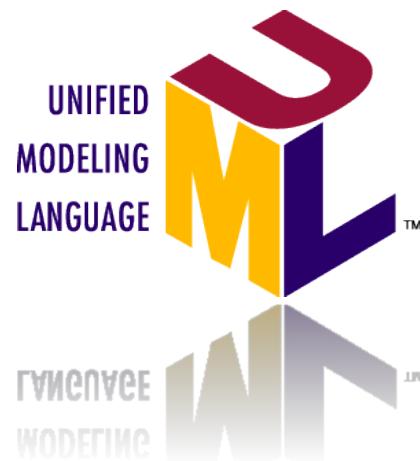




Conception de Systèmes d'Information

Le Langage UML :

Introduction et Modèles Principaux

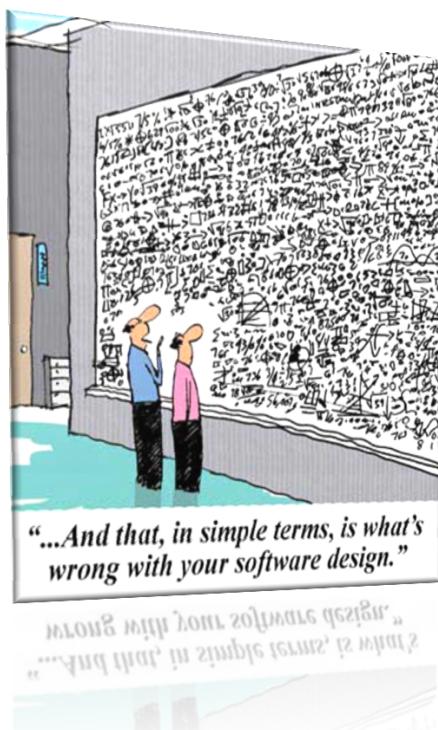


Formateur: Sarah Bouraga (sarah.bouraga@unamur.be)

Unified Modeling Language: UML

Table des matières

| | |
|--|----------------|
| Théorie..... | 1 |
| Chapitre 0: Introduction..... | 3 |
| Chapitre 1: Les Bases du Langage UML..... | 22 |
| Chapitre 2: La Modélisation des Comportements..... | 30 |
| Chapitre 3: Le Diagramme de Use Case | 34 |
| Chapitre 4: Le Diagramme d'Activité | 46 |
| Chapitre 5: Le Diagramme de Classe | 69 |
| Chapitre 6: Le Diagramme d'Interaction | 103 |
| Chapitre 7: Le Diagramme d'Etat..... | 121 |
| Annexes | 142 |
| Exercices..... | 163 |
| Check Lists..... | 165 |
| QCM..... | 168 |
| Exercices de Modélisation | 179 |
| Le Diagramme de Use Case..... | 179 |
| Le Diagramme d'Activités | 183 |
| Le Diagramme de Classe | 187 |
| Le Diagramme de Séquence | 190 |
| Le Diagramme d'Etat | 192 |
| Exercice Intégré | 197 |



UNIFIED MODELING LANGUAGE (UML)

Concepts de Base en UML pour la Modélisation de Systèmes d'Information

Sarah Bouraga

[1]

Objectifs de la Formation

- Savoirs:
 - Comprendre l'utilité du langage UML dans le processus de développement de logiciels
 - Connaitre les constructeurs et la grammaire du
 - Diagramme de Use Case
 - Diagramme de Classes
 - Diagramme d'Activité
 - Diagramme d'Interactions: le Diagramme de Séquence
 - Diagramme d'État
 - Comprendre et utiliser les liens entre les différents diagrammes

[2]

Objectifs de la Formation

- Savoir-faire:
 - Utiliser les différents diagrammes de conception dans des exercices simples
 - Utiliser et spécifier un ensemble d'exigences concernant un système d'information à créer

[3]

Table des Matières

- Chapitre 0: Introduction
- Chapitre 1: Les Bases du Langage UML
- Chapitre 2: La Modélisation des Comportements
- Chapitre 3: Le Diagramme de Use Cases
- Chapitre 4: Le Diagramme d'Activité
- Chapitre 5: Le Diagramme de Classes
- Chapitre 6: Les Diagrammes d'Interactions
- Chapitre 7: Le Diagramme d'État

[4]

Table des Matières

- *Chapitre 0: Introduction*
- Chapitre 1: Les Bases du Langage UML
- Chapitre 2: La Modélisation des Comportements
- Chapitre 3: Le Diagramme de Use Cases
- Chapitre 4: Le Diagramme d'Activité
- Chapitre 5: Le Diagramme de Classes
- Chapitre 6: Les Diagrammes d'Interactions
- Chapitre 7: Le Diagramme d'État

[5]

CHAPITRE 0: INTRODUCTION

Systèmes d'Information & Software Engineering
Introduction à UML

[6]

Introduction

- Systèmes d'Information
- Software Engineering
- Introduction à UML

[7]

Introduction

- *Systèmes d'Information*
- Software Engineering
- Introduction à UML

[8]

Systèmes d'Information

- Un Système d'Information (SI) est:
 - Un ensemble structuré
 - De ressources techniques (forte composante logicielle)
 - De ressources humaines
 - Destiné à acquérir, mémoriser, traiter, transmettre, retrouver et communiquer des informations
 - Pour satisfaire les besoins d'une organisation
- Un SI est composite par défaut:
 - Composants software, et
 - Composants hardware
 - (Implémentation dans un environnement existant)

[9]

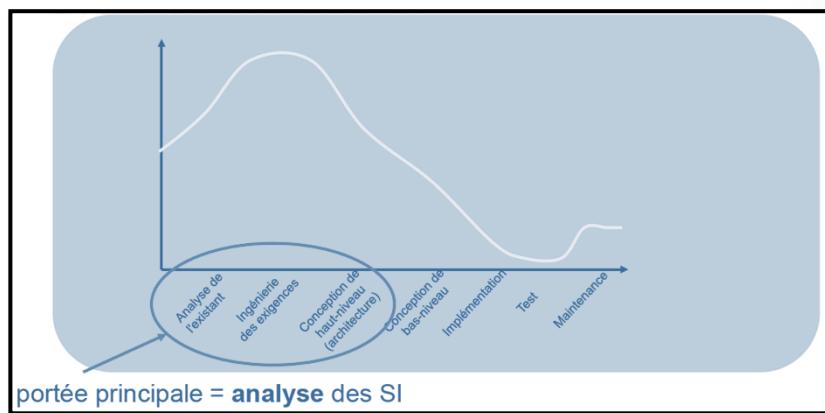
Introduction

- Systèmes d'Information
- **Software Engineering**
- Introduction à UML

[10]

Software Engineering

- Software Engineering:
 - *Processus de création d'un SI particulier*



[11]

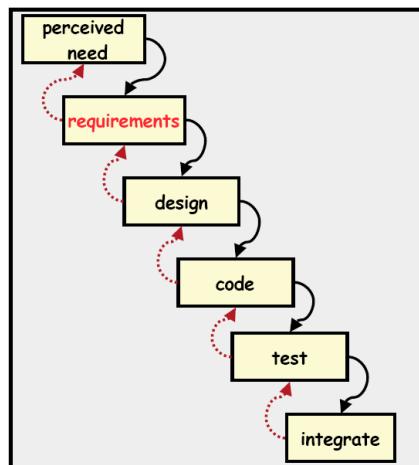
Software Engineering – Différents Processus

- Différents Processus:
 - Modèle Séquentiel:
 - Waterfall
 - Processus Itératifs:
 - Évolutionnaire
 - Incrémental

[12]

Software Engineering – Différents Processus

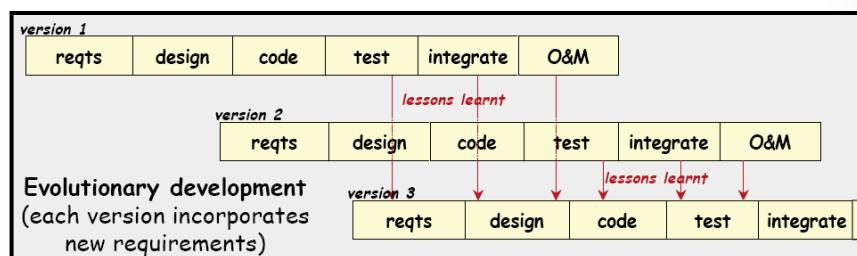
- Modèle Séquentiel: Waterfall



[13]

Software Engineering – Différents Processus

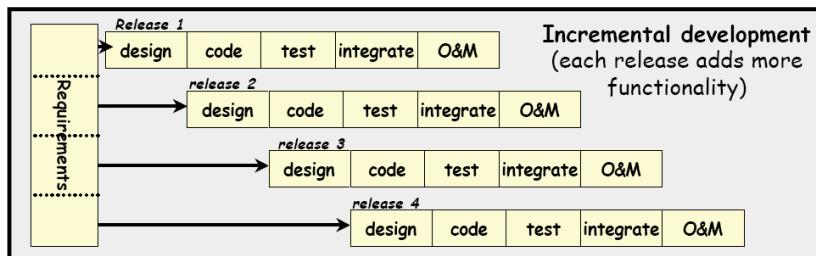
- Modèle Itératif: Évolutionnaire



[14]

Software Engineering – Différents Processus

- Modèle Itératif: incrémental



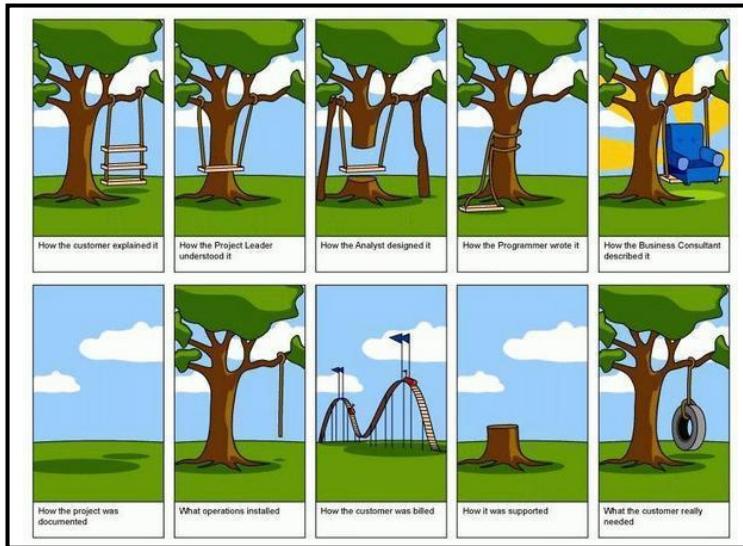
[15]

Software Engineering – Requirements Engineering

- Exigence (Requirement):
 - Besoin exprimé par une partie prenante devant être rencontré par le SI à créer (et ce, étant donné les hypothèses du domaine d'application)
- Modélisation et spécification:
 - Pourquoi
 - Analyser *correctement* le domaine d'application?
 - Récolter *correctement* les exigences?
 - Spécifier *correctement* le SI à créer?

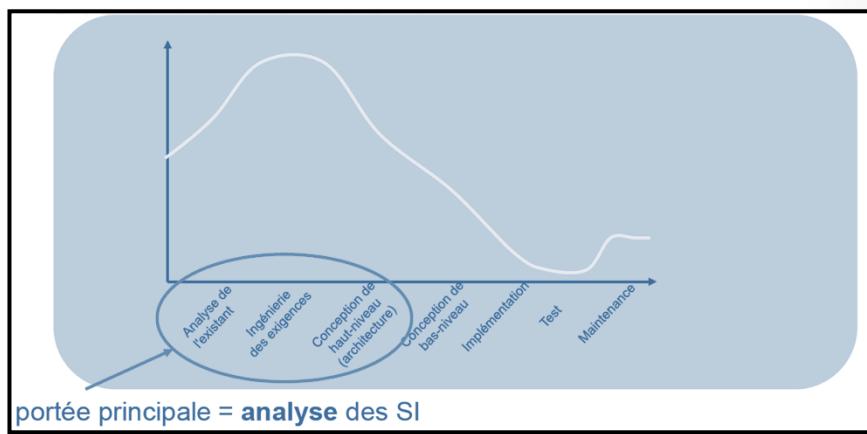
[16]

Software Engineering – Modélisation et Spécification



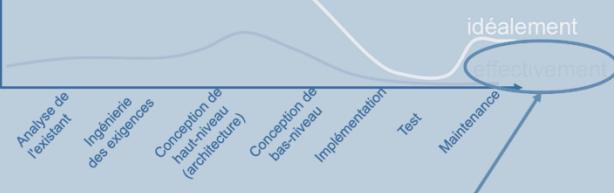
[17]

Software Engineering – L'Analyse Trop Souvent Absente



[18]

Software Engineering – L'Analyse Trop Souvent Absente

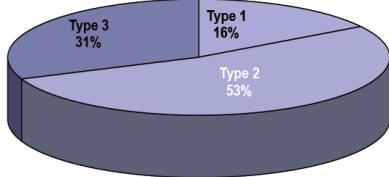


[19]

Quelques Chiffres

- Dépenses annuelles mondiales en logiciel, matériel et services liés aux TIC (Technologies de l'Information et de la Communication)
 - 1,2 milliards de \$ en 2006
 - 1,5 milliards de \$ prévu en 2010
- Taux de succès des projets informatiques (8380 projets analysés)

Type 3 – projets abandonnés en cours de réalisation



Type 1 – projets réussis à temps, avec le budget prévu et réalisant les fonctionnalités demandées

Type 2 – projets terminés, hors budget (>180% du coût prévu initialement), hors planning et n'offrant pas toutes les fonctionnalités requises

[20]

Alors, Pourquoi Négliger l'Analyse et Modélisation des SI à Créer?

- Pression croissante du marché
 - Le client veut plus, plus vite, moins cher
- Raisonnement à court terme
 - « Faisons toujours sans cahier des charges, on verra bien après »
- Méconnaissance technique
 - Ignorance collective
 - Génie Logiciel = discipline jeune (50 ans)
 - Pas encore de corpus de bonnes pratiques bien établi
 - Les mythes: ex. "Le logiciel, c'est immatériel et donc facile à modifier"
 - Ignorance individuelle
 - Pas d'accès à la profession
 - "Bidouilleurs" percevant l'analyse comme un frein à leur créativité
- Existence de chefs de projets ne percevant pas les enjeux et ne maîtrisant pas les techniques

[21]

Introduction

- Systèmes d'Information
- Software Engineering
- ***Introduction à UML***

[22]

Introduction à UML

- UML:
 - Notations standardisées
 - ≠ Méthodologie/processus de software engineering

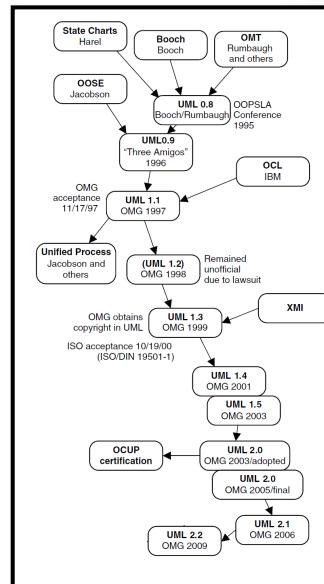
[23]

Introduction à UML

- Unified:
 - Réunion de plusieurs vues *intégrées* d'un SI à créer
- Modeling:
 - Modélisation de Systèmes d'Information
 - Modéliser ≠ dessiner
 - Modéliser ≠ écrire de la documentation pour "faire plaisir"
 - Modéliser = comprendre un domaine d'application et le communiquer efficacement
- Language:
 - UML donne une série de constructeurs et une grammaire permettant de spécifier un SI à créer sur base des exigences récoltées (= besoins des parties prenantes)

[24]

Introduction à UML – Historique



[25]

Introduction à UML – Diagrammes

- UML et ses différents diagrammes:
 - UML est un ensemble de vues sur le SI
 - Idem qu'en architecture:
 - Vue de l'urbaniste, Vue de l'architecte, Vue du client, Vue du maçon, Vue du plombier, Vue du cadastre, ...

[26]

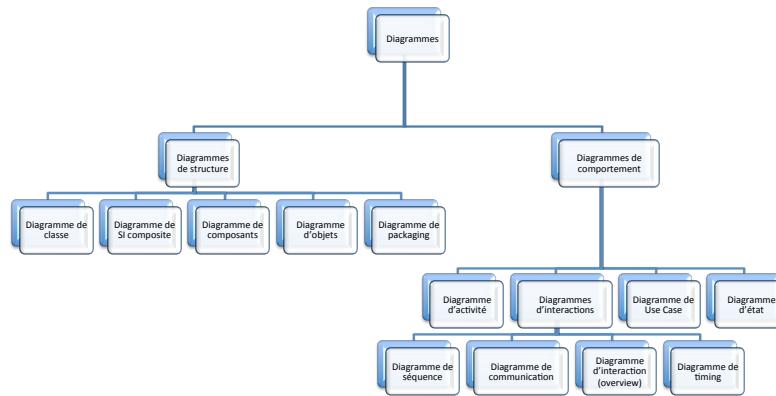
Introduction à UML – Principales Perspectives

- Principales perspectives d'UML:
 - Perspective informationnelle
 - Contenu: les données
 - Diagramme de Classes UML (autre: Entité-Association)
 - Perspective fonction
 - Contenu: les services, tâches, opérations,...
 - Diagramme de Use Case UML (autre: Features Diagrams)
 - Perspective dynamique
 - Contenu: le comportement des objets, acteurs, document,...
 - Diagramme de Séquence, de Collaboration, d'État et d'Activité UML (autre: les réseaux de Petri)
 - Perspective de l'Orienté Objet
 - Contenu: données + fonctions (dynamisme)
 - Diagramme de Classes UML

[27]

Introduction à UML – Diagrammes

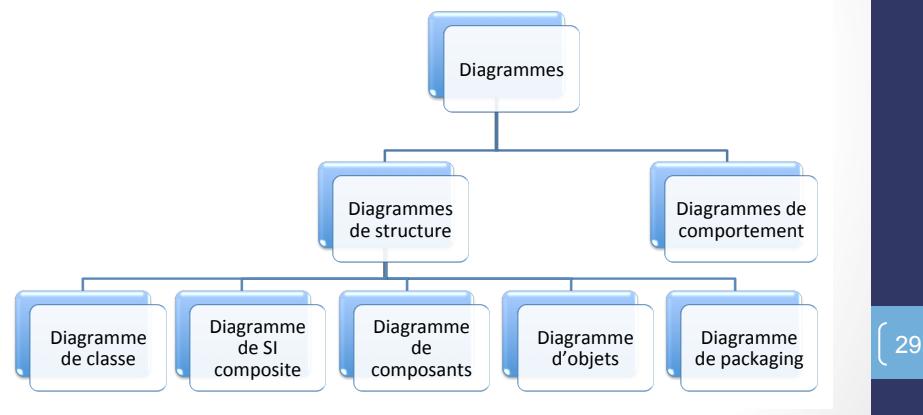
- Liens entre les diagrammes



[28]

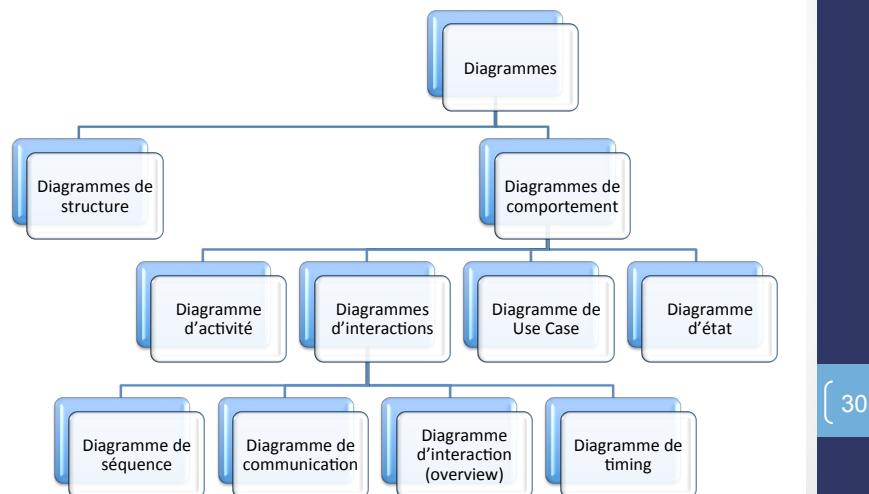
Introduction à UML – Diagrammes

- Les diagrammes de structure:



Introduction à UML – Diagrammes

- Les diagrammes de comportement:



Introduction à UML – Modèle vs Schéma

- Modèle:
 - Un système formel de représentation de certains aspects de tous les domaines d'applications d'un certain type
 - En SE: représenter des acteurs, des objets (+ attributs et méthodes), des états, des activités des acteurs, liens,...
 - Plusieurs points de vue
- Schéma:
 - Représentation des concepts abstraits d'un domaine d'application *particulier* suivant un point de vue spécifique
 - En UML: la schématisation d'un point de vue sur le domaine d'application donne lieu à la création d'un diagramme
 - Diagramme: représentation simplifiée/abstraite d'un aspect du SI:
 - Moyen de communiquer avec le client
 - Maîtrise des risques (réduction)
 - Diriger le processus de création du SI

[31]

Introduction à UML – Langage Naturel

- Et pourquoi pas le langage naturel?

| Avantages | Inconvénients |
|--------------------------------|--|
| Aucun apprentissage nécessaire | Verbosité: long à créer et à lire |
| Création facile par tous | Les 7 péchés du LN: |
| Lecture facile par tous | <ul style="list-style-type: none"> - Ambiguïté - Bruit - Silence - Sur-spécification - Contradiction - Référence en avant - Se repentir |

[32]

Introduction à UML – Langage Naturel

- Un bref exemple:
 - Un homme demande à son épouse pour aller acheter un bac de bière
 - La dame dit à son mari: « *D'accord, tu peux aller chercher un bac de bières, et s'il y a des œufs, prends-en six* »
 - Combien de bacs de bière peut prendre le mari et sous quelle(s) condition(s)?

[33]

Introduction à UML – Notations Semi-Formelles

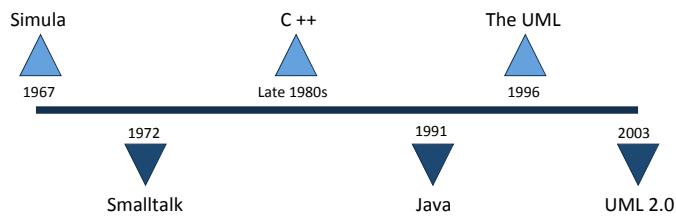
- UML est une notation semi-formelle

| Avantages | Inconvénients |
|---|--|
| Syntaxe bien définie | Risques d'ambiguïté persistants → langage formel? |
| Généralement graphique (donc intuitif?) | <ul style="list-style-type: none"> - Souvent très difficile et syntaxe peu flexible |
| Interprétation commune partielle | |
| Possible automatisation partielle du raisonnement et de la génération de code | |
| Raisonnement aisément d'apprentissage | |

[34]

Introduction à UML – Orienté Objet

- UML fait partie du paradigme Orienté Objet (OO)



[35]

Introduction à UML – Langage de Modélisation

- D'autres langages de modélisation
 - I*
 - KAOS
 - Tropos
- Ces langages peuvent être utilisés avant (et pendant) l'utilisation de UML pour aider au respect des exigences des parties prenantes au projet
 - ‘How question’

[36]

Introduction à UML – Langage de Modélisation

- i*:
 - Langage utilisé au tout début du processus de RE pour modéliser les acteurs, leurs objectifs et leurs relations
 - ‘Who’ et ‘Why questions’
 - Pas de ‘What question’
- KAOs et Tropos:
 - Langages permettant de modéliser toutes les exigences (utile durant l'entièreté du processus de RE)
 - ‘Who, Why, What questions’

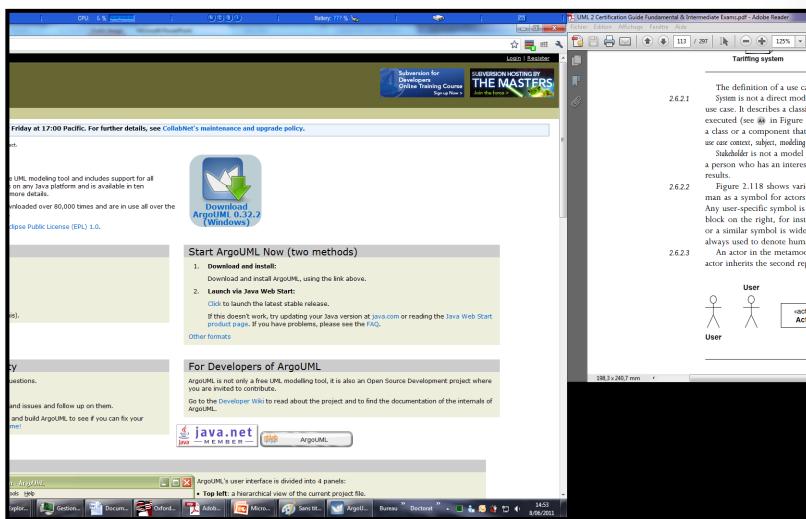
[37]

Introduction à UML – Un Outil



[38]

Introduction à UML – Un Outil



[39]

Introduction à UML – Outils

- D'autres outils:
 - Logiciels libres
 - Logiciels propriétaires

[40]

Introduction à UML – Outils

- Logiciels libres:
 - ATL open source
 - Dia
 - Umbrello un modeleur UML sous GPL
 - Gaphor un modeleur UML sous GPL la version actuelle est 0.7.1
 - BOUML, un modeleur UML sous GPL pour Windows, Linux, MacOS X et Solaris
 - Eclipse GMT-XUML
 - Eclipse UML2, Méta modèle UML2, sans interface graphique
 - Netbeans 5.5
 - Staruml, en version libre
 - Acceleo, générateur de code source à partir de modèles UML

[41]

Introduction à UML – Outils

- Logiciels propriétaires:
 - Rational Rose : Un des outil les plus important du marché - <http://www.rational.com/> / Racheté par IBM
 - Together (racheté par Borland/Inprise) : Outil fortement couplé avec Java
 - Visio : Outil (non complet) de Microsoft
 - Together
 - Poseidon, basé sur ArgoUml (version commerciale)
 - Rational Software Architect / Rational Software Modeler (et toujours Rose/XDE), de IBM Software Rational
 - PowerDesigner, de Sybase (outil de modélisation complet intégrant UML)
 - Objecteering de Softeam
 - Visual Paradigm for UML, de Visual Paradigm Internation Ltd.
 - SDE for Eclipse, un plugin UML pour Eclipse
 - Omondo EclipseUML, un plugin UML pour Eclipse
 - Jude [1], en Java
 - MagicDraw, un éditeur de diagrammes UML
 - Enterprise Architect, un outil de modélisation UML

[42]

Table des Matières

- Chapitre 0: Introduction
- ***Chapitre 1: Les Bases du Langage UML***
- Chapitre 2: La Modélisation des Comportements
- Chapitre 3: Le Diagramme de Use Cases
- Chapitre 4: Le Diagramme d'Activité
- Chapitre 5: Le Diagramme de Classes
- Chapitre 6: Les Diagrammes d'Interactions
- Chapitre 7: Le Diagramme d'État

[43]

CHAPITRE 1: LES BASES DU LANGAGE UML

[44]

Les Bases du Langage UML

- Le Typage des Données
- La Structure Générique d'un Diagramme
- Les Stéréotypes en UML

[45]

Les Bases du Langage UML

- *Le Typage des Données*
- La Structure Générique d'un Diagramme
- Les Stéréotypes en UML

[46]

Le Typage des Données

- Les types de données:
 - DataType
 - PrimitiveType
 - EnumerationType

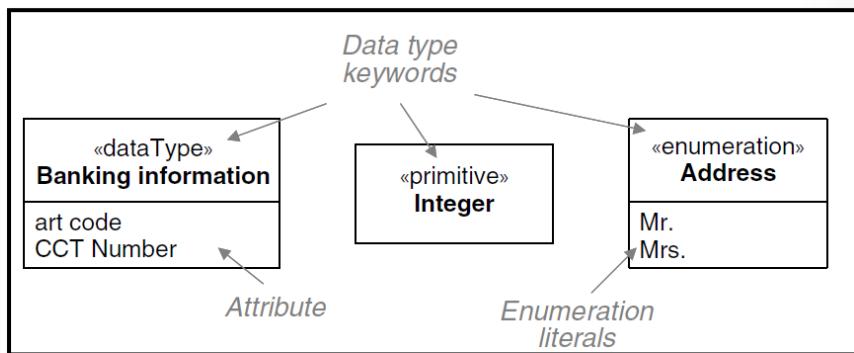
[47]

Le Typage des Données

- DataType (type simple):
 - Type de données dont les valeurs n'ont pas d'identité propre (eg, 1€)
- PrimitiveType (type primitif):
 - Type de données sans structure particulière et communément connu (eg, l'ensemble des entiers, les lettres de l'alphabet)
- EnumerationType:
 - Type de données ayant une valeur contenue dans un ensemble restreint (eg, état civil, adresse)

[48]

Le Typage des Données



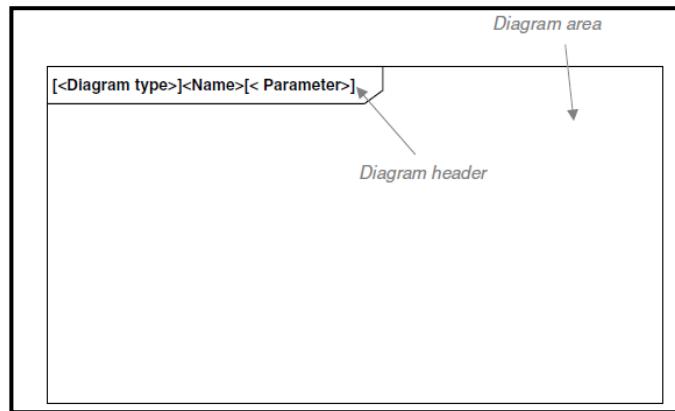
[49]

Les Bases du Langage UML

- Le Typage des Données
- *La Structure Générique d'un Diagramme*
- Les Stéréotypes en UML

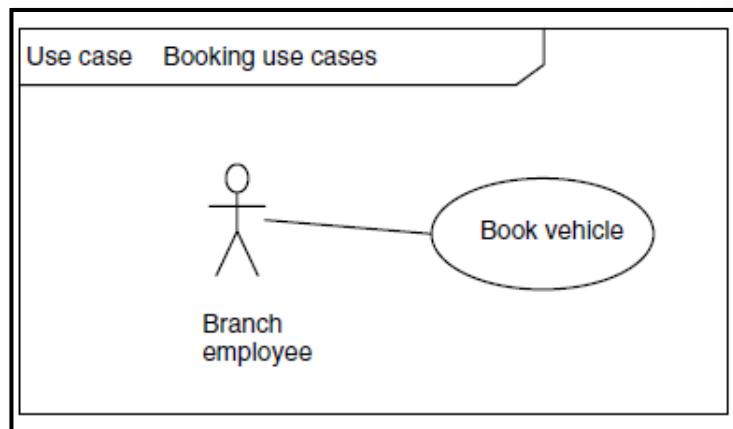
[50]

La Structure Générique d'un Diagramme



[51]

La Structure Générique d'un Diagramme



[52]

Les Bases du Langage UML

- Le Typage des Données
- La Structure Générique d'un Diagramme
- *Les Stéréotypes en UML*

[53]

Les Stéréotypes en UML

- Les stéréotypes:
 - Mécanisme d'extensibilité utilisé en UML
 - Extension du méta-modèle utilisé par et pour certains projets/certaines entreprises/certains consultants
 - Un stéréotype est dérivé de ce qui existe déjà dans le langage UML, mais avec des propriétés spécifiques
 - Avec un stéréotype, on crée un nouveau type d'élément de modélisation → on étend la sémantique du méta-modèle UML utilisé

[54]

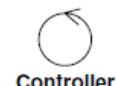
Les Stéréotypes en UML

- Utiliser:
 - La notation classique:

«stereotype»
name

Ou

- Un symbole défini au départ de la notation classique:



Controller

«control»
Controller

- Principe de réutilisabilité:

- Nouvelle classe "type" créée en plus de ce qui est déjà prévu dans UML

[55]

Les Stéréotypes en UML

- Les stéréotypes standards

| Stereotype | UML Element | Description |
|-----------------------|--------------------|---|
| «call» | Dependency (usage) | Call dependency between operations or classes. |
| «create» | Dependency (usage) | The source element creates instances of the target element. |
| «instantiate» | Dependency (usage) | The source element creates instances of the target element. <i>Note: This description is identical to the one of «create».</i> |
| «responsibility» | Dependency (usage) | The source element is responsible for the target element. |
| «send» | Dependency (usage) | The source element is an operation and the target element is a signal sent by that operation. |
| «derive» | Abstraction | The source element can, for instance, be derived from the target element by a calculation |
| «refine» | Abstraction | A refinement relationship (e.g., between a design element and a pertaining analysis element). |
| «trace» | Abstraction | Serves to trace of requirements. |
| «script» | Artifact | A script file (can be executed on a computer). |
| «auxiliary» | Class | Classes that support other classes («focus»). |
| «focus» | Class | Classes contain the primary logic. See «auxiliary». |
| «implementationClass» | Class | An implementation class specially designed for a programming language, where an object may belong to one class only. |

[56]

Les Stéréotypes en UML

- Les stéréotypes standards

| | | |
|------------------|--------------------|---|
| «metaclass» | Class | A class with instances that are, in turn, classes. |
| «type» | Class | Types define a set of operations and attributes, and they are generally abstract. |
| «utility» | Class | Utility classes are collections of global variables and functions, which are grouped into a class, where they are defined as class attributes/operations. |
| «buildComponent» | Component | An organizationally motivated component. |
| «implement» | Component | A component that contains only implementation, no specification. |
| «framework» | Package | A package that contains Framework elements. |
| «modelLibrary» | Package | A package that contains model elements, which are reused in other packages. |
| «create» | Behavioral feature | A property that creates instances of the class to which it belongs (e.g., constructor). |
| «destroy» | Behavioral feature | A property that destroys instances of the class to which it belongs (e.g., destructor). |

