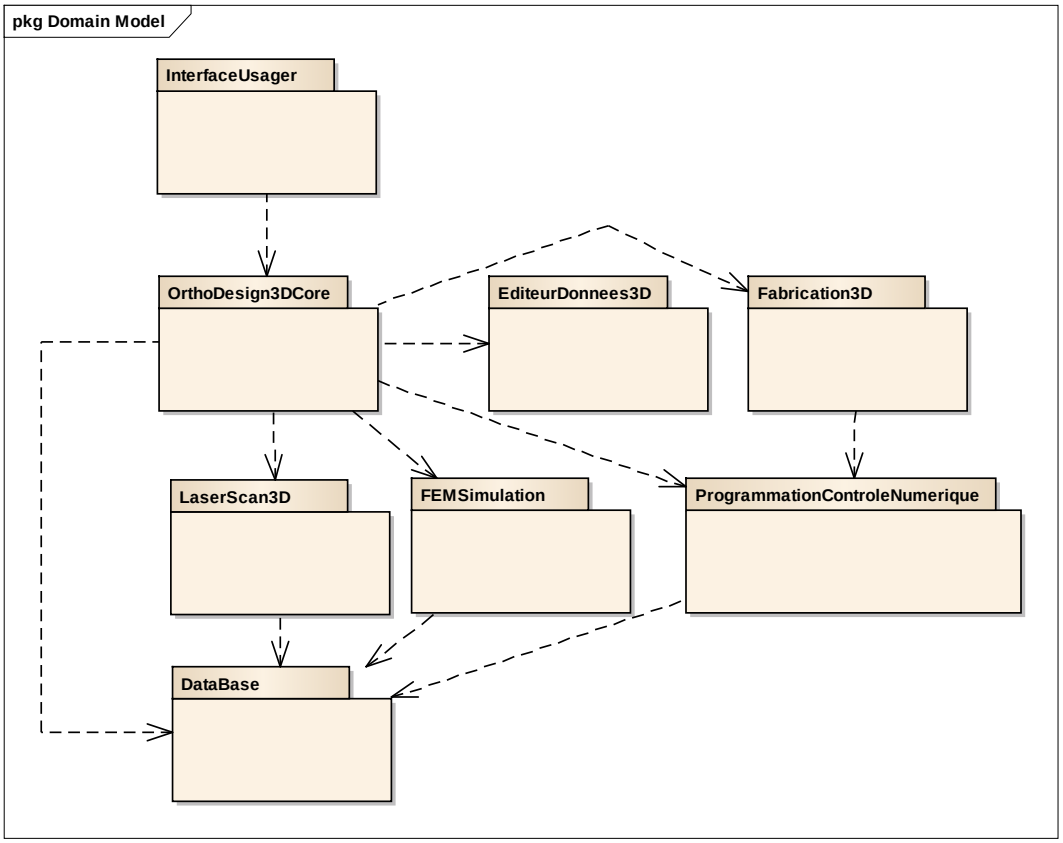
Question 2 - Architecture logique (7 points)

Les logiciels modernes sont développés selon des architectures logiques comportant plusieurs couches ou niveaux (*multi-tier architecture*).

- Indiquez deux avantages importants de la décomposition d'un logiciel en une architecture multi-niveaux (2 points),
  Au moins deux parmi :
  1. La séparation de la logique d'application dans des composantes séparées qui peuvent être réutilisé dans d'autres systèmes,
  2. La possibilité de répartir les niveaux sur différents nœuds de calcul, et dans différents processus,

3. L'assignation de développeurs à la construction de chaque niveau: parallèlisation des efforts et spécialisation des intervenants.

- Lors de la conception d'une architecture multi-niveaux, il est recommandé, pour accéder aux modules de la couche service, de passer par des *Façade*s. Expliquer le rôle de la Façade et discutez des avantages liés à son utilisation (2 points).
  Une façade fournit un point d'entrée clairement identifiable et simplifie l'utilisation d'un module. L'utilisation d'une façade simplifie le remplacement d'un module par un autre en isolant les clients des détails internes du module. La façade augmente donc la modularité du système.
- Proposez une décomposition architecturale multi-niveaux du logiciel *OrthoDesign3D*. Fournissez votre réponse sous la forme d'un diagramme de paquetages (3 points).



