GÉNIE LOGICIEL ORIENTÉ OBJET (GLO-2004)
ANALYSE ET CONCEPTION DES SYSTÈMES ORIENTÉS OBJETS (IFT-2007)

Automne 2016

Module 03 – Phase de conceptualisation / inception et modèle des cas d'utilisation

Martin.Savoie@ift.ulaval.ca

B. ing, Chargé de cours, département d'informatique et de génie logiciel

UNIVERSITÉ LAVAL

© Jonathan Gaudreault

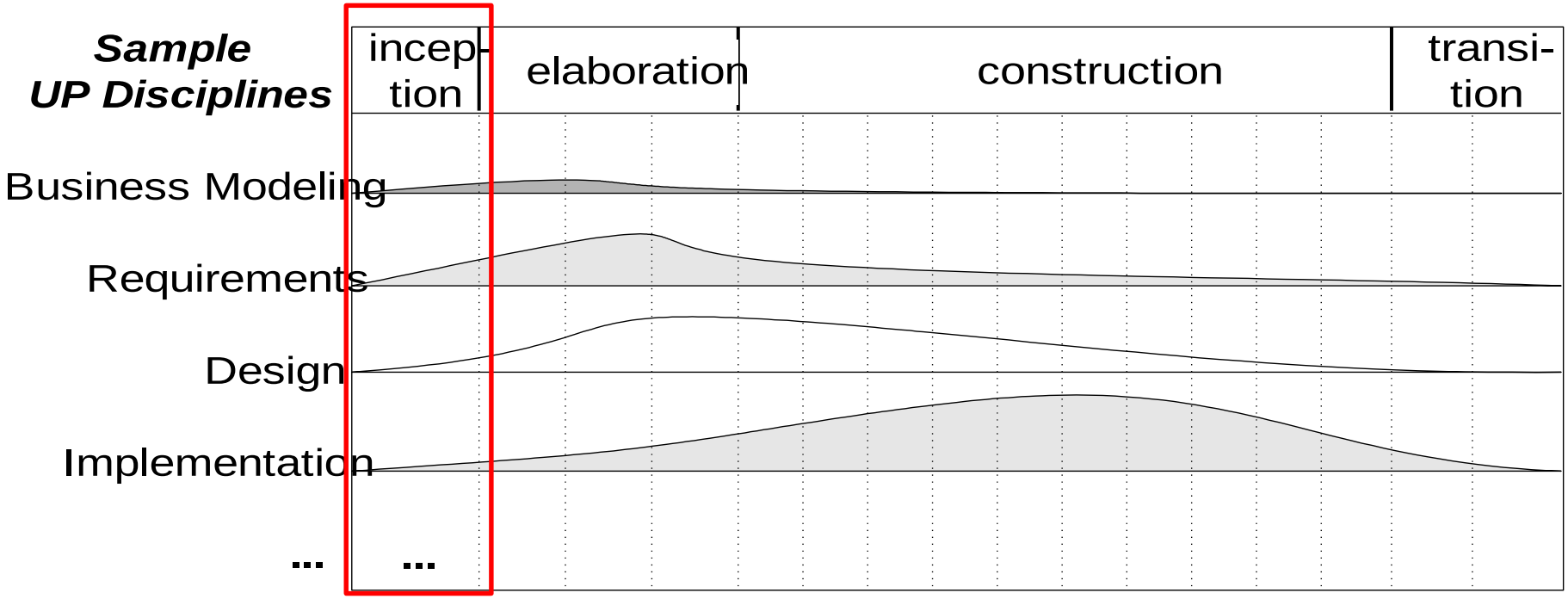
Question sur la matière du derniers cours?

© Jonathan Gaudreault

Cours d’aujourd’hui

© Jonathan Gaudreault

Phase de conceptualisation / inception



© Jonathan Gaudreault

Phase de conceptualisation / inception

- **But:**
  - Établir une définition du projet, les objectifs
  - Établir la faisabilité
  - Décider si on doit pousser plus loin (décider si on va réaliser le projet)
- **Autrement dit:**
  - Identifier le besoin
  - Rédiger un document décrivant la vision du projet / modèle d'affaires
  - Déterminer si le projet est-il réalisable
  - Peut-on acheter le système plutôt que le construire?
  - Estimer grossièrement les coûts
  - Échéancier approximatif
  - Go/no Go

© Jonathan Gaudreault

Artefacts de la phase de conceptualisation/inception

Discipline	Artifact Iteration	Incep. Il	Elab. EL.En	Const. CL.Cn	Trans. T1..T2
Business Modeling	Domain Model		s		
Requirements	Use-Case Model	*s*	r		
	Vision	*s*	r		
	Supplementary Specification	*s*	r		
	Glossary	*s*	r		
Design	Design Model SW Architecture Document Data Model		s s s	r r	
Implementation	Implementation Model (code)		s	r	r

S: start
R: refine

© Jonathan Gaudreault

	Activités (appelées *disciplines* dans le Processus Unifié)	Principaux **modèles** et artefacts générés
Analyse	Modélisation domaine d'affaires / Business modeling / Modélisation métier	**Modèle du domaine**: (1) diagramme de classe « conceptuel », (2) parfois un diagramme d'activités
	Analyse des besoins / Exigences / Requirements	(3) Énoncé de vision
		Modèle de cas d'utilisation / Use-case model : (4) diagramme des cas d'utilisation, (5) texte des cas d'utilisation, (6) diagramme de séquence système
		(7) Spécifications supplémentaires
		(8) Glossaire
	Design / Conception	**Modèle de conception / Design model** : (9) diagrammes de classes, (10) diagrammes d'interaction, (11) tout autre diagramme UML pertinent selon le contexte
	Implémentation	(12) Code
	…	