[57]

Table des Matières

- Chapitre 0: Introduction
- Chapitre 1: Les Bases du Langage UML
- ***Chapitre 2: La Modélisation des Comportements***
- Chapitre 3: Le Diagramme de Use Cases
- Chapitre 4: Le Diagramme d'Activité
- Chapitre 5: Le Diagramme de Classes
- Chapitre 6: Les Diagrammes d'Interactions
- Chapitre 7: Le Diagramme d'État

[58]

CHAPITRE 2: LA MODELISATION DES COMPORTEMENTS

[59]

La Modélisation des Comportements

- Les Comportements en UML: Introduction
- Les Diagrammes de Comportement en UML

[60]

La Modélisation des Comportements

- *Les Comportements en UML: Introduction*
- Les Diagrammes de Comportement en UML

[61]

Les Comportements en UML: Introduction

- La modélisation des comportements dans un SI comprend la description:
 - Des flux
 - Des dépendances temporelles
 - Des états et changements d'états
 - La gestion des événements
 - La gestion des échanges d'informations/données
 - ...
- Un comportement n'existe pas indépendamment à l'existence d'un objet spécifique:
 - Un comportement provient de l'action d'un objet
 - Un comportement peut provoquer un changement d'état pour l'un (ou plusieurs) des objets *impliqués*

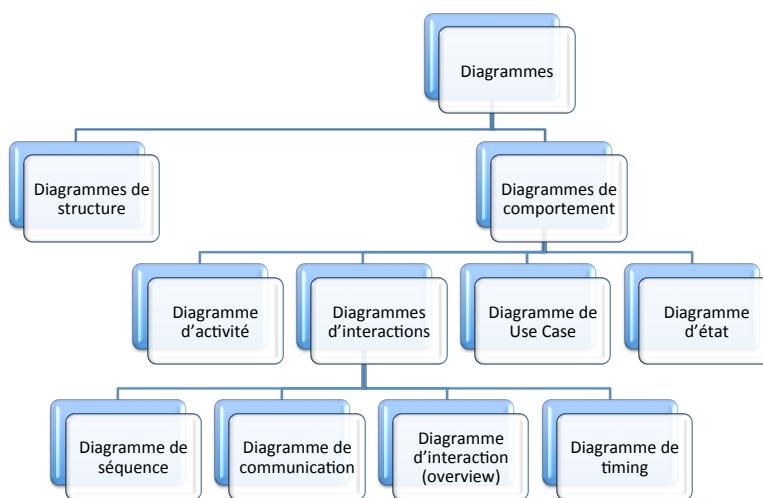
[62]

La Modélisation des Comportements

- Les Comportements en UML: Introduction
- *Les Diagrammes de Comportement en UML*

[63]

Les Diagrammes de Comportement en UML



[64]

Les Diagrammes de Comportement en UML

- Quatre manières de modéliser le comportement en UML:
 - Diagramme de Use Case
 - Diagrammes d'activités
 - Diagrammes d'états
 - Diagrammes d'interactions et de communication

[65]

Les Diagrammes de Comportement en UML

- Diagramme de Use Case
 - Accent: vue globale des interactions acteurs – SI
 - "Main stories for all actors"
- Diagrammes d'activités
 - Accent : séquencement/parallélisme, qui fait quoi, données/objets
 - "Some/all stories for some objects"
- Diagrammes d'états
 - Accent : changements d'états des objets
 - "All stories for one object"
- Diagrammes d'interactions et de communication
 - Accent : séquences de messages échangés entre objets
 - "One story for all objects"

[66]

Table des Matières

- Chapitre 0: Introduction
- Chapitre 1: Les Bases du Langage UML
- Chapitre 2: La Modélisation des Comportements
- ***Chapitre 3: Le Diagramme de Use Cases***
- Chapitre 4: Le Diagramme d'Activité
- Chapitre 5: Le Diagramme de Classes
- Chapitre 6: Les Diagrammes d'Interactions
- Chapitre 7: Le Diagramme d'État

[67]

CHAPITRE 3: LE DIAGRAMME DE USE CASES

[68]

Chapitre 3: Le Diagramme de Use Cases

- Use Case: Définition et Utilité
- Les Notations Utilisées pour Modéliser des Use Cases
- Description Textuelle des Use Cases

[69]

Chapitre 3: Le Diagramme de Use Cases

- ***Use Case: Définition et Utilité***
- Les Notations Utilisées pour Modéliser des Use Cases
- Description Textuelle des Use Cases

[70]

Use Case: Définition et Utilité

- Le diagramme de Use Case:
 - Utilisé pour spécifier un ensemble d'actions réalisées par le SI à créer et qui sont directement observables par un ou plusieurs acteurs
 - Chaque Use Case a un impact (direct ou indirect) sur au moins un des acteurs
 - Décrit les exigences fonctionnelles globales que le SI devra permettre
- Dans le diagramme:
 - Représentation du comportement du SI (sous certaines conditions) dans le but de satisfaire un objectif de l'un des acteurs

[71]

Chapitre 3: Le Diagramme de Use Cases

- Use Case: Définition et Utilité
- ***Les Notations Utilisées pour Modéliser des Use Cases***
- Description Textuelle des Use Cases

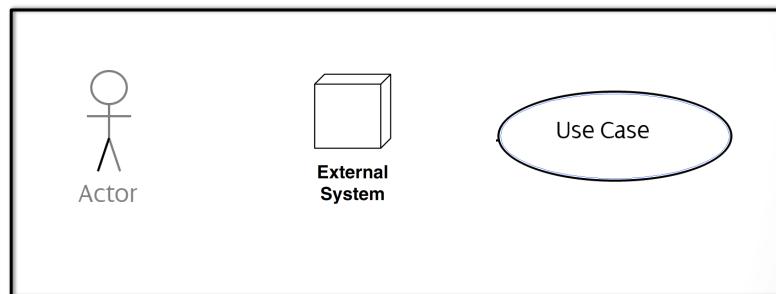
[72]

Les Notations Utilisées pour Modéliser des Use Cases

- La notion d'Acteur
 - Un acteur:
 - Utilisateur (ou groupe d'utilisateurs) interagissant avec le SI à créer
 - Stakeholders (personne ou groupe de personnes ayant un intérêt particulier par rapport au SI à créer et/ou dans ses résultats)
 - Un acteur peut être un SI externe
 - La dénomination d'un acteur doit contenir son rôle et/ou responsabilités
- Un Use Case:
 - Spécification d'un ensemble d'actions réalisées par le SI dont le résultat est valorisé et perceptible par un ou plusieurs acteurs

[73]

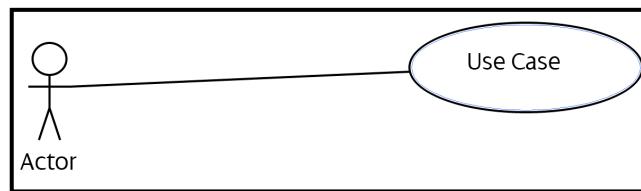
Les Notations Utilisées pour Modéliser des Use Cases



[74]

Les Notations Utilisées pour Modéliser des Use Cases

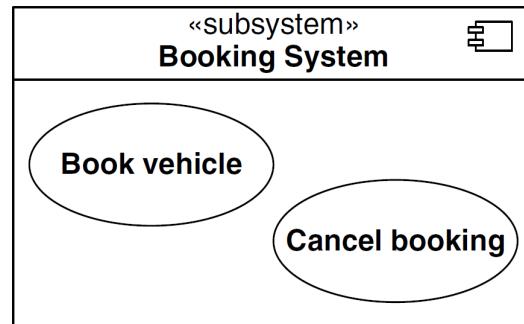
- La relation d'Association:
 - Le lien d'association entre un acteur et un Use Case indique quels sont les acteurs ayant un intérêt dans chaque Use Case
 - Multiplicité: par défaut 0..1



[75]

Les Notations Utilisées pour Modéliser des Use Cases

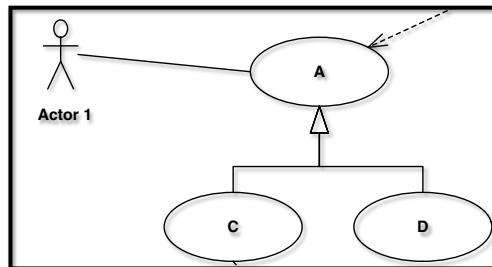
- Use Case par composant
 - Un SI peut être séparé en plusieurs composants
 - Les Use Cases sont rattachés à leur composant propre



[76]

Les Notations Utilisées pour Modéliser des Use Cases

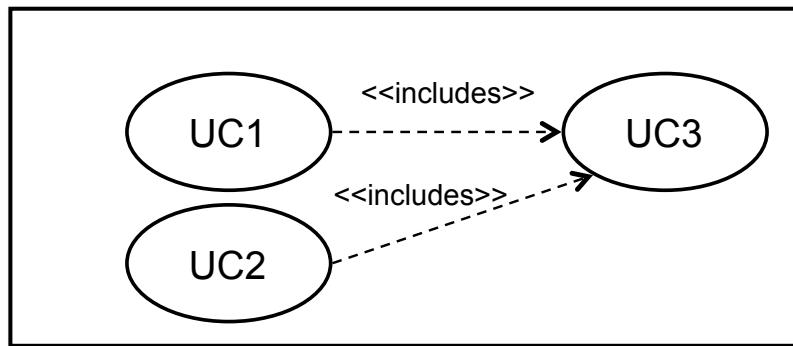
- La relation de Généralisation:
 - Principe d'abstraction destiné à structurer un modèle de manière hiérarchique
 - Relation entre un Use Case général et un Use Case spécialisé
 - Un Use Case peut être une spécialisation d'un autre Use Case
 - Le spécialisé ajoute des attributs au général
 - Relation entre Acteurs



[77]

Les Notations Utilisées pour Modéliser des Use Cases

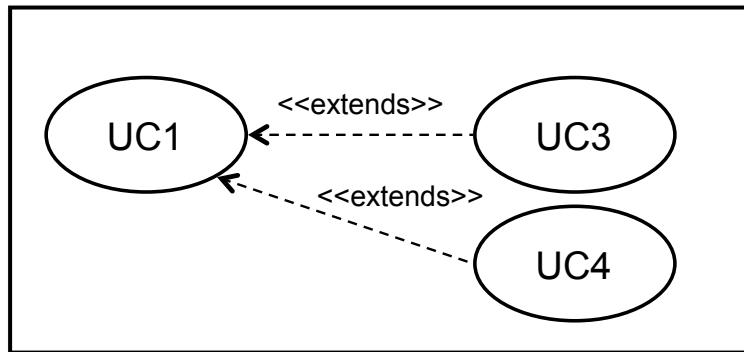
- La relation d'Inclusion (« *Include* »):
 - Intégration d'un Use Case dans un autre Use Case; le second étant une partie du premier
 - Un « *include* » Use Case peut *ne pas* être lié à un acteur



[78]

Les Notations Utilisées pour Modéliser des Use Cases

- La relation d'Extension (« Extend »):
 - Exprime le fait qu'un Use Case peut être étendu (augmenté) par un autre Use Case



[79]

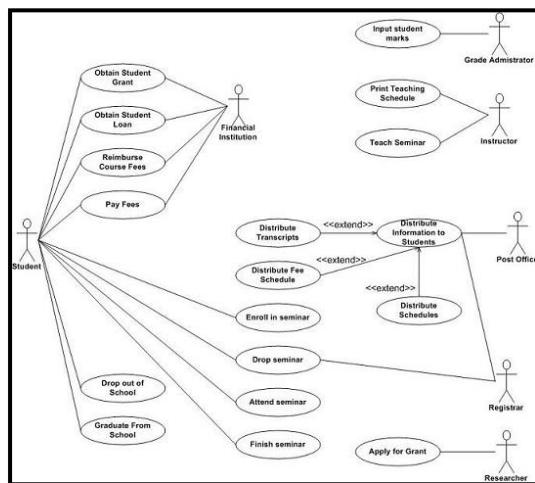
Les Notations Utilisées pour Modéliser des Use Cases

| « Extend » | « Include » |
|---|---|
| Inclusion d'un comportement sous conditions | Inclusion d'un comportement sans condition → toujours utilisé |
| S'ajoute à, ou remplace une partie du comportement précédent | Comportement purement additionnel |
| Demande la précision d'un point d'extension (= condition d'extension) | |

[80]

Les Notations Utilisées pour Modéliser des Use Cases

- Exemple de diagramme de Use Case



[81]

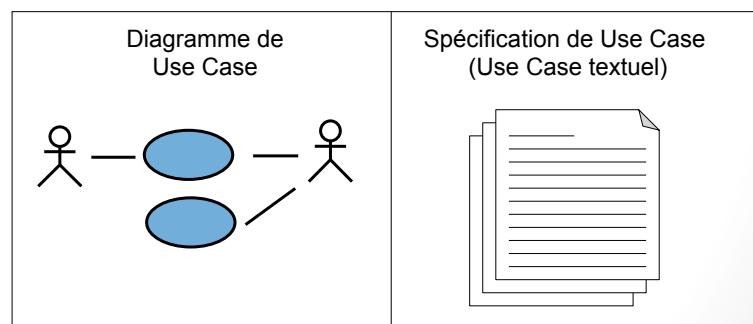
Chapitre 3: Le Diagramme de Use Cases

- Use Case: Définition et Utilité
- Les Notations Utilisées pour Modéliser des Use Cases
- **Description Textuelle des Use Cases**

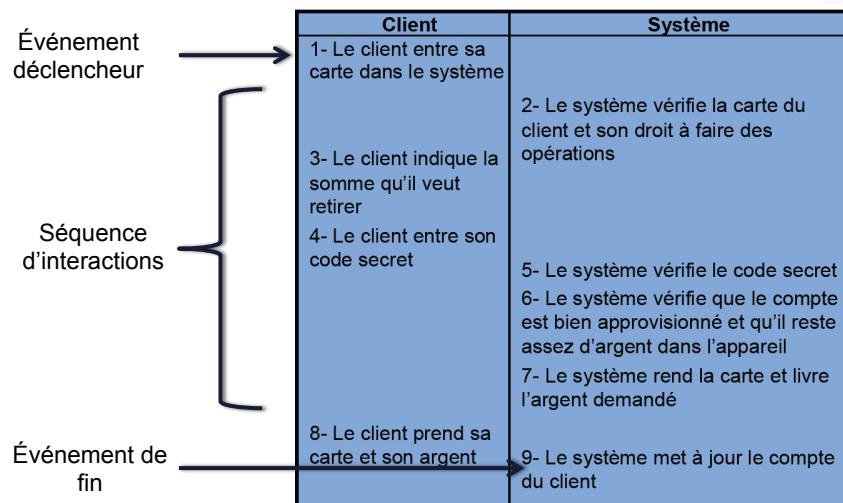
[82]

Description Textuelle des Use Cases

- Description du diagramme Use Case:
 - Use Case textuel:
 - Document regroupant les Use Cases en terme de flux d'évènements et décrivant les interactions entre les acteurs et le SI
 - Décrire le contenu de chaque Use Case:
 - Expression narrative de comment un système est utilisé



Description Textuelle des Use Cases



Description Textuelle des Use Cases

- Structure du document:
 1. Brève description générale du Use Case
 2. Cas normal
 3. Cas alternatif(s)

[85]

Description Textuelle des Use Cases

1. Brève description générale du Use Case: [texte]
 - Pré-condition(s) globale(s) : [texte]
 - Post-condition(s) globale(s) : [texte]
2. Cas normal: description [texte]
 - Pré-condition(s) : [texte]
 - Post-condition(s) : [texte]
 - Description narrative en deux colonnes: Acteur | SI

[86]

Description Textuelle des Use Cases

3. Cas alternatif(s): description [texte]

- Pré-condition(s) : [texte]
- Post-condition(s) : [texte]
- Description narrative en deux colonnes: Acteur | SI

[87]

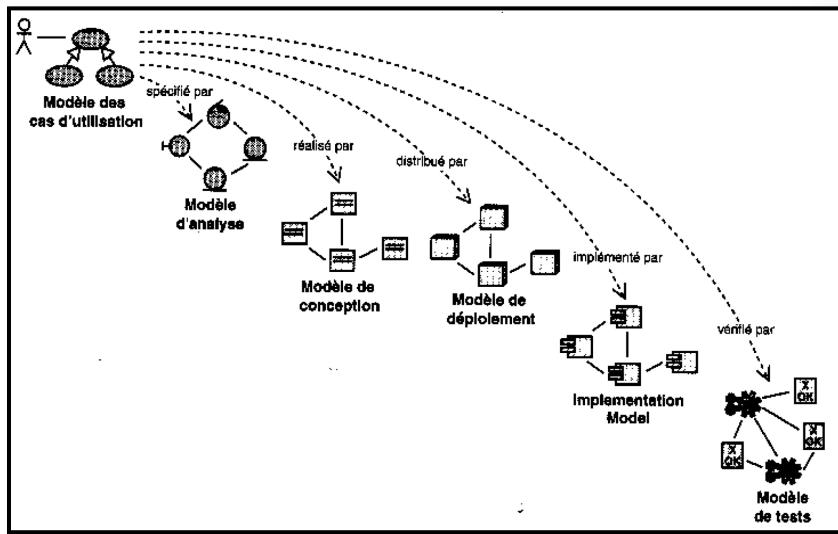
Description Textuelle des Use Cases

• Vue unifiée d'UML:

- Importance du diagramme de Use Case
- Les Use Cases garantissent la cohérence du processus de développement du SI

[88]

Description Textuelle des Use Cases



[89]

Exercices

- Exercices pour exploiter la notion de Use Case et les diagrammes de Use Case

[90]

Table des Matières

- Chapitre 0: Introduction
- Chapitre 1: Les Bases du Langage UML
- Chapitre 2: La Modélisation des Comportements
- Chapitre 3: Le Diagramme de Use Cases
- ***Chapitre 4: Le Diagramme d'Activité***
- Chapitre 5: Le Diagramme de Classes
- Chapitre 6: Les Diagrammes d'Interactions
- Chapitre 7: Le Diagramme d'État

[91]

CHAPITRE 4: LE DIAGRAMME D'ACTIVITE

[92]

Chapitre 4: Le Diagramme d'Activité

- Le Diagramme d'Activité: Définition et Utilité
- Les Notations Utilisées dans le Diagramme d'Activité
- Les Swimlanes et les Flux d'un Diagramme d'Activité
- Autres Notations

[93]

Chapitre 4: Le Diagramme d'Activité

- ***Le Diagramme d'Activité: Définition et Utilité***
- Les Notations Utilisées dans le Diagramme d'Activité
- Les Swimlanes et les Flux d'un Diagramme d'Activité
- Autres Notations

[94]

Le Diagramme d'Activité: Définition et Utilité

- Activité:
 - Une séquence d'actions capturant un comportement
- Diagramme d'Activité:
 - Modélise la partie dynamique (c.à.d. le comportement) de l'exécution d'une activité organisationnelle (e.g. workflow, business process) et/ou informatique (e.g. un algorithme)

[95]

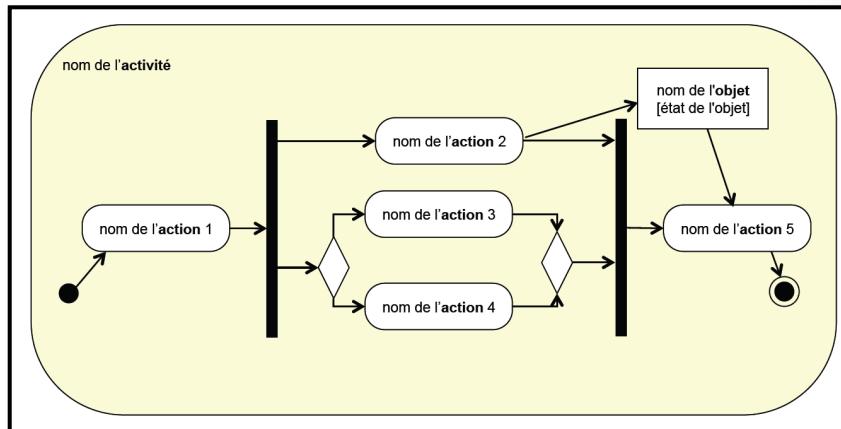
Le Diagramme d'Activité: Définition et Utilité

- Diagramme d'Activité:
 - Contenu du Diagramme d'Activité:
 - Les pas d'exécution (step)
 - Les relations précédentes et concurrentes
 - (L'acteur responsable d'un pas d'exécution)
 - (Les données/objets transmis, lus, modifiés,...)
 - Utilité du Diagramme d'Activité:
 - Analyser et modéliser les processus métiers d'une organisation pour les informatiser
 - Détailer et factoriser les Use Cases

[96]

Le Diagramme d'Activité: Définition et Utilité

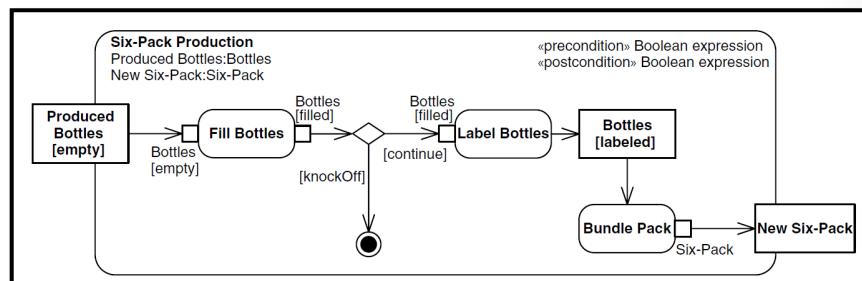
- Exemple:



[97]

Le Diagramme d'Activité: Définition et Utilité

- Exemple:



[98]

Chapitre 4: Le Diagramme d'Activité

- Le Diagramme d'Activité: Définition et Utilité
- ***Les Notations Utilisées dans le Diagramme d'Activité***
- Les Swimlanes et les Flux d'un Diagramme d'Activité
- Autres Notations

[99]

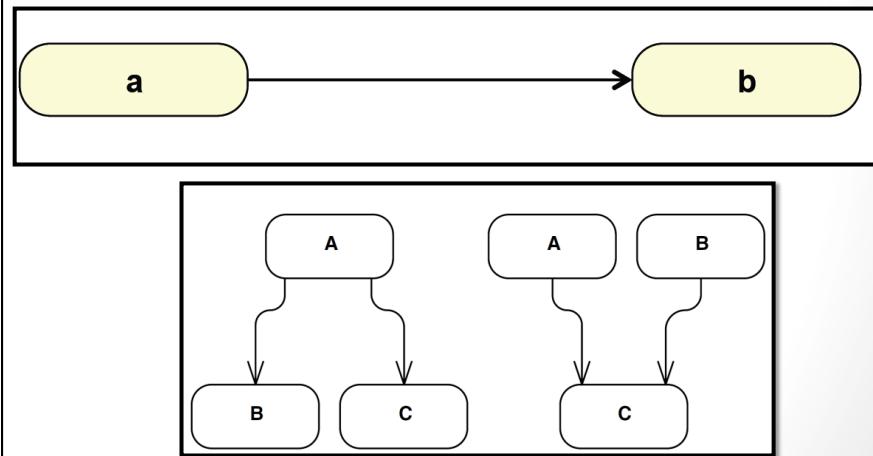
Les Notations Utilisées dans le Diagramme d'Activité

- Les Arcs (Edges):
 - Arcs de Flux:
 - Arcs d'Activités:
 - Condition de présence entre deux nœuds
 - Arcs d'Objets:
 - Un objet est fourni en sortie d'une action pour être (éventuellement) utilisé par l'action suivante
 - 3 notations possibles

[100]

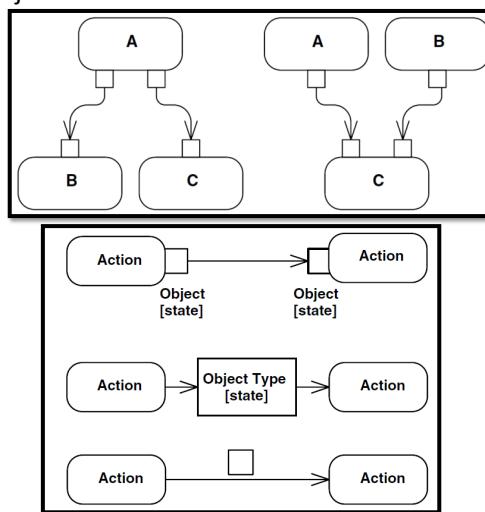
Les Notations Utilisées dans le Diagramme d'Activité

- Les Arcs de Flux:



Les Notations Utilisées dans le Diagramme d'Activité

- Les Arcs d'Objets:



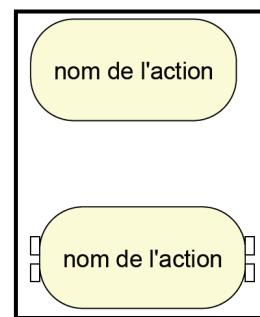
Les Notations Utilisées dans le Diagramme d'Activité

- Les Nœuds (Nodes):
 - Nœuds d'Action/d'Exécution
 - Nœuds de Contrôle
 - Nœuds d'Objets

[103]

Les Notations Utilisées dans le Diagramme d'Activité

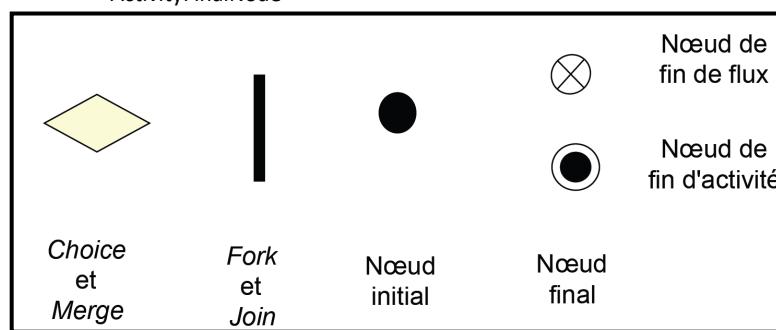
- Les Nœuds d'Action/d'Exécution:
 - Nœud fondamental d'exécution d'une activité
 - Avec ou sans *InputPin*/*OutputPin* (→ transmission d'objets)



[104]

Les Notations Utilisées dans le Diagramme d'Activité

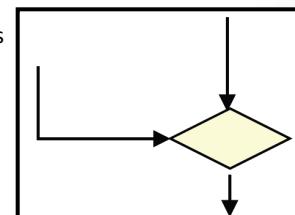
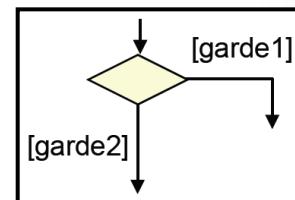
- Les Nœuds de Contrôle:
 - Dirige les objets/données à travers le diagramme d'Activité:
 - DecisionNode
 - MergeNode
 - InitialNode
 - ActivityFinalNode



[105]

Les Notations Utilisées dans le Diagramme d'Activité

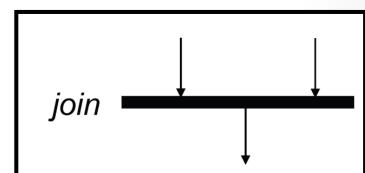
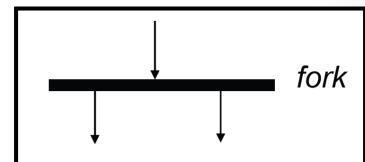
- Les Nœuds de Contrôle:
 - Représentation du 'Ou' (exclusif)
 - Choice Node:
 - Création de branchements conditionnels
 - 1 seul arc en entrée
 - ≥ 2 arcs de sortie
 - Indication des gardes:
 - [Condition]
 - [Else]
 - Merge Node:
 - Permet de fusionner différentes branches issues d'un choice:
 - ≥ 2 arcs en entrée
 - 1 seul arc en sortie



[106]

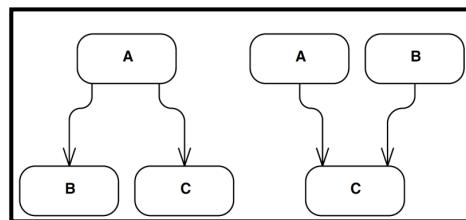
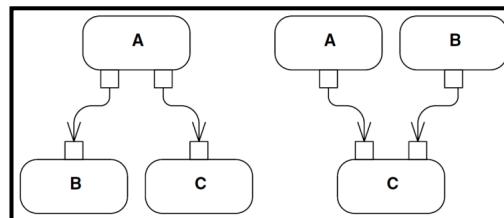
Les Notations Utilisées dans le Diagramme d'Activité

- Les Nœuds de Contrôle:
 - Représentation du 'Et'
 - Fork Node:
 - Création de deux flux concurrents
 - 1 arc en entrée
 - ≥ 2 arcs de sortie
 - Les flux continuent en parallèle
 - Join Node:
 - Permet de synchroniser plusieurs flux concurrents
 - ≥ 2 arcs en entrée
 - 1 seul arc en sortie



Les Notations Utilisées dans le Diagramme d'Activité

- Les Nœuds de Contrôle:



[107]

[108]

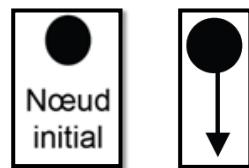
Les Notations Utilisées dans le Diagramme d'Activité

- Les Nœuds de Contrôle:
 - Nœud Initial
 - Nœud Final
 - Nœud de Fin de Flux

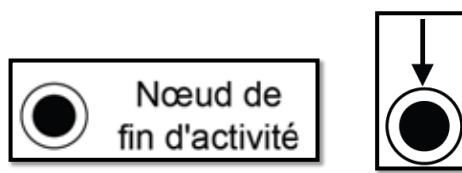
[109]

Les Notations Utilisées dans le Diagramme d'Activité

- Les Nœuds de Contrôle:
 - Nœud Initial:
 - Indique le point de départ (unique) d'une activité



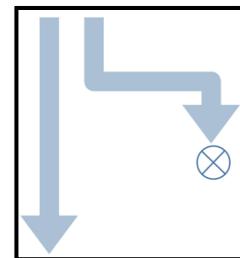
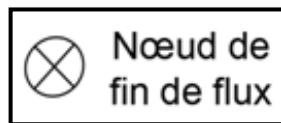
- Nœud Final:
 - Indique la fin d'une activité, c'est-à-dire, la fin de tous ses flux



[110]

Les Notations Utilisées dans le Diagramme d'Activité

- Les Nœuds de Contrôle:
- Nœud de Fin de Flux:
 - Indique la fin d'un flux
 - D'autres flux concurrents peuvent subsister et ainsi continuer l'activité



[111]

Les Notations Utilisées dans le Diagramme d'Activité

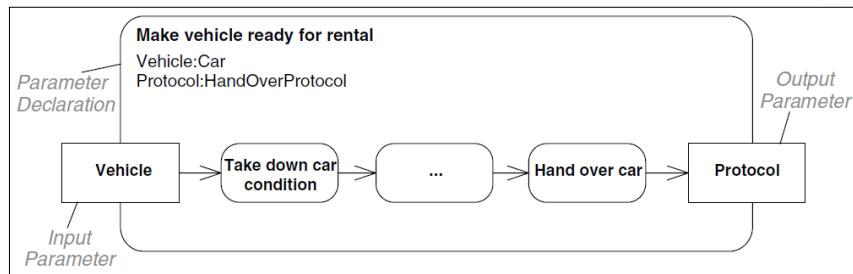
- Les Nœuds d'Objets:
 - Indique les objets et les données circulant dans le diagramme d'activité
 - Se note sur les arcs
 - Peut déclencher une activité

Nom de l'objet
[état de l'objet]

[112]

Les Notations Utilisées dans le Diagramme d'Activité

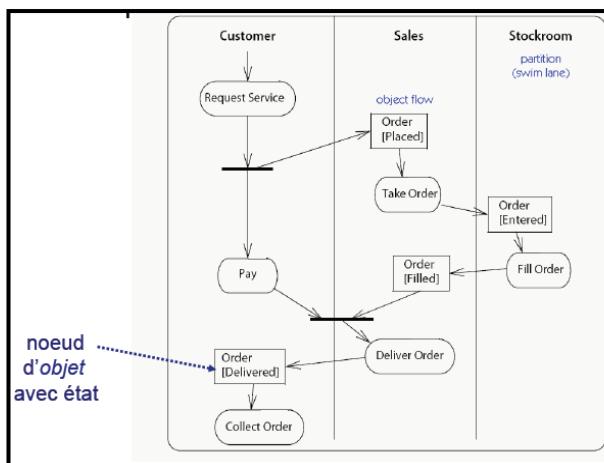
- Les Nœuds d'Objets:



[113]

Les Notations Utilisées dans le Diagramme d'Activité

- Les Nœuds d'Objets:



[114]

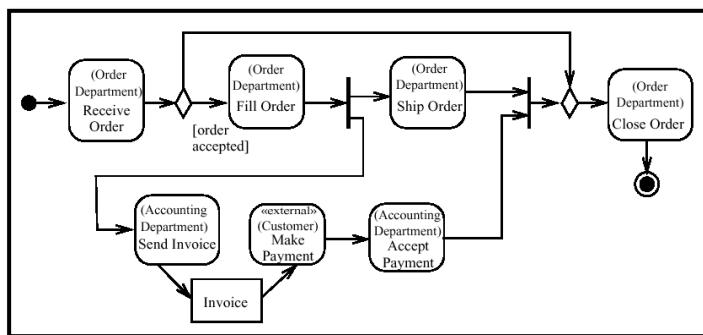
Chapitre 4: Le Diagramme d'Activité

- Le Diagramme d'Activité: Définition et Utilité
- Les Notations Utilisées dans le Diagramme d'Activité
- ***Les Swimlanes et les Flux d'un Diagramme d'Activité***
- Autres Notations

[115]

