UNIVERSITÉ LIBRE DE BRUXELLES
Faculté des sciences
DÉPARTEMENT D'INFORMATIQUE

Enseignants : Joël GOOSSENS, Christian HERNALSTEEN
Cours : INFO-F-209 – Projet d'année 2

CITYLORD : SOFTWARE REQUIREMENTS DOCUMENT

GROUPE 2
Zakaria AHARRAR – Hakim BOULAHYA – David FISHEL – Cédric ORINX – Gabriel ORTEGA – Kaio LOPES

Année académique 2014 – 2015

Table des matières

Chapitre 1

Introduction

1.1 But du projet

Nous allons implémenter CityLord, un jeu de gestion de ville multi-joueur. Celui-ci se jouera entre 2 à 8 joueurs différents qui pourront acheter, vendre, ou améliorer des bâtiments, dans une même villes. Le but du jeu est le même pour chaque joueur, être le dernier propriétaire dans la ville, ou bien être celui avec le plus grand capital (argent+valeur des bâtiments) à la fin du temps imparti.

Au départ, chaque joueur commence avec une parcelle prise au hasard, et une somme modeste lui permettant d'acheter un nombre limité de propriété (1 voir 2 parcelles ou bâtiments).

Pour gagner de l'argent, des visiteurs (non-controlé par les joueurs) doivent entrer dans un de leur établissement, y rester une durée déterminée, leur donner l'argent adapté à l'établissement où il se trouve, et disparaitra ensuite de la partie. Un visiteur ne peut apparaitre qu'à des endroits prédéfini sur la carte (ex : station de métro), il suivra alors un chemin également prédéfini (ex : vers la sortie de la ville), et entrera dans un établissement qu'il croise, si celui-ci l'intéresse. Ce sera donc aux joueurs de bien positionner leurs bâtiments là où ils ont le plus de chances d'attirer un grand nombre de visiteur.

Les bâtiments ne peuvent accueillir qu'un nombre limité de visiteur en même temps, accepter une même somme par visiteur, et se débarrasser d'un visiteur qu'après une certaine durée. Un joueur peut, s'il le souhaite et pour un certain coût, amélioré son bâtiment, et alors augmenter la capacité maximale, la recette par visiteur et diminuer la durée d'occupation d'un visiteur.

Si un joueur est en faillite, à cause d'un coût d'entretien trop élevé face aux recettes, il sera alors obligé de vendre des propriétés. Il mettra alors sa propriété (parcelle+bâtiment) dans le catalogue de vente, ouvert à tous les joueurs. Il peut également s'il le souhaite, supprimer son bâtiment, ce qui réduira grandement les coût d'entretien, mais rendra les recettes nulles. (Une parcelle vide à un coût d'entretien, mais pas de revenu, ceci est pour éviter qu'un joueur achète tous les emplacements simplement pour empêcher les autres de construire).

La partie s'achève lorsqu'il ne reste qu'un seul joueur en jeu, n'ayant pas fait faillite, ou que le temps est écoulé. Lorsque l'horloge arrive à 0, le jeu est arrêté et le joueur remportant la partie est celui dont le capital cumulé (argent sur le compte + prix réel des bâtiments possédés) est le plus élevé. En cas d'égalité, le vainqueur est choisi en comptant la notoriété des bâtiments, puis le nombre de bâtiments (celui qui en a le plus gagne), et enfin le nombre de visiteurs total lors de la partie.

1.2 Glossaire

— **Système :** Ensemble d'éléments interagissant entre eux selon certains principes ou règles. Dans ce cas, il s'agit du programme.
— **Serveur :** Ordinateur dédié à l'administration d'un réseau informatique. Il gère l'accès aux ressources et aux périphériques et les connexions des différents utilisateurs.
— **Résau Informatique :** Ensemble des moyens matériels et logiciels mis en œuvre pour assurer les communications entre ordinateurs.
— **Pseudo :** Nom d'emprunt qu'une personne porte pour exercer une activité sous un autre nom que celui de son identité officielle.
— **Carte :** Carte prédéfinie, représentant une ville.
— **Catalogue :** Liste reprenant toutes les propriétés mises en vente par les joueurs ou par la villes.
— **Améliorer :** Action d'améliorer un bâtiment et ces statistique, gain, durée de visite, etc.
— **Débug :** Action de chercher, trouver et retirer un bug d'un programme informatique.
— **Log In :** se connecter.
— **Log Out :** se déconnecter.
— **Join :** Action de rejoindre une partie.
— **Build :** Bâtiment qui permettra de faire tourner le jeu.

1.3 Historique du document

Date de modification	Version	Modification apportée	Auteur
08/04/2015	3.8	Diagramme : Ajout de quelques précisions.	Hakim
08/04/2015	3.7	Diagramme de classe : Correction des compositions et ajout des généralisations de thread.	Hakim
08/04/2015	3.6	Diagramme de composant : Version final du diagramme de composant.	Gabriel
08/04/2015	3.5	Mise à jour du Use Case Sabotage.	David
08/04/2015	3.4	Diagramme d'activité : Version final du diagramme d'activité.	Hakim
08/04/2015	3.3	Commentaire du diagramme de classe	Cédric
04/04/2015	3.2	UseCase de connection et Construction en Fr, ajout de mots dans l'index	David
26/03/2015	3.1	Index Latex et non plus "manuel", changement de mots anglais en français	David
23/03/2015	3.0	Insertion du UseCase Sabotage.	David
20/03/2015	2.10	Modification des figures des UseCases [Interface de connexion] & [Construire-Améliorer-Détruire].	David
19/03/2015	2.9	Correction de fautes d'orthographes. Ajout de mots dans le glossaire.	Cédric
15/03/2015	2.8	Modification de la description textuelle du UseCase [Construire-Améliorer-Détruire].	David
15/03/2015	2.7	Diagramme de classe : ajoute de UpdateSystem, RequestSystem et ClientManager + liaison entre packages.	Hakim
13/03/2015	2.7	Mise à jour du diagramme de classe - mise en avant des packages.	Hakim
12/03/2015	2.6	Correction de fautes d'orthographes. Remplacement des mots de l'index en anglais.	Cédric

Date de modification	Version	Modification apportée	Auteur
07/03/2015	2.5	Repositionnement des figures avec leur UseCase.	David
20/02/2015	2.4	Modification du use case "Login" et modification des exigences non fonctionnelles.	Zakaria
19/02/2015	2.3	Modification du glossaire, but du jeu, Historique (tableau), use cases, ajustement, orthographe, enrichissement des pré et post condition.	Zakaria
16/02/2015	2.2	Modification du Use Case de "Log In", plus de utilisateur hôte mais un serveur & modification des besoins non-fonctionnels	David
13/02/2015	2.1	Modification du glossaire, enrichissement des pré et post condition.	David - Cédric
06/02/2015	2.0	Modification du But du projet, ajout au glossaire & index +précondition "Construire".	David
15/12/2015	1.8	Mise à jour des exigences de domaine.	Cédric
14/12/2015	1.7	Ajout des descriptions textuelles des use case (Premiers achats,Construire-Améliorer-Détruire,Achats entre joueurs).	David
12/12/2015	1.6	Rajout et correction des descriptions textuelles des use case, ajout des éxigeances de domaine.	Hakim
11/12/2015	1.5	Description de Use Case interface de connexion.	Kaio
11/12/2015	1.4	Exigence fonctionnelles (Besoin de l'utilisateur) et ajout dans le glossaire et l'index des termes.	David
11/12/2015	1.3	Ajout du diagramme de classe.	Équipe
10/12/2015	1.2	Ajout des premières *use case*	Zakaria
10/12/2015	1.1	Première version. Contient les points 1.1, 1.2, 1.3, 2 et 2.1 (partiellement).	Zakaria
09/12/2015	1.0	Création du document.	Hakim