Les artefacts dont nous produisons une première version dans la phase de conceptualisation/inception

- Énoncé de vision
  - Description de haut niveau du projet, des objectifs, etc..
- Modèle des cas d'utilisation
  - Description des besoins en termes de "fonctionnalités"
- Spécifications supplémentaires
  - Autres besoins (non fonctionnels)
- Glossaire
  - Définition de la terminologie de base du domaine
- Plan sommaire du projet

© Jonathan Gaudreault

Autres choses à faire dans la phase de conceptualisation / inception

- Budget
- Plan sommaire du projet
  - Évaluer la durée des phases / itérations
  - Ressources nécessaires
  - Besoins en formation
- Plan détaillé de la prochaine itération (première itération de la phase d'élaboration)
- Principaux facteurs de risque et stratégies de gestion du risque
- Prototypes / preuve de concept (si nécessaire)

© Jonathan Gaudreault

Besoins / requirements

- **Besoins**: Conditions auxquelles un système (ou projet) doit satisfaire

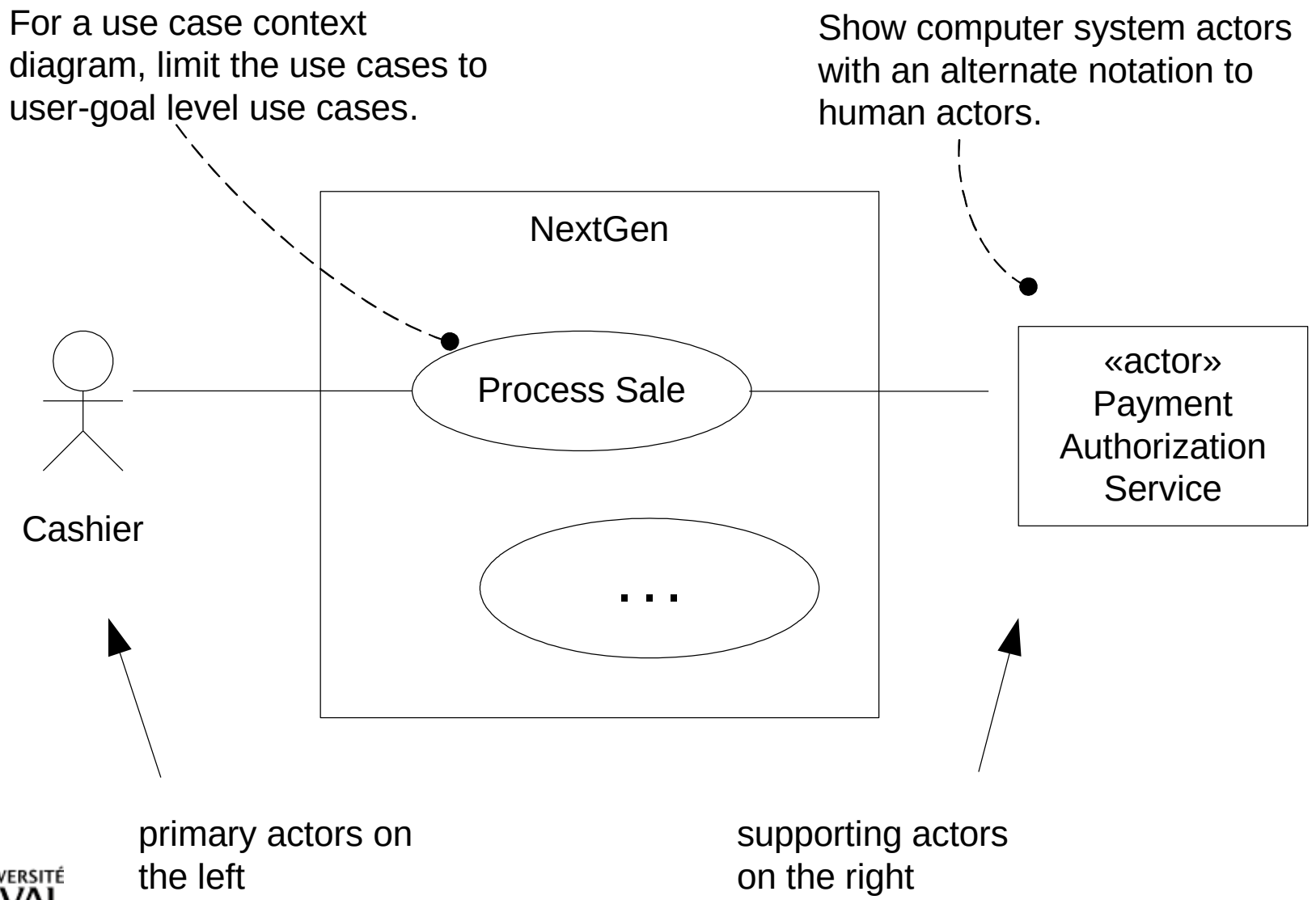
- **Besoins fonctionnels**
  - Fonctionnalités
  - Répertoriées et décrites dans le **modèle des cas d'utilisation**