Les Swimlanes et les Flux d'un Diagramme d'Activité

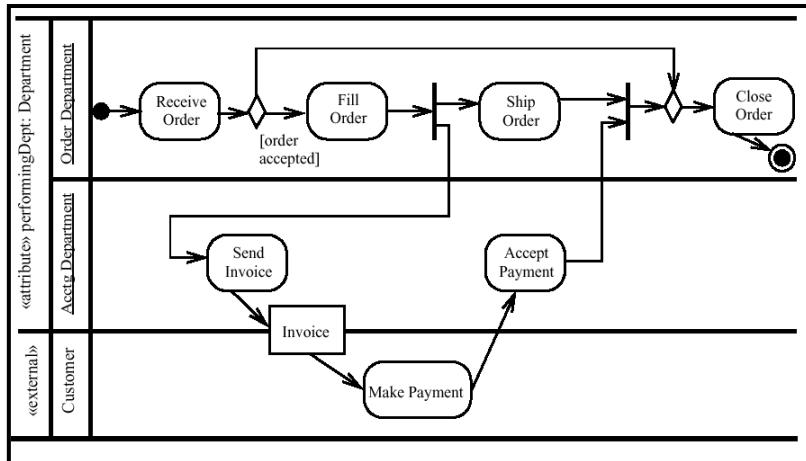
- Partitions (Swimlanes):
 - Utilité:
 - Séparer un diagramme d'activité par acteur responsable des actions à réaliser



[116]

Les Swimlanes et les Flux d'un Diagramme d'Activité

- Partitions (Swimlanes):



[117]

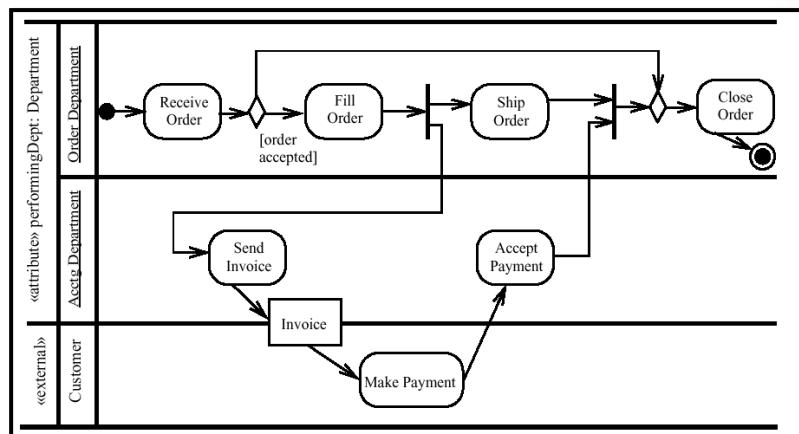
Les Swimlanes et les Flux d'un Diagramme d'Activité

- Flux et Tokens d'un Diagramme d'Activité (Sémantique):
 - Flux: déplacement de Tokens à travers les arcs et les nœuds du diagramme

[118]

Les Swimlanes et les Flux d'un Diagramme d'Activité

- Flux et Tokens d'un Diagramme d'Activité (Sémantique):



[119]

Les Swimlanes et les Flux d'un Diagramme d'Activité

- Propriétés des Tokens:
- Une action ne peut commencer que si un Token est disponible à tous ses arcs entrants
 - Synchronisation de flux
- Lorsqu'une action est terminée, un Token est libéré pour tous ses arcs sortants
 - Séparation en flux concurrents

[120]

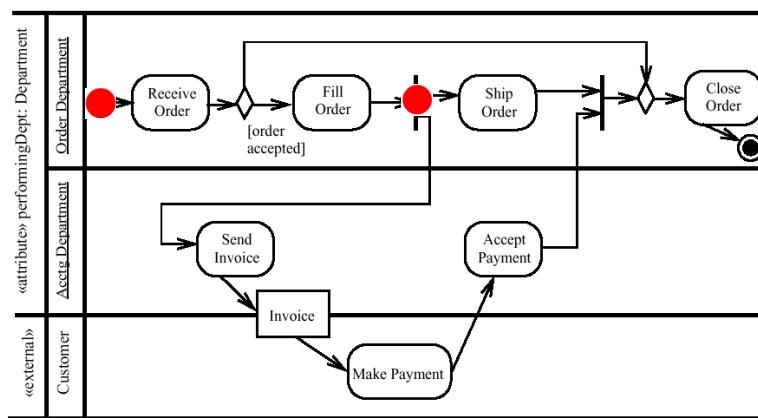
Les Swimlanes et les Flux d'un Diagramme d'Activité

- Propriétés des Tokens:
 - Un Token parcourt un arc d'une action vers la suivante aux conditions que:
 1. le Token est rendu libre par la première action de la transition
 2. l'action suivante est prête à prendre en compte le Token
 3. l'éventuelle garde (règle) indiquée sur l'arc admet la transition du Token
 - Token: transcrit les sémantiques 'And' et 'Or' (Fork / Choice)

[121]

Les Swimlanes et les Flux d'un Diagramme d'Activité

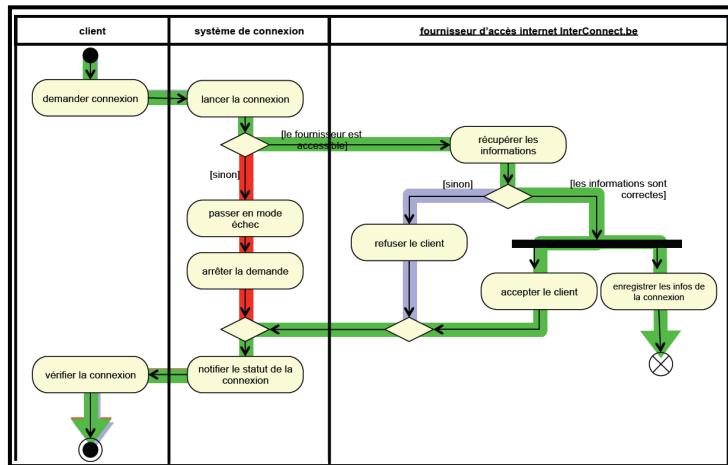
- Flux et Tokens d'un Diagramme d'Activité: Illustration du fonctionnement des Tokens



[122]

Les Swimlanes et les Flux d'un Diagramme d'Activité

- Les Flux:



[123]

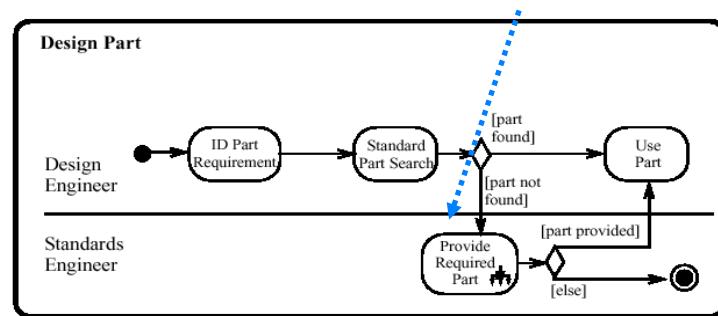
Chapitre 4: Le Diagramme d'Activité

- Le Diagramme d'Activité: Définition et Utilité
- Les Notations Utilisées dans le Diagramme d'Activité
- Les Swimlanes et les Flux d'un Diagramme d'Activité
- **Autres Notations**

[124]

Autres Notations

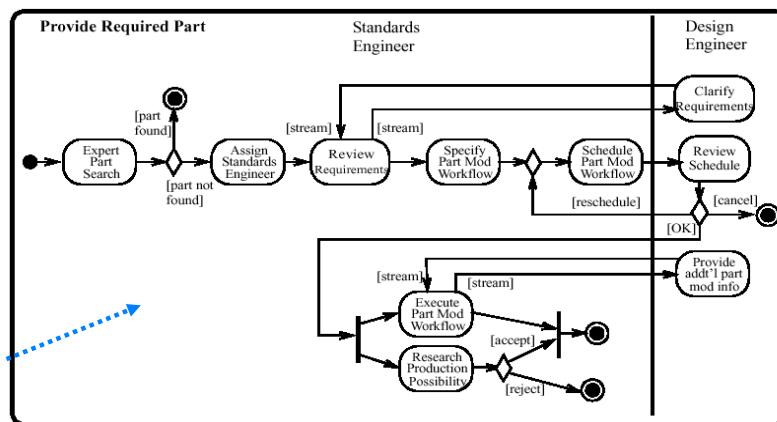
- Sous-Activité:
 - Utilité: représentation du diagramme d'activité au bon niveau d'abstraction (pas trop de détails, pas trop complexe)



[125]

Autres Notations

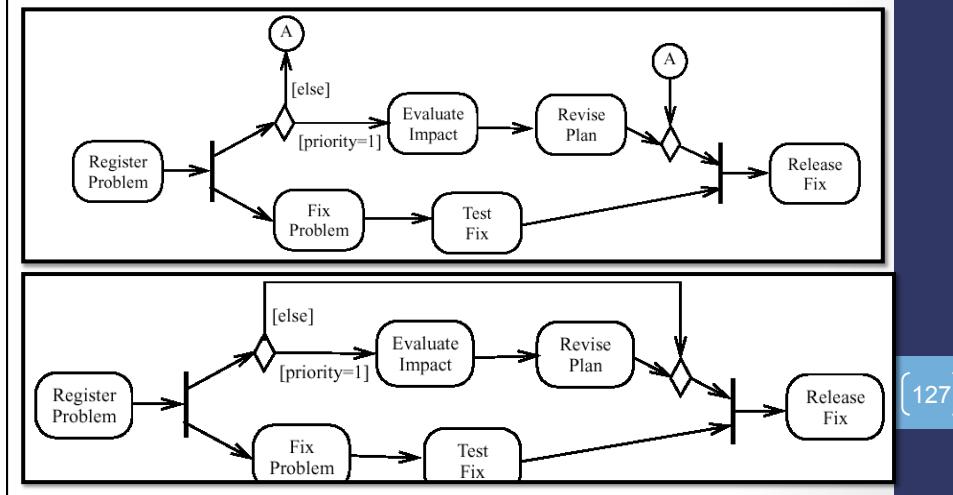
- Sous-Activité:
 - Utilité: représentation du diagramme d'activité au bon niveau d'abstraction (pas trop de détails, pas trop complexe)



[126]

Autres Notations

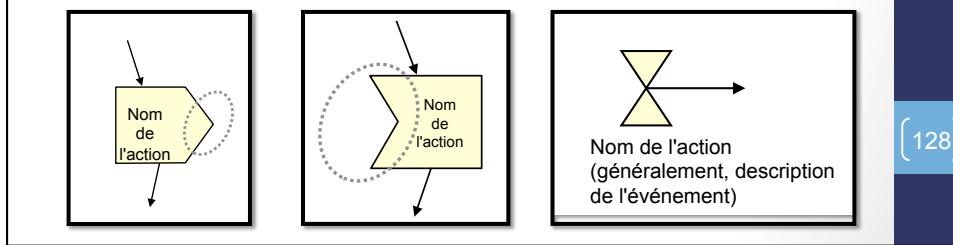
- Connecteur:



Autres Notations

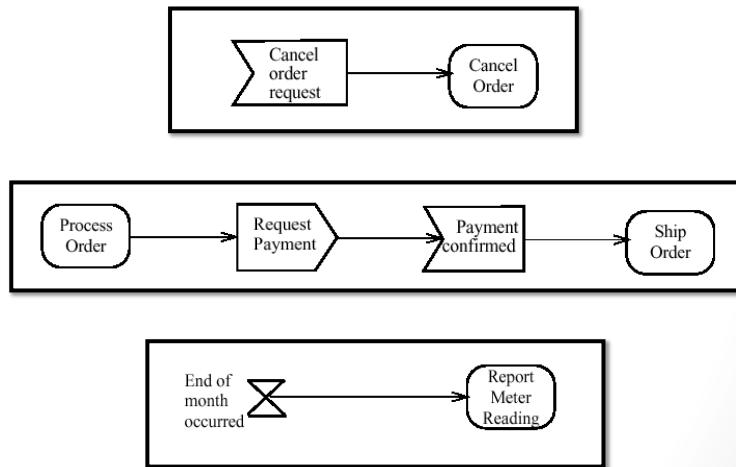
- Signaux:

- Action d'envoi de signal:
 - Sert à déclencher le comportement défini à la suite d'une action de réception de signal dans un diagramme d'activité
- Action de réception de signal:
 - Peut ne pas avoir d'arc entrant
 - Action de réception de signal temporel



Autres Notations

- Exemples de signaux:



[129]

Autres Notations

- Nœuds structurés:
 - Nœuds contenant un sous-ensemble des nœuds et des arcs du diagramme d’activité
 - Sous-ensemble appelé « région »
 - 2 types:
 - Régions interruptibles:
 - Région dont on peut quitter en même temps tous les flux en cours
 - Régions d’expansion:
 - Région que l’on répète autant de fois qu'il y a d'inputs

[130]

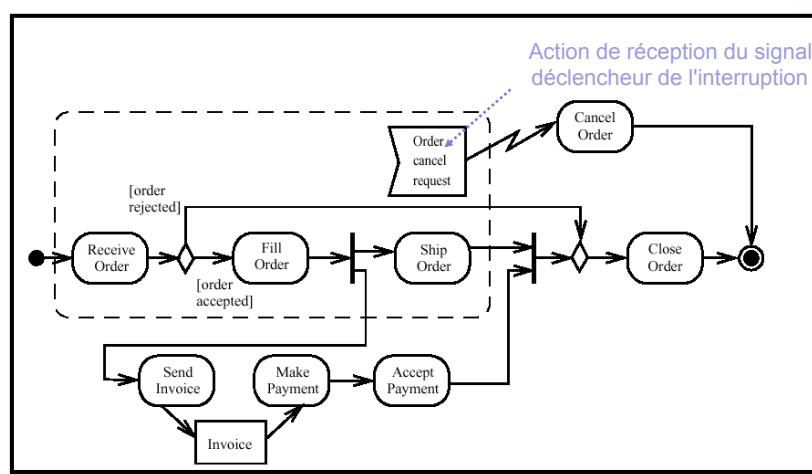
Autres Notations

- Nœuds structurés – Région interruptible:
 - Tous les jetons présents dans la région interruptible disparaissent lorsque l'interruption intervient
 - Permet la modélisation de plusieurs scénarios d'exception à la fois

[131]

Autres Notations

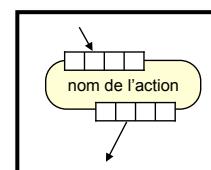
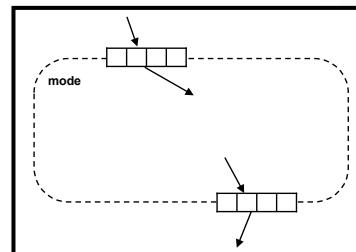
- Nœuds structurés – Région interruptible:



[132]

Autres Notations

- Nœuds structurés – Région d’expansion:
 - Mode =
 - << concurrent >> (= parallèle): les exécutions de la région peuvent se chevaucher
 - << iterative >>: les exécutions de la région se déroulent en séquence
 - Si la région ne contient qu’une seule action



[133]

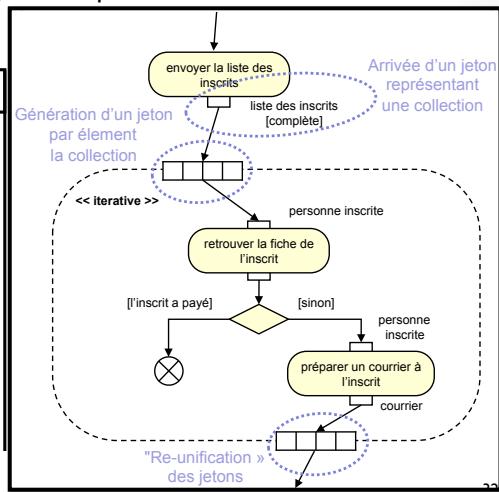
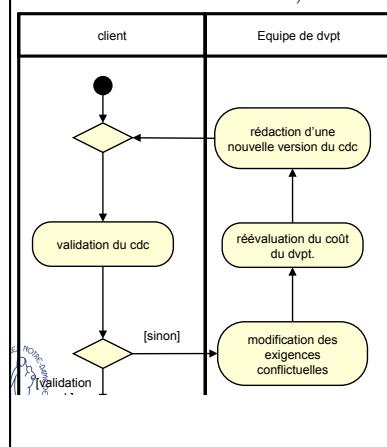
Autres Notations

- Nœuds structurés – Région d’expansion vs boucle:
 - Exemples:
 - Validation du cahier des charges avec le client:
 - Nombre d’itérations inconnu en entrant dans la boucle
 - Envoi d’un courrier aux inscrits qui n’ont pas encore payé:
 - Nombre d’itérations connu en entrant dans la « boucle »

[134]

Autres Notations

- Nœuds structurés – Région d’expansion vs boucle:
- Exemples:



[135]

Autres Notations

- Exercices sur les Diagrammes d’Activité

[136]

Table des Matières

- Chapitre 0: Introduction
- Chapitre 1: Les Bases du Langage UML
- Chapitre 2: La Modélisation des Comportements
- Chapitre 3: Le Diagramme de Use Cases
- Chapitre 4: Le Diagramme d'Activité
- ***Chapitre 5: Le Diagramme de Classes***
- Chapitre 6: Les Diagrammes d'Interactions
- Chapitre 7: Le Diagramme d'État

[137]

CHAPITRE 5: LE DIAGRAMME DE CLASSES

[138]

Chapitre 5: Le Diagramme de Classe

- ***Le Diagramme de Classe: Définition et Utilité***
- Les Classes: Définition et Structure
- Les Contraintes
- La Notion d'Objet et la Classification
- Les Caractéristiques des Objets: les Attributs et les Opérations
- Les Relations
- Les Packages et Interfaces de Classes

[139]

Le Diagramme de Classe: Définition et Utilité

- Définition:
 - Un Diagramme de Classe représente les propriétés statiques d'éléments (eg, objets) présents dans le domaine d'application, ou dans le domaine machine à construire (eg, application)
- Finalités et principaux cas d'utilisation du Diagramme de Classe:
 - Analyse et cartographie du domaine d'application
 - Conception logicielle
 - Conception de base de données
 - Méta-modélisation
- Un exemple de classes, leurs attributs et relations – Vente de matériel sportif par correspondance

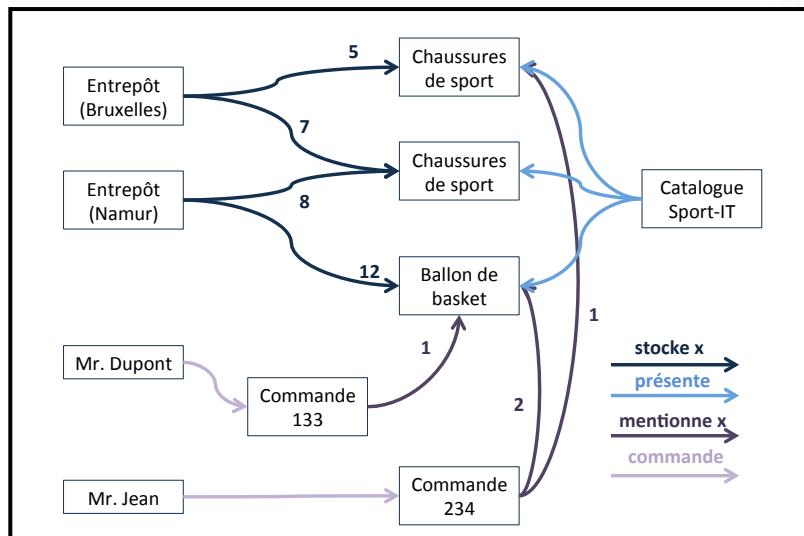
[140]

Le Diagramme de Classe: Définition et Utilité

- Un exemple de classes, leurs attributs et relations – Vente de matériel sportif par correspondance:
 - Pour les clients:
 - Accès à un catalogue décrivant les différents articles commandables
 - Tous les types d'articles disponibles sont décrits dans le catalogue, accompagnés de leur prix
 - Le stock de Bruxelles contient uniquement des chaussures (5+7)
 - Le stock de Namur possède des chaussures bleues (8) ainsi que des ballons de basket (12)
 - (Le domaine d'application d'un commerce par correspondance est évidemment plus étendu : fournisseurs, service de livraison et tarification, fichiers clients,...)

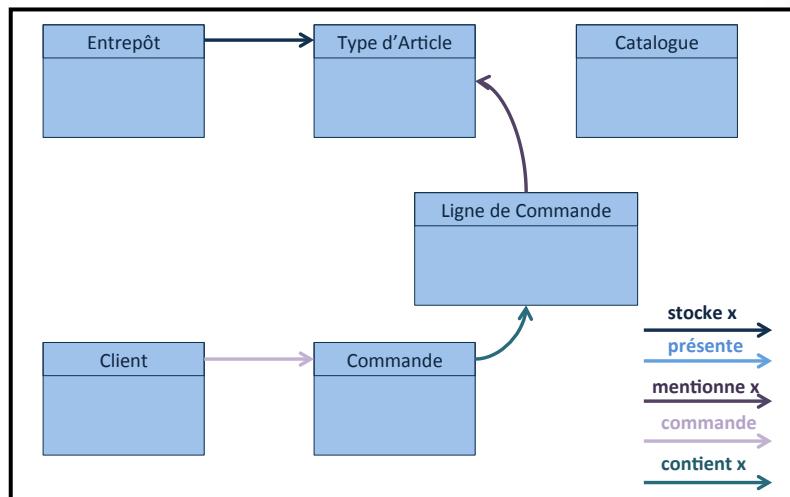
[141]

Le Diagramme de Classe: Définition et Utilité



[142]

Le Diagramme de Classe: Définition et Utilité



[143]

Chapitre 5: Le Diagramme de Classe

- Le Diagramme de Classe: Définition et Utilité
- ***Les Classes: Définition et Structure***
- Les Contraintes
- La Notion d'Objet et la Classification
- Les Caractéristiques des Objets: les Attributs et les Opérations
- Les Relations
- Les Packages et Interfaces de Classes

[144]

Les Classes: Définition et Structure

- La notion de Classe:
 - Une classe:
 - Représente un concept générique
 - Mêmes caractéristiques
 - Mêmes contraintes
 - Même sémantique
 - Reprend les propriétés communes d'un ensemble d'éléments
 - Donne le domaine de définition (typage) d'un ensemble d'objets
 - Objectif des classes:
 - Classer l'ensemble des instances par rapport à leurs caractéristiques

| Chien |
|--------------------|
| +Date_de_naissance |
| +Pédigree |
| +Sexe |

(145)

Les Classes: Définition et Structure

- La notion de Classe: Structure d'une Classe

Comportiment de la classe

- Nom de la classe
- Les **attributs** de la classe
- Les **opérations** effectuées sur les objets de la classe

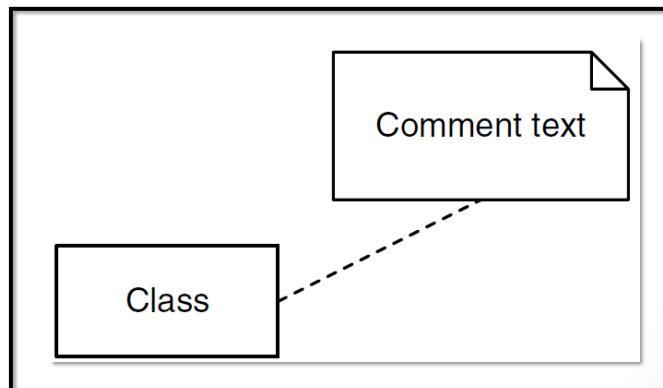
Classe

| |
|---|
| [vis] Attribut_1 [: type] [mult] [= valeur initiale] [[contr]] [vis] Attribut_2 [: type] [mult] [= valeur initiale] [[contr]] ... [vis] Attribut_m [: type] [mult] [= valeur initiale] [[contr]] |
| [vis] Opération_1 ([par:]type,..., [par:]type) [: type] [vis] Opération_2 ([par:]type,..., [par:]type) [: type] ... [vis] Opération_n ([par:]type,..., [par:]type) [: type] |

(146)

Les Classes: Définition et Structure

- Commenter une classe:
 - Tout élément UML peut être commenté



[147]

Chapitre 5: Le Diagramme de Classe

- Le Diagramme de Classe: Définition et Utilité
- Les Classes: Définition et Structure
- ***Les Contraintes***
- La Notion d'Objet et la Classification
- Les Caractéristiques des Objets: les Attributs et les Opérations
- Les Relations
- Les Packages et Interfaces de Classes

[148]

Les Contraintes

- Une contrainte est une expression qui restreint le domaine de valeur d'un élément (pouvant être une relation):
 - Langage naturel (LN)
 - Langage Semi-formel
 - Langage Formel
- Notation graphique UML: le XOR
 - Modélise le 'Ou' exclusif

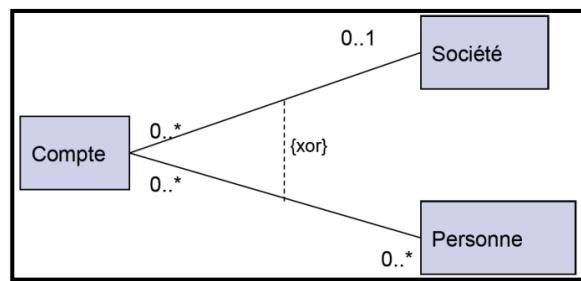
[149]

Les Contraintes

- Pour les autres contraintes:


```
{[<name> : ] < Boolean expression > }
```

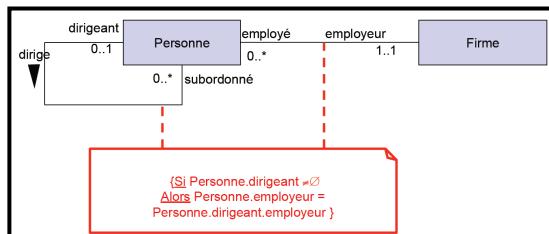
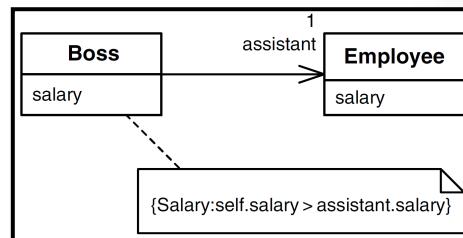
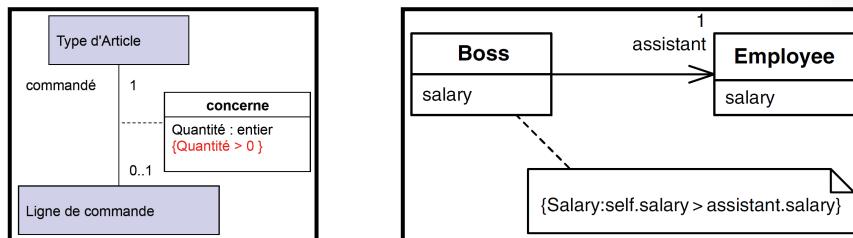
 - Utiliser un commentaire ("post-it")
 - À la suite de la définition d'un élément



[150]

Les Contraintes

- Exemples:



[151]

Chapitre 5: Le Diagramme de Classe

- Le Diagramme de Classe: Définition et Utilité
- Les Classes: Définition et Structure
- Les Contraintes
- La Notion d'Objet et la Classification***
- Les Caractéristiques des Objets: les Attributs et les Opérations
- Les Relations
- Les Packages et Interfaces de Classes

[152]

La Notion d'Objet et la Classification

- Les Objets (ou Instances):
 - Une instance/un objet:
 - Élément concret présent dans le domaine du SI modélisé (domaine d'application ou machine)
 - Rappel: UML entre dans le paradigme orienté objets

[153]

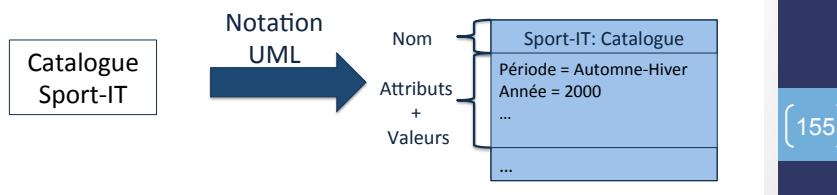
La Notion d'Objet et la Classification

- Les Objets (ou Instances):
 - Une instance/un objet dispose de:
 - Une identité: c'est une "chose" ayant une existence propre dans le monde modélisé (application ou machine)
 - (Souvent) différent(e)s actions ou événements
 - Une durée de vie: "chose" qui existe pendant un certain temps
 - Un état: à un moment donné, l'objet a des valeurs pour des caractéristiques propres
 - Une pertinence et une signification acceptées dans le domaine (décrise dans le lexique)

[154]

La Notion d'Objet et la Classification

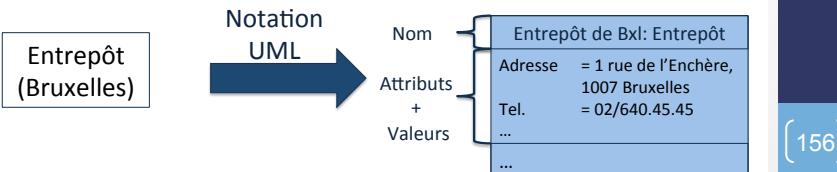
- Exemple d'Objets:
 - Identité: Sport-IT, collection automne-hiver 2000
 - Durée de vie: [conception du catalogue, ∞[
 - État: valeur des attributs + relations avec autres objets
 - Pertinence: présentation des différents articles (+ prix) aux clients. La période de validité: [septembre 2000, février 2001]



[155]

La Notion d'Objet et la Classification

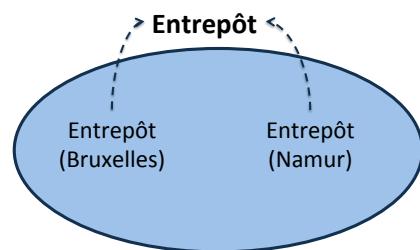
- Exemple d'Objets:
 - Identité: Entrepôt de Bruxelles
 - Durée de vie: tant que cet entrepôt existe et est utilisé
 - État: attributs + relations avec autres objets (ensemble des couples <article, quantité> entreposés)
 - Pertinence: sert à entreposer les différents articles



[156]

La Notion d'Objet et la Classification

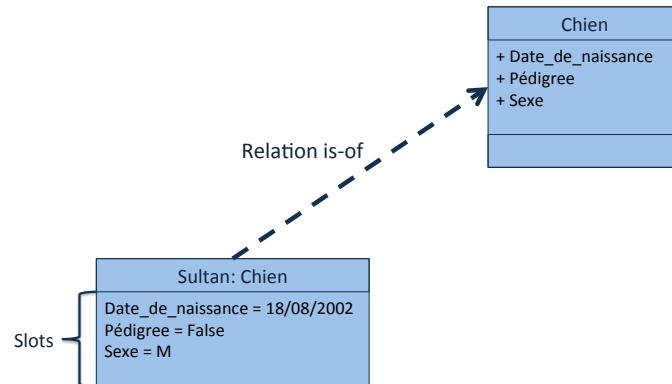
- La Classification des Objets:
 - Les classes décomposent le monde modélisé en ensembles d'objets ayant des caractéristiques communes
 - Mécanisme d'abstraction



[157]

La Notion d'Objet et la Classification

- La Classification d'un Objet: spécification d'une instance



[158]

La Notion d'Objet et la Classification

- La granularité d'une classification:
 - Échelle définissant le niveau de détails d'un élément ou ensemble d'éléments
- Les classes:
 - Représentation d'ensembles d'éléments ayant des caractéristiques communes, mais des différences subsistent entre eux
 - Définissent la granularité intéressante en fonction de la vue du monde modélisé dans chaque classe

[159]

Chapitre 5: Le Diagramme de Classe

- Le Diagramme de Classe: Définition et Utilité
- Les Classes: Définition et Structure
- Les Contraintes
- La Notion d'Objet et la Classification
- ***Les Caractéristiques des Objets: les Attributs et les Opérations***
- Les Relations
- Les Packages et Interfaces de Classes

[160]

Les Caractéristiques des Objets: les Attributs et les Opérations

- Attribut:
 - Caractéristique partagée par (tous?) les objets d'une classe
 - Syntaxe d'un attribut ([optionnel]):
**[visibility][/]name[:type][multiplicity][=initial value]
[{property string}]**

[161]

Les Caractéristiques des Objets: les Attributs et les Opérations

- Propriété d'un attribut:
 - Visibilité (utilisé uniquement lors de la représentation d'éléments logiciels)
 - -: private (visible uniquement par les méthodes de la classe)
 - +: public (visible partout à l'intérieur et à l'extérieur de la classe)
 - ~: package (visible au sein du package uniquement)
 - #: protected (visible au sein de la classe et par les descendants de cette classe)

[162]

Les Caractéristiques des Objets: les Attributs et les Opérations

- Propriété d'un attribut:
 - Type: spécification de l'ensemble des valeurs pour l'élément typé (= attribut)
 - De base (e.g., int, String, boolean,...)
 - Défini par l'utilisateur

[163]

Les Caractéristiques des Objets: les Attributs et les Opérations

- Propriété d'un attribut:
 - Multiplicité: définition d'un intervalle d'éléments spécifiant une cardinalité
 - $[min.. max]$: eg, $[0.. 1]$, $[3.. 3] = 3$, $[0.. *] = *, \dots$
 - Par défaut: attribut monovalué obligatoire, soit $[1]$
 - Si $min = 0$: attribut facultatif
 - Si $max > 1$: attribut multivalué

[164]

Les Caractéristiques des Objets: les Attributs et les Opérations

- Propriété d'un attribut:
 - Propriétés:
 - {readonly}: L'attribut peut être lu, mais pas modifié
 - {union}: L'attribut est l'union de sous-ensemble d'attributs
 - {ordered}, {unordered}: L'ensemble est ordonné ou non
 - {unique}, {nonunique}, or {bag}: Un ensemble d'attribut peut, ou ne peut pas contenir plusieurs éléments identiques
 - {sequence}: Une liste ordonnancée d'éléments
 - {composite}: L'attribut est le composant d'un autre attribut
 - ...