Chapitre 2

Besoins de l'utilisateur

2.1 Exigences fonctionnelles

2.1.1 Interface de connexion

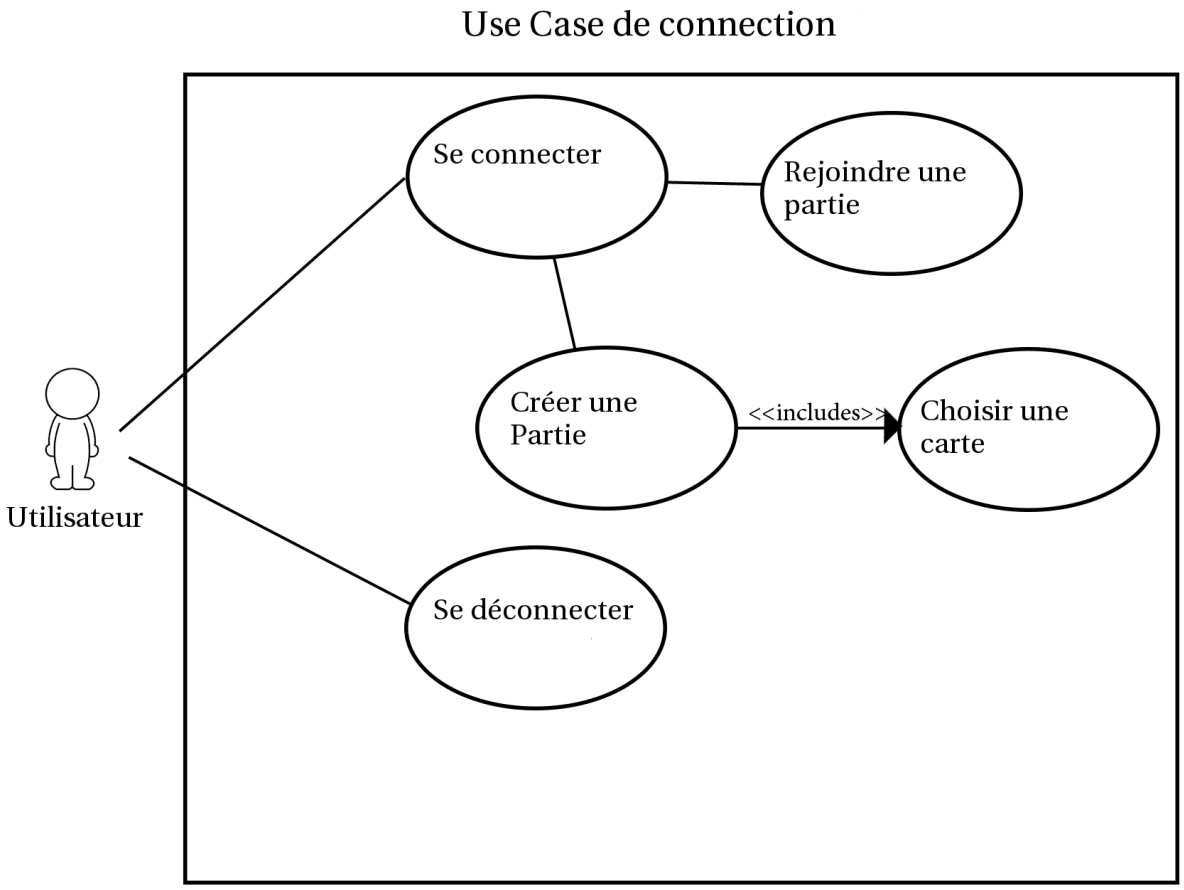

FIGURE 2.1 – Use Case Connection, Déconnection, Rejoindre Partie

2.1.1.1 Connection

Préconditions
- Le jeu doit être ouvert.
- L'utilisateur ne doit pas être connecté.

Postconditions
- L'utilisateur est connecté.

Cas général Une fois le jeu démarrer, l'utilisateur peut se connecté insérant un nom d'utilisateur existant et le mot de passe associé [Exception : mauvais mot de passe ou pseudo non-existant]. Une fois toutes les informations entré, l'utilisateur appuie sur le bouton « *Connect* », et l'utilisateur est connecté. Un utilisateur non-enregistré peut également créer un nouveau pseudo unique et lui associé mot de passe en appuyant sur le bouton « *Create* »[Exception : pseudo déjà utilisé]. Une fenêtre va ensuite s'ouvrir et lui demandé d'entrer les informations nécessaires [Exception : aucun pseudo et/ou mot de passe entré]. Ceci ajoutera sur le *server* un nouvel utilisateur avec comme pseudo et mot de passe associés ceux entrés dans les zones appropriées, et connectera l'utilisateur automatiquement.

Exceptions
- *Mauvais mot de passe ou pseudo non-existant :* Après la pression du bouton « *Connect* »,sur la même vue, une remarque apparait, en une couleur fortement perceptible, et indique à l'utilisateur que les informations entrées sont incorrectes. La fenêtre de connexion reste ouverte et demande à l'utilisateur de réentrer son pseudo et son mot de passe.
- *Pseudo déja utilisé :* Après la pression du bouton « *Create* », sur la même vue, une remarque apparait, en une couleur fortement perceptible, et indique à l'utilisateur que le pseudo demandé est déjà utilisé. La fenêtre de création reste ouverte et demande à l'utilisateur de réentrer un nouveau pseudo.
- *Aucun pseudo et/ou mot de passe entré :* Dans les deux cas - de connexion ou de création - si l'utilisateur manque d'entrer une information, après du bouton associé à la fenêtre en premier plan, sur la même vue, une remarque apparait, en une couleur fortement perceptible, et indique à l'utilisateur l'information manquante. La fenêtre reste ouvert et demande à l'utilisateur d'entrer cette information pour continuer la connexion ou la création.

2.1.1.2 Déconnection

Préconditions
- L'utilisateur doit être connecté.

Postconditions
- L'utilisateur est déconnecté.
- La fenêtre du jeu est fermé.

Cas général Un joueur peut se déconnecté à n'importe quel moment. Si il est en partie, il la quitte automatiquement. L'utilisateur est ensuite déconnecté et la fenêtre du jeu se ferme.

Exceptions Néant.

2.1.1.3 Rejoindre Partie

Préconditions
— L'utilisateur est connecté.

Postconditions
— L'utilisateur est dans une partie.

Cas général Un utilisateur client peut rejoindre une partie en appuyant sur le bouton « *Join Game* »du menu principal. Il est ensuite envoyé dans l'écran de sélection du jeu, ou il lui est possible de voir toutes les parties lancées sur le même *serveur* et visualiser les différentes informations sur chacune d'elles, comme le temps passé depuis le lancement de la partie ou le status de celle-ci. [Exception : la partie est complète] Après avoir rejoins une partie, si celle-ci n'est pas commencé le joueur est placé dans l'accueil où il peut voir tous les joueurs en attente - du début de la partie -, et la carte choisie. Il lui est également possible de chatter avec les autres joueurs. [Exception : Hôte ferme la partie] Dans le cas où la partie est déjà commencé le joueur reçoit de l'argent.

Exceptions
— *La partie est complète :* Une fenêtre s'ouvre indiquant a l'utilisateur que la partie est complète et donc injoignable. Une fois cette fenêtre fermée l'utilisateur est ramené au menu de sélection.
— *Hôte ferme la partie :* L'utilisateur client est ramené au menu de sélection [pour plus d'informations, voir Host Game].

2.1.1.4 Créer Partie

Obligations spéciales
— Inclut Choisir Carte.

Préconditions
— L'utilisateur doit être connecté.

Postconditions
— Une partie est ajoutée sur le *serveur*.
— Au moins 2 joueurs participent.