- **Besoins non fonctionnels (« URPS »)**
  - **U**sability (Help, documentation, …), **R**eliability (Frequency of failure, recoverability, …), **P**erformance (Response times, availability, …), **S**upportability (Adaptability, maintainability, …)

- **Rappel**: le processus unifié suppose une notion évolutive des besoins

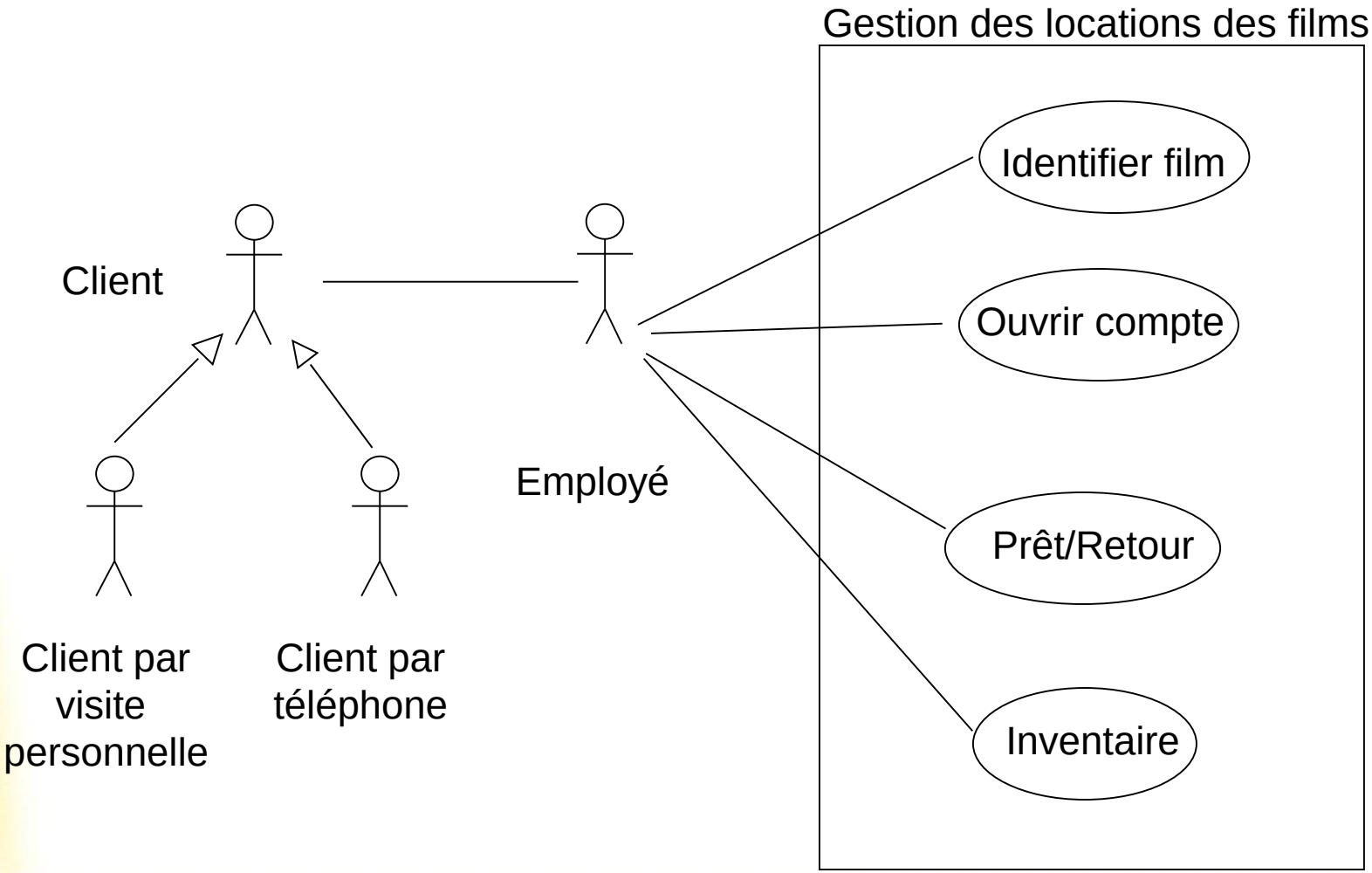
© Jonathan Gaudreault

Représenter les besoins fonctionnels à l'aide du diagramme des cas d'utilisation



© Jonathan Gaudreault

Représenter les besoins fonctionnels à l'aide du diagramme des cas d'utilisation

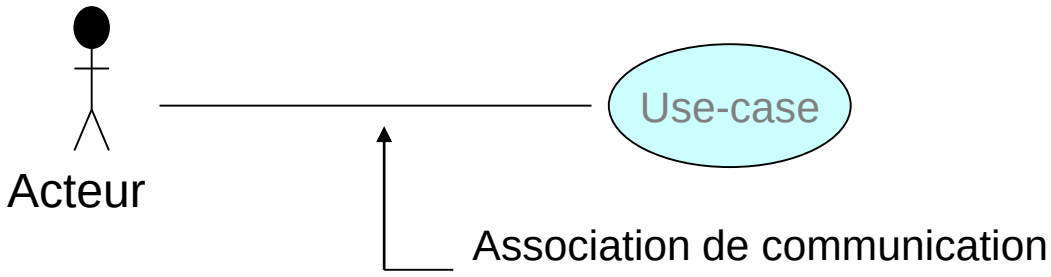


Identifier les acteurs

- Se poser les questions suivantes pour la détermination des acteurs :
  - **Acteurs primaires**
    - Qui va utiliser la fonctionnalité principale du système ?
    - Qui aura besoin du système pour réaliser les tâches qui lui sont dédiées ?
  - **Acteurs secondaires**
    - Qui est intéressé par les résultats retournés par le système ?
    - Qui aura besoin de maintenir, administrer et laisser le système fonctionner ?
  - **Systèmes externes**
    - Quels sont les moyens physiques dont le système a besoin pour faire les traitements ? (systèmes externes)
    - Avec quels systèmes le système interagit ? (systèmes initiant le contact avec le système et systèmes contactés par celui-ci)