[165]

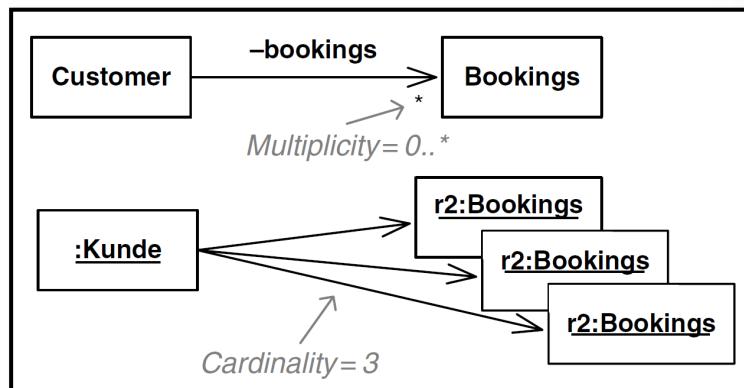
Les Caractéristiques des Objets: les Attributs et les Opérations

- Attributs de classe vs Attributs d'instance
 - Attribut d'instance: la valeur évolue indépendamment pour chaque objet de la classe
 - Attribut de classe: une valeur unique qui est identique pour tous les objets d'une classe donnée
- Multiplicité vs. Cardinalité
 - Multiplicité: définition d'un intervalle d'éléments spécifiant une cardinalité
 - Cardinalité: nombre d'éléments concrets dans un ensemble

[166]

Les Caractéristiques des Objets: les Attributs et les Opérations

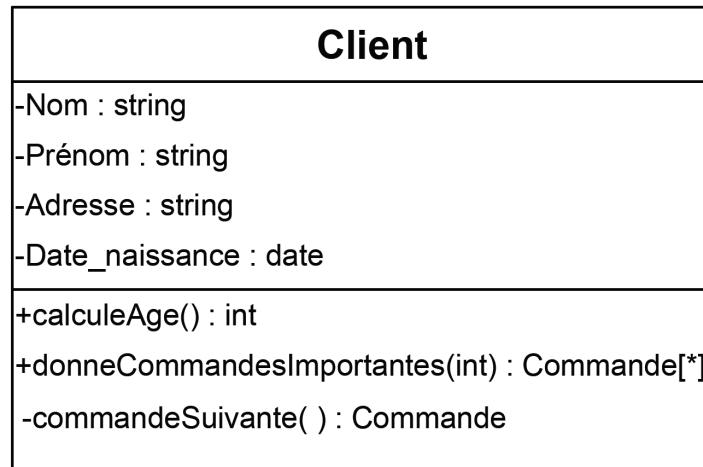
- Multiplicité vs. Cardinalité



[167]

Les Caractéristiques des Objets: les Attributs et les Opérations

- Les opérations: exemple



[168]

Les Caractéristiques des Objets: les Attributs et les Opérations

- Les opérations
 - Une opération représente un service/une fonction permis par les instances de la classe
 - Syntaxe:
[visibility] name (parameter list) [: type] [{property string}]
 - Caractéristiques
 - Visibilité: idem attribut (-, #, +, ~)
 - Paramètres et leur type, ainsi que le résultat et son type

[169]

Les Caractéristiques des Objets: les Attributs et les Opérations

- Les opérations
 - Pré-condition(s), post-condition(s) et body conditions (contraintes spécifiées par "post-it")
 - Certaines opérations demandent la définition d'une méthode (décrit la manière de réaliser/implémenter l'opération)
 - Diagrammes d'activités, diagrammes d'états, langage de programmation (hors UML), pseudo-code (hors UML),...

[170]

Chapitre 5: Le Diagramme de Classe

- Le Diagramme de Classe: Définition et Utilité
- Les Classes: Définition et Structure
- Les Contraintes
- La Notion d'Objet et la Classification
- Les Caractéristiques des Objets: les Attributs et les Opérations
- ***Les Relations***
- Les Packages et Interfaces de Classes

[171]

Les Relations

- La notion de Relation:
 - La relation est un concept abstrait permettant de modéliser un lien et structurer les concepts entre eux
 - Un lien peut exister:
 - Entre des objets du domaine d'application et/ou du monde machine, et
 - Entre éléments dans un métamodèle
 - Les deux éléments/objets mis en relation ont soit un rôle "target" (*client*) ou un rôle "source" (*supplier*)
 - Quatre types de relations:
 - L'association
 - L'agrégation et la composition
 - La généralisation/spécialisation
 - La dépendance

[172]

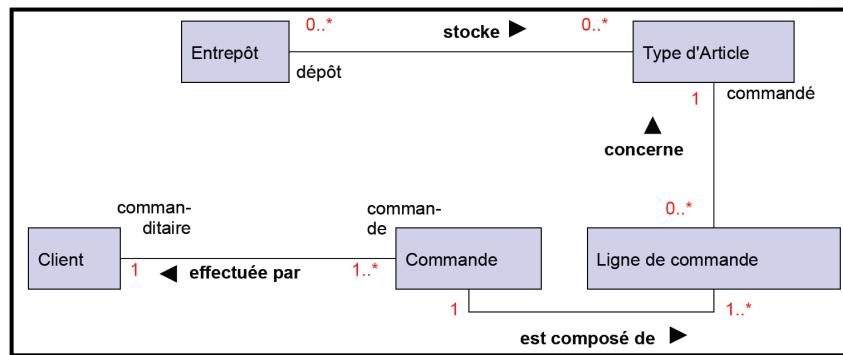
Les Relations

- L'Association:
 - Une association décrit un lien particulier entre des objets
 - Caractéristiques:
 - Nom
 - Une multiplicité: définition d'un intervalle d'entiers positifs pour la détermination des cardinalités autorisées
 - Optionnel [0.. x]
 - Obligatoire [1..x]
 - Rôle (optionnel)
 - Nom
 - Visibilité (-, #, ~, +)

[173]

Les Relations

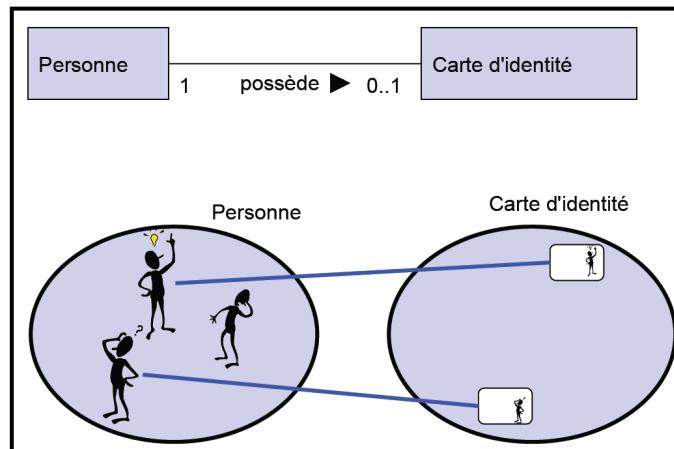
- L'Association:



[174]

Les Relations

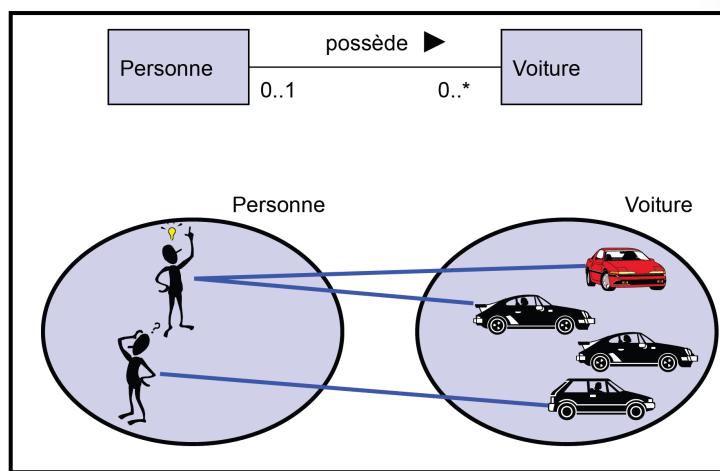
- Association « one-to-one »



[175]

Les Relations

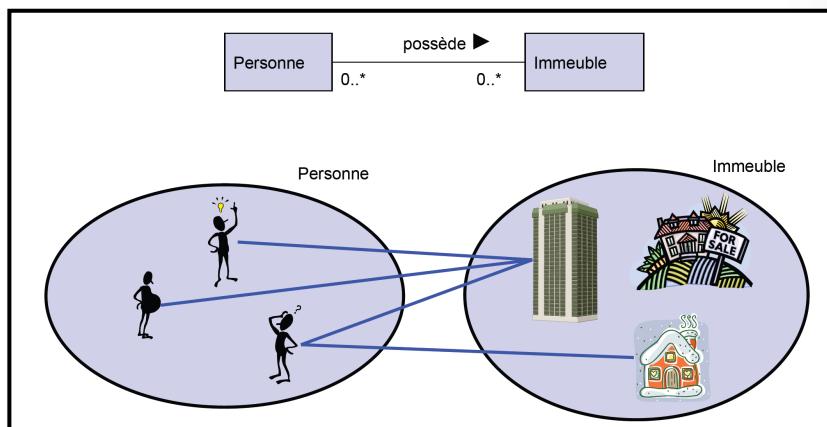
- Association « one-to-many »



[176]

Les Relations

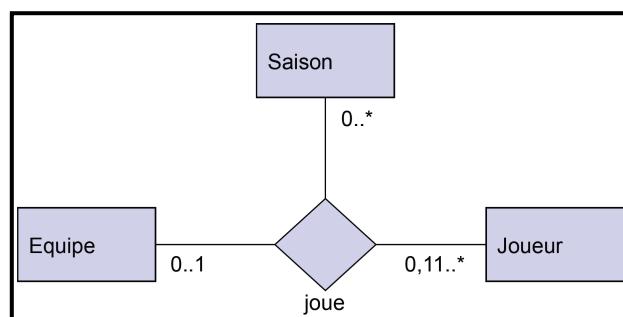
- Association « many-to-many »



[177]

Les Relations

- Association « n-aire »
 - Le diamant est le point de connexion entre les différents rôles de l'association



[178]

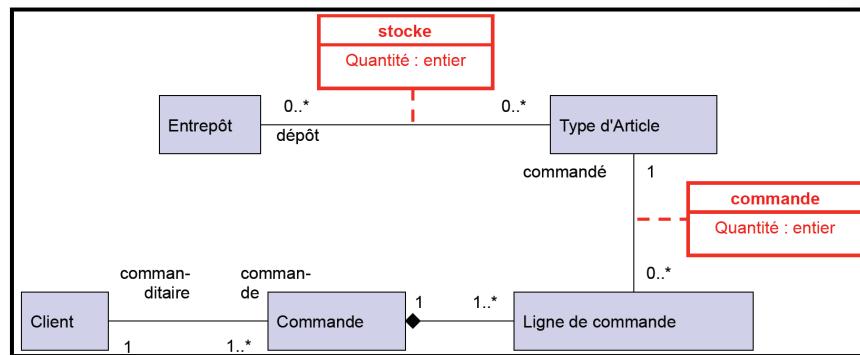
Les Relations

- Les Classes d'Association
 - Une classe d'association est une classe dont les entités sont des relations du monde réel → permet de rajouter des attributs à des associations:
 - Dénomination identique à l'association
 - Multiplicité implicite entre la classe d'association et l'association: one-to-one
 - Association binaire et n-aire

[179]

Les Relations

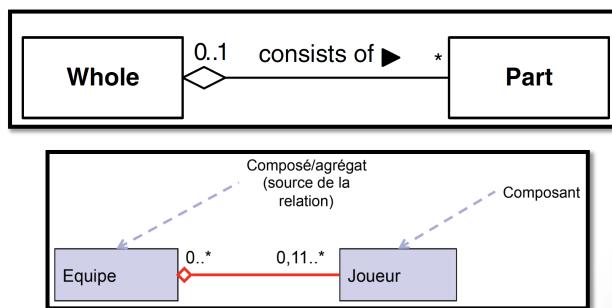
- Les Classes d'Association



[180]

Les Relations

- L’Agrégation
 - Une agrégation est une association particulière
 - Indique que les instances d’une classe sont les agrégats d’instances de l’autre classe
 - Le tout et la partie



[181]

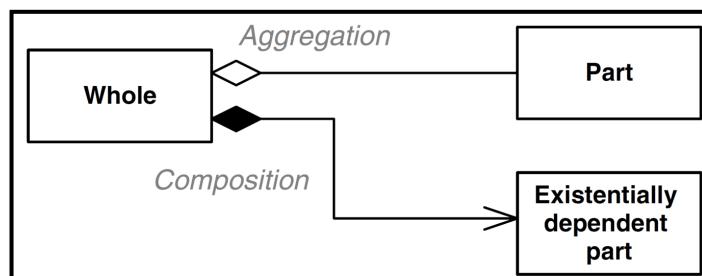
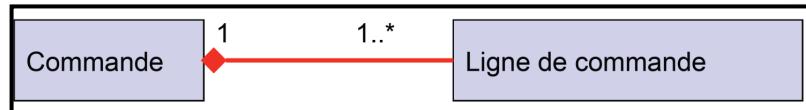
Les Relations

- La Composition:
 - La composition est une forme très forte d’agrégation
 - L’existence du composant dépend strictement de l’existence du composé/agrégat
 - La multiplicité du côté du composé vaut 1
 - Le composant compose obligatoirement un seul objet
 - Le composant ne peut être impliqué que dans une seule composition

[182]

Les Relations

- La Composition



[183]

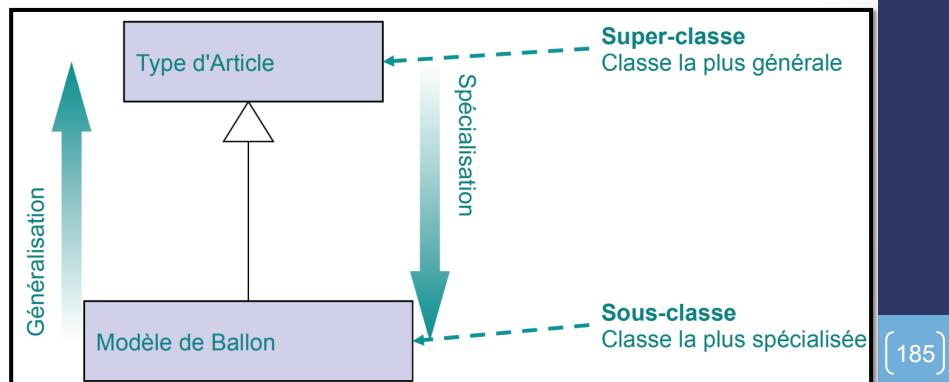
Les Relations

- La Généralisation et la Spécialisation:
 - La relation de généralisation/spécialisation est une relation de hiérarchisation des classes d'entités
 - Gestion de la complexité
 - Plusieurs niveaux d'abstraction (de granularité)
 - Classification des objets dans différents classes positionnées dans différents niveaux hiérarchiques

[184]

Les Relations

- La Généralisation et la Spécialisation:



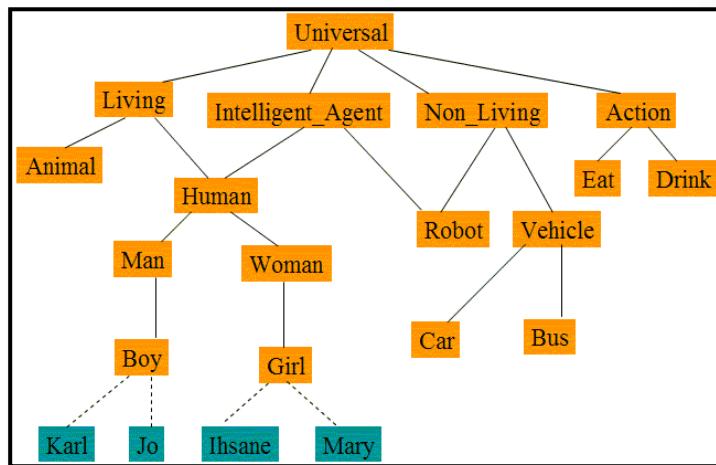
Les Relations

- La Généralisation et la Spécialisation:
 - Spécialisation:
 - Ajout de caractéristiques *spécifiques* aux sous-classes
 - Généralisation:
 - Factorisation de caractéristiques communes (abstraction des détails)
 - Inclusion ensembliste

[186]

Les Relations

- La Généralisation et la Spécialisation:



(187)

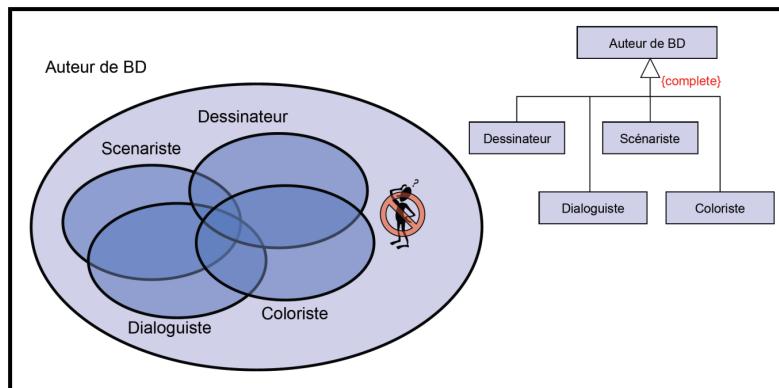
Les Relations

- Complet vs. Incomplet:
 - Complet:
 - Toutes les entités de la super classe sont spécialisées
 - Incomplet:
 - Des entités peuvent n'appartenir à aucune sous-classe

(188)

Les Relations

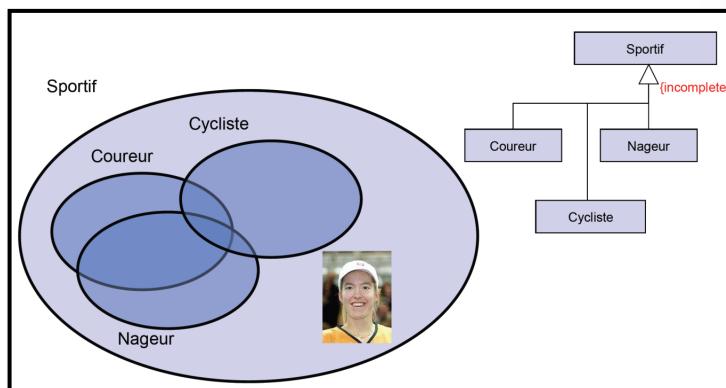
- Complet vs. Incomplet:
 - Complet:



[189]

Les Relations

- Complet vs. Incomplet:
 - Incomplet:



[190]

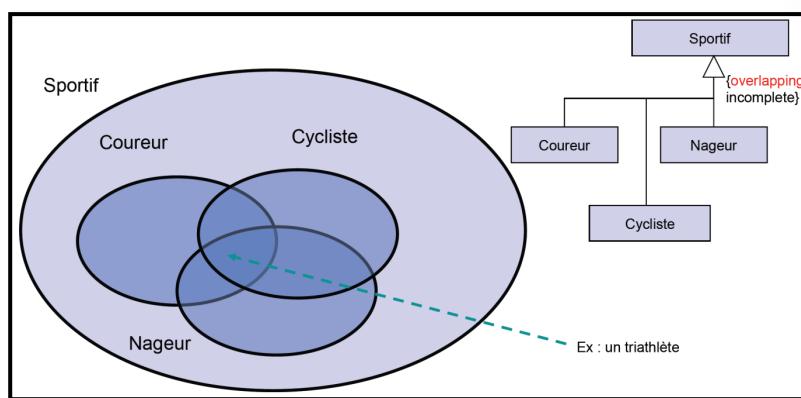
Les Relations

- Disjoint vs. Overlapping:
- Overlapping:
 - Des entités peuvent être classées dans plusieurs sous-classes
- Disjoint:
 - Une entité ne peut pas appartenir à plus d'une sous-classe

[191]

Les Relations

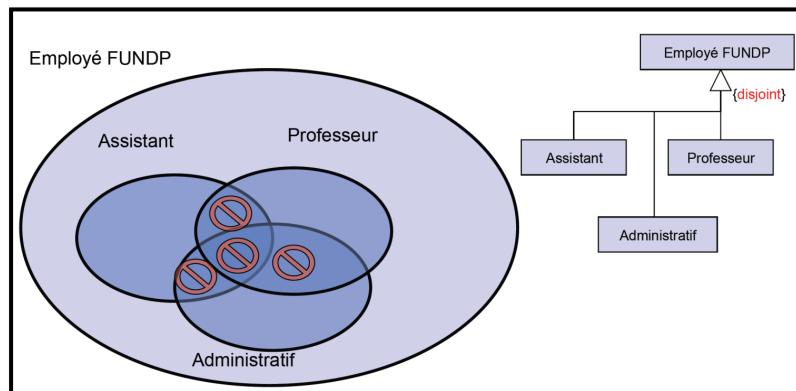
- Disjoint vs. Overlapping:
 - Overlapping:



[192]

Les Relations

- Disjoint vs. Overlapping:
 - Disjoint:



[193]

Les Relations

- Contraintes de la relation de Spécialisation/Généralisation:
 - Types de contrainte:
 - Complet vs. Incomplet
 - Disjoint vs. Overlapping
 - 4 possibilités:
 - {complet, disjoint} = *partition*
 - {complet, overlapping}
 - {incomplet, disjoint}
 - {incomplet, overlapping} "Contrainte" par défaut
 - Classe concrète:
 - La spécialisation est *incomplète*
 - Classe abstraite:
 - La spécialisation est *complète*
 - « *Classifier* »

[194]

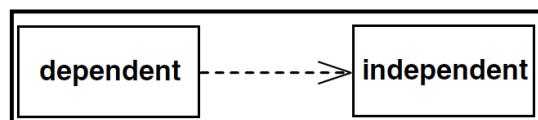
Les Relations

- La relation de Dépendance:
 - Une relation de dépendance modélise un lien entre un ou plusieurs éléments *source* et un ou plusieurs éléments *target*
 - Quelques stéréotypes de cette relation de dépendance (relation abstraite): «call», «create», «derive», «instantiate», «permit», «realize», «refine», «trace», «use», «substitute»,...

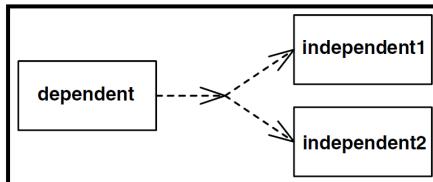
[195]

Les Relations

- La relation de Dépendance:



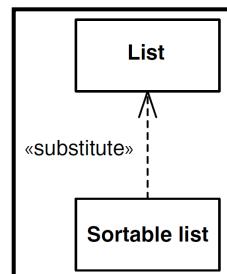
« type de dépendance »



[196]

Les Relations

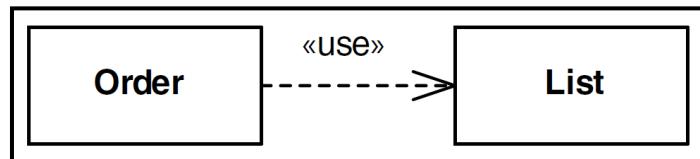
- Quelques Stéréotypes communs:
- «substitute»:
 - Substitution des instances de l'élément indépendant par des instances de l'élément dépendant en runtime
 - Ex: implémentation de la même interface



[197]

Les Relations

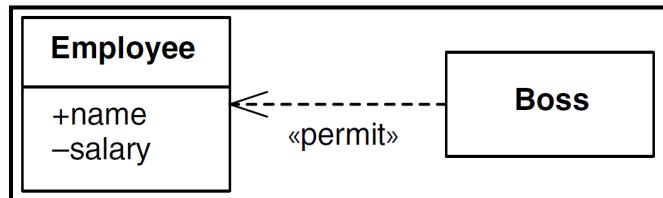
- Quelques Stéréotypes communs:
- «use»:
 - L'élément dépendant a besoin de l'élément indépendant pour son implémentation
 - Dépendance limitée à l'implémentation (pas pour la spécification)



[198]

Les Relations

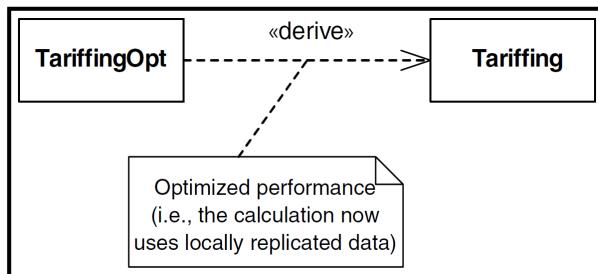
- Quelques Stéréotypes communs:
 - «permit»:
 - L'élément dépendant autorise l'élément indépendant à accéder à ces caractéristiques (ex: attribut privé)



[199]

Les Relations

- Quelques Stéréotypes communs:
 - «derive»:
 - L'élément dépendant est issu de l'élément indépendant
 - Utilisation de notes pour commenter les stéréotypes utilisés



[200]

Exercices

- Exercices sur le diagramme de classe

[201]

Chapitre 5: Le Diagramme de Classe

- Le Diagramme de Classe: Définition et Utilité
- Les Classes: Définition et Structure
- Les Contraintes
- La Notion d'Objet et la Classification
- Les Caractéristiques des Objets: les Attributs et les Opérations
- Les Relations
- ***Les Packages et Interfaces de Classes***

[202]

Les Packages et Interfaces de Classes

- Cfr. Annexes

[203]

Table des Matières

- Chapitre 0: Introduction
- Chapitre 1: Les Bases du Langage UML
- Chapitre 2: La Modélisation des Comportements
- Chapitre 3: Le Diagramme de Use Cases
- Chapitre 4: Le Diagramme d'Activité
- Chapitre 5: Le Diagramme de Classes
- **Chapitre 6: Les Diagrammes d'Interactions**
- Chapitre 7: Le Diagramme d'État

[204]

CHAPITRE 6: LES DIAGRAMMES D'INTERACTIONS

[205]

Chapitre 6: Les Diagrammes d'Interactions

- Les Différents Diagrammes d'Interactions
- Le Diagramme de Séquence: Structure et Notations
- Overview des Autres Diagrammes d'Interactions
 - Diagramme de Communication
 - Diagramme de Timing
 - Diagramme d'Interactions Global

[206]

Chapitre 6: Les Diagrammes d'Interactions

- *Les Différents Diagrammes d'Interactions*
- Le Diagramme de Séquence: Structure et Notations
- Overview des Autres Diagrammes d'Interactions
 - Diagramme de Communication
 - Diagramme de Timing
 - Diagramme d'Interactions Global

[207]

Les Différents Diagrammes d'Interactions

- Utilité des Diagrammes d'Interactions
 - Un Diagramme d'Interaction décrit les processus d'échanges entre acteurs/objet durant une activité
 - Rappel: une activité est une série de tâches (ici dans un processus → ordonnancement temporel)
 - Objectif: expliquer comment les objets communiquent entre eux

[208]

Les Différents Diagrammes d'Interactions

- 4 types de diagrammes (4 points de vue sur les interactions):
 - Diagramme de Séquence: spécification de la séquence dans laquelle les messages sont échangés
 - Diagramme de Communication: spécification des relations entre les participants à une interaction donnée
 - Diagramme de Timing: spécification des changements d'état des participants relativement au temps et aux messages échangés
 - Diagramme d'Interactions Général: spécification de l'ordre des interactions (notations similaires au diagramme d'activité)

[209]

Chapitre 6: Les Diagrammes d'Interactions

- Les Différents Diagrammes d'Interactions
- ***Le Diagramme de Séquence: Structure et Notations***
- Overview des Autres Diagrammes d'Interactions
 - Diagramme de Communication
 - Diagramme de Timing
 - Diagramme d'Interactions Global

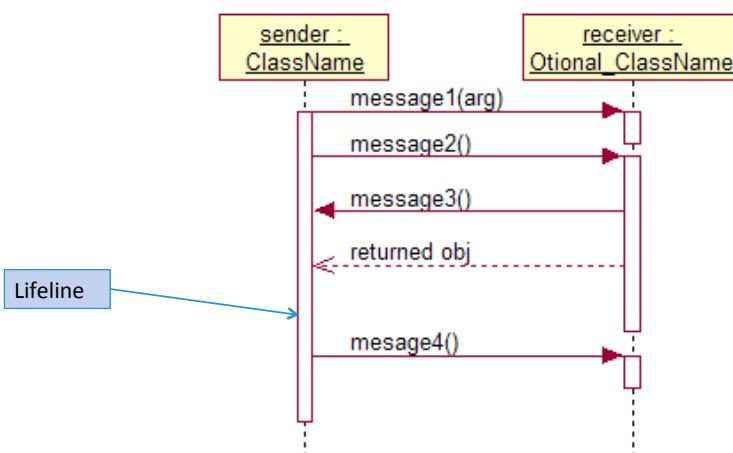
[210]

Le Diagramme de Séquence: Structure et Notations

- Un Diagramme de Séquence décrit une série de messages échangés entre un ensemble d'agents (dans une limite temporelle):
 - Les objets (\neq classes) participants aux interactions
 - La séquence des messages échangés
- Interaction: situation pendant le runtime d'un système
- Lifeline: ensemble des éléments relatifs à un participant
 - Implique un objet/acteur (= rôle)
 - Pas de Multi-objets
- Message: une communication spécifique entre Lifelines d'une Interaction

[211]

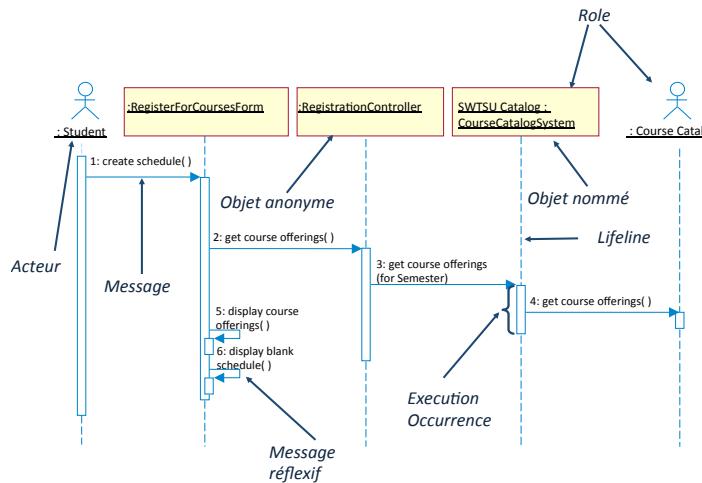
Le Diagramme de Séquence: Structure et Notations



[212]

Le Diagramme de Séquence: Structure et Notations

- Les différents symboles du Diagramme de Séquence:



Le Diagramme de Séquence: Structure et Notations

- Les différents symboles du Diagramme de Séquence:

- Les Occurrences d'Exécution représentent l'exécution d'une opération de l'objet; la participation active d'un objet/acteur dans une séquence
 - Début et fin définis par des événements (envoi et réception d'un message)
 - Représenté par des rectangles sur la lifeline
- Un Rôle est attaché à chaque lifeline (nom:type)
 - Un rôle est un objet ou un acteur
 - Ne pas modéliser les interactions directes entre acteurs (le SI n'y est pas impliqué)
 - Un acteur est souvent l'instigateur d'une séquence
 - Un rôle peut-être nommé ou non-nommé

[214]

Le Diagramme de Séquence: Structure et Notations

- Les différents symboles du Diagramme de Séquence:
 - Un Message est une communication entre deux rôles dans le but de transmettre des informations
 - La flèche représentant un message est labélisée avec le nom du message échangé ainsi que ses paramètres
 - Existence de messages réflexifs
 - On peut indiquer sur les messages leur séquence à l'aide de nombre
 - Les Événements dans un Diagramme de séquences sont les envois et réceptions de messages
 - Un message peut conduire à la création/destruction d'un objet (nouvelle/destruction d'une lifeline à cet endroit)

[215]

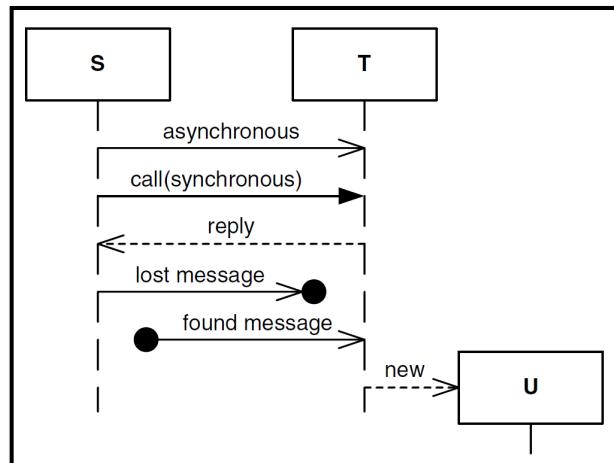
Le Diagramme de Séquence: Structure et Notations

- Message synchrone et message asynchrone:
 - Synchrone: Effet bloquant pour l'émetteur (attente de la réponse)
 - Asynchrone: L'émetteur peut continuer ses tâches (souvent, pas de réponse prévue)
- Message "perdu" et message "trouvé":
 - Perdu: émetteur connu, mais pas de récepteur connu
 - Trouvé: pas d'émetteur connu, mais le récepteur l'est
- Syntaxe: **[attribute =] name [(arguments)] [:return value]**
 - 'attribute': variable locale de l'interaction (seulement pour les messages synchrone avec message retour prévu)
 - 'arguments': liste des paramètres passés par le message (**[parameter=value]**)
 - Joker pour n'importe quel message: argument = '*'

[216]

Le Diagramme de Séquence: Structure et Notations

- Les Messages:



[217]

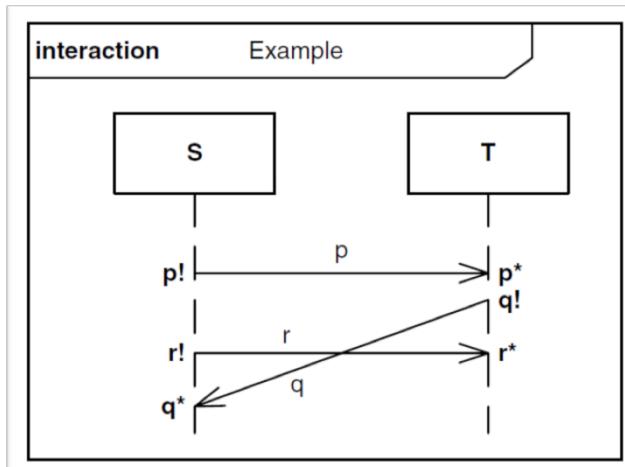
Le Diagramme de Séquence: Structure et Notations