Cas général L'utilisateur est capable de créer une partie en appuyant sur le bouton « *Create game* »dans le menu principal. Il choisit d'abords une carte et optionnellement d'autres options ; la partie est alors créée, mais n'est pas lancée tant qu'il n'y a pas au moins 2 joueurs qui participent. Quand il y a assez de joueur - minimum 2 et maximum 8 - le bouton « *Start game* »s'active. Quand il est cliqué, la partie démarre.

Exceptions Néant.

2.1.1.5 Choisir Carte

Obligations spéciales
— Est inclut dans Créer Partie.

Préconditions
— L'utilisateur doit être dans le menu de création de partie.

Postconditions
— La carte choisie est chargée.

Cas général L'utilisateur peut choisir une carte dans une liste prédéfinie de carte. Quand tous les éléments pour démarrer la partie sont réunis [pour plus d'informations, voir Host Game], la carte est chargée.

Exceptions Néant.

2.1.2 Achats libres

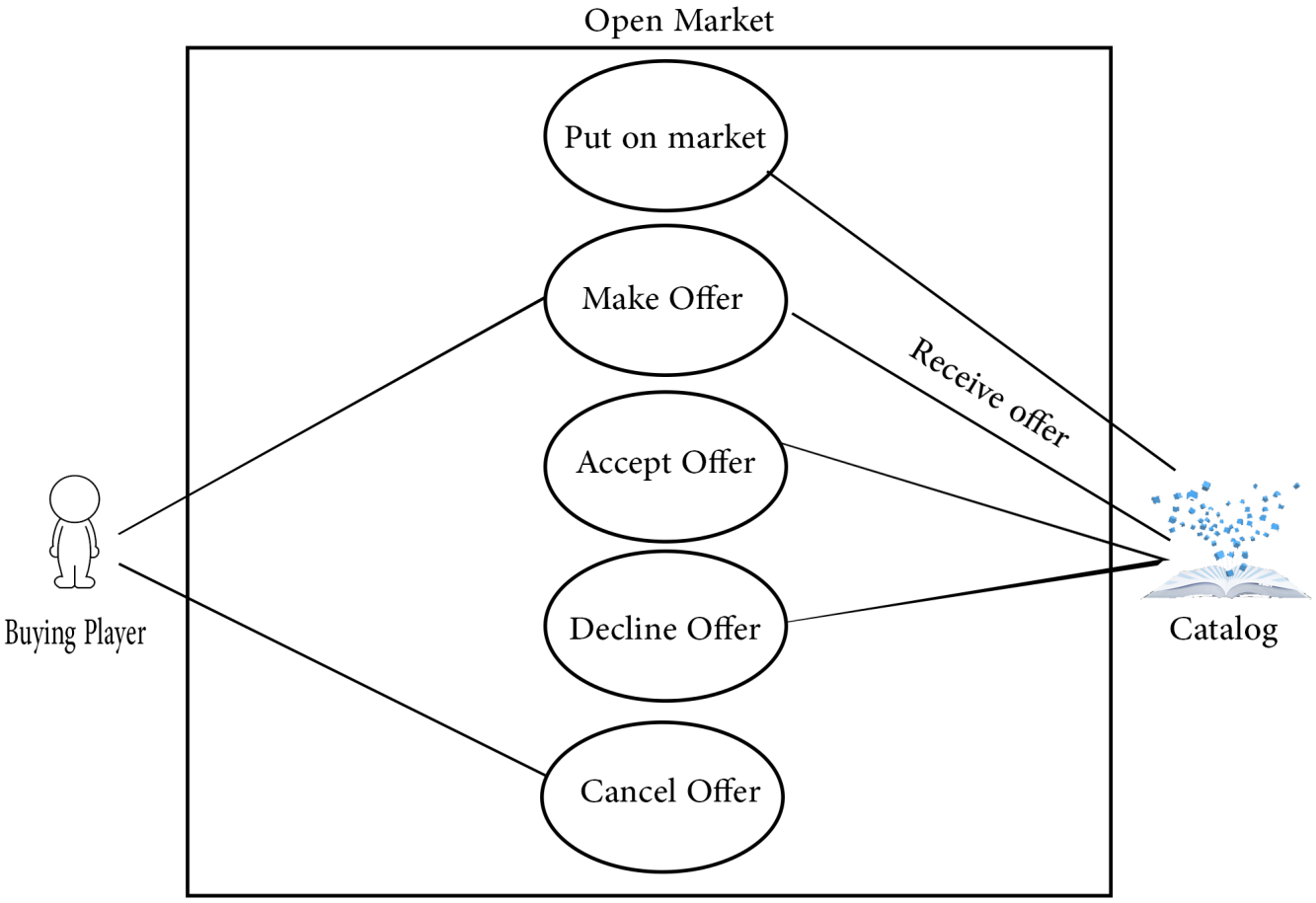

FIGURE 2.2 – Use Case Achats libres

2.1.2.1 Faire une offre

Préconditions

— La partie est en cours.
— La propriété voulue n'appartient à personne.

Postconditions

— Une offre pour une propriété est envoyée.

Cas général A tout moment de la partie, n'importe quel joueur peut sélectionner une propriété dans le *catalogue* qui n'a pas de propriétaire, et faire une offre sur celle-ci. Après avoir sélectionner la propriété, ses informations sont affichées, et le joueur peut cliquer sur acheter.[Exception : Le joueur n'a pas assez d'argent]

Exceptions

— *Le joueur n'a pas assez d'argent :* Une fenêtre s'ouvre indiquant au joueur qu'il ne possède pas assez d'argent.

2.1.2.2 Annuler une offre

Préconditions
- La partie est en cours.
- Le joueur à fait une offre qui n'as pas encore été validée.

Postconditions
- L'offre est annulée et retirée.

Cas général Dès que le joueur a mit une offre et tant que celle-ci n'a pas été acceptée, ce joueur peut annuler cette offre.

Exceptions Néant.

2.1.2.3 Accepter une offre

Préconditions
- La partie est en cours.
- Un joueur a fait une offre.

Postconditions
- Le joueur paie le prix de la propriété.
- Le joueur devient propriétaire de la propriété.

Cas général Lorsque le *catalogue* reçoit une offre sur un bâtiment, il accepte automatiquement celle-ci.[Exception : Deux joueur font la demande en même temps] Une fois l'offre acceptée, le joueur doit confirmer son achat ou l'annuler.

Exceptions
- *Deux joueur font la demande en même temps :* La première offre reçue sera prise en compte, l'autre sera annulée.

2.1.2.4 Mettre en vente

Préconditions
- La partie est en cours.
- Il existe un bâtiment sans propriétaire.

Postconditions
- Le bâtiment est en vente libre.

Cas général Dès qu'un bâtiment n'as pas de prioritaire, le *catalogue* le met en vente libre. Les bâtiments créés en début de partie seront mit dans le *catalogue*, ainsi que les bâtiments d'un joueur qui vient de perdre qui sont mit en vente libre.

Exceptions Néant.