Les cas d'utilisation / use-case

- Un *use-case* est une fonctionnalité complète telle que perçue par un acteur
- Il représente une séquence d'actions que le système exécute en réponse à une activation venant d'un acteur par exemple
- Sur le schéma, un use-case est connecté à un acteur par une association de communication



Identifier les cas d'utilisation…

- Pour chaque acteur (primaires et secondaires), identifier ses buts (on suppose qu'une fonctionnalité du système est liée à chacun de ces buts)
- Définir un "cas d'utilisation" pour chacun de ces buts
- Réfléchir aux frontières du système
- Identifier les composantes / acteurs externes

© Jonathan Gaudreault

Identifier les cas d'utilisation… pertinents

- Répondant à un but d'un utilisateur

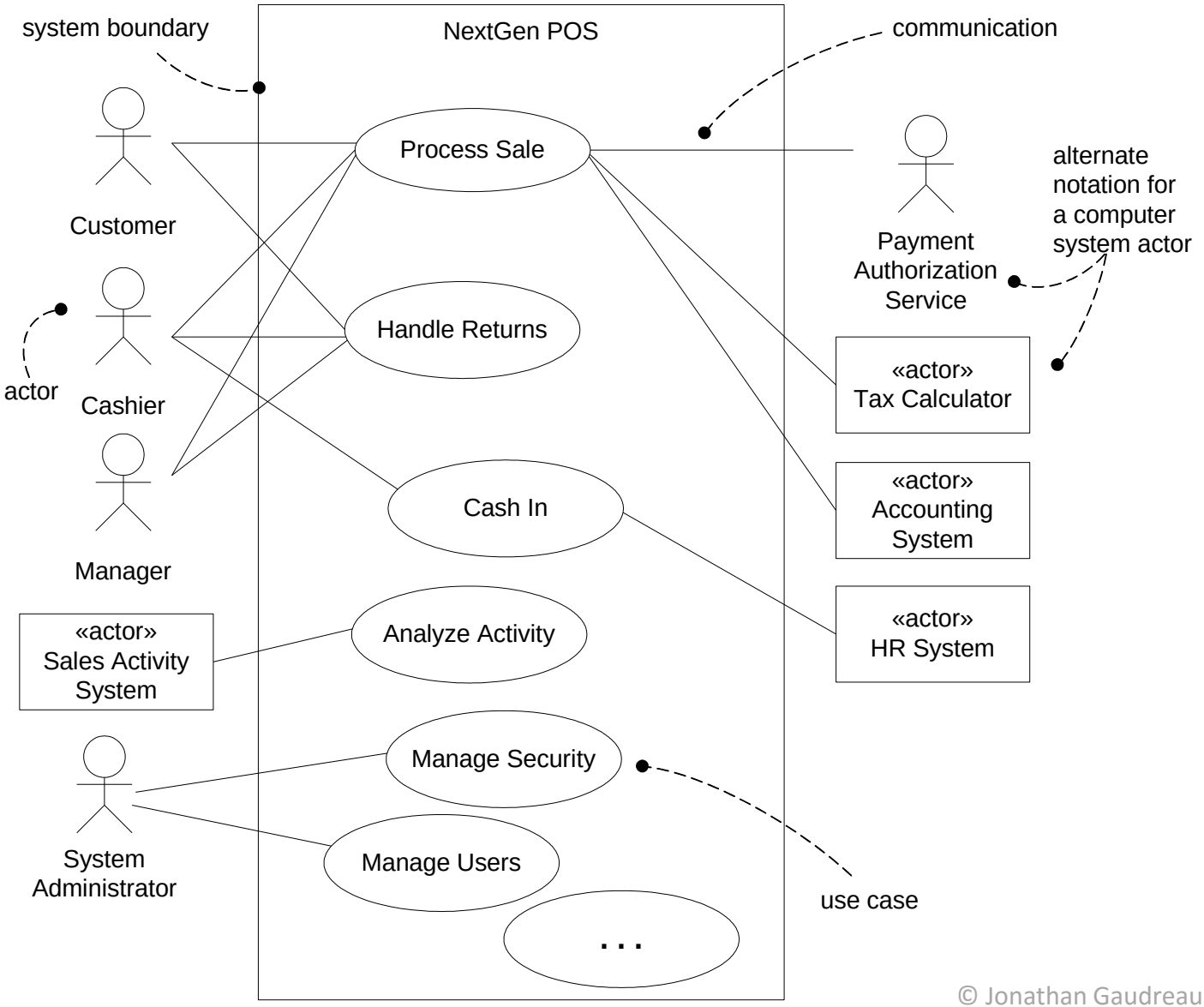
- Permettant à l'utilisateur d'accomplir une tâche quelque chose ayant de la valeur dans le contexte de l'entreprise ("test du patron")