- Validité d'un Diagramme de Séquence:
 - Deux règles d'or à respecter:
 - Avant de recevoir un message (événement), il doit toujours y avoir un message y étant lié et envoyé précédemment dans la lifeline
 - La séquence dans la lifeline est très importante (dimension temporelle)
 - Par contre, il n'est pas pertinent de comparer le positionnement des éléments d'une lifeline à une autre → Elles ne sont pas liées temporellement

[218]

Le Diagramme de Séquence: Structure et Notations

- Validité d'un Diagramme de Séquence:



[219]

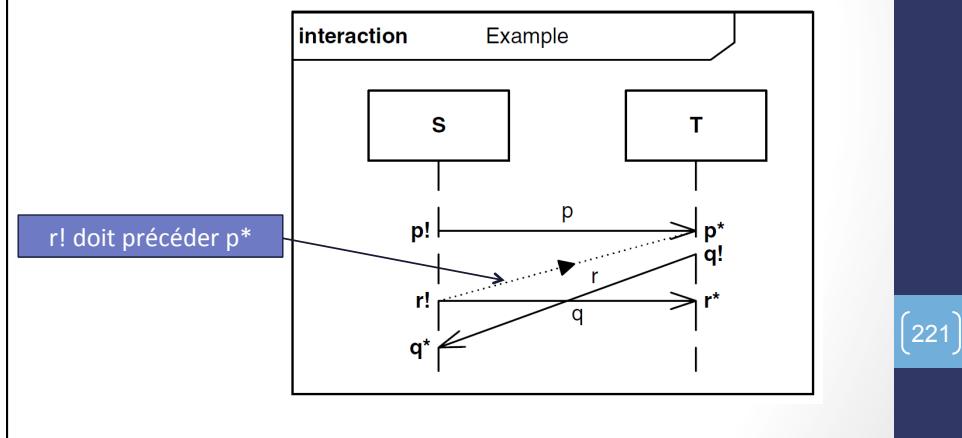
Le Diagramme de Séquence: Structure et Notations

- Ordonnancement d'un Diagramme de Séquence:
 - Pour donner une relation temporelle entre les lifelines, il faut les ordonner (GeneralOrdering)
 - Les événements sont alors mis dans une relation temporelle <premier,second>
 - La flèche utilisée pointe vers l'événement le plus lointain dans la relation temporelle
 - Important dans la gestion des messages perdus et trouvés

[220]

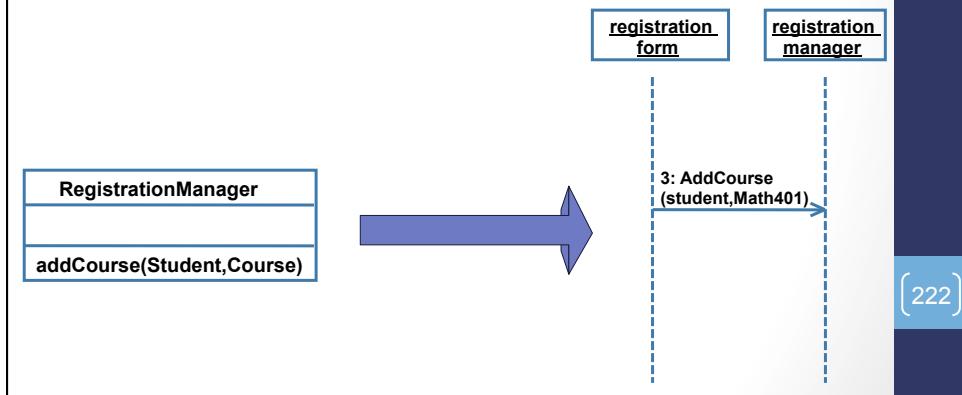
Le Diagramme de Séquence: Structure et Notations

- Ordonnancement d'un Diagramme de Séquence:



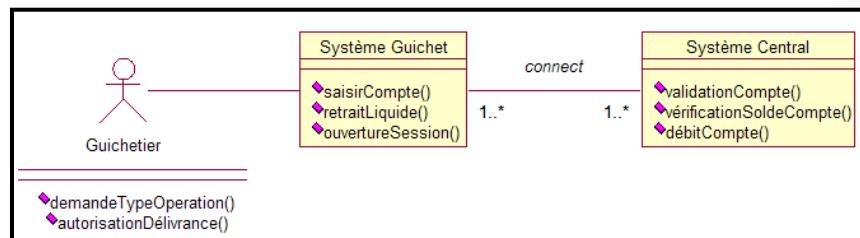
Le Diagramme de Séquence: Structure et Notations

- Les comportements d'un Diagramme de Séquence:
 - Les méthodes du diagramme de classe sont les comportements possibles dans le diagramme de séquence



Le Diagramme de Séquence: Structure et Notations

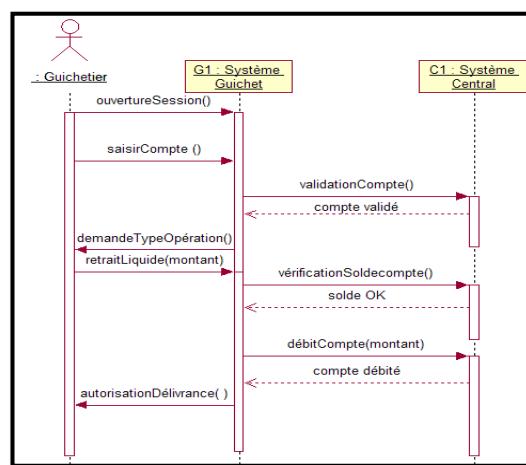
- Autre exemple: le retrait d'argent



[223]

Le Diagramme de Séquence: Structure et Notations

- Autre exemple: le retrait d'argent



[224]

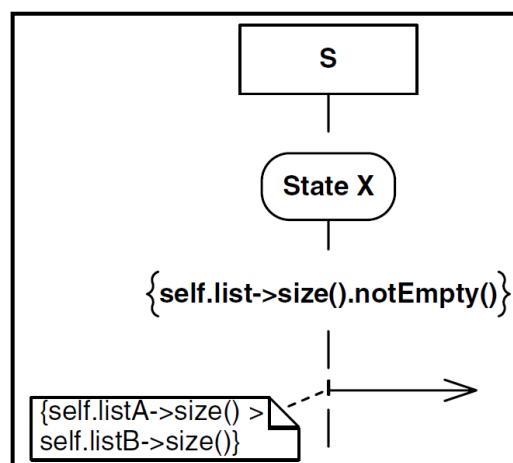
Le Diagramme de Séquence: Structure et Notations

- Invariant d'État:
 - L'échange de messages peut provoquer un changement d'état de l'objet rattaché à la lifeline
 - Un invariant d'état définit dans quelle mesure l'objet peut changer d'état
 - Un invariant d'état est évalué juste avant l'occurrence de l'événement suivant
 - Si invariant d'état violé → la séquence n'est pas valide

[225]

Le Diagramme de Séquence: Structure et Notations

- Invariant d'État:



[226]

Chapitre 6: Les Diagrammes d'Interactions

- Les Différents Diagrammes d'Interactions
- Le Diagramme de Séquence: Structure et Notations
- ***Overview des Autres Diagrammes d'Interactions***
 - Diagramme de Communication
 - Diagramme de Timing
 - Diagramme d'Interactions Global

[227]

Chapitre 6: Les Diagrammes d'Interactions

- Les Différents Diagrammes d'Interactions
- Le Diagramme de Séquence: Structure et Notations
- ***Overview des Autres Diagrammes d'Interactions***
 - ***Diagramme de Communication***
 - Diagramme de Timing
 - Diagramme d'Interactions Global

[228]

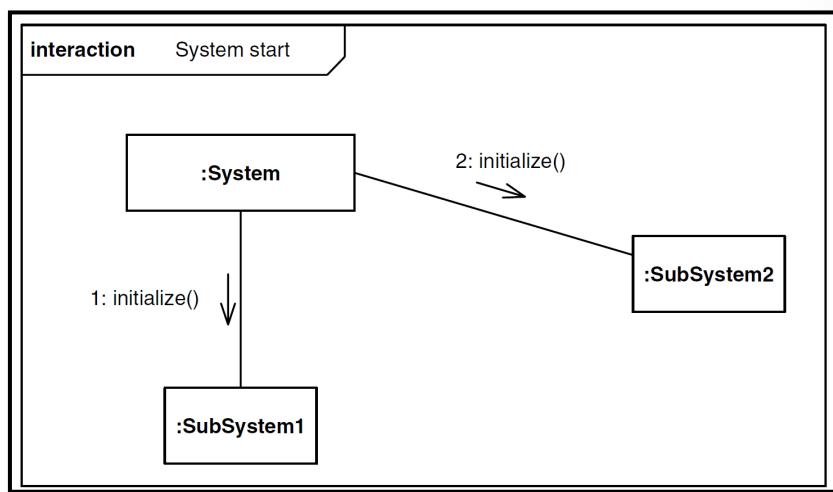
Diagramme de Communication

- Overview:
 - (Anciennement appelé *diagramme de collaboration* dans UML 1.x)
 - Le Diagramme de Communication spécifie plus clairement les relations entre les individus participants et la structure de ces relations
 - Éléments principaux modélisés:
 - Les objets participants
 - Les liens entre les objets
 - Les messages passés entre les objets

[229]

Diagramme de Communication

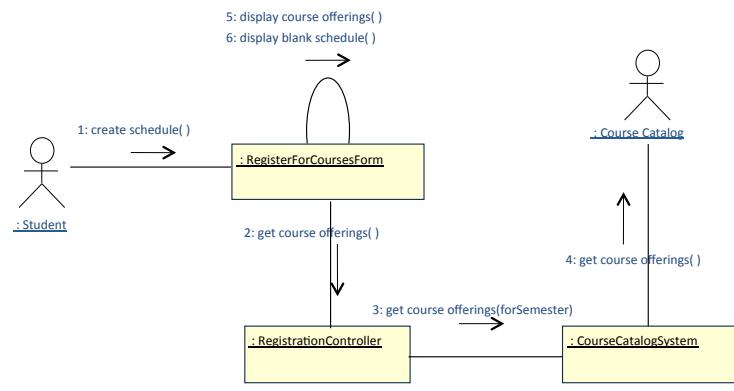
- Exemple:



[230]

Diagramme de Communication

- Exemple:



[231]

Diagramme de Communication

- Comparaison: Diagramme de Séquence et Diagramme de Communication

| Diagramme de Séquence | Diagramme de Communication |
|---|--|
| Rend explicite la séquence des messages | Montre clairement les relations entre objets/acteurs |
| Montre l'exécution des occurrences | Meilleure visualisation du pattern des échanges de messages |
| Facilite la compréhension générale des flux | Meilleure visualisation des effets sur un objet donnée |
| Plus facile pour modéliser les processus temps-réel et les scénarios plus complexes | Plus souple et plus facile à utiliser en séance de brainstorming |

[232]

Chapitre 6: Les Diagrammes d'Interactions

- Les Différents Diagrammes d'Interactions
- Le Diagramme de Séquence: Structure et Notations
- ***Overview des Autres Diagrammes d'Interactions***
 - Diagramme de Communication
 - ***Diagramme de Timing***
 - Diagramme d'Interactions Global

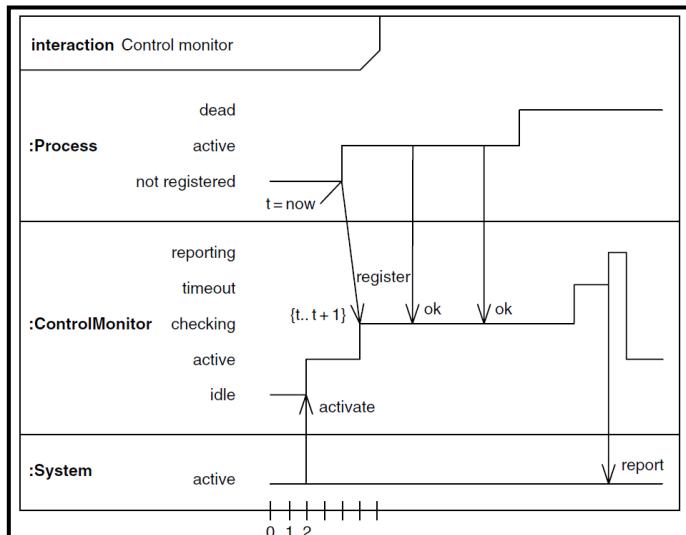
[233]

Le Diagramme de Timing

- Le Diagramme de Timing:
 - Souligne davantage les aspects temporels des différentes interactions entre les objets/acteurs

[234]

Le Diagramme de Timing



[235]

Chapitre 6: Les Diagrammes d'Interactions

- Les Différents Diagrammes d'Interactions
- Le Diagramme de Séquence: Structure et Notations
- ***Overview des Autres Diagrammes d'Interactions***
 - Diagramme de Communication
 - Diagramme de Timing
 - ***Diagramme d'Interactions Global***

[236]

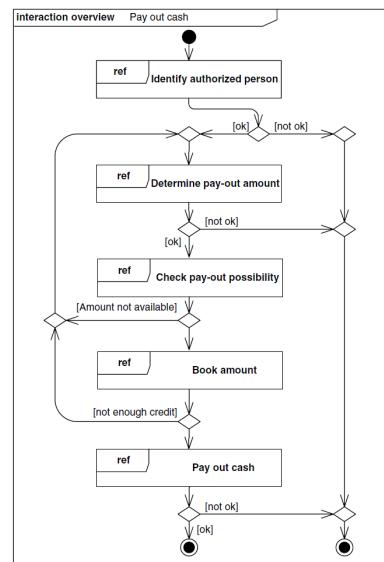
Le Diagramme d'Interactions Global

- Overview:
 - Très similaire à un diagramme d'activité au niveau des symboles utilisés
 - Le Diagramme d'Interactions permet de spécifier l'ordre d'exécution des interactions présentes dans le système d'information à créer

[237]

Le Diagramme d'Interactions Global

- Exemple:
 - Les noeuds sont des noeuds d'interactions



[238]

Exercices

- Exercices sur le Diagramme de Séquence

[239]

Table des Matières

- Chapitre 0: Introduction
- Chapitre 1: Les Bases du Langage UML
- Chapitre 2: La Modélisation des Comportements
- Chapitre 3: Le Diagramme de Use Cases
- Chapitre 4: Le Diagramme d'Activité
- Chapitre 5: Le Diagramme de Classes
- Chapitre 6: Les Diagrammes d'Interactions
- **Chapitre 7: Le Diagramme d'État**

[240]

CHAPITRE 7: LE DIAGRAMME D'ETAT

[241]

Chapitre 7: Le Diagramme d'État

- Le Diagramme d'État: Définition et Utilité
- L'État: Définition et Notations
- La Transition: Définition et Notations
- Les Machines à États: Structure et Notations

[242]

Chapitre 7: Le Diagramme d'État

- *Le Diagramme d'État: Définition et Utilité*
- L'État: Définition et Notations
- La Transition: Définition et Notations
- Les Machines à États: Structure et Notations

[243]

Le Diagramme d'État: Définition et Utilité

- Un diagramme de classe permet de modéliser les différents états d'un objet, mais pas les transitions (conditionnelles) entre ces états (ordonnancement temporel)
- Le diagramme d'état définit comment représenter:
 - Le mode d'emploi des objets
 - Le cycle de vie des objets
 - Le comportement des opérations
- Théorie basée sur les machines à états

[244]

Le Diagramme d'État: Définition et Utilité

- Décrit la structure de transitions éventuellement conditionnelles entre les différents états dans lesquels un objet donné peut exister
- Spécification des:
 - Événements et conditions qui causent les transitions d'état des objets
 - Actions lorsque un état particulier est atteint
- Notions du Diagramme d'État:
 - État
 - Transition
 - Événement
 - Action, activité
 - Garde

[245]

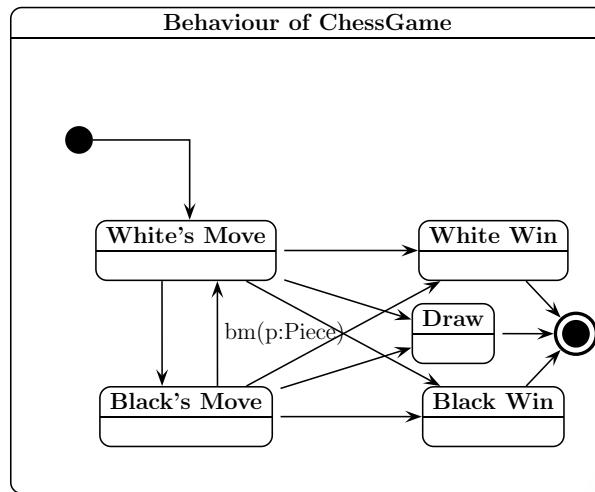
Le Diagramme d'État: Définition et Utilité

- Principe:
 - Le comportement continu d'un objet est *discretisé* en états pertinents pour les opérations modélisées
 - Un objet n'est que d'un état à la fois
- Exemple: un jeu d'échecs
 - Dans une partie d'échecs, les blancs jouent le premier coup puis les joueurs jouent à tour de rôle en déplaçant à chaque fois une de leurs pièces
 - Le jeu peut se terminer quand c'est le tour des blancs à jouer, ou quand c'est le tour des noirs
 - La partie peut se terminer par une victoire d'un joueur (échec et mat, ou abandon de l'autre joueur), ou par une partie nulle

[246]

Le Diagramme d'État: Définition et Utilité

- Principe: exemple



[247]

Chapitre 7: Le Diagramme d'État

- Le Diagramme d'État: Définition et Utilité
- *L'État: Définition et Notations*
- La Transition: Définition et Notations
- Les Machines à États: Structure et Notations

[248]

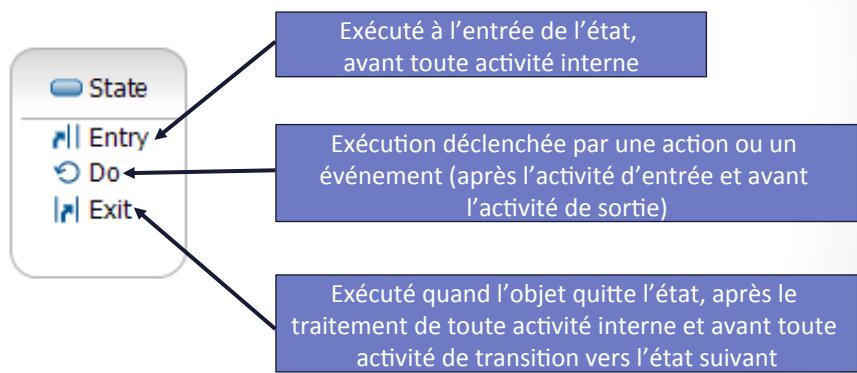
L'État: Définition et Notations

- Notion d'État:
 - Nœud du graphe
 - Un état est une configuration particulière d'un objet
 - Un état est labellisé par un nom, et par (éventuellement) une activité à l'*entrée*, à la *sortie* et pendant l'*existence de l'état*
 - L'état initial et l'état final ont des formes spécifiques
 - Un état simple: pas de sous-région
- Notion de Transition:
 - Arc du graphe
 - Labellisé par: événement, garde, action

[249]

L'État: Définition et Notations

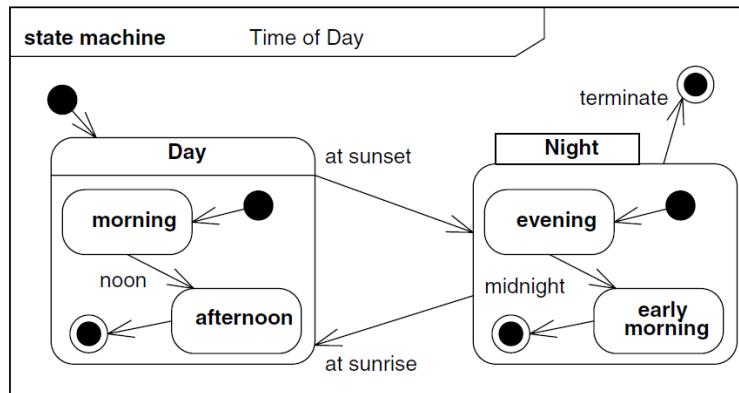
- Notion d'État:



[250]

L'État: Définition et Notations

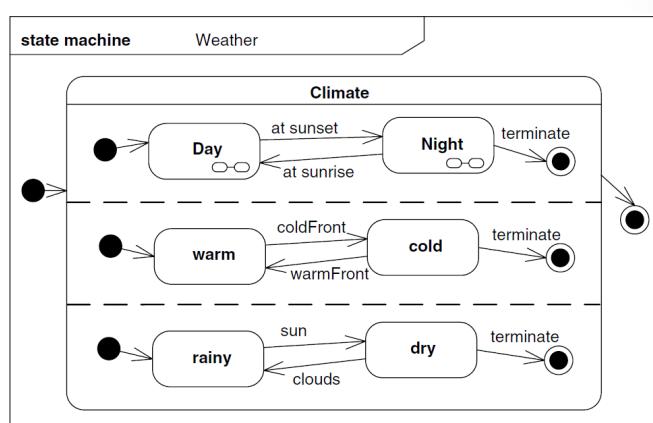
- État Composite: plus d'une région
 - Les régions sont dépendantes



[251]

L'État: Définition et Notations

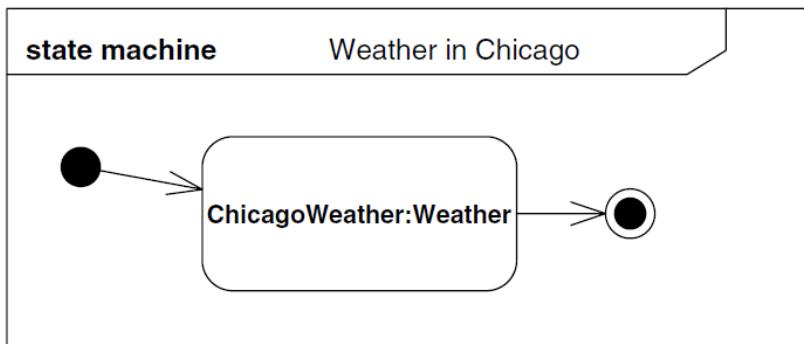
- État Orthogonal: plus de deux régions
 - Les régions sont indépendantes
 - Utile lorsqu'une multitude d'états (et de transitions) existent pour un objet



[252]

L'État: Définition et Notations

- Instanciation de Machine à États
 - Utilisation et instantiation d'un état (ici: l'état instancié est 'weather')



[253]

Chapitre 7: Le Diagramme d'État

- Le Diagramme d'État: Définition et Utilité
- L'État: Définition et Notations
- ***La Transition: Définition et Notations***
- Les Machines à États: Structure et Notations

[254]

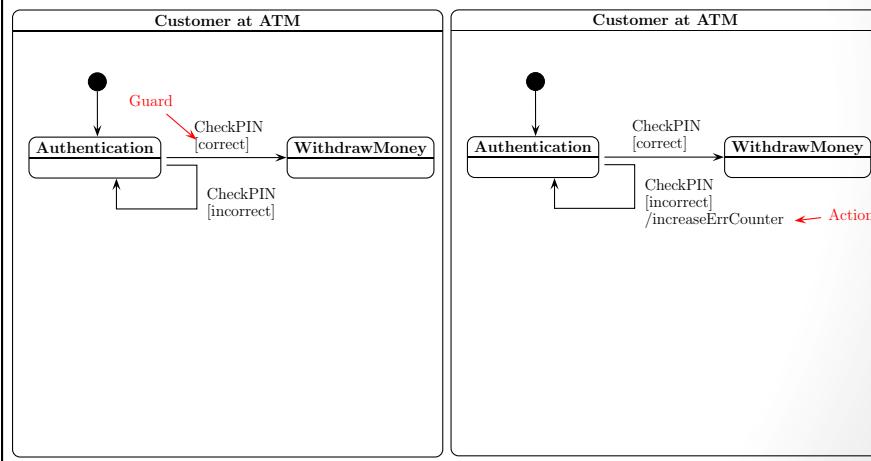
La Transition: Définition et Notations

- Notion de Transition: Quand l'objet est dans l'état source, que l'événement déclencheur se produit et que la garde est vraie, l'objet se retrouve dans l'état cible
 - Une transition a:
 - Un état source
 - Un événement déclencheur (ex: réception d'un appel d'opération)
 - Une *garde* pouvant empêcher une transition (= expression booléenne)
 - Un état cible
 - Propriétés:
 - Décrit un changement d'un état à un autre
 - Rôles dans le diagramme:
 - Déclenchement contrôlé par: événements, gardes, et conditions de sortie d'état
 - L'exécution peut causer des actions

[255]

La Transition: Définition et Notations

- Notion de Transition:



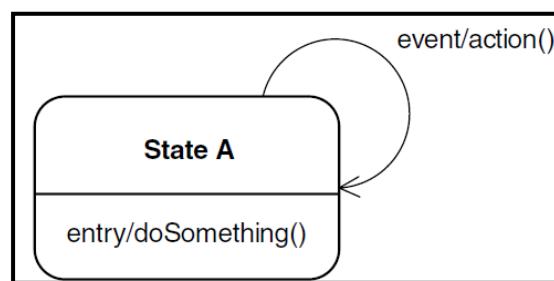
[256]

La Transition: Définition et Notations

- Notion d'Activité:
 - Notation:
 - do/*Action*
 - Écrit dans le symbole de l'état
 - Sémantique:
 - A une durée
 - Peut être terminée par un événement pour une transition vers un autre état

La Transition: Définition et Notations

- Notation des Transitions et des États:
 - Transitions:
 - Événement/*action*
 - Activité d'un état:
 - Moment/*action*



[258]

La Transition: Définition et Notations

- Les différents types d'événements:

| Type d'événement | Description | Syntaxe utilisée |
|------------------|--|--------------------------------------|
| Call Event | Réception d'un appel synchrone provenant d'un objet | op(a:T) |
| Change Event | Changement dans la valeur d'une expression booléenne | When(expression) |
| Signal Event | Réception d'un signal (asynchrone, nommée et explicite) provenant d'un autre objet | SignalName(a:T) |
| Time Event | - Arrivée à une date absolue - Délai de temps écoulé | - at(AbsoluteTime) - after(delay) |

[259]

Chapitre 7: Le Diagramme d'État

- Le Diagramme d'État: Définition et Utilité
- L'État: Définition et Notations
- La Transition: Définition et Notations
- ***Les Machines à États: Structure et Notations***

[260]

Les Machines à États: Structure et Notations

- Machines à États Hiérarchiques:
 - Utilité:
 - Décrire progressivement le comportement d'un objet (c.à.d.: ses états successifs) → un état peut contenir une machine à états
 - Top-down approche (raffinement)

[261]

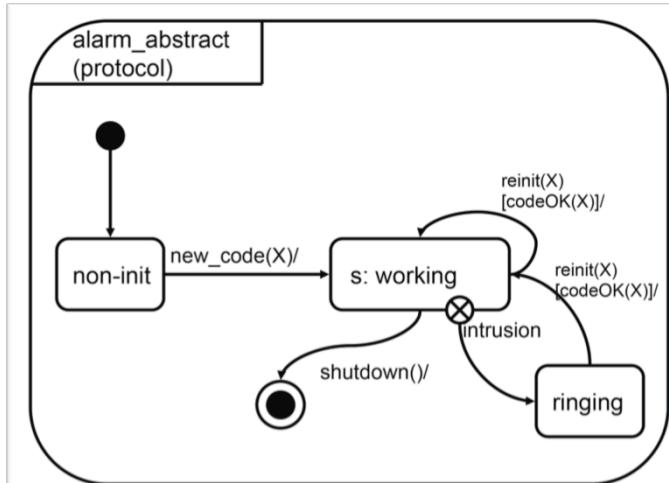
Les Machines à États: Structure et Notations

- Machines à États Hiérarchiques:
 - Exemple: Un système d'alarme
 - Lorsqu'il vient d'être installé, aucun code n'est mémorisé. Avant la première utilisation, il faut qu'un code soit introduit
 - Lors de son fonctionnement, il est toujours possible de le réinitialiser
 - Un événement particulier ("shutdown") permet de stopper le système

[262]

Les Machines à États: Structure et Notations

- Exemple: l'alarme

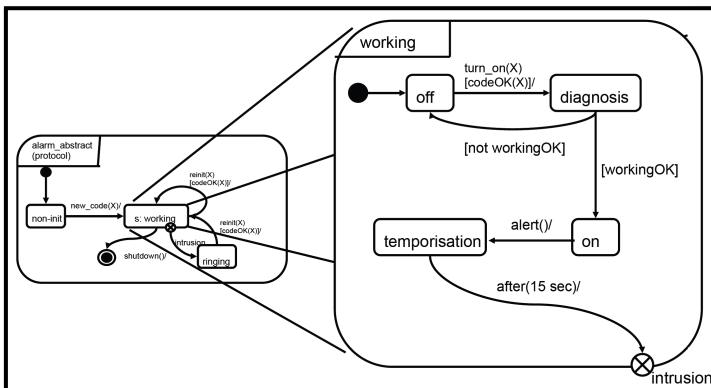


[263]

Les Machines à États: Structure et Notations

- Exemple: l'alarme

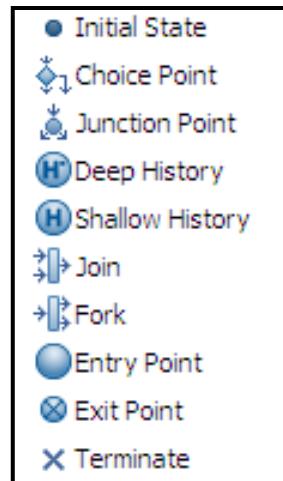
- Remarque: les points d'entrée et de sortie évitent les transitions inter-niveaux



[264]

Les Machines à États: Structure et Notations

- Sémantique et Notations



[265]

Les Machines à États: Structure et Notations

- Sémantique et Notations:
 - État Initial:
 - Entrée d'une machine à état/d'une région (un seul par machine à états)
 - Point de Décision:
 - Branches de transition multiple (uniquement séparation)
 - Point de Jonction:
 - Joindre plusieurs transitions ensemble (regroupement ou séparation)
 - 'Shallow History':
 - Retourne à l'état le plus récent

[266]

Les Machines à États: Structure et Notations

- Sémantique et Notations:
 - 'Deep History':
 - Retourne au sous-état le plus récent
 - Join & Fork:
 - Similaire au diagramme d'activités
 - Point d'Exit et Point d'Entrée:
 - Désigne l'entrée/la sortie dans une machine à états composite
 - Point de Terminaison:
 - Marque la fin de l'exécution d'une machine à états

[267]

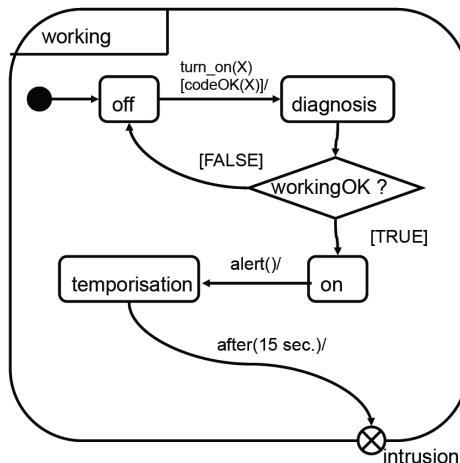
Les Machines à États: Structure et Notations

- Point de Décision:
 - Transitions avec un même déclencheur, mais plusieurs transitions possibles (on indique des gardes sur chacune des transitions)
 - Toutes les gardes sont évaluées avant de lancer la transition
 - Les gardes sont mutuellement exclusives
 - Les valeurs utilisées dans les gardes proviennent de l'état

[268]

Les Machines à États: Structure et Notations

- Point de Décision:



[269]

Les Machines à États: Structure et Notations

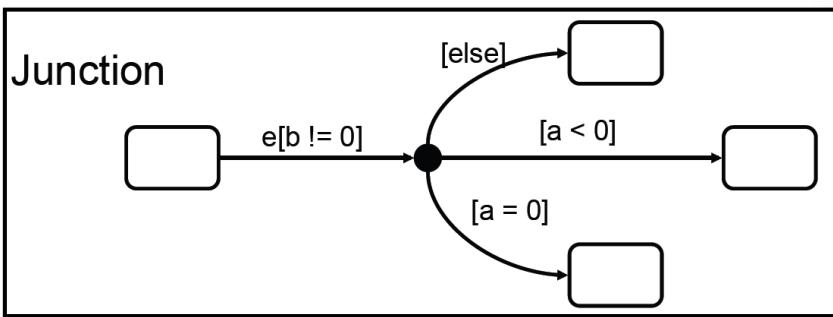
- Point de Jonction:

- Utiliser pour séparer ou rejoindre plusieurs transitions
- Les éventuelles évaluations en cas de séparation d'une transition ne dépendent pas de l'état source (>< point de décision) → indépendant de l'état source

[270]

Les Machines à États: Structure et Notations

- Point de Jonction:



[271]

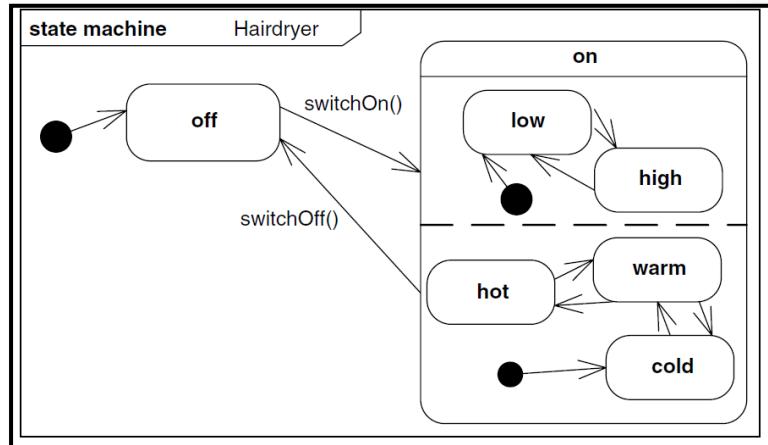
Les Machines à États: Structure et Notations