2.1.3 Construire-Améliorer-Détruire

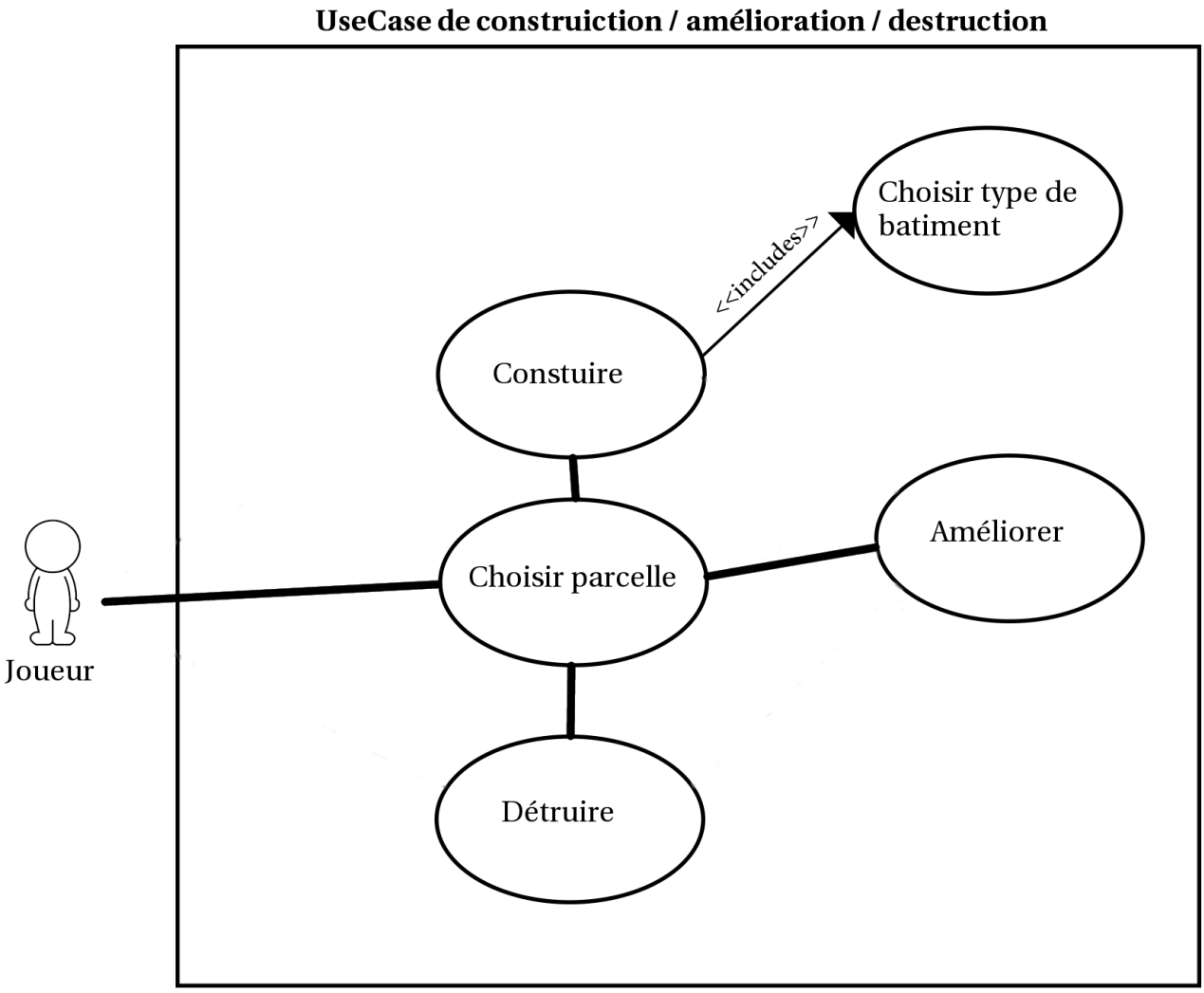

FIGURE 2.3 – Use Case Buid/Upgrade/Destroy

2.1.3.1 Choisir lieu

Préconditions

— Le joueur est dans une partie, sur l'écran de jeu.

Postconditions

— Une propriété est sélectionnée.
— Les informations sur cette propriété sont affichées.
— Les boutons <Buy>,<Build>,<Upgrade>,<Sell>,<Demolish> s'activent ou se désactivent selon la propriété sélectionnée.

Cas général Le joueur doit simplement cliquer, sur la carte, sur le terrain exigé.[Exception : Sélection invalide]

Exceptions

— *Sélection invalide :* Si l'emplacement sélectionné n'est pas un terrain/bâtiment (et donc une route ou un obstacle), la sélection n'est pas faite.

2.1.3.2 Construire

Obligations spéciales

— Inclut Choisir type de Bâtiment.

Préconditions

— La partie est en cours.
— Le joueur a sélectionné une parcelle qui lui appartient et est vide. (Le bouton <Build> est activé).

Postconditions

— Un bâtiment est construit et affiché sur la carte.
— Le joueur possède ce nouveau bâtiment.
— Le joueur a payé le prix de construction.

Cas général Lorsque un joueur veut construire un nouveau bâtiment, il lui faudra simplement sélectionner une parcelle qui lui appartient et est vide. Le bouton « <*Build*> »s'activera. Le joueur devra cliquer dessus, il entre alors en mode Construction ; Un tableau s'affichera, permettant au joueur de sélection un type de bâtiment et affichant les informations sur ce type. Il peut à tout moment quitter ce mode et annuler la construction, en cliquant sur « *Annuler* ». Pour valider l'opération, le joueur cliquera sur « <*Ok*> ».[Exception : Le joueur n'as pas assez d'argent] Il devra payer ensuite le prix indiqué.

Exceptions

— *Le joueur n'as pas assez d'argent :* Une fenêtre s'ouvre indiquant au joueur qu'il ne possède pas assez d'argent.

2.1.3.3 Améliorer

Préconditions

— La partie est en cours.
— Le joueur possède au moins un bâtiment.
— Le joueur a sélectionné une parcelle qui lui appartient et qui a un bâtiment améliorable. (Le bouton <Upgrade> est activé).

Postconditions

— Un bâtiment sélectionné est amélioré d'un niveau.
— Le joueur a dépensé de l'argent pour l'amélioration.

Cas général Lorsque un joueur veut améliorer un bâtiment, il lui faudra simplement sélectionner une parcelle qui lui appartient et qui possède un bâtiment améliorable. Le bouton « <*Upgrade*> »s'activera. Le joueur devra cliquer dessus. Un tableau s'affichera, permettant au joueur de voir les informations sur le niveau supérieur de ce bâtiment.[Exception : Bâtiment amélioré au max] Pour valider l'opération, le joueur cliquera sur « <*Ok*> ».[Exception : Le joueur n'as pas assez d'argent] Il devra payer ensuite le prix indiqué.

Exceptions

— *Le joueur n'as pas assez d'argent :* Une e s'ouvre indiquant au joueur qu'il ne possède pas assez d'argent.

— *Bâtiment amélioré au max :* Les bâtiments ont un niveau maximum d'amélioration Une fenêtre s'ouvre indiquant au joueur qu'il ne peut plus améliorer ce bâtiment.

2.1.3.4 Détruire

Préconditions

— La partie est en cours.

— Le joueur possède au moins un bâtiment.

— Le joueur a sélectionné une parcelle qui lui appartient et qui a un bâtiment. (Le bouton <Destroy> est activé).

Postconditions

— Un bâtiment est détruit et n'existe plus.

— Le propriétaire gagne une somme d'argent équivalente à la moitié de la valeur du bâtiment détruit.

Cas général Un joueur peut également détruire un de ses bâtiments. Il lui suffit de sélectionner une parcelle qui lui appartient et qui possède un bâtiment et ensuite de cliquer sur le bouton « <*Destroy*> ». Un tableau s'affichera montrant le gain de la destruction. Il peut à tout moment quitter ce mode et annuler la destruction en cliquant sur « <*Annuler*> ». Pour valider l'opération, le joueur cliquera sur « <*Ok*> ». Une fois détruit, le bâtiment rapporte la moitié de la valeur de celui-ci à son propriétaire.

2.1.3.5 Choisir type de Bâtiment

Obligations spéciales

— Est inclut dans construire.

Préconditions

— Le joueur est entré en mode Construction.

Postconditions

— Un type est sélectionné pour la construction.
— Les information lié à ce type sont affiché dans le tableau de construction.

Cas général Un *catalogue* de types prédéfinis de bâtiment(bar,magasin,night-club,...) apparaît, pour chaque type les statistiques (prix, coûts, capacité, heures,..) sont affichés. Il suffit au joueur alors de cliquer sur le type désiré.

Exceptions Néant.