© Jonathan Gaudreault

Exemple Système POS NexGen (livre)

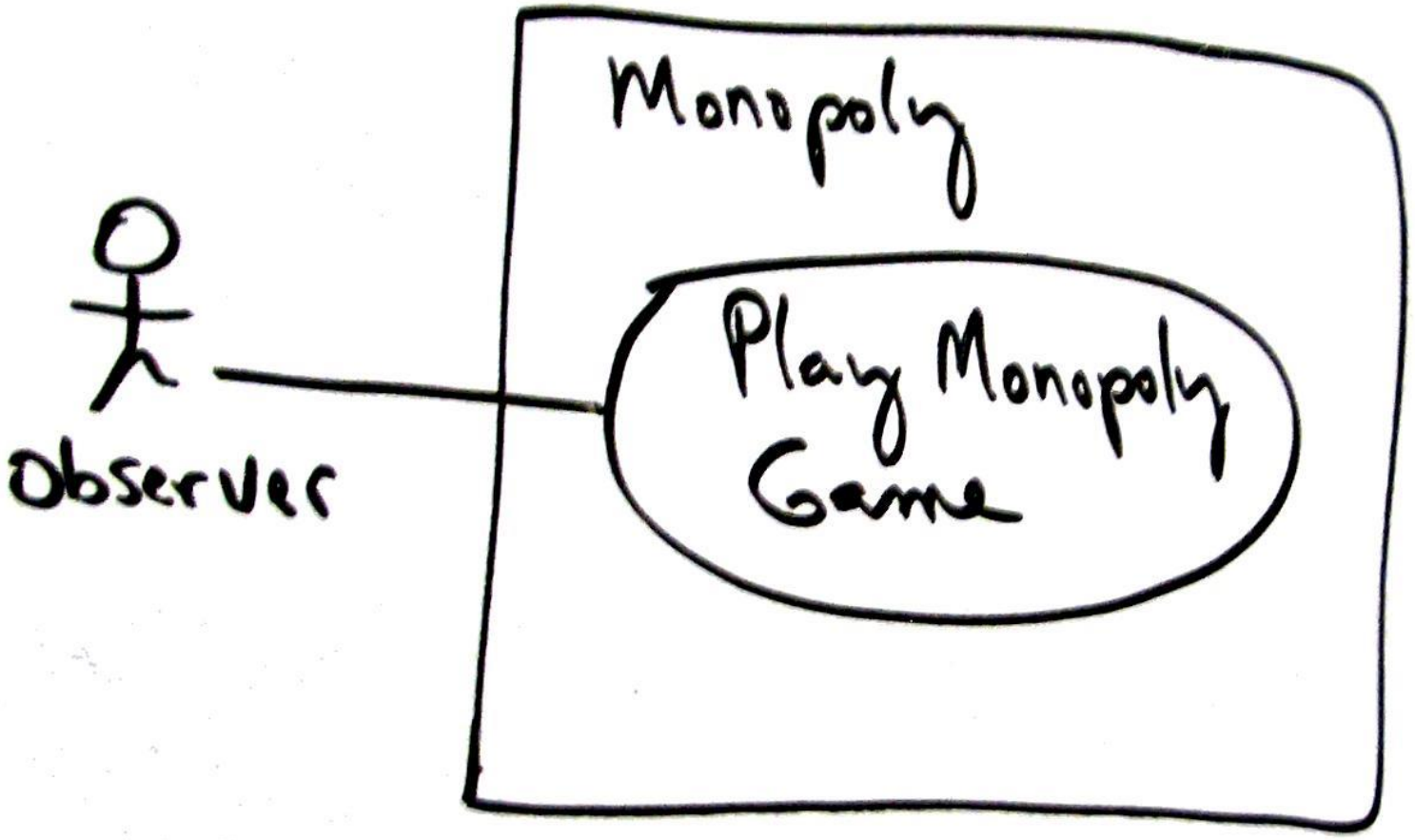
© Jonathan Gaudreault

Exemple Système POS NexGen (livre)



© Jonathan Gaudreault

Exemple ridicule !



