Université de Sherbrooke, Département d'informatique

IGL301 : Spécification et vérification des exigences
Professeur : Marc Frappier, Mercredi 16 mars 2005, 8h30 à 11h20

Documentation permise. La correction est, entre autres, basée sur le fait que chacune de vos réponses soit *claire*, c'est-à-dire lisible et compréhensible pour le lecteur; *précise*, c'est-à-dire exacte et sans erreur; *concise*, c'est-à-dire qu'il n'y ait pas d'élément superflu; *complète*, c'est-à-dire que tous les éléments requis sont présents.

Pondération :

Question	1	2	3	4	5	6	7	8	9	total
Pondération	5	5	5	35	5	15	5	10	85	170

Nom : _____ Prénom : _____

Signature : _____ Matricule : _____

1) (5 pts) Expliquez pourquoi on distingue l'activité d'analyse de l'activité de spécification dans le processus d'ingénierie des exigences. Expliquez les dangers de confondre analyse et spécification.

Réponse : L'analyse permet de décrire le problème à résoudre. La spécification permet de décrire une solution au problème décrit par l'analyse. Il peut y avoir plusieurs solutions au problème; la spécification en choisit une. Si on ne distingue pas le problème de la solution, on risque de choisir une solution qui ne correspond pas au problème, ou bien d'éliminer inutilement des solutions potentielles.

---

2) (5 pts) Y a-t-il une distinction entre le domaine du problème et une exigence? Justifiez votre réponse.

Réponse : Oui. L'exigence est une propriété ou un service requis par l'utilisateur. Le domaine du problème constitue l'univers

3) (5 pts) Quelle différence y a-t-il entre une exigence, un service et une propriété?

Réponse : Une exigence est soit un service, soit une propriété.

4) (35 pts) Soit les activités suivantes du processus d'ingénierie des exigences : élicitation, analyse, spécification, conception interface personne-machine, validation. Soit les diagrammes suivants de la notation UML : diagramme de classe, diagramme de cas d'utilisation. Considérez les questions suivantes et justifiez chacune de vos réponses.

a) Le diagramme de cas d'utilisation peut-il être produit dans l'analyse?
Réponse : oui, car il peut décrire les interactions entre le système et le client si cela fait partie de la définition du domaine du problème.

b) Le diagramme de cas d'utilisation peut-il être produit dans la spécification?

Réponse : oui, car il peut décrire les interactions attendues entre le système et le client.
c) Le diagramme de classe est-il typiquement produit dans la spécification?

Réponse : oui, car il peut décrire les structures de données du domaine du problème.
d) Le diagramme de classe est-il typiquement produit dans l'élicitation?

Réponse : c'est peu probable. C'est plutôt dans l'activité d'analyse et de spécification. L'élicitation est une activité d'acquisition de l'information à analyser.
e) Le diagramme de classe est-il typiquement produit dans conception de l'interface personne-machine?

Réponse : c'est peu probable. La description de l'interface p-m décrit la structure de l'interface et les actions associés aux évènements permis par l'interface.
f) Le diagramme de classe est-il typiquement produit pour la description des exigences?

Réponse : c'est peu probable. La description des exigences comporte principales une description des services et des propriétés attendues.

g) Le diagramme de classe est-il typiquement produit pour la description des caractéristiques du domaine du problème?

Réponse : c'est tout à fait possible. Il sert alors de définition des données du problème.

5) (5 pts) Est-ce que la norme IEEE 830-1998 établit une distinction nette entre l'activité d'analyse et l'activité de spécification, au même titre que les cas d'étude présentés dans Bray (chapitre 15 à 18)?

Réponse : non. Un document de spécification satisfaisant IEEE 820-1998 comprend à la fois des éléments d'analyse et de spécification.

6) (15 pts) Vous démarrez une entreprise de développement de logiciel et votre premier produit est un logiciel de production de rapports d'impôt pour les particuliers dont les rapports d'impôt sont très simples (salarié avec retenues à la source et déductions simples). Le système doit être accessible sur le web.

a) Déterminez les sources pour l'élicitation.

Réponse : - formulaire d'impôt de Revenu Canada et de Revenu Québec, loi sur l'impôt, autres documents d'interprétation de la loi sur l'impôt, systèmes des compétiteurs, expert en fiscalité.

b) Identifiez la meilleure technique d'élicitation et justifiez pourquoi.