2.1.4 Achats entre joueurs

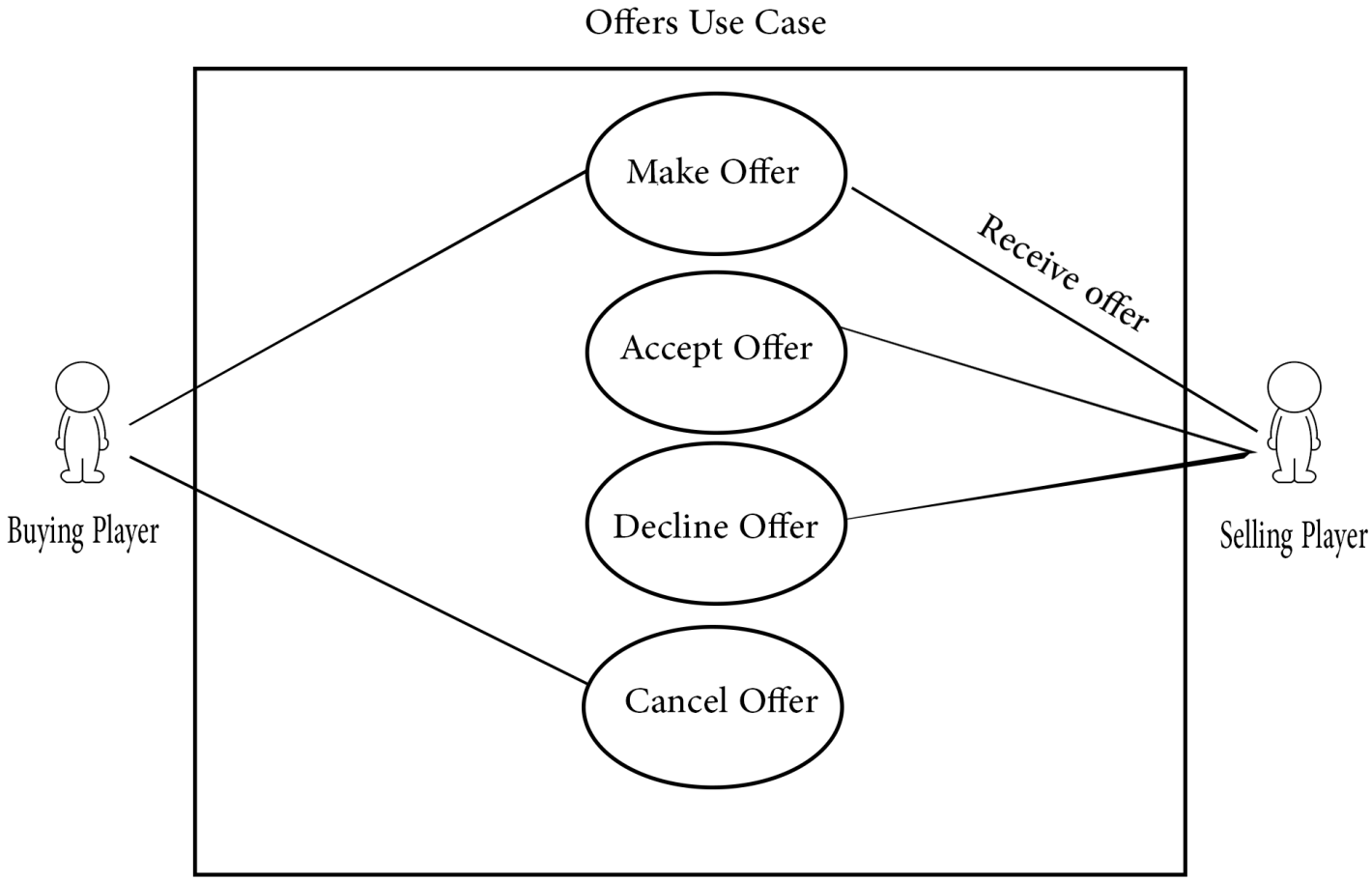

FIGURE 2.4 – Use Case Achats entre joueurs

2.1.4.1 Faire une offre

Préconditions

— La partie est en cours.
— La propriété voulue appartient à un joueur.

Postconditions

— Une offre pour une propriété est envoyé à un autre joueur.

Cas général A tout moment de la partie, un joueur peut tenter d'acheter le bâtiment d'un autre. Il lui doit simplement de faire une offre auprès de ce joueur, pour ce bâtiment.[Exception : Le joueur a fait trop d'offre] Il suffit à l'acheteur de sélectionner le bâtiment voulu et de cliquer sur « *Faire une offre* », une fenêtre s'ouvre pour que le joueur puisse donner le prix qu'il est prêt a payer et confirme son offre.[Exception : Le joueur n'a pas assez d'argent]

Exceptions
- — *Le joueur n'a pas assez d'argent :* Une fenêtre s'ouvre indiquant au joueur qu'il ne possède pas assez d'argent.
- — *Le joueur a fait trop d'offre :* Afin d'éviter tout spam, une limite d'offre sur un temps donné est instauré.

2.1.4.2 Annuler une offre

Préconditions
- — La partie est en cours.
- — Le joueur à fait une offre qui n'a pas encore été validée.

Postconditions
- — L'offre est annulée.

Cas général Dès que le joueur a mit une offre et tant que celle-ci n'a pas été acceptée, ce joueur peut annuler cette offre.

Exceptions Néant.

2.1.4.3 Accepter une offre

Préconditions
- — La partie est en cours.
- — Un joueur a reçu une offre.

Postconditions
- — L'acheteur devient propriétaire de la propriété.
- — Le vendeur reçoit une compensation monétaire et perd sa propriété.

Cas général Lorsque un joueur reçoit une offre sur un bâtiment, une fenêtre s'affiche avec les informations sur cette offre, il peut accepter celle-ci en cliquant sur « *Accepter* ».

Exceptions Néant

2.1.4.4 Refuser une offre

Préconditions
- — La partie est en cours.
- — Un joueur a reçu une offre.

Postconditions
- — L'offre est annulée.

Cas général Lorsque un joueur reçoit une offre sur un bâtiment, une fenêtre s'affiche avec les informations sur cette offre, il peut refuser celle-ci en cliquant sur « *Refuser* ».