- "Shallow History":
 - On entre dans le sous-état qui était actif lors de la dernière sortie de l'état étant 'Shallow History'
 - Le 'H' peut ne pas être "connecté" et se trouver simplement dans l'état ayant la caractéristique "Shallow History"

[272]

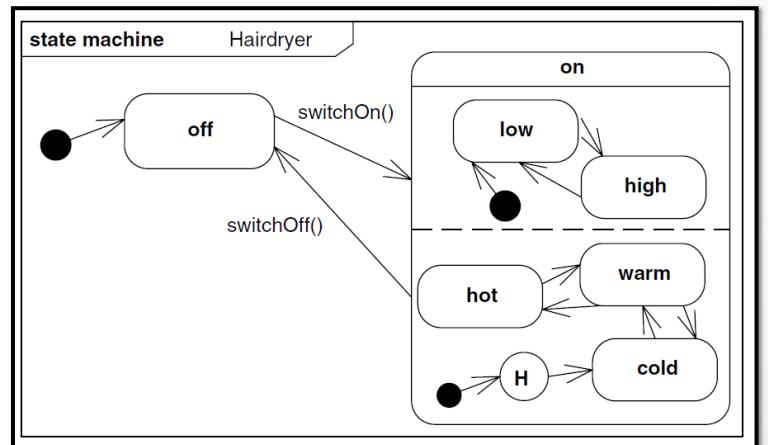
Les Machines à États: Structure et Notations

- "Shallow History":



Les Machines à États: Structure et Notations

- "Shallow History":



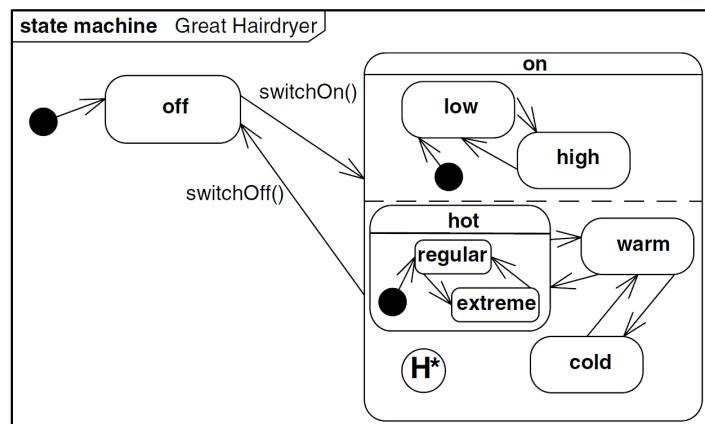
Les Machines à États: Structure et Notations

- "Deep History":
- Identique au "Shallow History", sauf que le mécanisme de mémorisation du dernier état est également valable pour les sous-états composites de l'état (→ tous les niveaux de profondeur)

[275]

Les Machines à États: Structure et Notations

- "Deep History":



[276]

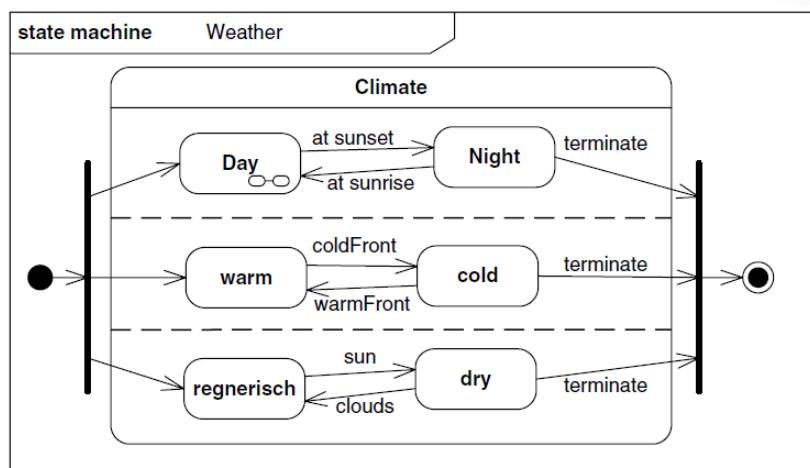
Les Machines à États: Structure et Notations

- Join & Fork:
 - Permet la création de sous-états parallèles
 - Pas de condition/garde ni trigger

[277]

Les Machines à États: Structure et Notations

- Join & Fork:



[278]

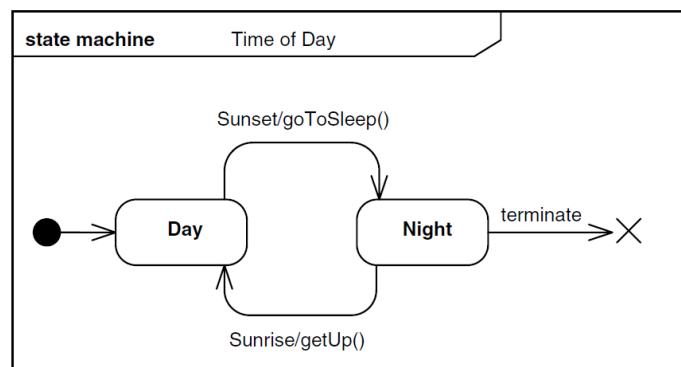
Les Machines à États: Structure et Notations

- Point de Terminaison:
 - Un Point de Terminaison met un terme à un comportement décrit par la machine à états, sans que l'objet ne prenne une nouvelle valeur → marque la fin de l'exécution de l'objet possédant la machine à états (autodestruction)
 - Un Point d'Exit: seul l'exécution de la machine à états se termine

[279]

Les Machines à États: Structure et Notations

- Point de Terminaison:



[280]

Exercices

- Exercices sur le Diagramme d'États

[281]

Fin de Cette Formation

Un Grand merci!

[282]

Sources Principales

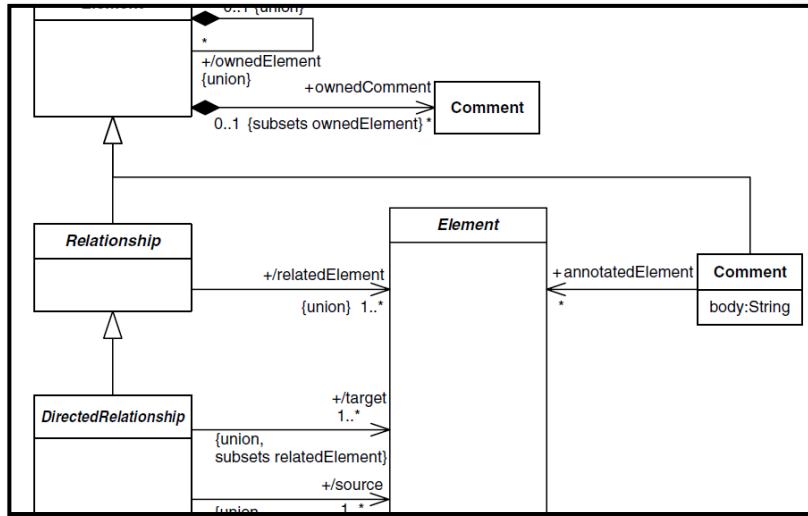
- T. Weilkiens, B. Oestereich, (2010) « *UML 2 Certification Guide: Fundamental & Intermediate Exams* »
- B. Beckert, « *Formal Specification of Software* »
- P. Thiran, « *Analyse et modélisation de systèmes d'information* »

[283]

ANNEXES

[284]

Le Méta-Modèle de Base d'UML



[285]

Check List

1. Which data types are known in UML, and what are they called?
2. What's the main difference between a class and a datatype?
3. How does the basic graphical representation of a diagram, including diagram header, look?
4. What is a stereotype? Does it define a new metamodel element?

[286]

Check List

1. What forms can behaviors have?
2. Can behavior be defined independently as a class?
3. Can a classifier own behavior?
4. What is the context of a behavior?
5. How is the execution of behavior triggered?

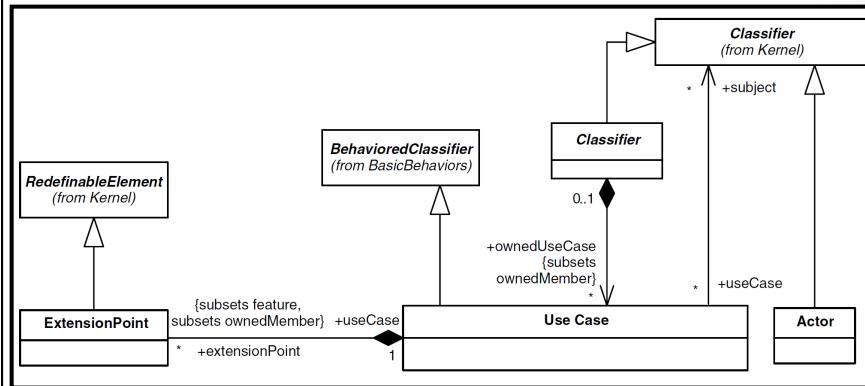
[287]

Check List

6. What is the Request Object in behavior calls?
7. What are the two ways to invoke behavior?
8. Is the sender or (*XOR*) the receiver responsible for selecting the behavior to be executed?
9. How can behaviored classifiers be defined as active objects?

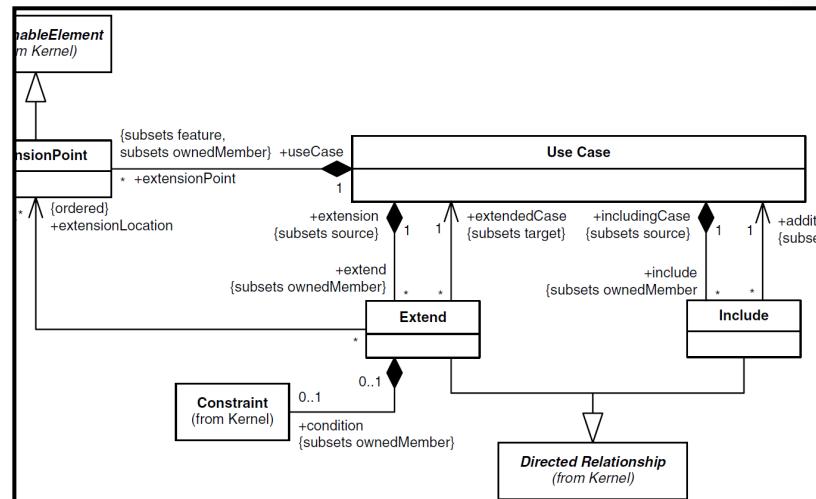
[288]

Le Méta-Modèle Use Case (Use Case & Actor)



[289]

Le Méta-Modèle Use Case (Use Case Relationship)



[290]

Check List

1. What is the subject of a Use Case?
2. Can an actor be only a human user?
3. Name the base class of an actor in the metamodel
4. Which notations are there for an actor?
5. Name the base class of a Use Case in the metamodel
6. Who can own a Use Case?

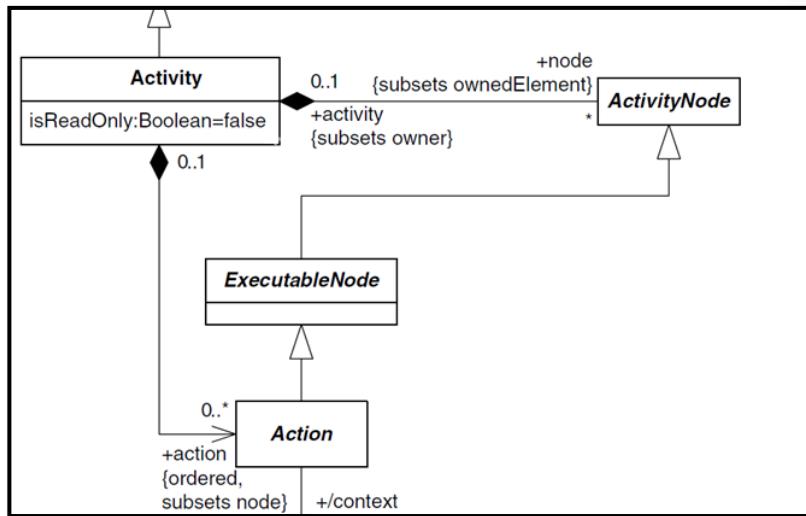
[291]

Check List

7. In what direction does the include relationship point
8. Does an included Use Case have to have an actor?
9. In what direction does the extend relationship point
10. How many extension points can a Use Case define
11. Does an extend relationship have to specify a condition?
12. Does an extended Use Case have to have an actor?

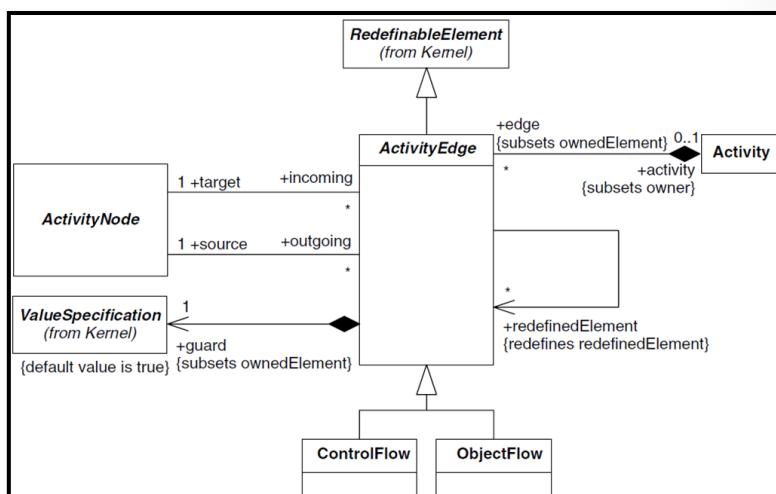
[292]

Méta-Modèle du Diagramme d'Activité



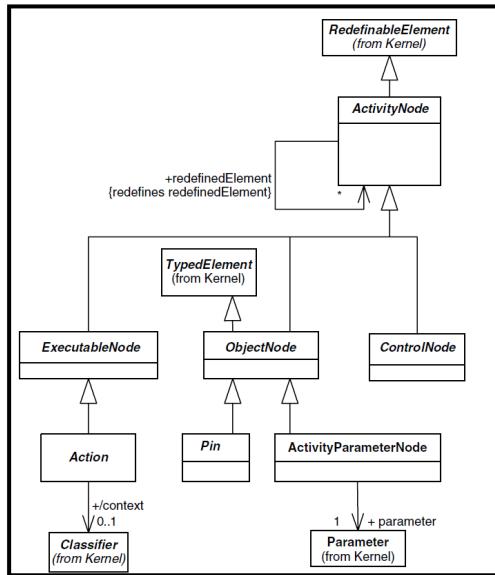
[293]

Méta-Modèles des Arcs et Noeuds



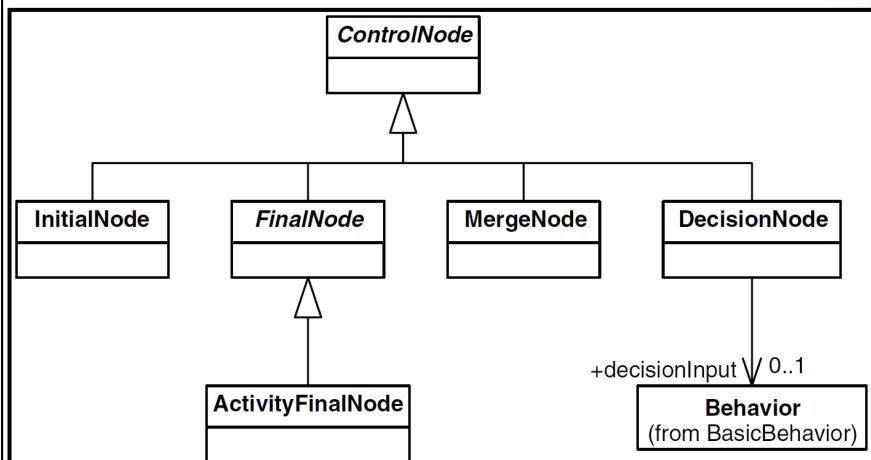
[294]

Méta-Modèles des Arcs et Noeuds



[295]

Méta-Modèles des Arcs et Noeuds



[296]

Check List

1. What does an activity represent?
2. What does an action represent?
(Compare actions with activities)
3. What kinds of edge are there?
4. What are pins?
5. What kinds of activity node are there?
6. What prerequisite must be met for an action to execute?

[297]

Check List

7. At which outgoing edges are tokens made available when an action terminates?
8. When does the or apply to object flows?
9. How many incoming edges can an initial node have?
10. What do several edges outgoing from one initial node signify?
11. What do several edges incoming into one final node signify?
12. What is the meaning of behavior (decision input) that can be defined by a decision?

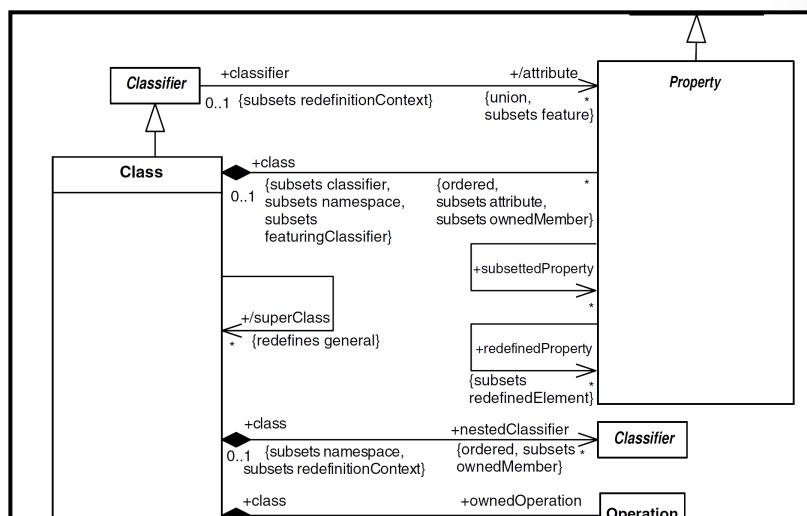
[298]

Check List

13. Within an activity diagram having object nodes,
 - a. What happens at the input parameters of an activity when that activity is called?
 - b. What happens at the output parameters of an activity when that activity is terminated?
14. How many pins can an action have?
15. When may pins be modeled alternatively as large rectangles with incoming and outgoing edges?

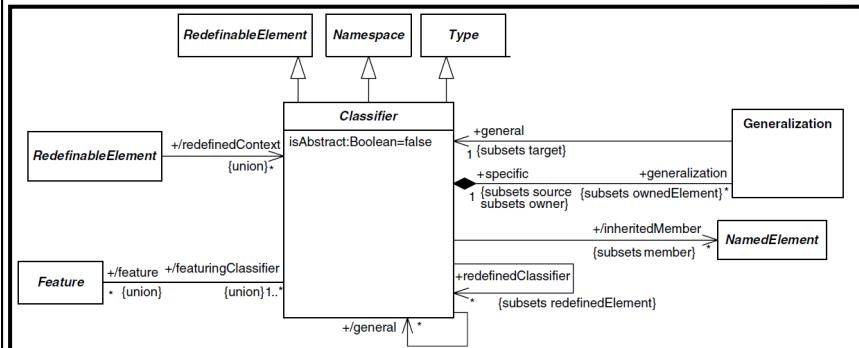
[299]

Méta-Modèle de Classe



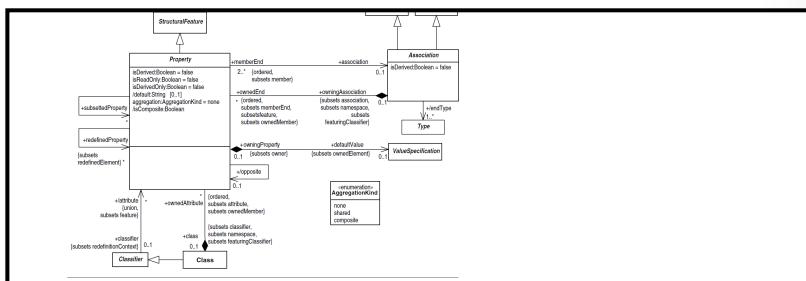
[300]

Méta-Modèle du Classifieur



[301]

Méta-Modèle de l'Association



[302]

CHAPITRE 5: LE DIAGRAMME DE CLASSE

Suite et Fin

[303]

Chapitre 5: Le Diagramme de Classe

- Le Diagramme de Classe: Définition et Utilité
- Les Classes: Définition et Structure
- Les Contraintes
- La Notion d'Objet et la Classification
- Les Caractéristiques des Objets: les Attributs et les Opérations
- Les Relations
- *Les Packages et Interfaces de Classes*

[304]

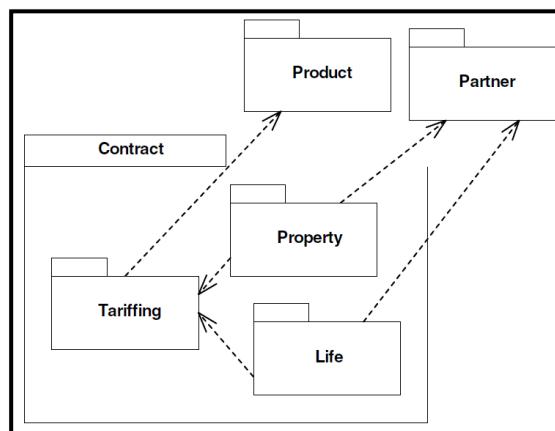
Les Packages et Interfaces de Classes

- Les Packages:
 - Un package est une collection de schémas (c'est-à-dire de modélisation de domaine d'application avec une perspective particulière)
 - Utiliser pour structurer une modélisation importante en plus petite unité, plus facile à gérer
 - Le nom des éléments (eg, diagramme de classes, use case,...) du package doivent être uniques
 - Sémantique: *PackageName::ClassName*

[305]

Les Packages et Interfaces de Classes

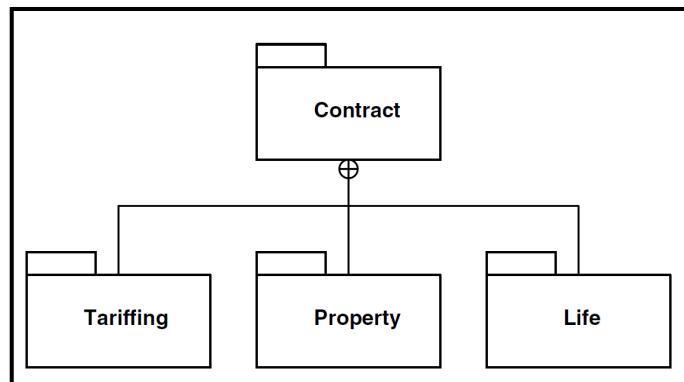
- Deux notations pour les Packages:
 - Notations prenant en compte les dépendances:



[306]

Les Packages et Interfaces de Classes

- Deux notations pour les Packages:
 - La notation  signifie "consists", "contains"



[307]

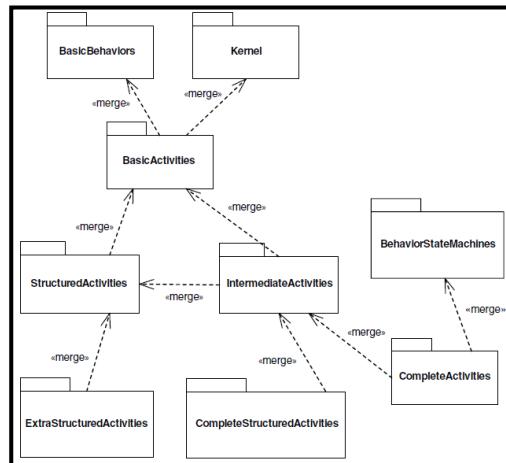
Les Packages et Interfaces de Classes

- Composition de Package (Merge Relationship):
 - Package merge:
 - Relation entre deux packages indiquant la fusion du contenu des packages en question (*target package* vers *source package*)
 - Les noms identiques dans les packages font référence aux mêmes concepts

[308]

Les Packages et Interfaces de Classes

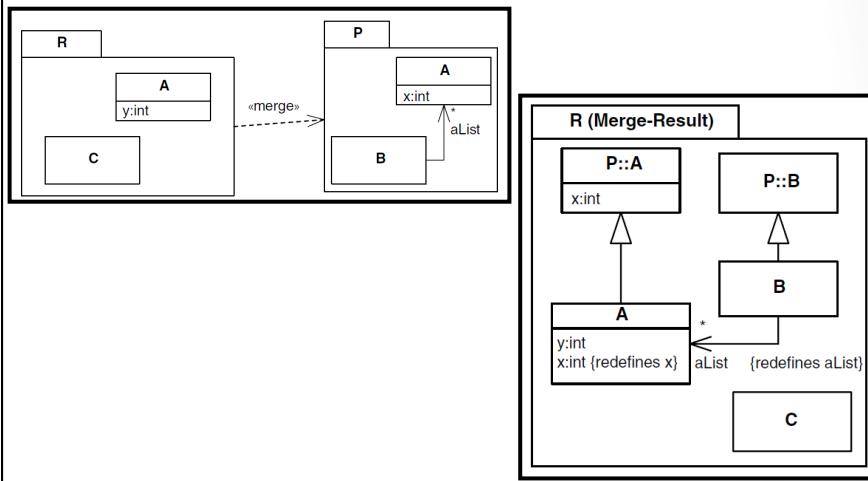
- Composition de Package (Merge Relationship):



[309]

Les Packages et Interfaces de Classes

- Exemple: composition de Packages



[310]

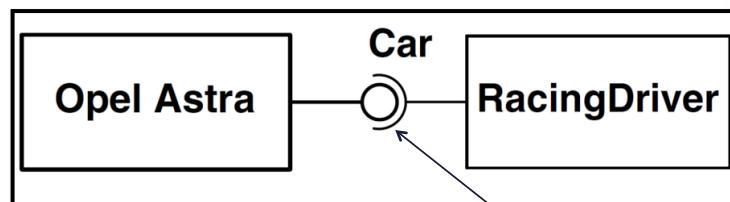
Les Packages et Interfaces de Classes

- Les Interfaces:
 - Une Interface est un ensemble d'attributs et de méthodes publiques que des classes peuvent s'engager à fournir ou à exiger → spécification de l'ensemble des comportements visibles d'une classe (ou d'un composant)
 - Sémantique: «interface»

[311]

Les Packages et Interfaces de Classes

- Les Classes Exigeant vs Fournissant



Notation Ball-Socket

[312]

Les Packages et Interfaces de Classes

- Les Classes Exigeant vs Fournissant
 - La classe fournissant l'interface:



- La classe demandant l'interface:



[313]

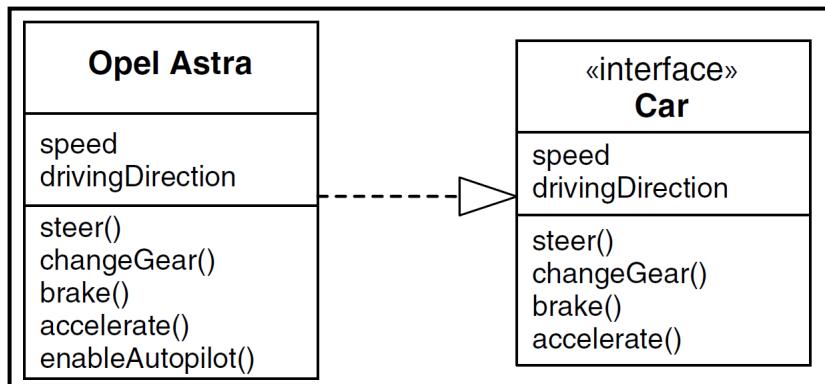
Les Packages et Interfaces de Classes

- Implémentation d'une Interface:
 - Une Relation d'Implémentation indique une relation de réalisation: la *classe exigeant* implémente les caractéristiques la *classe fournissant*
 - Implémentation d'une interface → toutes les caractéristiques sont reprises

[314]

Les Packages et Interfaces de Classes

- Implémentation d'une Interface:



[315]

Check List

1. For Namespace, what kind of elements can be imported by the element import?
2. What is the difference between private import and public import?
3. Give an example of a type and of a typed element.
4. What value range is described by a multiplicity?
5. What is the difference between multiplicity and cardinality?
6. Which constraint is predefined in UML?

[316]

Check List

7. How does a slot differ from an instance specification?
8. What subclasses does a classifier have?
9. What is the superclass of Class in the UML metamodel?
What is the type of an operation?
10. Explain pre-, post-, and body condition
11. What happens to the conditions if an exception occurs during the processing of an operation?

[317]

Check List

12. What do a property and an attribute have in common?
13. How do an attribute and an association relate?
14. Who owns a property?
15. Explain the meaning of “tuple” in the context of an association.
16. What does an association describe?

[318]

Check List

17. What are the semantics of an aggregation?
18. What are the semantics of a composition
19. What are the forms of interpreting the navigability in dependency relationship?
20. What does the generalization relationship have to do with packages?
21. How does the package merge work?

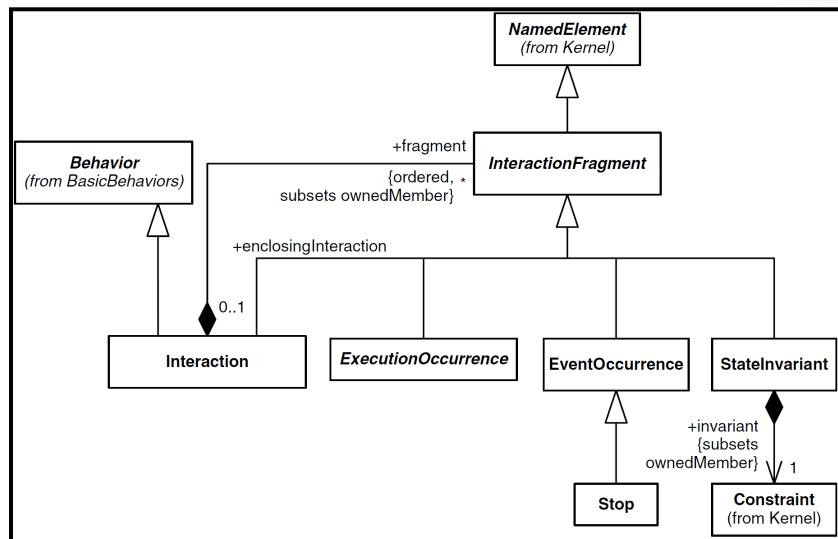
[319]

Check List

22. In what context is the term “incomplete” used for a dependency relationship?
23. In which direction does the arrow point that indicates the relationship between dependent and independent elements?
24. What is a(n)
 - abstraction relationship?
 - realization relationship?
 - substitution relationship?
 - usage relationship?
 - permission relationship?
25. What specifies an interface?
26. What is a ball-socket notation?

[320]

Méta-Modèle d'une Interaction



[321]

Check List

1. What is an interaction?
2. What does a lifeline represent?
3. What is an execution occurrence?
4. What types of message are there?
5. What is the special event occurrence *stop*?
6. What events are triggered by an exchange of messages?

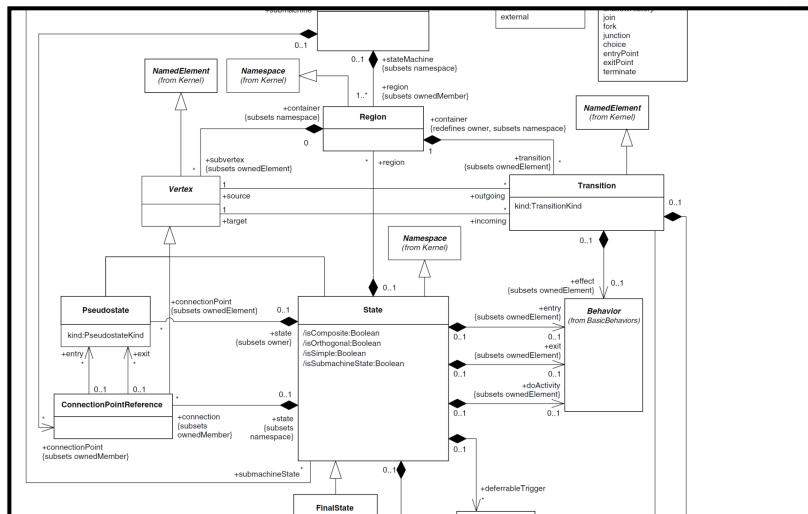
[322]

Check List

7. How is the semantics of an interaction defined?
8. What rules determine valid event sequences?
9. What does the *GeneralOrdering* relationship express?
10. What is a state invariant?
11. How are state invariants denoted?