Réponse : Entrevue avec un expert de la production de rapports d'impôt. Il permettra d'identifier les meilleures techniques pour la production rapide d'un rapport et l'organisation de l'information sur le site web.

c) Donnez un exemple d'exigence pour chaque catégorie suivante :

- fonctionnelle (ordinaire, ie de comportement)
  Réponse : le système doit permettre de gérer les contributions à un REER.
- de performance
  Réponse : le système doit permettre de traiter jusqu'à 1000 utilisateurs en même temps.
- d'utilisabilité
  Réponse : le système doit pouvoir être utilisé par un citoyen ne maîtrisant pas la loi de l'impôt sur le revenu et qui dispose seulement de ses relevés.
- de fiabilité :
  Réponse : le système doit satisfaire la loi sur l'impôt sur le revenu du Québec et du Canada.
- de conception interne
  Réponse : le système doit utiliser la technologie Tomcat.

7) (5 pts) La méthode d'analyse structurée oblige-t-elle le concepteur à faire des choix de conception interne particuliers (par exemple, à utiliser telle méthode de conception interne plutôt qu'une autre, ou à choisir une implémentation plutôt qu'une autre)?

Réponse : non. Elle décrit les fonctions attendues du système. La seule restriction, c'est que l'implémentation respecte la spécification (entrée, sortie, relation entre les deux déterminée par le pseudo-code).

8) (10 pts) Comparer les notions de cas d'utilisation et de diagramme de flux de données.

Réponse : Un cas d'utilisation décrit une séquence d'interactions entre un utilisateur (ou l'environnement) et le système. Les cas d'utilisation peuvent être reliés par des associations comme *extends* ou *includes*. Un diagramme de flux de données décrit les fonctions offertes par le système à l'environnement ainsi que les interactions avec la base de données. Il permet de décomposer le fonctions du système de manière hiérarchique. Au plus bas niveau d'un DFD, on retrouve des fonctions qui correspondent à des évènements assez abstraits d'un cas d'utilisation. On omet en général certains évènement très concrets d'un DFD (sélection d'éléments dans une liste, click de bouton, etc). Le DFD se concentre sur la fonction globale d'une interface graphique qu'un utilisateur peut invoquer. Il ne décrit pas des séquences d'évènements comme peut le faire un cas d'utilisation. Chaque fonction de base d'un DFD est décrite par un pseudo-code dans le dictionnaire de données qui donne une description complète du traitement effectué par une fonction en réponse à un évènement, alors qu'une description complète de la réponse d'un système à un évènement est répartie sur l'ensemble des cas d'utilisation utilisant cet évènement. Les relations *extends* et *includes* n'ont pas d'équivalent direct en DFD. Les préconditions et postconditions d'un cas d'utilisation sont décrites de manière équivalente dans le pseudo-code des fonction concernées.

9) (85 pts) Utilisez la méthode d'analyse structurée (Bray chapitre 4.3, 13.1, 14.3) pour faire l'ingénierie d'un système de carte magnétique pour la gestion de l'accès à des salles dans un édifice. Chaque porte dispose d'un lecteur de carte magnétique. Chaque personne dispose d'une carte pour accéder aux salles qui sont pertinentes pour ses fonctions. Chaque carte identifie une personne de manière unique. L'administrateur dispose d'un appareil pour initialiser une carte magnétique. L'administrateur du système dispose d'une console lui permettant d'indiquer à quelles salles une personne peut accéder. Le système doit permettre la gestion de profils. Un profil indique un ensemble de salles accessibles. Un profil ne peut référencer un autre profil; il référence seulement des salles. Une personne peut être associée à plusieurs profils, lui donnant ainsi accès aux salles de ces profils. L'administrateur doit pouvoir spécifier les accès d'une personne sans nécessairement utiliser un profil, c'est-à-dire en donnant simplement les salles accessibles à une personne. Pour simplifier sa tâche, l'administrateur peut créer des *salles virtuelles* ; une salle virtuelle est une salle constituée de salle physiques ou de salles virtuelles, récursivement. Par exemple, pour donner accès à toutes les salles du Département d'informatique, l'administrateurs peut créer une salle virtuelle département_d'informatique et lui associer toutes les salles physiques du département. Pour donner accès à toutes les salles de la Faculté des sciences, il peut créer la salle faculté_des_sciences, qui contient la salle département_d'informatique et les autre salles virtuelles représentant les autres département. Une salle peut faire partie de plusieurs salles virtuelles. Le système doit conserver l'historique des accès effectués pour chaque salle physique.

Fournissez les éléments suivants :

- diagramme de contexte;
- modèle conceptuel de données;
- diagrammes de flux de données (DFD) :
    - mentionnez seulement les fonctions mentionnées dans le paragraphe ci-dessus;
    - regroupez les fonctions de base de type créer-modifier-supprimer en une seule fonction de base;
- dictionnaire de données :
    - décrire les entités externes;
    - décrire tous les flux;
    - décrire le pseudo-code de la fonction qui autorise l'accès à une salle suite à la lecture de la carte magnétique d'une personne;
    - **omettez** la description des dépôts de données.