Exceptions Néant

2.1.5 Sabotage

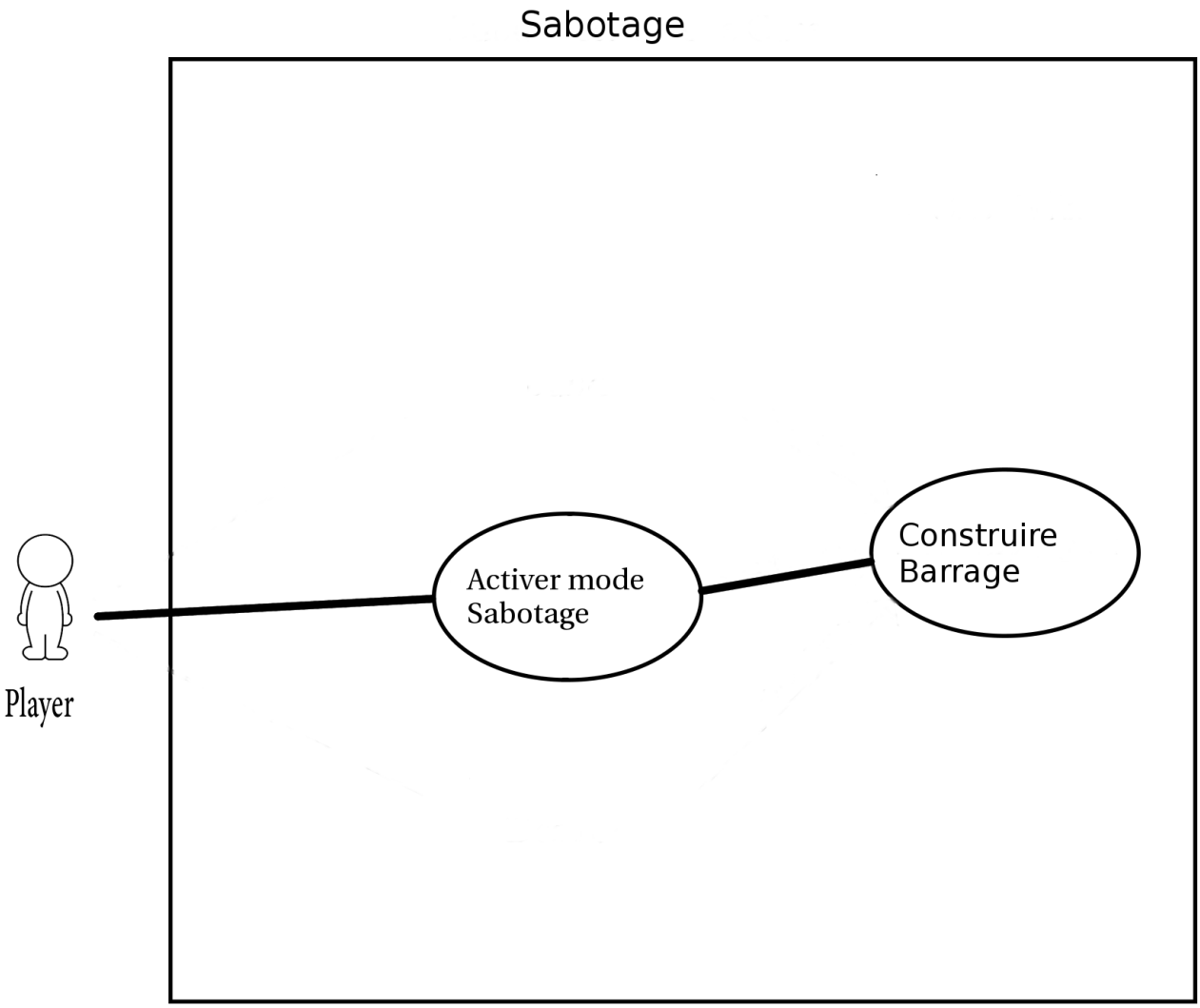

FIGURE 2.5 – Use Case Buid/Upgrade/Destroy

2.1.5.1 Activer mode sabotage

Préconditions

— Le joueur est dans une partie, sur l'écran de jeu.

Postconditions

— Le joueur est en mode sabotage.

Cas général Le joueur doit simplement cliquer, le bouton <Sabotage>, celui-ci change de couleur. Tous les autres boutons se désactivent, le joueur ne peut plus séléctionné de parcelle. Par contre, maintenant il peut construire des barrages. Pour quitter ce mode, il suffit au joueur de

cliquer à nouveau sur le bouton <Sabotage>, celui-ci reprendra sa couleur normale et le mode sabotage se désactive.

Exceptions Néant.

2.1.5.2 Construire Barrage

Préconditions

— Le joueur est en mode sabotage.

Postconditions

— Un barrage est construit et affiché sur la carte à l'emplacement séléctionné.
— Le joueur a payé le prix de construction du barrage.

Cas général Lorsque un joueur veut construire un nouveau barrage, il lui faudra simplement cliquer sur une route vide. Un barrage se construit imédiatement.[Exception : Le joueur n'as pas assez d'argent][Exception : Un barrage se trouve déjà sur cette route] Il devra payer ensuite le prix indiqué.

Exceptions

— *Le joueur n'as pas assez d'argent :* Une fenêtre s'ouvre indiquant au joueur qu'il ne possède pas assez d'argent.
— *Un barrage se trouve déjà sur cette route :* Une fenêtre s'ouvre indiquant au joueur qu'un barrage s'y trouve déjà.

2.2 Exigences non fonctionnelles

2.2.1 Service

Ergonomie :
Le *système* sera crée de telle sorte qu'il disposera d'une apparence graphique concise, afin que le joueur puisse repérer les options dont il a besoin dans le menu du jeu.

La map sera aura comme particularité que chaque joueur aura sa couleur qui sera affichée sur ses propriétés.

L'utilisation sera facilitée par l'interface graphique, grâce à des icônes, boutons et aides affichées tout au long du jeu.

Aptitude à la maintenance :
Le concept de programmation orientée objet étant suivit avec rigueur pour l'implémentation du jeu, celui-ci permettra la mise à jour ou le débuggage du jeu.

Compatibilité :
Le jeu est implémenté sur et pour un *système* d'exploitation Linux seulement.

Rétablissement :
Dans le cas d'un crash ou d'arrêt de *serveur*, il faut faire en sorte que les joueurs puissent reprendre leur partie. Le *serveur* fera donc une sauvegarde automatique à chaque debut de journée(de la partie en cours, 7h du matin). Vu qu'une seule sauvegarde est nécessaire, la nouvelle remplacera la précédente. Si la partie se termine, la sauvegarde lié à celle-ci sera supprimée.

Quand le *serveur* re-démarre, il vérifie si une partie en cours a été interrompue ; si oui, il relance la partie grâce à sa sauvegarde.

Disponibilité :
Le jeu sera disponible 24h/24 7j/7, tant que le serveur tourne. Celui ne sera arrêté que brièvement lors de possible maintenance de débuggage et mise à jour.

2.2.2 Contraintes

Afin de fonctionner correctement, et d'avoir une expérience de jeu optimale, le *système* devra être lancé sur un ordinateur possédant la configuration nécessaire. En cas d'environnement peu performant, il est possible que des ralentissements, ou qu'un manque de fluidité se fasse ressentir.

Il est également nécessaire que le *système* soit accompagné d'une connexion internet stable et rapide. Sans cela, la communication avec le *serveur* et entre les joueurs serait endommagée, et endommagerait la jouabilité de tous les joueurs.

2.3 Exigences de domaine

— Le jeu est multi-joueur, et doit donc permettre aux différents utilisateurs de communiquer entre eux.
— Une partie doit être composé d'au minimum 2 joueurs et au maximum 8 joueurs.
— Les 8 premiers joueurs qui rejoignent la partie seront les seules 8 joueurs capable de la rejoindre.
— Pour qu'un joueur puisse continuer à jouer, il doit posséder soit des terrains, des batiments ou encore avoir un capital.
— Le capitale de chaque joueur varie tout au long de la partie en fonction des frais et des gains de chaque bâtiment.
— Les gains qu'apporte chaque bâtiment seront représentés par des visiteurs, qui, en suivant un chemin, pourront rentrer dans un bâtiment qui augmentera le capital de son propriétaire.
— Si le joueur est en faillite, la partie se terminera pour ce joueur et il ne sera plus possible pour lui de continuer de jouer dans cette partie, mais il pourra toute fois la regarder.
— La partie se finie uniquement si il ne reste plus qu'un joueur propriétaire.

Chapitre 3

Besoins du système

3.1 Exigences fonctionnelles

3.2 Exigences non fonctionnelles

3.3 Design et fonctionnement du système

3.3.1 Diagramme de classe

3.3.1.1 Précision du diagramme de classe

Une map est une matrice contenant des *Case*. Une *Case* est soit une parcelle (*Field*) qui peut être achetée par un joueur où il pourra y construire des bâtiments, soit une route (*Road*) permettant aux *Visitors* de se déplacer, soit un obstacle (décoration pour les joueurs), soit une zone d'entré pour les *Visitors*.

3.3.1.2 Précision concernant les classes sans relation

Les classes ne possédant aucune relation avec d'autres classes sont simplement utilisées comme simple attribut dans plusieurs classes ou comme simple paramètre de méthode, ce qui permet de faire circuler des données plus aisément. Par exemple : `SocketMessage` ou `Location`.

3.3.2 Diagramme d'activité

3.3.2.1 Précision concernant l'arret de CityLordServer

Le serveur tourne de manière infini, il ne s'arrête donc jamais. Le seul moyen de l'arreter est de le *kill*. Dans ce cas, tous les clients connéctés seront également arretés.