[323]

Méta-Modèle du Diagramme d'États (ou Diagramme de Machines à États)



[324]

Check List

1. Explain the structure of a state machine
2. List the different kinds of state and their properties
3. Explain the difference between Pseudostate and State
4. Explain the internal behavior of a state
5. List the characteristics of *FinalState*
6. List the characteristics of *Transition*
7. Explain the functionality of *InitialState*

[325]

Check List

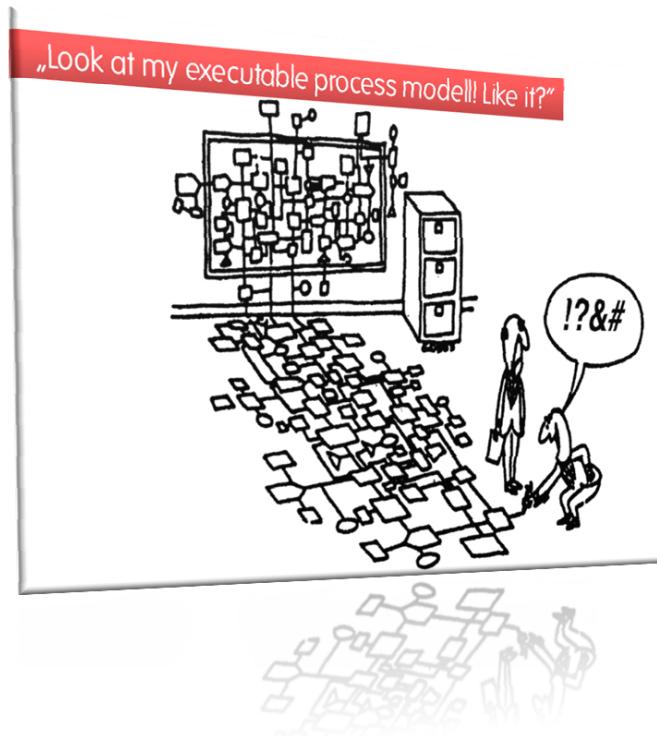
8. Explain the functionality of history state. What is the difference between 'deep history' and 'shallow history'?
9. Explain the functionality of static branching (junction)
10. Explain the functionality of choice
11. Explain the difference between *FinalState* and *Terminate*
12. Explain the functionality of splitting (fork) and synchronization (join)
13. Explain the functionality of entry and exit points
14. Explain the functionality of a connection point reference?

[326]

UML: EXERCICES – CHECKLISTS ET SPECIFICATION D'UN SI AVEC UML

Table des matières

| | |
|---|------------|
| CHECK LISTS | 165 |
| QCM | 168 |
| EXERCICES DE MODELISATION: SPECIFIER UN SYSTEME D'INFORMATION AVEC UML 2.0 | 179 |
| LE DIAGRAMME DE USE CASE | 179 |
| LE DIAGRAMME D'ACTIVITE | 183 |
| LE DIAGRAMME DE CLASSE | 187 |
| LE DIAGRAMME DE SEQUENCE | 190 |
| LE DIAGRAMME D'ETAT | 192 |
| EXERCICE INTEGRE | 197 |



UML EXERCICES

Check Lists

1. Which data types are known in UML, and what are they called?
2. What's the main difference between a class and a datatype?
3. How does the basic graphical representation of a diagram, including diagram header, look?
4. What is a stereotype? Does it define a new metamodel element?
5. What forms can behaviors have?
6. Can behavior be defined independently as a class?
7. Can a classifier own behavior?
8. What is the context of a behavior?
9. How is the execution of behavior triggered?
10. What is the Request Object in behavior calls?
11. What are the two ways to invoke behavior?
12. Is the sender or (*XOR*) the receiver responsible for selecting the behavior to be executed?
13. How can behaviored classifiers be defined as active objects?
14. What is the subject of a Use Case?
15. Can an actor be only a human user?
16. Name the base class of an actor in the metamodel.
17. Which notations are there for an actor?
18. Name the base class of a Use Case in the metamodel
19. Who can own a Use Case?
20. In what direction does the include relationship point?
21. Does an included Use Case have to have an actor?
22. In what direction does the extend relationship point?
23. How many extension points can a Use Case define?
24. Does an extend relationship have to specify a condition?
25. Does an extended Use Case have to have an actor?
26. What does an activity represent?
27. What does an action represent? (Compare actions with activities)
28. What kinds of edge are there?
29. What are pins?
30. What kinds of activity node are there?
31. What prerequisite must be met for an action to execute?
32. At which outgoing edges are tokens made available when an action terminates?
33. When does the or apply to object flows?
34. How many incoming edges can an initial node have
35. What do several edges outgoing from one initial node signify?
36. What do several edges incoming into one final node signify?
37. What is the meaning of behavior (decision input) that can be defined by a decision?
38. Within an activity diagram having object nodes,
39. What happens at the input parameters of an activity when that activity is called?

40. What happens at the output parameters of an activity when that activity is terminated?
41. How many pins can an action have?
42. When may pins be modeled alternatively as large rectangles with incoming and outgoing edges?
43. For Namespace, what kind of elements can be imported by the element import?
44. What is the difference between private import and public import?
45. Give an example of a type and of a typed element.
46. What value range is described by a multiplicity?
47. What is the difference between multiplicity and cardinality?
48. Which constraint is predefined in UML?
49. How does a slot differ from an instance specification?
50. What subclasses does a classifier have?
51. What is the superclass of Class in the UML metamodel? What is the type of an operation?
52. Explain pre-, post-, and body condition.
53. What happens to the conditions if an exception occurs during the processing of an operation?
54. What do a property and an attribute have in common?
55. How do an attribute and an association relate?
56. Who owns a property?
57. Explain the meaning of “tuple” in the context of an association.
58. What does an association describe?
59. What are the semantics of an aggregation?
60. What are the semantics of a composition?
61. What are the forms of interpreting the navigability in dependency relationship?
62. What does the generalization relationship have to do with packages?
63. How does the package merge work?
64. In what context is the term “incomplete” used for a dependency relationship?
65. In which direction does the arrow point that indicates the relationship between dependent and independent elements?
66. What is a(n)
- a) abstraction relationship?
 - b) realization relationship?
 - c) substitution relationship?
 - d) usage relationship?
 - e) permission relationship?
67. What specifies an interface?
68. What is a ball-socket notation?
69. What is an interaction?
70. What does a lifeline represent?
71. What is an execution occurrence?
72. What types of message are there?
73. What is the special event occurrence *stop*?
74. What events are triggered by an exchange of messages?
75. How is the semantics of an interaction defined?
76. What rules determine valid event sequences?
77. What does the *GeneralOrdering* relationship express?
78. What is a state invariant?

79. How are state invariants denoted?
80. Explain the structure of a state machine.
81. List the different kinds of state and their properties.
82. Explain the difference between Pseudostate and State.
83. Explain the internal behavior of a state.
84. List the characteristics of *FinalState*.
85. List the characteristics of *Transition*.
86. Explain the functionality of *InitialState*.
87. Explain the functionality of history state. What is the difference between 'deep history' and 'shallow history'?
88. Explain the functionality of static branching (junction).
89. Explain the functionality of choice.
90. Explain the difference between *FinalState* and *Terminate*.
91. Explain the functionality of splitting (fork) and synchronization (join).
92. Explain the functionality of entry and exit points.
93. Explain the functionality of a connection point reference?

QCM

Source: UML 2 Certification Guide Fundamental & Intermediate Exams

1. Which of the following diagram types are defined in UML?
 - a) Composite structure diagram
 - b) Message sequence chart
 - c) Data flow diagram
 - d) Activity diagram

2. The Use Case Diagram is a
 - a) behavior diagram.
 - b) interaction diagram
 - c) structure diagram
 - d) context diagram
 - e) requirements diagram.

3. What is a relationship ? Select the best answer.
 - a) Relationship is an abstract concept that specifies the kind of relationship between elements.
 - b) Relationship is an arrow between two elements.
 - c) Relationship is an abstract concept that specifies some kind of relationship between two elements.
 - d) Relationship defines an association between elements

4. Which statement(s) about Data Types is/are correct ?
 - a) A DataType has no attributes.
 - b) A DataType has no operations.
 - c) Instances of a DataType are different even if their attribute values are

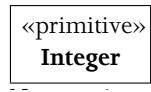
equal.

- d) A DataType is a specialized class.
- e) An instance of a DataType has no identity.

5. Which primitive types are defined in UML ?

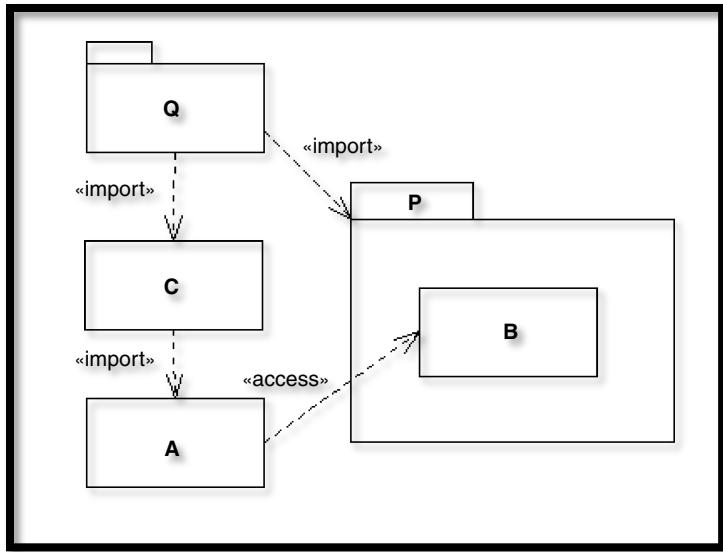
- a) Boolean
- b) Integer
- c) Real
- d) UnlimitedNatural
- e) Double
- f) String

6. Which is the correct notation for the PrimitiveType Integer ?

- a) 
- b) No notation
- c) □
- d) Integer

7. Which statements are correct ?

- a) The import relationship between Q and P is an ElementImport.
- b) B is known in Q.
- c) B is known in A.
- d) B is known in C.
- e) P is known in C
- f) It's possible to define an alias at the import relationship between Q and P.



8. Which of the following are valid multiplicities ?

- a) 0,1
- b) *
- c) 0..*
- d) 23..42
- e) 9..1
- f) 1
- g) -5..0

9. Which properties can be defined by a MultiplicityElement ?

- a) Lower and upper values
- b) Iterator
- c) Ordering

10. What is a constraint ? Select the best answer.

- a) An expression that is always true
- b) A condition expressed in natural language text or in a machine- readable

language for the purpose of declaring some of the semantics of an element

- c) A structured tree of symbols that denotes a (possibly empty) set of values when evaluated in a context
- d) A boolean expression that restricts the values of an attribute

11. Which statements about an InstanceSpecification are true ?

- a) InstanceSpecification is an object.
- b) InstanceSpecification represents an instance in a modeled system.
- c) InstanceSpecification represents an entity at a point in time.
- d) InstanceSpecification has a set of slots that contain the values of the structural features.

12. Which elements may be annotated by a comment ?

- a) Class
- b) Operation
- c) Diagram
- d) Comment
- e) Association

13. StructuralFeature is a specialized

- a) NamedElement
- b) SuspectElement
- c) PackageableElement
- d) TypedElement
- e) Element
- f) Namespace

14. A class is a specialized

- a) NamedElement
- b) SuspectElement
- c) PackageableElement
- d) TypedElement
- e) Element
- f) Namespace

15. Which of the following statements are true ?

- a) A NamedElement is a specialized PackageableElement
- b) A PackageableElement is a specialized NamedElement
- c) A TypedElement is a specialized NamedElement
- d) An Element is a specialized NamedElement
- e) A PackageableElement is a specialized TypedElement

16. What is a BehavioredClassifier?

- a) A classifier that can have operations
- b) A classifier with behavior that is described by a state machine.
- c) A classifier that can have behavior specifications in its namespace.
- d) A classifier that classifies instances that have operations in common.

17. A generalization is a relationship between... (There is more than one correct answer.)

- a) Classifiers
- b) UseCases
- c) an Actor and a UseCase

d) DataTypes

18. The instance of an association is called a/an

- a) Object
- b) Link
- c) LinkObject
- d) LinkSpecification
- e) ObjectLink

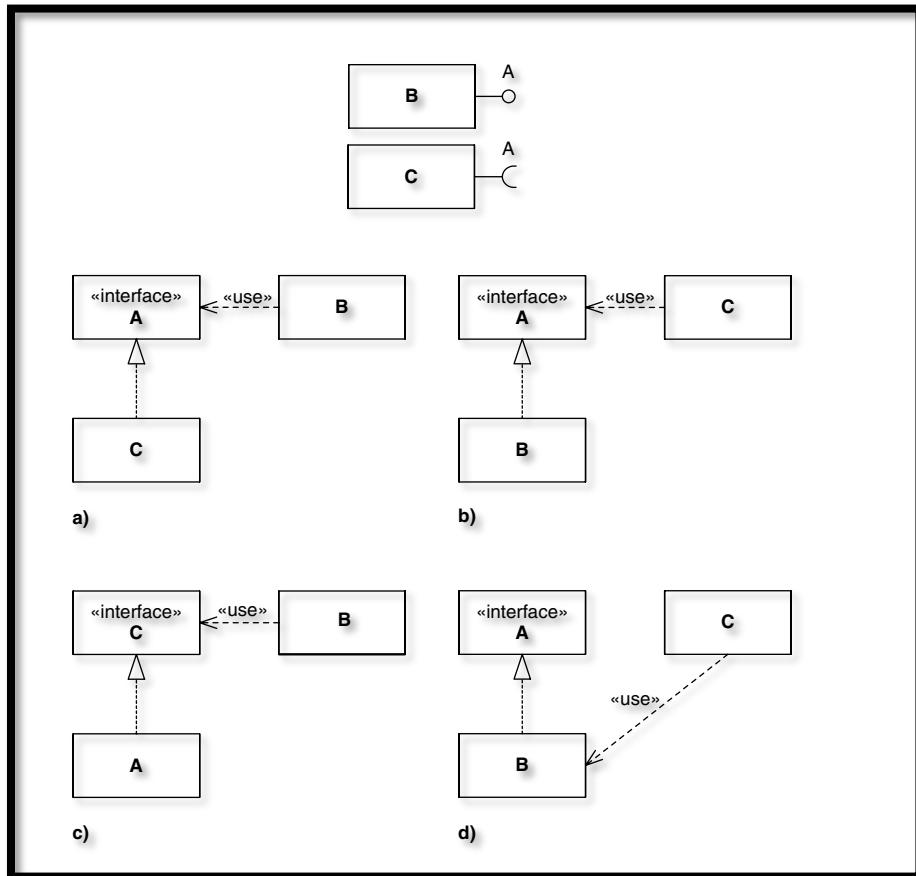
19. Which of the following statements about dependency relationships are true?

- a) A change in the supplier requires an update of the client.
- b) The client is incomplete without the supplier
- c) The client sends data to the supplier.
- d) The supplier sends data to the client.
- e) The client requires the supplier for its specification or implementation.

20. Which statements about usage relationships are true?

- a) A change in the supplier requires an update of the client.
- b) The client is incomplete without the supplier.
- c) The client requires the supplier for its implementation.
- d) The supplier sends data to the client.
- e) The client requires the supplier for its specification or implementation.

21. Which of the four solutions shows the following interface configuration?



22. What kind of ActivityNodes are defined in UML?

- a) ObjectNode
- b) SleepNode
- c) SuspendNode
- d) ControlNode

23. An activity has an ActivityFinalNode with two incoming edges. When does the activity terminate?

- a) One token arrives at the ActivityFinalNode
- b) Two tokens arrive at the ActivityFinalNode
- c) The terminate condition evaluates to true.

- d) The user has pressed the stop button.

24. A token

- a) is a label in the activity diagram.
- b) contains an object or focus of control and is present in the activity diagram at a particular node
- c) is a state that is valid while an activity is active.
- d) is a condition to trigger activity edges.

25. A MergeNode has three incoming edges. Which statement(s) is/are true?

- a) The MergeNode maps the incoming edges to one or more outgoing edges.
- b) The MergeNode waits for tokens at all incoming edges and offers a token to the outgoing edge.
- c) The MergeNode offers all tokens from the incoming edges to the outgoing edge
- d) The MergeNode offers tokens to outgoing edges depending on a condition.

26. An interaction is

- a) communication between two objects.
- b) a call of an operation.
- c) a unit of behavior.
- d) a set of messages.
- e) described by a sequence diagram.

27. A message in an interaction could be

- a) Asynchronous
- b) Concurrent
- c) Synchronous

d) Iterative

28. A message is an interaction

- a) is a call of an operation.
- b) defines a communication between lifelines.
- c) must be defined in an interface.
- d) executes an action for sending a signal.

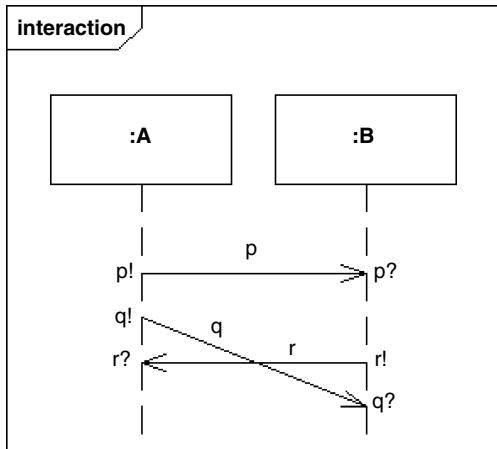
29. What kind of events can occur in an interaction ?

- a) Send events
- b) Receive events
- c) Flow events
- d) Destruction events
- e) Creation events
- f) Decision events

30. The GeneralOrdering relationship

- a) defines an order between two or more message events
- b) is a special call of an operation
- c) defines an order between lifelines
- d) defines an order between two message events.

31. Which traces are valid for the interaction shown here?



- a) <p!,p?,q!,r?,r!,q?>
- b) <p!,q!,r?,p?,r!,q?>
- c) <p!,p?,q!,r!,r?,q?>
- d) <p!,p?,r!,q!,r?,q?>
- e) <p?,r!,q?,p!,q!,r?>
- f) <p!,r!,p?,q!,r?,q?>

32. What describes a UseCase best?

- a) A UseCase is an ordered list of actions
- b) A UseCase is the specification of a set of actions performed by a system
- c) A UseCase describes an interaction between a user and a system
- d) A UseCase is a specialized operation

33. What describes an actor best?

- a) An actor is a user of a system.
- b) An actor is a user or any other system that interacts with the subject.
- c) An actor specifies a role played by a user or any other system that interacts with the subject.
- d) An actor is an object that may execute its own behavior without requiring

method invocation.

34. An include relationship is a specialized

- a) Dependency.
- b) DirectedRelationship.
- c) Relationship.
- d) Association
- e) Element.

Exercices de modélisation: Spécifier un système d'information avec UML 2.0

Consignes: pour chacun des exercices suivants, modélez le point de vue décrit dans le cas à l'aide d'un diagramme UML, et plus précisément en utilisant le diagramme portant le nom de la section dans laquelle le cas est écrit.

Le Diagramme de Use Case

- 1) Gestion de salles de cours
(NIVEAU Facile)

Dans un établissement scolaire, on désire gérer la réservation des salles de cours ainsi que du matériel pédagogique (ordinateur portable ou/et Vidéo projecteur).

Seuls les enseignants sont habilités à effectuer des réservations (sous réserve de disponibilité de la salle ou du matériel). Le planning des salles peut quant à lui être consulté par tout le monde (enseignants et étudiants).

Par contre, le récapitulatif horaire par enseignant (calculé à partir du planning des salles) ne peut être consulté que par les enseignants.

Enfin, il existe pour chaque formation un enseignant responsable qui seul peut éditer le récapitulatif horaire pour l'ensemble de la formation

- 2) Le processus de vente
(NIVEAU Moyen)

Dans un magasin, le processus de vente est le suivant : le client entre, passe dans les rayons, demande éventuellement des renseignements ou procède à des essais, prend des articles (si le stock est suffisant), passe à la caisse où il règle ses achats (avec tout moyen de paiement accepté). Il peut éventuellement bénéficier d'une réduction.

- 3) Connexion à un serveur
(NIVEAU Facile)

Considérons la connexion d'un client à un serveur fournissant les protocoles *HTTP*, *mail*, *telnet* et *FTP*. Décrire les cas d'utilisation de ce serveur en fonction des capacités de chaque protocole à transférer des données, exécuter des commandes, identifier un client...

- 4) Le distributeur de billets
(NIVEAU Facile)

Déterminer les cas d'utilisation d'un distributeur de billets. On considère les scénarios où un client désire retirer de l'argent en euros ou en dollars. Il faut traiter la situation où le stock de billets est insuffisant. On s'intéresse également à la procédure d'identification (de la carte et du client).

- 5) L'ATM : distributeur de billets avancé

(NIVEAU Moyen)

Les principales fonctions d'un ATM sont les suivantes :

- distribution d'argent à tout porteur d'une carte de la banque (autorisation d'un certain montant par le Système d'Information de la banque) ou d'une carte VISA (autorisation à distance par le Système d'Autorisation VISA)
- consultation du solde, dépôt en numéraire et de chèques pour les possesseurs d'une carte de la banque.

Toutes les transactions sont sécurisées (code personnel vérifié avec le code enregistré sur la puce de la carte ; la carte est avalée après trois échecs).

Il faut parfois recharger l'ATM avec de l'argent ainsi que et récupérer des cartes avalées ou les chèques.

6) L'organisation de formation

(NIVEAU Difficile)

Une entreprise souhaite modéliser avec UML le processus de formation de ses employés afin d'informatiser certaines tâches.

Le processus de formation est initialisé quand le responsable formation reçoit une demande de formation d'un employé. Cet employé peut éventuellement consulter le catalogue des formations offertes par les organismes agréés par l'entreprise. Cette demande est instruite par le responsable qui transmet son accord ou son refus à l'employé.

En cas d'accord, le responsable cherche la formation adéquate dans le catalogue des formations agréées qu'il tient à jour. Il informe l'employé du contenu de la formation et lui soumet la liste des prochaines sessions prévues. Lorsque l'employé a fait son choix, le responsable formation inscrit l'employé à la session retenue auprès de l'organisme de formation concerné.

En cas d'empêchement, l'employé doit avertir au plus vite le responsable formation pour que celui-ci demande l'annulation de l'inscription.

A la fin de la formation, l'employé transmet une appréciation sur le stage suivi et un document attestant sa présence.

Le responsable formation contrôle la facture envoyée par l'organisme de formation.

7) Projet de recherches en viticulture

(NIVEAU Difficile)

Dans le cadre d'un projet de recherche en viticulture, on désire collecter les temps de travaux sur des exploitations agricoles pilotes, pour travailler en particulier sur les opérations phytosanitaires.

Pour ce faire, un glossaire des opérations culturales types a été mis en place (afin que tout le monde ait le même cadre analytique).

Des contraintes assez fortes sont apparues sur le projet : les ouvriers agricoles des exploitations pilotes n'ont pas accès aux outils informatiques et la lourdeur

d'enregistrement des temps de travaux a donc de fait écarté l'utilisation d'un outil informatique.

La procédure suivante a ainsi été définie : chaque ouvrier agricole saisit ses temps de travaux sur un cahier au format prédéfini (dans ce cahier, il peut consulter en annexe le glossaire afin d'identifier l'opération culturelle type). Pour les opérations de type phytosanitaire, les informations complémentaires sont demandées : liste des maladies visées, stade phénologique, méthodes de traitements et observation.

En fin de mois, le chef d'exploitation vérifie la saisie effectuée sur le cahier et apporte d'éventuelles corrections. Il saisit ensuite les opérations du mois sur une application internet connectée à une base de données.

Le chercheur en charge du projet reçoit automatiquement un mail qui lui indique que la saisie mensuelle a été effectuée. Après avoir vérifié la pertinence de la saisie, il notifie au chef d'exploitation que tout s'est bien passé et que les données intégrées dans la base de données sont valides et prêtes à être exploitées.

Le chef d'exploitation imprime alors 2 documents sur le mois écoulé :

- l'état mensuel des travaux pour chaque salarié (qui est remis à chaque salarié)
- l'état des opérations phytosanitaires (état Terravitis)

En fin d'année, le chercheur analyse toutes les opérations saisies et rédige une synthèse générale sur les temps de travaux dans les différentes exploitations. Cette synthèse est alors transmise à tous les chefs d'exploitation.

Exemple d'impression Terravitis :

LISTE DES INTERVENTIONS

| No intervention | Parcelles | Opérateur | Date | Opération | Durée | Matériel | Maladies visées (stade phéno) | Méthode de traitement | Observations |
|--------------------------|--|------------------|------------|---------------------------|-------|---------------------------------|-------------------------------|-----------------------|--|
| Modifier | Merlots, Blancs, Petits verdots et Cabernets Franc | Pichot Yann | 11-05-2006 | TraITEMENT phytosanitaire | 9.00 | Enjameur Bobard 896 Pulvé Hardy | Mildiou Oidium (?) | Face / Face Localisé | 5 rangs traités par passage |
| Modifier | Cabernets Sauvignons | Strubel Frédéric | 17-05-2006 | TraITEMENT phytosanitaire | 8.00 | Enjameur Bobard 896 Pulvé Hardy | Mildiou Oidium (?) | Face / Face | 5 rangs traités par passage |
| Modifier | Merlots, Blancs, Petits verdots et Cabernets Franc | Pichot Yann | 26-05-2006 | TraITEMENT phytosanitaire | 8.00 | Enjameur Bobard 896 Pulvé Hardy | Mildiou Oidium (?) | Face / Face | stade 21 / 5 rangs traités par passage |
| Modifier | Cabernets Sauvignons | Pichot Yann | 02-06-2006 | TraITEMENT phytosanitaire | 8.00 | Enjameur Bobard 896 Pulvé Hardy | Mildiou Oidium (?) | Face / Face | stade 21 / 5 rangs traités par passage |
| Modifier | Parcelles de multiplication | Pichot Yann | 06-06-2006 | TraITEMENT phytosanitaire | 0.00 | Enjameur Bobard 896 Pulvé Hardy | Fd (?) | Face / Face | stade 21 / 5 rangs traités par passage |

Exemple d'un état mensuel :
 RELEVE MENSUEL POUR Strubel Frédéric pour le mois de Avril 2006

| Jour | Operations | Total heures |
|------|-----------------------------------|--------------|
| 1 | | 0 h |
| 2 | | 0 h |
| 3 | Taille : 24.00 h | 24 h |
| 4 | Taille : 15.00 h | 15 h |
| 5 | Taille : 24.00 h | 24 h |
| 6 | Tirage des bois : 24.00 h | 24 h |
| 7 | Pliage/Liage : 32.00 h | 32 h |
| 8 | | 0 h |
| 9 | | 0 h |
| 10 | Pliage/Liage : 24.00 h | 24 h |
| 11 | Pliage/Liage : 16.00 h | 16 h |
| 12 | Decavallonnage mecanique : 5.00 h | 5 h |

8) Les emprunts dans une bibliothèque universitaire *(NIVEAU Moyen)*

Il s'agit de spécifier un système d'information de gestion des emprunts dans une bibliothèque universitaire. La bibliothèque prête des livres aux emprunteurs (étudiants et personnel de l'université) inscrits à la bibliothèque. Lors de l'inscription à la bibliothèque, une vérification de l'identité du nouvel emprunteur est faite auprès du SI (existant) de l'université.

Les livres les plus "populaires" peuvent exister en plusieurs exemplaires.

Il est possible de réserver des exemplaires de livres. Les réservations peuvent également être annulées, soit explicitement (via une procédure spécifique), soit le jour suivant la date de la réservation, soit lorsque l'emprunteur vient chercher ce qu'il a réservé. Les emprunts sont payants pour tous.

Si hypothèses et exigences doivent être précisées pendant la résolution, notez les hypothèses/exigences que vous avez dû faire pour poursuivre votre travail. On commencera par traiter la problématique des emprunts sans réservation (le cas le plus simple).

Le Diagramme d'Activité

9) Identité des enfants

L'obtention d'une carte d'identité pour enfant de moins de 12 ans, d'un certificat d'identité provisoire et d'un passeport demandent un certain nombre de démarches décrites dans le document ci-après. Modélez ces démarches.

- Considérer toute l'administration comme étant le système (*NIVEAU facile*).
- Considérer tous les services de manière indépendante (*NIVEAU Moyen*).



V/Cor : service population
Tél : 02.279.35.60
02.279.60.61

Baudoin
Stéphanie

Rue Saint-Quentin, 22
1000 Bruxelles

IMPORTANT

Bruxelles, le 26 juin 2009

Madame, Monsieur,

Concerne: **Document d'identité pour enfants belges de moins de 12 ans (kids-eID)**
Demande urgente ou très urgente de documents d'identité
Certificat d'identité provisoire
Passeport pour enfant

En complément des annonces publiques qui ont déjà été faites, je souhaite vous rappeler les procédures relatives au document pour enfants belges de moins de 12 ans (la kids - eID) aux demandes urgentes ou très urgentes de documents d'identité, au certificat d'identité provisoire, et au passeport pour enfant.

Ceci dit, si votre enfant possède un document de voyage encore valable (ancien document carton), un document d'identité électronique n'est pas obligatoire.

La kids-eID

Sur décision du Ministre de l'Intérieur, depuis le 27 avril 2009, le document d'identité électronique pour enfants belges de moins de 12 ans (kids-eID) peut être demandé au service « Population ». Cette carte électronique remplace l'ancien document papier. Ce document d'identité électronique est inspiré de la carte pour adultes (eID). Il constitue un document de voyage valable dans la plupart des pays européens et certains pays hors Union Européenne (s'informer auprès de son agence de voyage ou de l'ambassade du pays). Il est difficilement falsifiable et donc nettement plus sécurisé que le certificat d'identité actuel. En outre, il offre des services supplémentaires au citoyen grâce à sa puce électronique.

La Kids-eID est délivrée à la demande d'une personne ayant autorité parentale sur l'enfant concerné. La présence de l'enfant est obligatoire ! Il faudra une photo récente sur fond blanc. La Kids-eID coûte 5 euros et a une durée de validité de trois ans maximum. Il faut d'abord faire la demande de la carte et tenir compte d'un délai de 3 à 4 semaines pour obtenir du Service Public Fédéral Intérieur les codes personnels liés à la carte commandée. Dès que vous recevez ces codes à domicile par courrier, vous pourrez retirer la Kids-eID au service « Population ». Demandez donc votre carte à temps si vous comptez partir en vacances !

La demande et le retrait de la carte d'identité électronique pour enfant belge peuvent se faire au Centre Administratif sis au n° 6 boulevard Anspach et dans tous les bureaux de liaison (sauf demande de carte urgente ou très urgente – voir ci-dessous).

Demande urgente ou très urgente de documents d'identité

Pour les pays où le certificat d'identité provisoire n'est pas accepté (s'informer auprès de son agence de voyage ou de l'ambassade du pays), une demande de procédure urgente (délivrance en 5 jours ouvrables – coût : 100 Euros) ou très urgente (délivrance en 3 jours ouvrables – coût : 150 Euros) pour l'obtention d'une carte d'identité électronique est possible. Ces demandes se font uniquement au Centre Administratif, service « Population » (2^e ét.).

VILLE DE BRUXELLES
DEMOGRAPHIE
POPULATION



STAD BRUSSEL
DEMOGRAFIE
BEVOLKING

Le certificat d'identité provisoire

Au cas où vous n'auriez pas fait la demande de la kids-eID dans les délais et que vous vous rendez dans un pays où le passeport n'est pas exigé (s'informer auprès de son agence de voyage ou de l'ambassade du pays), un certificat d'identité provisoire dont la durée de validité est limitée à deux mois pourra être délivré **gratuitement** par le Service Public Fédéral Intérieur.

Toutefois, avant de faire la démarche auprès du SPF Intérieur pour le certificat d'identité provisoire, vous êtes tenu de vous présenter auprès du service population pour commander une kids-eID et obtenir le document permettant de faire la demande de certificat d'identité provisoire auprès du SPF Intérieur, Direction générale - Institutions et Population (Park Atrium, rue des Colonies, 11 à 1000 Bruxelles tél. : 02/518 21 81) Vous pouvez également consulter le site web www.ibz.rn.fgov.be sous la rubrique Documents d'identité et cartes électroniques->'carte d'identité provisoire'.

Passeport pour enfant

Pour les pays qui n'acceptent pas la kids-eID (s'informer auprès de son agence de voyage ou de l'ambassade du pays), vous pouvez demander un passeport au service « Passeports » (2^{ème} étage).

Le passeport est délivré à la demande d'une personne ayant autorité parentale sur l'enfant concerné. La présence de l'enfant est obligatoire ! Il faudra deux photos identiques récentes sur fond blanc. Le passeport pour mineur (0-18) coûte 53,50 euros et a une durée de validité de cinq ans maximum. Il faut d'abord faire la demande du passeport et tenir compte d'un délai de 10 jours ouvrables pour la délivrance.

Alors que la demande d'un passeport ne peut se faire qu'au Centre Administratif, vous pouvez retirer le passeport au Centre Administratif mais aussi dans tous les bureaux de liaison.

Pour les adresses et heures d'ouverture des bureaux de liaison et pour toute information complémentaire, je vous invite à consulter le site web de la Ville de Bruxelles : www.bruxelles.be

Veuillez agréer, Madame, Monsieur, l'assurance de notre considération distinguée.

Le Secrétaire de la Ville

L'Echevin de l'Etat Civil

Luc SYMOENS

Hamza FASSI - FIHRI

10)Le fonctionnement d'un distributeur de billets

(*NIVEAU Moyen*)

Décrire le fonctionnement d'un distributeur de billets. Le client introduit sa carte dont la validité est immédiatement vérifiée. Il est ensuite invité à saisir le code de la carte. Après trois tentatives infructueuses, la carte est avalée. Sinon le client peut indiquer le montant qu'il désire retirer, le solde de son compte bancaire est alors consulté pour s'assurer que le retrait est possible. En cas de solde insuffisant, le client en est informé et peut alors saisir un montant inférieur. Si le solde du compte est suffisant, le distributeur restitue la carte et délivre alors les billets accompagnés d'un reçu.

11)Connexion à Telnet

(*NIVEAU Difficile*)

Décrire la connexion d'un client à un serveur *telnet*. On considère trois protagonistes: le client, le démon *telnet* (i.e. le serveur logiciel) et la machine serveur. Une fois la connexion établie entre le client et le serveur, le démon demande un mot de passe au client, ce dernier dispose de trois tentatives avant que la connexion ne soit rompue. Les tentatives infructueuses sont enregistrées dans un fichier sur le serveur. Une fois l'identification faite, un terminal est ouvert et l'utilisateur peut alors saisir des commandes qui sont interprétées par le démon et exécutées sur le serveur. La commande *exit* déconnecte le client du serveur.

12)Gestion des sinistres dans une compagnie d'assurance

(*NIVEAU Difficile*)

Les déclarations de sinistres peuvent soit provenir directement de l'assuré, soit provenir d'un bureau de courtage agissant comme intermédiaire entre le client et la compagnie. Une déclaration de sinistre provenant directement d'un client est prise en charge par un employé du service de gestion des sinistres qui procède à la vérification de la couverture : il vérifie s'il existe dans le système une police souscrite par ce client et couvrant le risque décrit par le sinistre. Si le sinistre n'est pas couvert, il retourne la déclaration au client accompagnée d'une lettre explicative. S'il est couvert, l'employé enregistre le sinistre dans le S.I.