Réponse

dictionnaire de données

LecteurCarteMagnétiquePorte ::=

(* Appareil associé à une porte qui permet de lire une carte magnétique*)

SerrurePorte ::=

(* Appareil acceptant une commande d'ouverture qui déverrouille une porte durant un certain laps de temps *);

AdministrateurSystème ::=
(* Personne qui gère les accès aux salles en spécifiant les droits d'accès de chaque personne *);
VerrouPorte ::=
(* mécanisme qui permet de déverrouiller momentanément une porte *);
demandeAccès ::=
noSalle, noCarte;
profil ::=
((ajout | modification), idProfil, {noSalle}) | (suppression, noProfil);
salle ::=
((ajout | modification), noSalle, typeSalle, {sallesIncluses}) | (suppression, noSalle);
typeSalle ::=
physique | virtuelle;
sallesIncluses ::=
{noSalle};
personne ::=
((ajout | modification), noCarte, {profil}, sallesIncluses) | (suppression, noCarte);
commandeOuverturePorte ::=
(* signal envoyé au verrou d'une porte pour la déverrouiller momentanément *)
accèsPossible ::=
noCarte, {salle}
accèsAutorisé ::=
noCarte, noSalle, date
OuvrirPorte ::=
vérifier si le noCarte existe dans l'entité personne
s'il existe
calculer les accès possibles comme suit:
accèsPossible = sallesAccessibles de la personne;
accèsPossible = accèsPossible union salles des profils de la personne;
remplacer récursivement chaque salle virtuelle de accèsPossible par ses sallesContenues;
si noSalle appartient à accèsPossible
autoriser l'accès;
ajouter nosalle, noCarte, dateCourante à l'entité Accès.