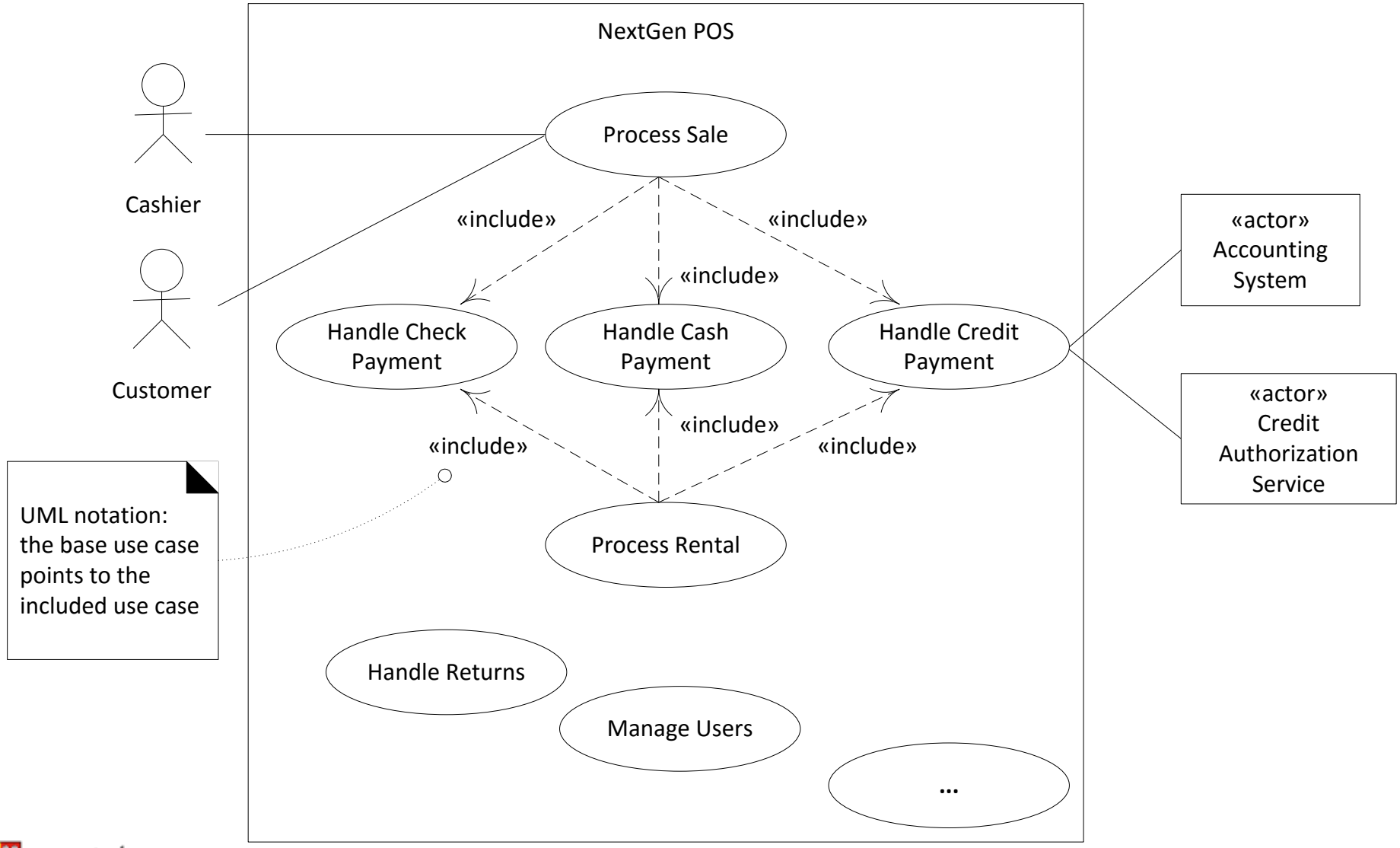
© Jonathan Gaudreault

Relation d'inclusion (ou d'utilisation): « include »

- Lorsque plusieurs use-cases possèdent un comportement commun, ce dernier peut être modélisé dans un seul use-case qui sera utilisé par tous les autres use-cases
- La relation « include » permet de définir des comportements partageables entre plusieurs use-cases
- Lorsqu'un use-case utilise un autre use-case, cette utilisation est quasi-obligatoire pour que le use-case source réalise son travail

© Jonathan Gaudreault

Exemple – NextGen POS

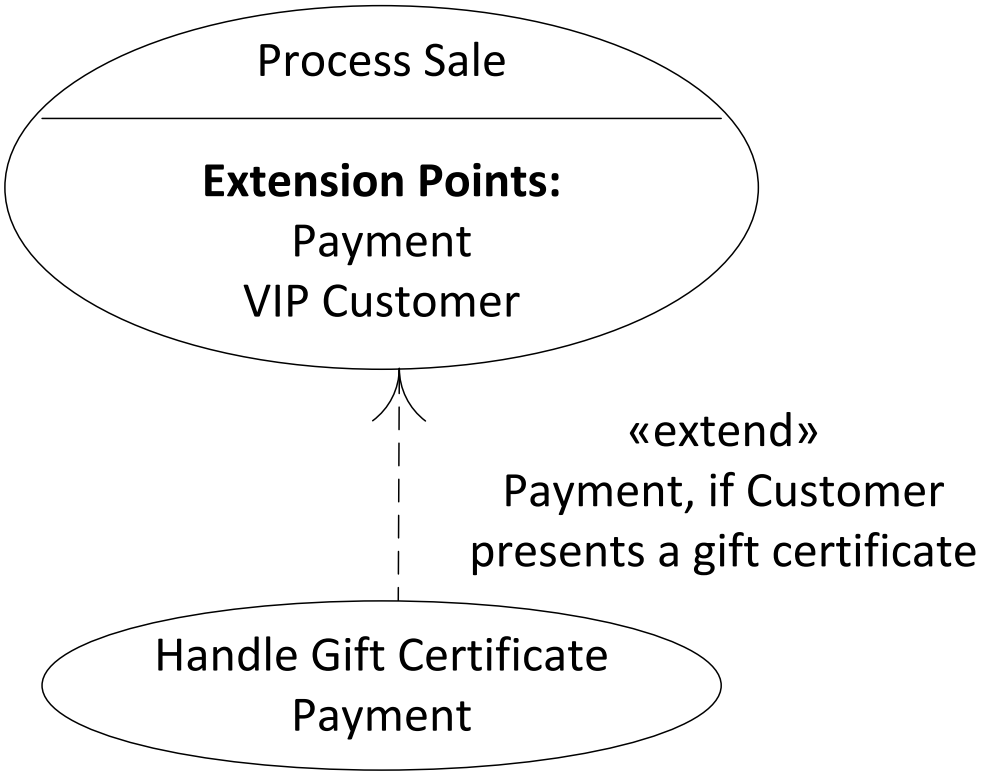


Relation d'extension: « extend »

- C'est une relation qui indique qu'un use-case peut étendre un autre use-case en y ajoutant de nouvelles actions
- Le use-case source ajoute son comportement au use-case destination
- Le use-case destination peut cependant être utilisé seul

© Jonathan Gaudreault

Exemple – NextGen POS



UML notation:
1. The extending use case points to the base use case.