Question 3 - Patrons de conception (18 points)

Durant la phase d'analyse et conception du logiciel *OrthoDesign3D*, voici trois problèmes qui ont été identifiés et que l'on vous demande de résoudre en utilisant les patrons de conception :

1. Afin de représenter dans un catalogue les catégories d'orthèses permettant de traiter différents types de fractures, les orthèses doivent être représentées comme des assemblages de plusieurs composantes. Chaque composante peut elle-même comprendre plusieurs pièces ou composantes. Les pièces ou composantes peuvent être utilisées dans plusieurs orthèses. Les pièces, les composantes et les orthèses doivent pourvoir être manipulées de façon identique, dans le système OrthoDesign3D, et ce quelle que soit leur complexité.
2. Afin de définir une procédure de validation complète des paramètres de définition d'une orthèse (requis #9), certaines étapes de validation doivent être effectuées au niveau de l'orthèse elle-même, alors que d'autres étapes de validation doivent être effectuées par les composantes ou les pièces de l'orthèse. On veut cependant s'assurer que toutes les étapes

de validation soient effectuées, quel que soit le type de l'orthèse et des pièces qui la composent.

3. Lors de la manipulation interactive des nuages de points, de leur fusion ou de la définition des paramètres de l'orthèse (requis #3 à #8), il doit toujours être possible pour l'usager d'annuler une action effectuée ou de réappliquer une opération annulée (requis #10).

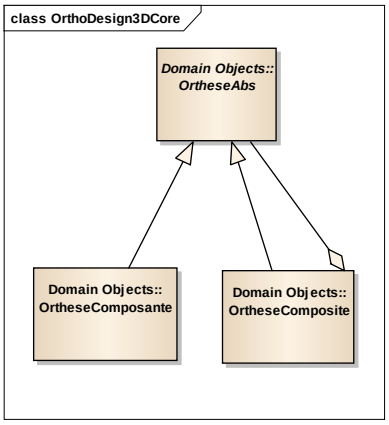
Le premier problème peut être résolu en appliquant le patron **Composite**. Le second problème peut être résolu en appliquant le patron **Template Method**. Le troisième problème peut être résolu en appliquant le patron **Commande**.

Pour chacun des trois problèmes de conception, on vous demande :

a) D'expliquer en vos propres mots l'intention du patron suggéré (1 point pour chaque problème).
b) D'appliquer le patron suggéré en fournissant un diagramme de classes annoté, où les rôles identifiés dans le patron sont explicitement associés aux classes appropriées du système *OrthoDesign3D* (3 points pour chaque problème).
c) De discuter des avantages, des inconvénients et des conséquences liés à l'utilisation du patron suggéré pour résoudre le problème de conception (2 points pour chaque problème).

Patron Composite

a) Intention : Traiter les objets individuels et les objets multiples, composés récursivement, de façon uniforme.
b) Diagramme :



c) Avantage, 1 parmi :
    1. Uniformité: traite les composants uniformément sans égard à leur complexité.
    2. Extensibilité: les nouvelles sous-classes de Component fonctionnent partout où les anciennes fonctionnent.

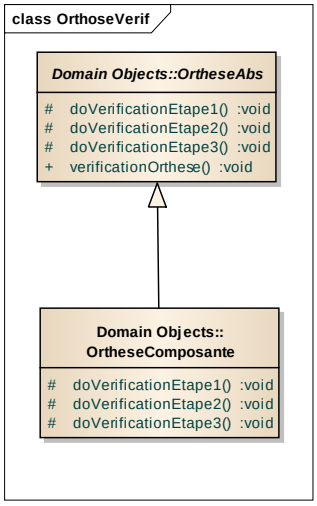
Inconvénient :
    1) Coût: peut nécessiter un grand nombre d'objets

Décision d'implantation, 1 parmi :
    1) Les Components connaissent-ils leur parent ?
    2) L'interface est-elle uniforme entre les Leaves et Composites ?
    3) On ne doit pas allouer d'espace de stockage pour les enfants dans la classe de base Component
    4) Qui est responsable de détruire les enfants?