3.3.2.2 Précision concernant l'arret de ClientUpdater

Le gestionnaire de mise à jour d'un client est arrêté par le *thread* principal de l'application du client, lorsque celui-ci quitte le jeu en fermant la fenêtre.

3.3.3 Diagramme de composant

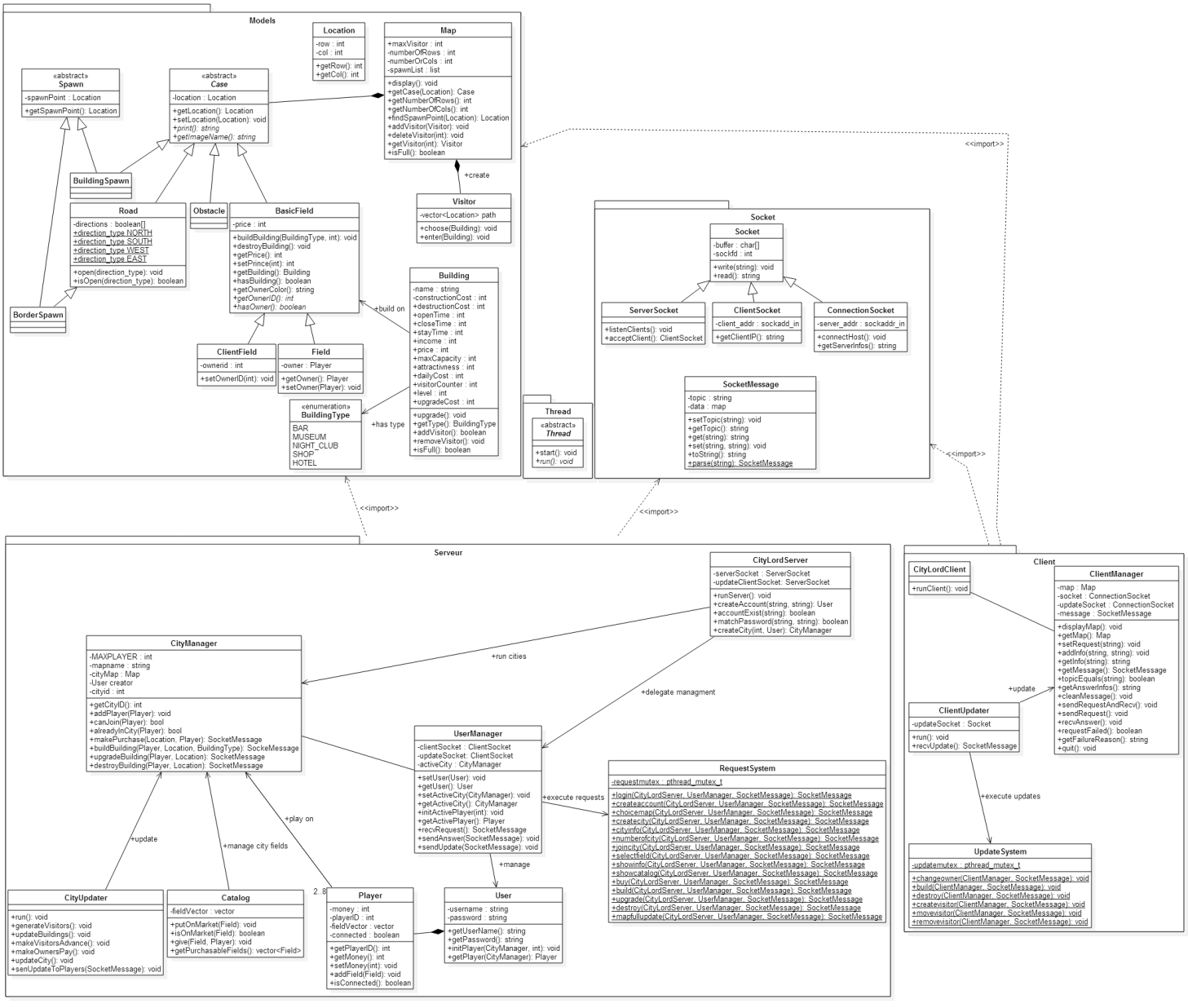