2. The condition and extension point can be shown on the line.

Relations entre les cas d'utilisation

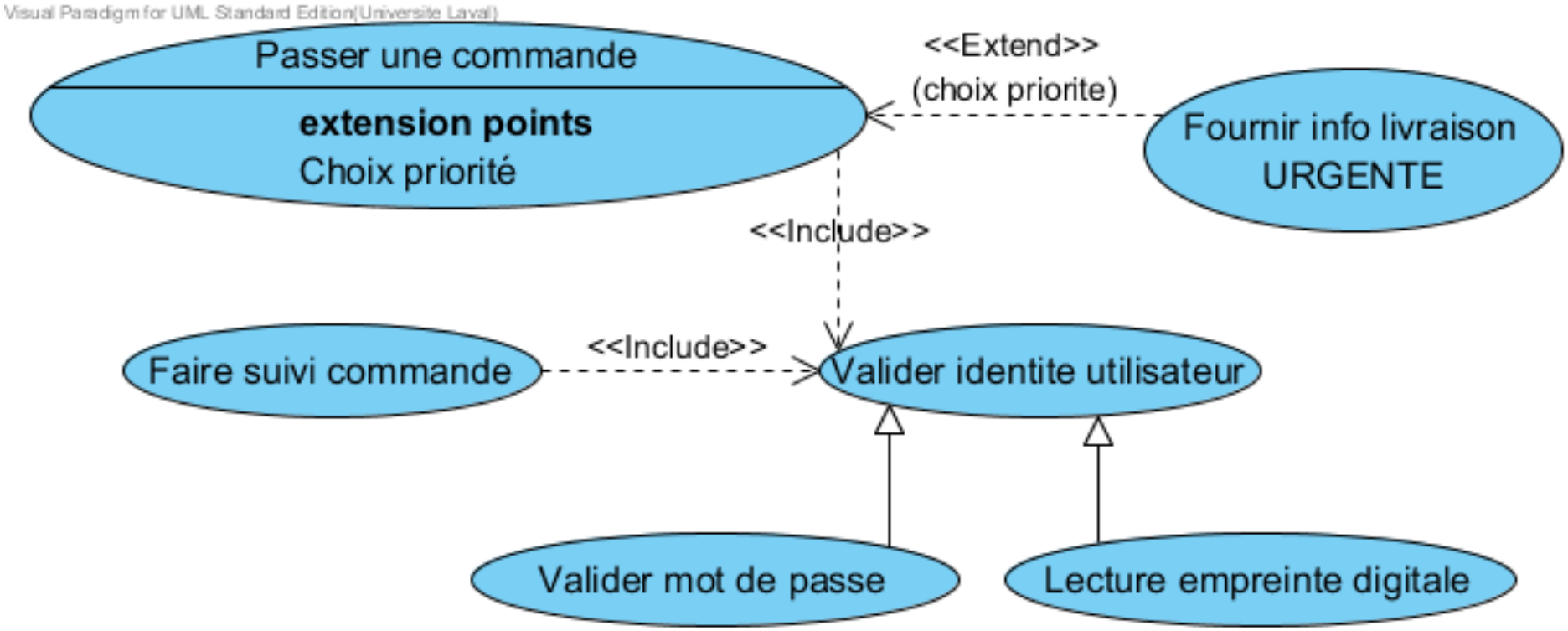
- **<<include>>** : un cas comprends explicitement le comportement d'un autre cas à un endroit spécifié dans le texte du cas de base. Le cas qui est inclus peut aussi être exécuté de manière autonome.

- **<<extend>>** : Le cas de base (le cas « étendu ») peut être utilisé de manière autonome, mais dans certaines circonstances son comportement est étendu par un autre cas. Le cas « étendant » ne peut généralement pas être exécuté de manière autonome.

- **Héritage** : définir une version modifiée d'un cas d'utilisation.

© Jonathan Gaudreault

Exemple « extend » et « include »