Patron Template Method

a) Intention : Définir le squelette d'un algorithme dans une opération, et laisser les sous-classes définir certaines étapes.

b) Diagramme :



c) Avantage, 1 parmi :

1) mène à une inversion de contrôle («Principe Hollywood: ne nous appelez pas, nous vous appellerons »)
2) favorise la réutilisation de code
3) permet d'imposer des règles de surcharge

Inconvénient

1) il faut sous-classer pour spécialiser le comportement.

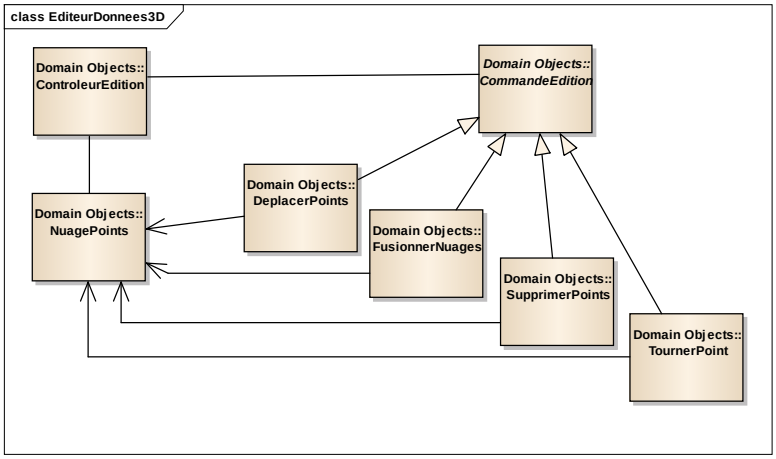
Décision d'implantation,

1) méthode Template virtuelle ou non-virtuelle
2) peu ou beaucoup d'opérations primitives
3) convention de noms (préfix do- ou faire-)

Patron Commande

a) Intention : Encapsuler une requête dans un objet de façon à permettre de supporter facilement plusieurs types de requêtes, de définir des queues de requêtes et de permettre des opérations « annuler ».

b) Diagramme :



c) Avantage, 1 parmi :

a. Découple l'objet qui invoque la requête de celui qui sait comment la satisfaire.
b. Les commandes sont encapsulées dans des objets qui peuvent être manipulées comme tout objet. L'utilisation d'objets amène également plus de flexibilité.
c. On peut facilement créer de nouvelles commandes.
d. Les commandes peuvent être assemblées en des commandes composites si nécessaires. Le patron Command peut, en effet, être combiné au patron Composite pour représenter des commandes qui sont un ensemble d'autres commandes.

Inconvénient, 1 parmi :

a) Nombre et coût des objets

Décision d'implantation, 1 parmi :

a) Quel est le niveau d'intelligence d'une commande ?
b) Veut-on supporter les opérations Undo/Redo ?
c) Comment conserver l'état pour les opérations Undo/Redo ?

Question 4 – Gestion de ressources (2 points)

Dans l'extrait de code suivant, la fonction `utiliserSocket( void )` ouvre un socket de communication, l'utilise, et le referme :

```
 1. #include <sys/socket.h>
 2. int utiliserSocket( void )
 3. {
 4.     int retour = 0;
 5.     // appel de fonction pour ouvrir un socket
 6.     int socket_id = socket( PF_LOCAL, SOCK_STREAM, 0);
 7.     if( socket_id != -1 )
 8.     {
 9.        // utiliser le socket
10.        ...
11.        // fermer le socket
12.        shutdown( socket_id, 2);
13.     }
14.     else
15.        retour = errno;
16.     return retour;
17. };
```

Dans le cas spécifique de la fonction `utiliserSocket( void )`, décrivez en vos propres mots le problème qui se produira si une exception est lancée durant l'utilisation du socket (ligne 10) et expliquez de quelle façon le principe « une acquisition est une initialisation » permet de corriger ce problème (2 points).

Si une exception est lancée à la ligne 10, le socket ne sera jamais fermé est sera probablement inutilisable. En initialisant un objet spécifique qui encapsule l'acquisition du socket, quelle que soit la raison pour laquelle la fonction se termine, le socket sera toujours libéré.

BONUS: Fournissez une nouvelle version de la fonction `utiliserSocket( void )` en appliquant le principe « une acquisition est une initialisation » afin de régler le problème (2 points).