FIGURE 3.1 – Diagramme de classe : Architecture du système

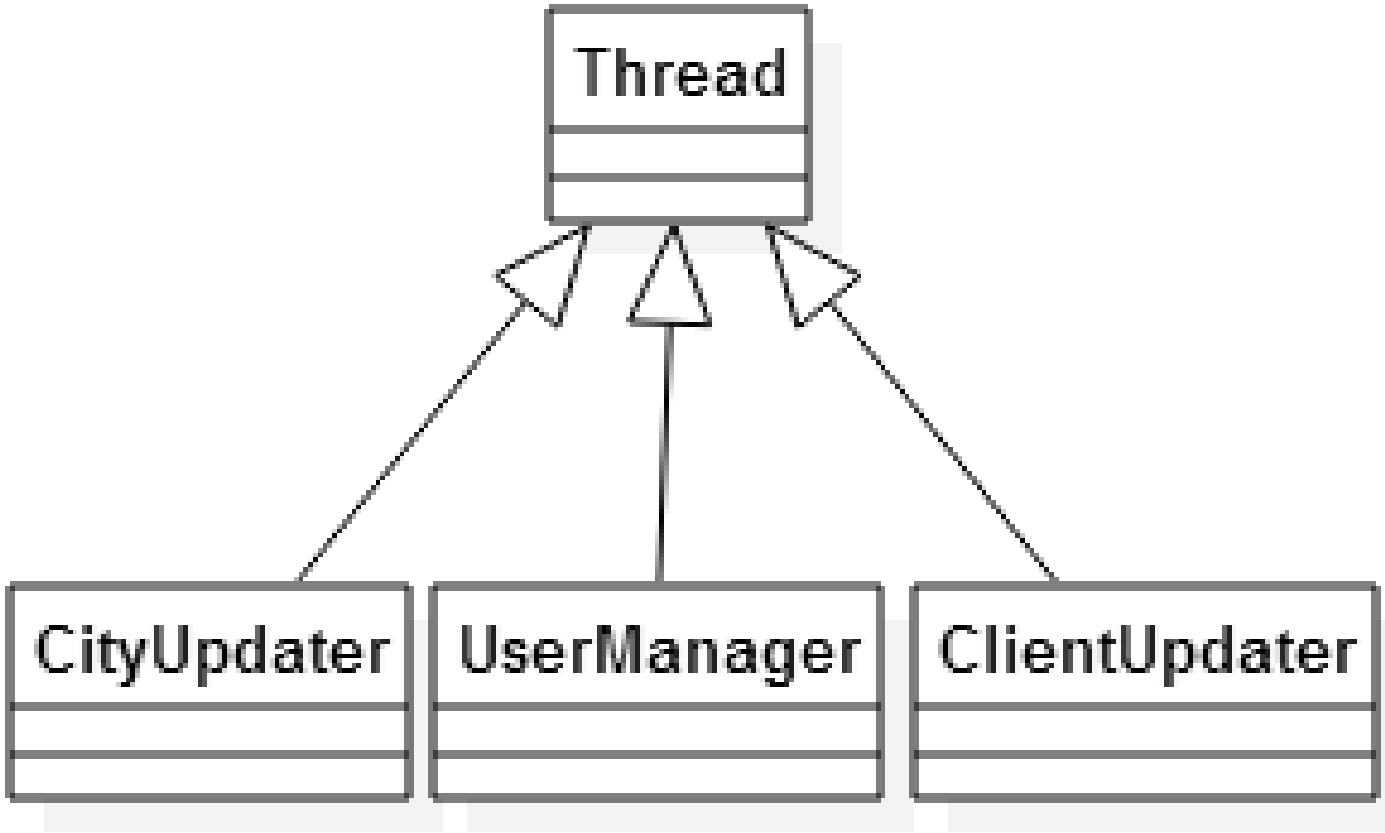

FIGURE 3.2 – Diagramme de classe : multithreading

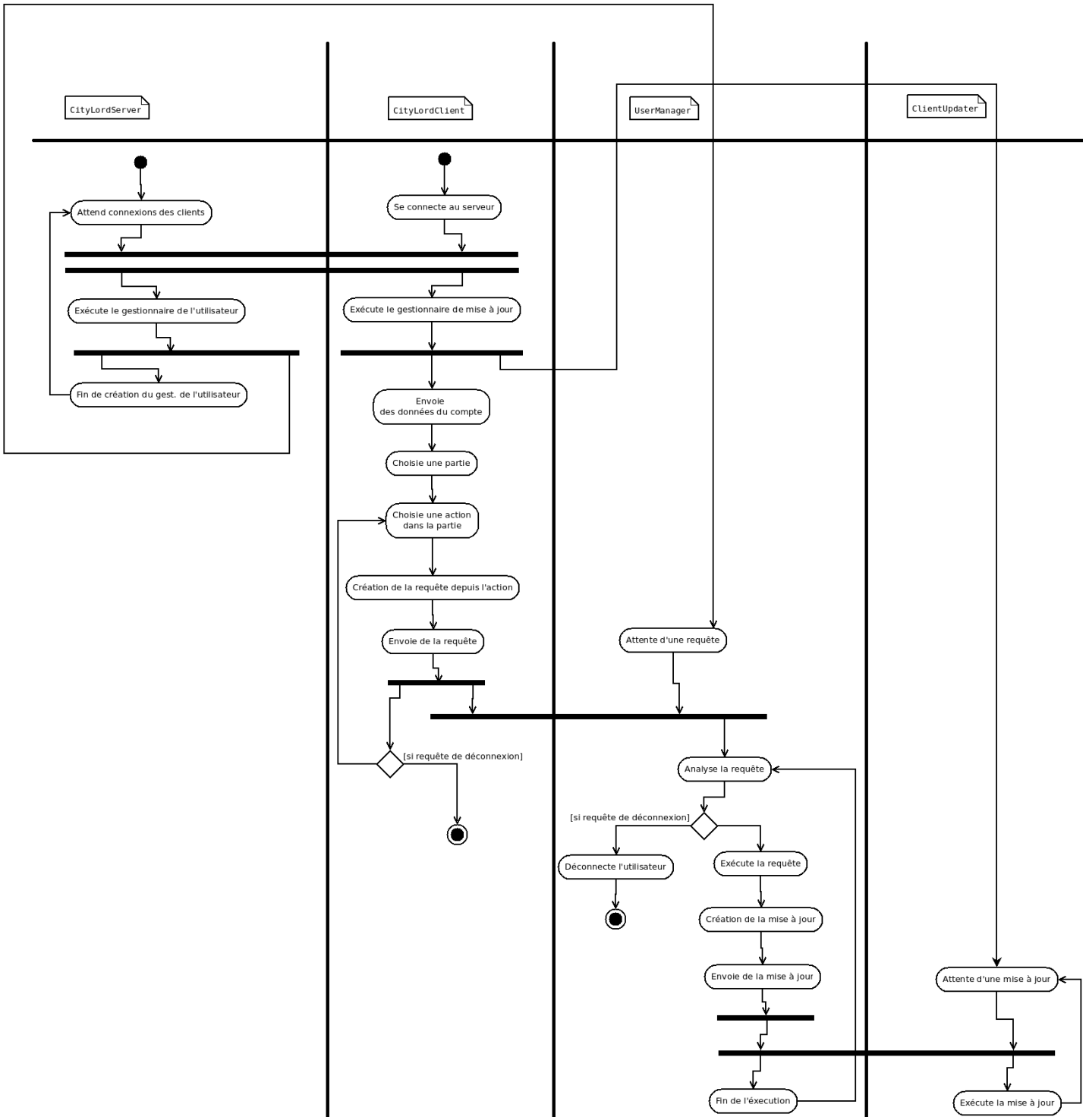

FIGURE 3.3 – Diagramme d'activité

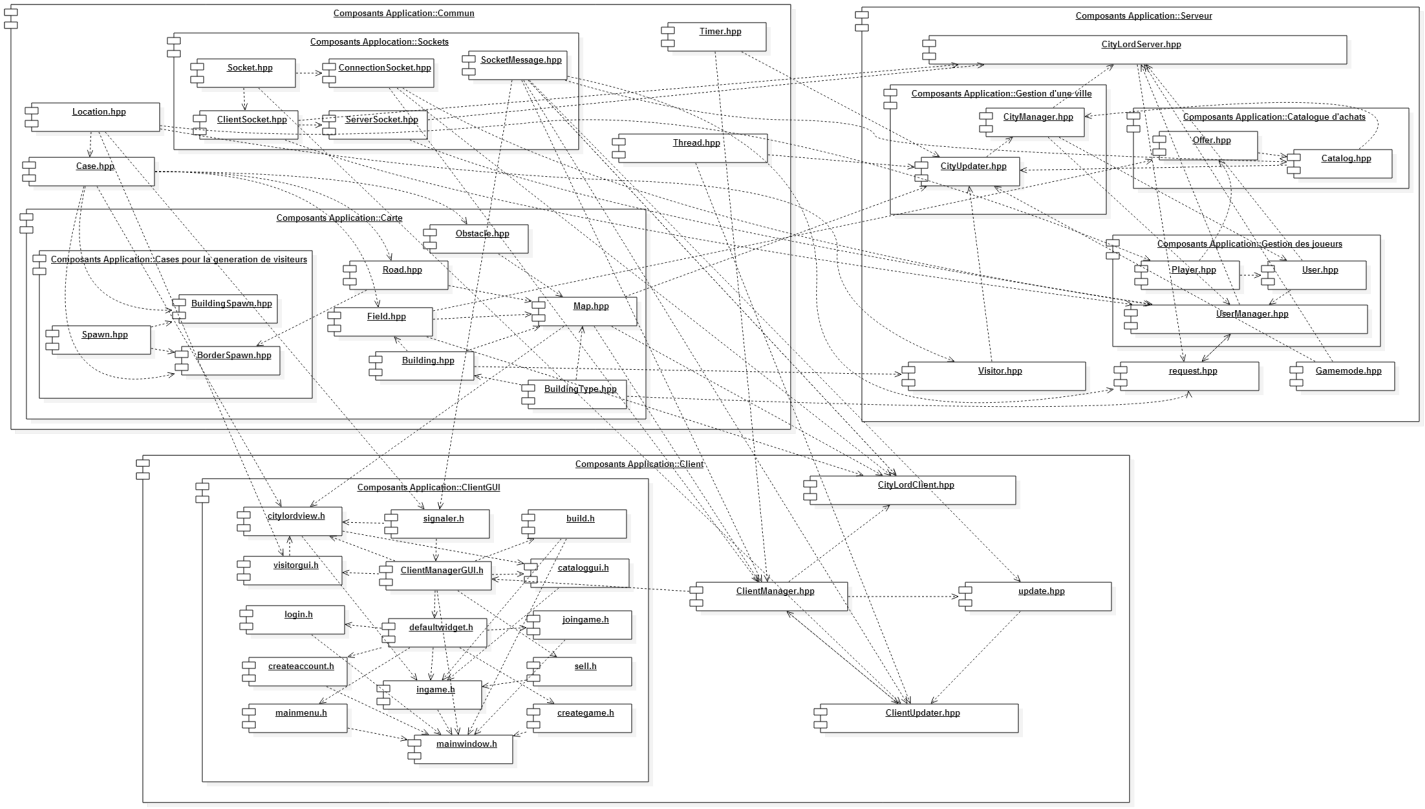

FIGURE 3.4 – Diagramme de composants

Index