Exemple – Guichet automatique

© Jonathan Gaudreault

Exemple – Guichet automatique

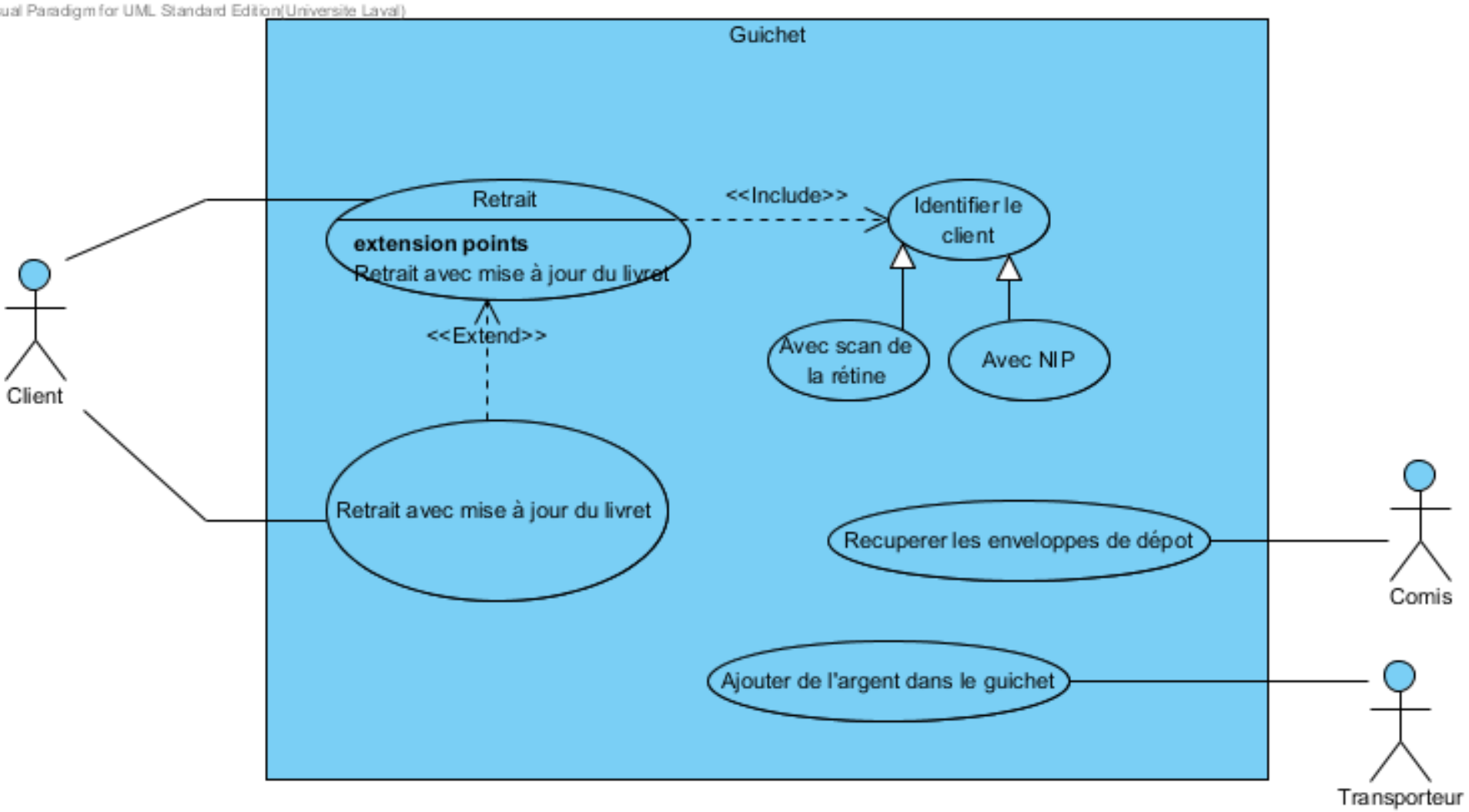


Diagramme des cas d'utilisation

≠

Cas d'utilisation

- **Diagramme des cas d'utilisation**: identifie **les** cas d'utilisation et les place dans un contexte
- Chaque **bulle** dans le diagramme identifie/nomme/réfère à un cas d'utilisation
- Mais **le cas d'utilisation est décrit par un texte** « à part ».
- Ce texte peut décrire un ou plusieurs « **scénarios** » associés à ce use case.
  - Un scénario principal/primaire
  - Des scénarios alternatifs

© Jonathan Gaudreault

Description du cas d'utilisation

- Possibilité de le décrire avec différents niveaux de détail:
  - **Abrégé**: simple paragraphe décrivant le scénario principal.
  - **Informel**: On décrit en plus les scénarios alternatifs.
  - **Détaillé**: On décrit en détail chaque scénario. On spécifie en plus les détenteurs d'intérêt, on en plus des préconditions et garantie de succès, etc.

© Jonathan Gaudreault

Abrégé

Cas d'utilisation:	**Effectuer un retrait**
Acteur(s):	Client
Type:	Primaire
Description:	Un client se présente au guichet automatique, s'identifie, retire un montant d'argent et quitte avec son argent.

© François Guibault – 2006

Détaillé

Cas d'utilisation:	**Effectuer un retrait**
Système:	Guichet automatique
Acteurs:	Client
Parties prenantes et intérêts:	Client: Il désire repartir avec son argent. Gestionnaire de la banque: Il veut que le système enregistre la transaction.
Préconditions:	Le client a un compte bancaire actif.
Garanties en cas de succès:	Le client a reçu son argent et son compte a été débité.
Scénario principal:	1. Un client se présente au guichet automatique 2. Il s'identifie. 3. …
Scénarios alternatifs:	…

© François Guibault – 2006

Détaillé en version deux colonnes

1. Un client se présente au guichet	
2. Le client s'identifie	3. Présentation des options
4. Le client choisit de faire un retrait.	5. Sollicitation du compte à débiter
6. Le client choisit le compte	7. Sollicitation du montant du retrait.
8. Le client entre le montant à retirer.	9. Validation de la disponibilité des fonds.
	10. Débite le compte
	11. Remet l'argent
12. Le client prend l'argent et quitte.	

© François Guibault – 2006

Concepts avancés

- Un use-case peut être décrit par un **diagramme d'activités** (plutôt qu'un texte)
- Exemple :

Use-case : Acheter boisson