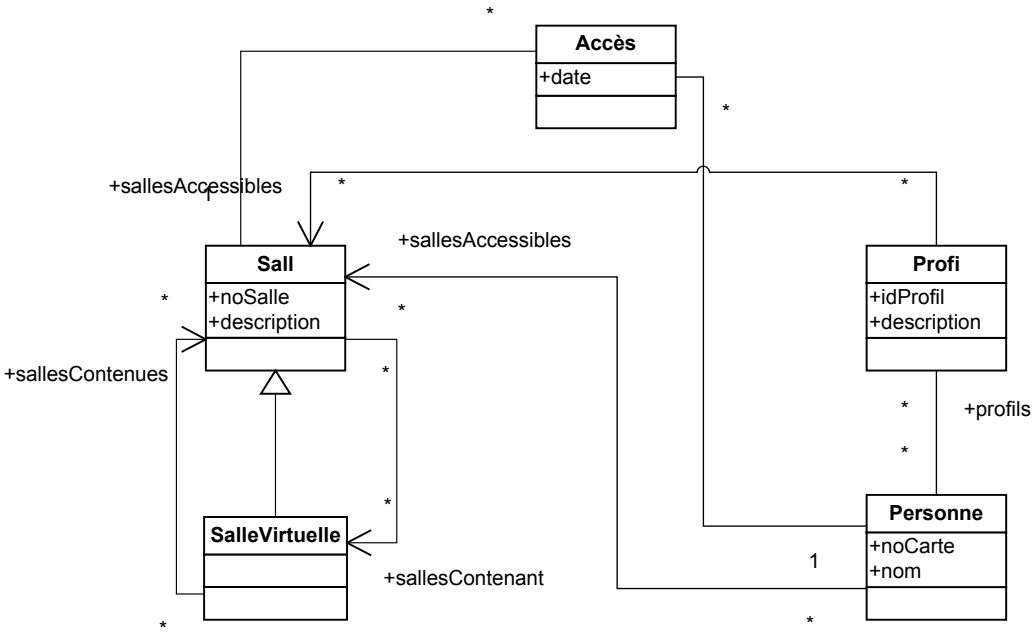
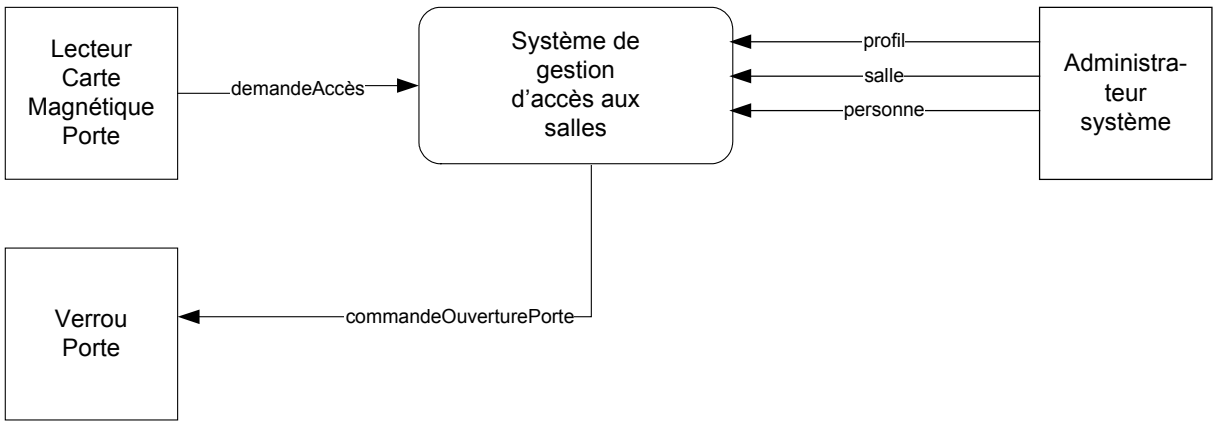
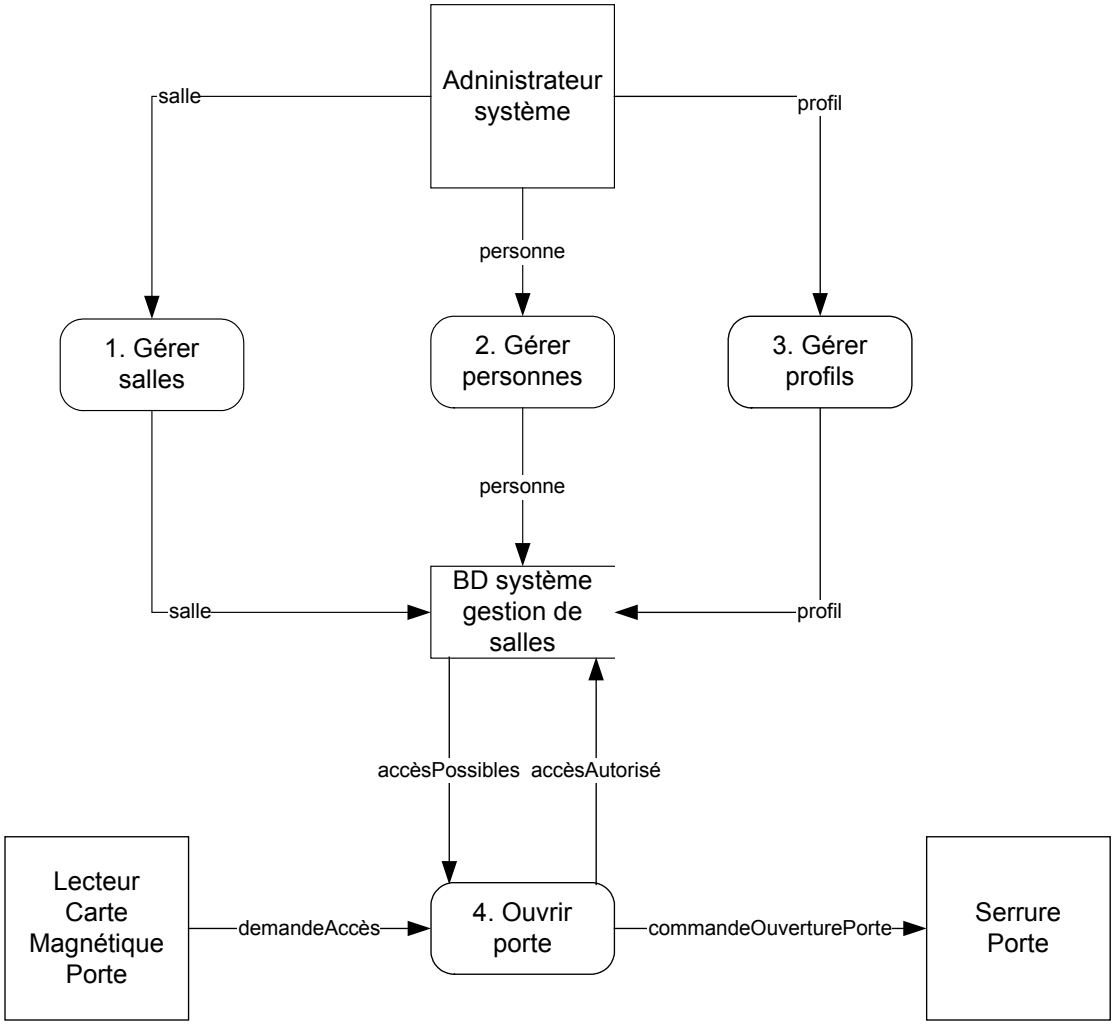


Diagramme de contexte



DFD 0


Adninistrateur système
salle
profil
personne
1. Gérer salles
2. Gérer personnes
3. Gérer profils
personne
BD système gestion de salles
salle
profil
accèsPossibles
accèsAutorisé
Lecteur Carte Magnétique Porte
demandeAccès
4. Ouvrir porte
commandeOuverturePorte
Serrure Porte