Une déclaration provenant d'un bureau de courtage n'est pas vérifiée au sein de la compagnie (le bureau de courtage s'en est chargé), elle est directement enregistrée.

Les déclarations qui ont été enregistrées restent sur le bureau de l'employé, jusqu'au moment où elles sont prises en charge par un adjoint au directeur. Cette prise en charge a lieu à 10h, 12h, 15h et 16h30. L'adjoint du directeur analyse la déclaration et décide sur base de l'appréciation des responsabilités d'indemniser le client ou de demander la contribution du service d'expertise.

Dans le premier cas (indemnisation du client), il transmet au client, pour approbation, une lettre lui annonçant le montant de l'indemnisation qui lui sera octroyée. Si la réponse du client qu'il reçoit en retour est favorable, il transmet au service financier, une demande d'indemnisation indiquant le montant de l'indemnisation à transmettre au client. La déclaration de sinistre correspondante et la réponse favorable du client sont transmises en fin de journée au service de classement qui les rangera dans le dossier du client. Si la réponse du client est défavorable, l'adjoint constitue un dossier comprenant

la déclaration d'accident et la lettre du client. Le dossier sera transmis au directeur qui se chargera du litige.

Dans le second cas (intervention du service d'expertise), la déclaration de sinistre est reproduite en 3 exemplaires : l'un est conservé par l'adjoint du directeur, un exemplaire (l'original) est transmis au directeur et deux exemplaires sont transmis au service d'expertise qui se chargera de la poursuite de la procédure.

Au service financier, un employé est chargé du traitement des demandes d'indemnisation. Lorsqu'il reçoit une demande d'indemnisation, cet employé met à jour, à l'aide d'un terminal; les informations relatives au sinistre concerné : il indique le montant payé par la compagnie. Cette opération met également à jour le fichier des indemnisations à effectuer : sur base des informations relatives au client et au sinistre une nouvelle opération d'indemnisation est enregistrée. Ce fichier des indemnisations subit un traitement automatique à 12h et 16H ou dès que 10 opérations d'indemnisation y ont été enregistrées. Ce traitement consiste à produire un listing reprenant, pour chaque client, le montant qui lui est dû et à supprimer le contenu du fichier. Sur base de ce listing, un employé du service financier rédige les chèques correspondants qui sont transmis, pour signature, à l'adjoint au directeur. Les chèques signés sont envoyés aux clients.

Le Diagramme de Classe

13) Propriétaire de voiture

(NIVEAU facile)

Soient un ensemble de personnes et un ensemble de voitures. Une personne est caractérisée par un numéro qui l'identifie et par les voitures dont elle est l'unique propriétaire. Une voiture est caractérisée par un numéro de plaque, une marque et une date de mise en circulation.

14) Classification des employés

(NIVEAU facile)

Les différents départements d'une entreprise occupent des employés. Un employé est décrit par son numéro matricule (unique dans l'entreprise), son nom, son grade et le département dans lequel il travaille. Un département est décrit par son numéro dans l'entreprise et par son directeur qui doit être un de ses employés.

15) Organisation des départements

(NIVEAU facile)

Toujours dans une organisation départements/employés, un département est identifié par un numéro et caractérisé par une localisation ; un employé est caractérisé par un numéro (unique dans son département mais pas dans l'entreprise), son nom, son grade et le département dans lequel il travaille.

16) Le complexe de cinéma

(NIVEAU Moyen)

Un exploitant possède un complexe de cinéma qui comporte plusieurs salles. Un film peut généralement être au programme de plusieurs séances par jour et être projeté dans plusieurs salles. Une séance est caractérisée par une tranche horaire, les séances sont communes à toutes les salles du complexe. Décrire un schéma qui permette à l'exploitant d'obtenir des renseignements sur le chiffre d'affaire d'un film, d'une salle ou d'un jour particulier.

17) Les types de pièces

(NIVEAU facile)

Définir un schéma décrivant une nomenclature de type de pièces. Chaque type de pièces est caractérisé par son numéro, les types de pièces qui la composent et ceux dont il est le composant.

18) Les comptes bancaires

(NIVEAU facile)

Soit un ensemble de personnes (identifiées par un numéro et caractérisées par un nom) et un ensemble d'organismes bancaires (identifiés par un numéro) ; une personne peut

ouvrir un ou plusieurs comptes dans un organisme bancaire ; chaque organisme bancaire affecte à chacun de ses comptes un numéro identifiant pour lui seul.

19) Relations familiales

(NIVEAU Moyen)

Définir un schéma décrivant les liens familiaux d'une population de personnes identifiables par leur numéro de registre national.

20) Réservation de vols

(NIVEAU Moyen)

On souhaite gérer les réservations de vols effectuées dans une agence. D'après les interviews réalisées avec les membres de l'agence, on sait que :

- Les compagnies aériennes proposent différents vols
- Un vol est ouvert à la réservation et refermé sur ordre de la compagnie
- Un client peut réserver un ou plusieurs vols, pour des passagers différents
- Une réservation concerne un seul vol et un seul passager
- Une réservation peut être confirmée ou annulée
- Un vol a un aéroport de départ et un aéroport d'arrivée
- Un vol a un jour et une heure de départ, et un jour et une heure d'arrivée
- Un vol peut comporter des escales dans un ou plusieurs aéroport(s)
- Une escale a une heure de départ et une heure d'arrivée
- Chaque aéroport dessert une ou plusieurs villes

Ajoutez tout attribut que vous jugez pertinent et qui n'a pas été décrit ci-dessus !

21) Le magasin d'électroménagers

(NIVEAU Difficile)

Première partie :

Pour améliorer le service à la clientèle, un vendeur d'appareils électroménagers envisage d'automatiser certaines de ses activités. En particulier, il désire faciliter la consultation de son catalogue (Il s'agit d'un catalogue interne destiné à faciliter les approvisionnements).

Le catalogue porte à la fois sur les appareils électroménagers et sur les pièces de rechange. Il reprend tous les types d'appareils et de pièces qui peuvent être fournis aux clients par le vendeur. Les appareils électroménagers sont décrits dans le catalogue par une marque (ex: PHILIPS, SIEMENS, ...), un libellé (ex: lave-vaisselle, surgélateur, ...), un modèle (ex: SP375, XZ374, ...), le prix de vente recommandé, les différents fournisseurs auxquels on peut s'adresser pour les commander ainsi que le prix d'achat et le délai de livraison spécifique à chaque fournisseur. Le prix de vente recommandé dépend exclusivement de l'appareil : il correspond à un prix standard établi par consensus entre tous les fournisseurs susceptibles de fournir ce type d'appareil. Parmi les appareils d'une même marque, certains peuvent avoir le même libellé. Dans ce cas, ils sont nécessairement de modèles différents.

Les pièces de rechange sont caractérisées par un nom (ex: pompe de vidange, joint de porte, ...), un modèle (ex : PP445, RTE-56, ...), un prix unitaire recommandé. On connaît

également les appareils électroménagers pour lesquelles elles peuvent convenir, les différents fournisseurs chez qui on peut se les procurer et pour chacun d'eux le prix d'achat correspondant. Le prix de vente unitaire recommandé dépend exclusivement de la pièce: il correspond à un prix standard établi par consensus entre tous les fournisseurs susceptibles d'approvisionner le vendeur. Le délai de livraison des pièces de rechange est également connu. Contrairement au cas des appareils, ce délai dépend uniquement du fournisseur; pour un fournisseur donné, toutes les pièces qu'il peut fournir le sont dans un délai de livraison unique (fréquence des tournées des camions livreurs). Des pièces différentes peuvent porter le même nom. Dans ce cas, elles ne peuvent être du même modèle.

Pour chaque appareil et chaque pièce repris au catalogue, on connaît au moins un fournisseur. Un fournisseur d'appareil peut également fournir des pièces et vice-versa. Il n'est connu du vendeur que s'il est susceptible de l'approvisionner (en pièces et/ou appareils). Il est possible que pour certains appareils aucune pièce de rechange ne puisse être fournie. Une pièce de rechange n'est répertoriée que si elle convient à au moins un type d'appareil.

Afin de joindre les fournisseurs facilement, on gardera trace de leur nom, adresse, numéro de téléphone et de Fax. En plus d'être identifiés par leurs nom et adresse, les fournisseurs sont aussi identifiés par un code de référence propre à l'entreprise.

Seconde partie :

Le vendeur d'appareils désire également faciliter la consultation de son stock. Il dispose de deux types de stock: un stock d'appareils et un stock de pièces de rechange.

Pour chaque appareil électroménager en stock on connaît sa marque (ex: PHILIPS, SIEMENS, ...), sa nature (ex: lave-vaisselle, surgélateur, ...), son modèle (ex: SP375, XZ374, ...), sa date de fabrication en usine, sa date d'entrée en stock, le code de référence du fournisseur chez qui l'appareil a été acheté, le prix d'achat (prix que le vendeur a payé au fournisseur), sa localisation (en magasin ou en entrepôt) et un n° d'appareil qui permet de l'identifier parmi tous les appareils de même marque, de même nature et de même modèle. Pour chaque fournisseur on connaît son numéro de téléphone ainsi que ses périodes de fermeture.

Pour chaque pièce de rechange en stock, on connaît le nom (ex: pompe de vidange, joint de porte, ...), le modèle (ex : PP445, RTE-56, ...), le prix d'achat unitaire (qui peut être différent du prix recommandé) et un numéro identifiant attribué par le vendeur.

Il est à noter que :

- Les appareils (respectivement, les pièces de rechange) se trouvant en stock correspondent nécessairement à des types d'appareils (respectivement, de pièces de rechange) répertorié(e)s dans le catalogue.
- Les types d'appareils et de pièces de rechange au catalogue n'ont pas nécessairement d'équivalent dans les stocks. En effet, tout appareil ou pièce au catalogue qui ne se trouverait pas en stock peut-être commandé par le vendeur pour répondre aux besoins particuliers des clients.

Le Diagramme de Séquence

22) La conversation téléphonique

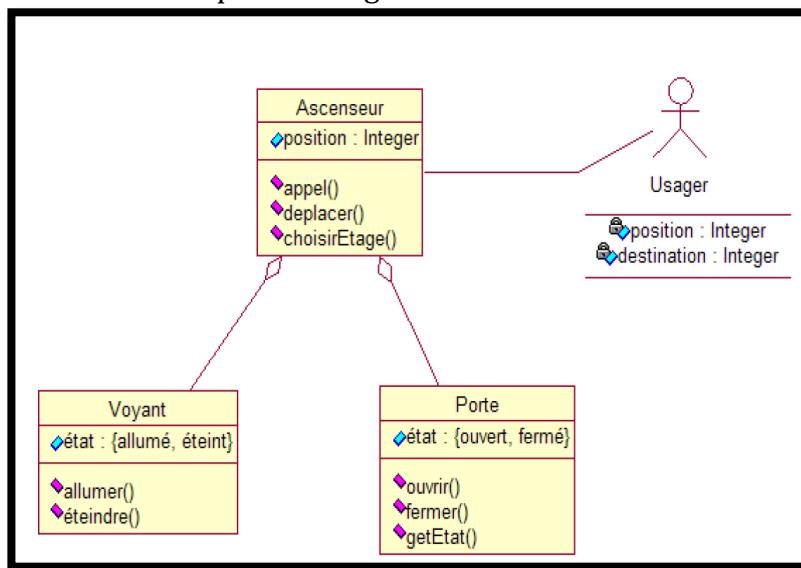
(NIVEAU Facile)

Décrire par un diagramme de séquences une conversation téléphonique (je décroche, tonalité, je compose le numéro, sonnerie...).

23) L'ascenseur

(NIVEAU Facile)

Sur base du diagramme de classes ci-dessus, réalisez le diagramme de séquence pour modéliser le scénario dans lequel un usager voudrait monter en utilisant l'ascenseur.



24) La caisse enregistreuse

(NIVEAU Facile)

Le déroulement normal d'utilisation d'une caisse de supermarché est le suivant :

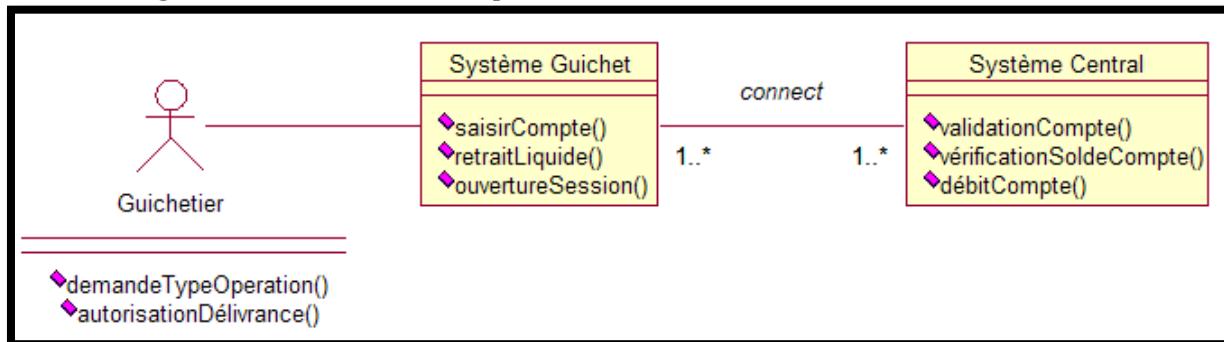
- un client arrive à la caisse avec ses articles à payer
- le caissier enregistre le numéro d'identification de chaque article, ainsi que la quantité si elle est supérieure à 1
- la caisse affiche le prix de chaque article et son libellé
- lorsque tous les achats sont enregistrés, le caissier signale la fin de la vente
- la caisse affiche le total des achats
- le caissier annonce au client le montant total à payer
- le client choisit son mode de paiement
 - liquide : le caissier encaisse l'argent, la caisse indique le montant à rendre au client
 - carte de crédit : la demande d'autorisation est envoyée avant la saisie
- la caisse enregistre la vente et l'imprime
- le caissier donne le ticket de caisse au client

25) Le retrait d'argent

(NIVEAU Moyen)

Le guichetier ouvre une session et saisit le numéro de compte du client ; après validation par le système central, le guichetier spécifie le type d'opération à l'invite du système guichet et sélectionne le montant du retrait. Le système central vérifie si le compte est suffisamment approvisionné à la demande du système guichet ; si tel est le cas, ce dernier demande au système central de débiter le compte et notifie au guichetier qu'il peut délivrer le montant demandé.

Voici le diagramme de classe correspondant :



26) Assistance téléphonique

Donner le diagramme de séquences correspondant au scénario suivant. Un utilisateur désire poser des questions à une assistance téléphonique. Soit un opérateur décroche dans les 10 secondes de l'appel téléphonique, et à ce moment il dialogue directement avec l'utilisateur; soit aucun opérateur n'est disponible et les 10 secondes s'écoulent, l'utilisateur est alors basculé sur un serveur vocal qui va enregistrer ses questions. Un opérateur disponible pourra ensuite consulter le serveur vocal, écouter les questions et, après réflexion, rappeler l'utilisateur. Entre le moment du premier appel et les réponses aux questions, il ne doit pas s'écouler plus d'une heure.

27) Projet de recherches en viticulture (diagramme de séquence)

Reprendre dans les exercices sur les Use Cases le cas : « Projet de recherches en viticulture » et réaliser le diagramme de séquence correspondant.

Le Diagramme d'Etat

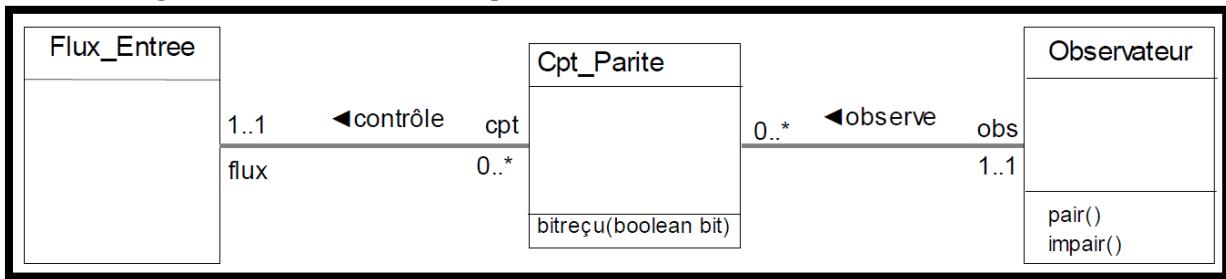
28) La parité (NIVEAU Facile)

Un objet doit maintenir la parité des bits reçus sur un flux d'entrée, en comptant le nombre de 1 reçu. Lorsque le nombre de 1 reçu est pair, il doit envoyer un événement "pair" et il doit envoyer vers un observateur un événement "impair" lorsque le nombre de 1 reçu est impair. De plus, l'objet ne peut contenir de variable entière.

Il faut présenter deux machines de comportement modélisant cet objet :

1. Une qui utilise des actions pour produire son résultat.
2. Une qui utilise des "activity entry".

Voici le diagramme de classe correspondant :



29) La boîte de vitesse automatique (NIVEAU Facile)

On considère une boîte de vitesse automatique de voiture. La boîte au démarrage est au point mort. La marche arrière ainsi que la position *parking* peuvent être enclenchées à partir du point mort. La première marche avant peut également être enclenchée à partir du point mort. En revanche, les autres marches avant, la seconde et la troisième, sont enclenchées en séquence: 1 -> 2 -> 3 pour une accélération, et 3 -> 2 -> 1 pour une décélération. Seule la marche arrière, la position *parking* et la première marche avant peuvent être ramenées directement au point mort.

30) La montre digitale

(NIVEAU Moyen)

Partie 1 :

On désire modéliser le mécanisme d'une montre digitale. Une montre digitale simple comporte un affichage et deux boutons de réglage. On considère pour l'instant la montre avec deux modes de fonctionnement (affichage et réglage). Le mode réglage possède deux sous-modes (réglage des minutes et réglage des heures). Le bouton A est utilisé pour changer de mode, ce qui s'effectue de manière cyclique :

affichage -> réglage minutes -> réglage heures -> affichage

Dans les deux sous-modes de réglage, le bouton B permet d'augmenter d'une minute ou d'une heure chaque fois qu'il est appuyé. On ajoute ensuite les modes chronomètre et alarme à la montre. L'alarme se programme avec le bouton B (de la même manière que le réglage simple de la montre). Le chronomètre est lancé et stoppé également avec le bouton B. Le passage d'un mode à l'autre s'effectue toujours avec le bouton A :

affichage -> réglage -> alarme -> chronomètre -> affichage

Le chronomètre fonctionne en parallèle avec les autres modes, et l'alarme possède un état interne (activée ou désactivée), indépendant des autres états, qui se règle avec le bouton B.

Partie 2 :

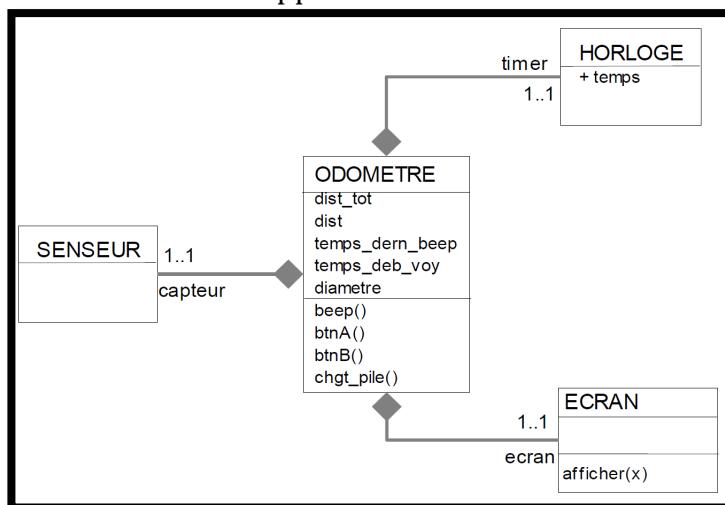
On rajoute un bouton C à la montre pour étendre les fonctionnalités du chronomètre. Le bouton B sert alors à la mise en route, à l'arrêt et à la reprise du compteur. Le bouton C permet de suspendre ou de reprendre l'affichage, il remet également le compteur à zéro si le chronomètre est arrêté.

31) L'odomètre (NIVEAU Moyen)

L'odomètre est un instrument placé sur un vélo et permettant de mesurer la vitesse actuelle et la distance parcourue. Il y a deux boutons, A et B.

- Initialement, l'odomètre est en mode *Distance*. Chaque fois que le bouton A est pressé, l'odomètre passe successivement en mode *Vitesse*, *Distance Totale*, *Temps* et *Distance*.
- En mode *Distance*, la distance à partir du début du voyage actuel est affichée toutes les 200 ms avec une précision de 100 mètres. Le bouton B permet de remettre la distance à zéro.
- En mode *Vitesse*, la vitesse actuelle est affichée toutes les 100 ms.
- En mode *Distance Totale*, la distance totale est affichée toutes les 500 ms, avec une précision d'1 km. La *Distance Totale* est remise à zéro lors d'un changement de piles.
- En mode *Temps*, le temps depuis le début du voyage actuel est affiché toutes les secondes.

Voici le diagramme de classe de cet appareil :



Quelques questions supplémentaires :

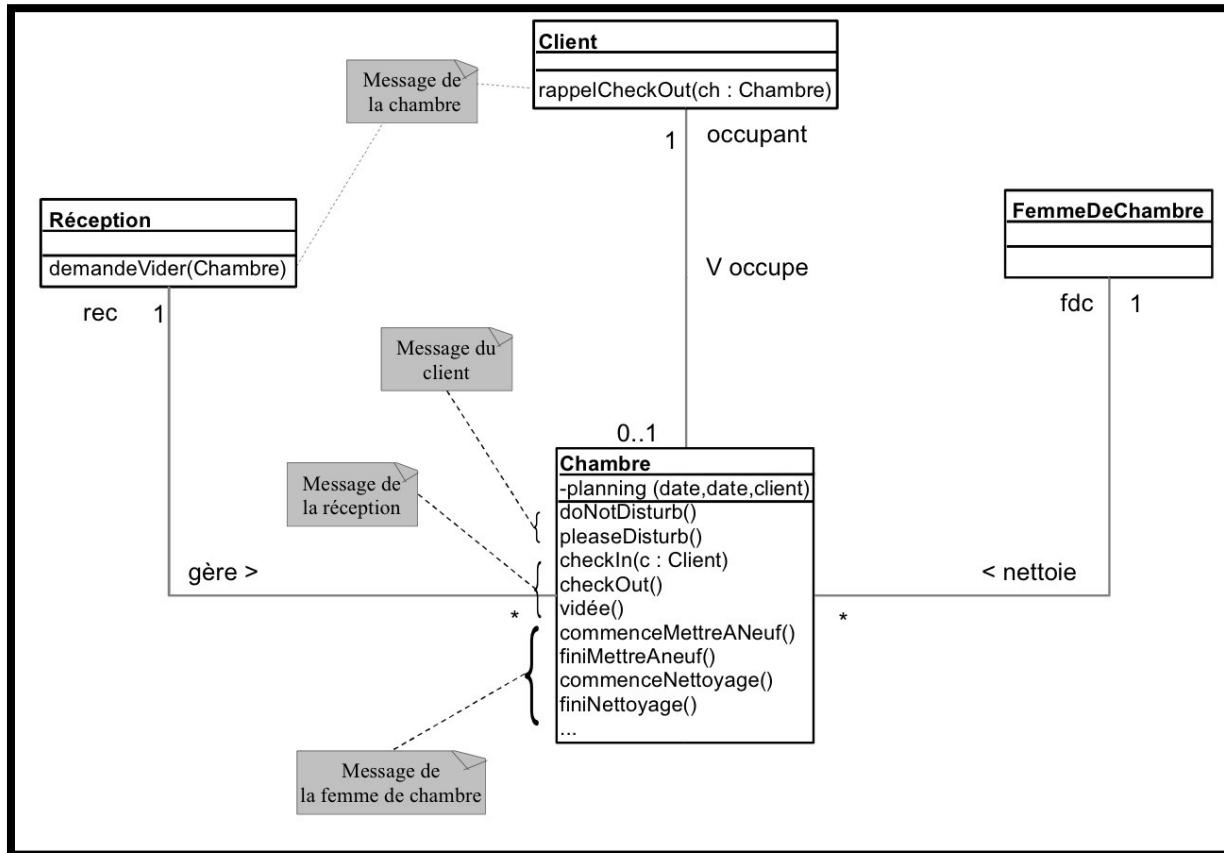
Pouvez-vous imaginer la modélisation du comportement du même odomètre :

- Si celui-ci est équipé d'un bouton On/Off (notez qu'après une mise à On, il revient sur le même écran que lors de son arrêt).
- Si l'affichage de la distance ne s'effectue plus toutes les 200 ms mais après avoir parcouru 100 mètres

32) Gestion des chambres d'un hôtel

(NIVEAU Difficile)

Dans le cadre du développement d'un SI de gestion hôtelière, vous devez spécifier le comportement des objets de la classe dont voici le schéma (diagramme de classe) :



planning est un triple <date arrivée prévue, date départ prévue, nom client>. Les méthodes spécifiées dans ce diagramme de classe sont "vides" : elles ne font qu'envoyer des messages entre objets sans aucun effet sur les valeurs des attributs de ces objets.

En plus, il faut savoir que toutes les chambres de l'hôtel sont 'single'.

Check-in/check-out :

Pour le check-in, un client n'est admis que si la chambre est libre, qu'elle a été mise en ordre et qu'il est prévu dans le planning de la chambre que le client arrive aujourd'hui (voir ci-dessous). De plus, les clients ne sont admis au check-in qu'entre 14h00 et 23h00. Si, à 23h00, un client dont l'arrivée était planifiée n'a toujours pas effectué son check-in, sa planification est tout simplement annulée (c.à.d. effacée du SI). Notez que c'est le seul cas où une planification peut-être effacée du SI.

Les check-out s'effectuent normalement entre 4h00 et 12h00. Une marge de tolérance est cependant admise jusque 13h00. Un client n'est admis au check-out que le jour planifié de son départ. On suppose, par simplification, que les départs anticipés ne sont pas admis dans cet hôtel. De même, les demandes de prolongation de séjour ne seront pas supportées par le système à créer.

Le seul cas où une prolongation est effectuée : si à 13h00, le jour du départ prévu d'un client, la chambre n'est toujours pas libérée et qu'il n'est pas prévu qu'un autre client l'occupe le même jour, alors une prolongation automatique d'un jour du séjour du dudit client est effectuée. Si, par contre, il est prévu que la chambre soit occupée par un autre

client ce jour-là, la chambre est alors vidée à 13h00 : une demande de vider la chambre est envoyée à la réception ; la réception signalera quand ce sera fait. Cette procédure est sensée être exceptionnelle. Dès lors, pour éviter d'en arriver à cet extrême, des rappels sont envoyés aux clients occupants une chambre devant être libérée le jour même. Ces rappels sont envoyés à 11h et à 12h si le check-out n'a pas déjà été effectué.

Nettoyage et remise en ordre des chambres :

Pendant le séjour d'un client, les chambres sont normalement nettoyées (nettoyage du sol, sanitaire et changer le linge) tous les jours. On suppose que les femmes de chambre ne viennent effectuer leurs tâches qu'à des heures raisonnables.

Une chambre ne peut être occupée par un nouveau client (check-in) que si elle a été remise en ordre. Il s'agit d'un nettoyage plus complet que le simple nettoyage journalier : en plus, on remet des savons, des serviettes, des sucreries pour l'accueil, etc. Une chambre ne peut être *remise en ordre* qu'entre le check-out d'un client (forcé ou non) et l'arrivée du client suivant.

Une chambre ne peut être nettoyée ou remise à neuf si le client qui l'occupe a demandé à ne pas être dérangé. Si à 14h00 une chambre devait être nettoyée, qu'elle ne l'a pas été et qu'elle ne peut l'être car le client a demandé à ne pas être dérangé, alors cette chambre est considérée comme nettoyée. Cela n'est évidemment pas possible s'il y a un changement de client dans la chambre en question.

Notez qu'il ne faut pas prendre en compte le processus de réservation dans votre modélisation.

Exercice Intégré

33) Système d'information d'une société de location de voitures

Une société de location de voitures souhaite informatiser son système de réservation et de facturation. Cette société opère à partir d'une seule localisation. Les voitures sont prises et ramenées à cette localisation.

Dans sa flotte de véhicules, la société dispose de plusieurs modèles. Afin de pouvoir au mieux satisfaire le client, la société considère qu'il est important de pouvoir lui fournir des renseignements sur les options disponibles sur certains modèles, comme la boîte à vitesse automatique, l'autoradio, 2 ou 5 portes, climatisation, ... Afin de standardiser ces options, la société leur a affecté en plus d'un libellé (texte libre), un code unique. Outre ces informations, les modèles sont caractérisés et identifiés par une marque (ex: Ford, VW, Toyota, ...), un type (Ex: Escort, Golf; Corolla, ...) et une puissance. Ils sont groupés dans des classes de tarification. Chaque modèle référencé contient au moins une voiture. Une voiture est caractérisée par un numéro de véhicule propre à la société (et unique dans la société) sa date et son prix d'achat, s'il est en réparation ou en entretien la date de restitution prévue. Pour chaque classe de tarification, en plus d'un code identifiant, on connaît quels sont les contrats d'assurances souscrits pour les véhicules correspondants aux modèles couverts par cette classe. Un contrat d'assurance est caractérisé par un type (ex: bris de glace, vol, responsabilité civile, ...) et un assureur dont on connaît les coordonnées : nom, adresse, n° téléphone et n° fax (il n'existe pas deux assureurs ayant même nom et même adresse). Tous les contrats d'assurance d'un même type sont réalisés auprès d'un même assureur. Un assureur peut gérer plusieurs types de contrats.

Trois formules de locations sont offertes au client :

- la location à la journée (de l'heure H du jour J, à l'heure H du jour J+1)
- la location à la semaine (de l'heure H du jour J de la semaine S à l'heure H du jour J de la semaine S+1)
- la location de week-end (du Vendredi après 19 heures au Lundi avant 7H30)

A chaque formule de location correspond un kilométrage forfaitaire. Celui-ci précise le nombre de kilomètres que le client peut effectuer sous le couvert du montant forfaitaire appliqué à ce mode location. Le kilométrage forfaitaire ne dépend que de la formule de location, tandis que le montant forfaitaire dépend non seulement du mode de location, mais également de la classe de tarification correspondante. A chaque classe de tarification est associé un prix au kilomètre, une amende journalière ainsi qu'un code unique. Le prix au kilomètre correspond au montant qui sera facturé au client pour chaque kilomètre qu'il aura parcouru au delà du kilométrage forfaitaire spécifié par la formule de location choisie. L'amende journalière correspond au supplément qui lui sera journallement facturé s'il ne restitue pas le véhicule à la date de restitution prévue.

Un client est connu de la société par son nom, prénom et adresse, un numéro identifiant est attribué par compostage à chaque nouveau client.

Lorsqu'un client s'adresse au service réservation, il spécifie ses différentes contraintes (ex: options souhaitées, modèle, formule de tarification, dates de retrait et restitution du véhicule). Par l'utilisation du système, l'employé devrait être capable de le renseigner sur la possibilité de répondre à sa demande en fonction de la disponibilité

des véhicules. Il est évident que si cela s'avère impossible, le client peut réduire ses contraintes. Dans ce cas le processus de vérification est reconduit.

Au terme de cette analyse, la liste des véhicules disponibles répondants aux désiderata du client lui est fournie. Elle stipule également, pour chaque véhicule, un **prix indicatif**¹ que le client est susceptible de payer pour la location, s'il ne dépasse le kilométrage maximum couvert par le tarif forfaitaire et s'il restitue le véhicule avant ou à la date prévue. Sur base de ces indications le client précise le véhicule qu'il désire louer. La réservation du véhicule (comprenant éventuellement la création d'un nouveau client) est alors enregistrée par l'employé, chaque nouvelle réservation se voit attribuer un numéro par compostage. Une réservation porte sur un seul véhicule, sur un seul mode de location et est relative à un seul client. A la fin de l'enregistrement de la réservation, le client reçoit un bon de réservation qui stipule ses choix (dates, véhicule et prix indicatif).

Quand un client se présente, avec son bon de réservation, pour retirer un véhicule réservé, un réceptionniste du garage vérifie (de visu) que le véhicule est effectivement disponible. Il se peut en effet qu'un client ait omis de se présenter à la date de restitution prévue et que de ce fait, le véhicule promis (réservé) ne soit pas disponible.

Si le véhicule est disponible, le réceptionniste procède à l'enregistrement de la location et à l'impression en double exemplaire d'un contrat de location que le client doit signer. Un exemplaire est remis au client avec les clefs du véhicule, tandis qu'un autre est archivé en fin de chaque journée. Sinon, le client est invité à se présenter au service de réservation ou la procédure de réservation est éventuellement reconduite (analyse de la demande et s'il y a lieu enregistrement d'une nouvelle réservation). Dans ce cas, il bénéficie d'un dédommagement équivalent à l'amende journalière prévue pour la catégorie de modèles à laquelle appartient le véhicule qu'il avait réservé. S'il est possible de lui fournir un véhicule, il peut alors retirer le véhicule réservé (et disponible) auprès du réceptionniste du garage.

L'enregistrement d'une location consiste à mémoriser le kilométrage du véhicule lors du retrait, le numéro de permis du conducteur et éventuellement le paiement de la caution. Une caution équivalente à 20 % du prix indicatif est demandée lors du retrait du véhicule. Si celle-ci n'est pas payée à ce moment, la facture est majorée d'un montant correspondant à 3 % du prix indicatif.

Lorsqu'un client se présente pour restituer un véhicule, un réceptionniste au garage vérifie l'état du véhicule et clôture la location : il stipule le kilométrage du véhicule lors de la restitution, la date effective de restitution et s'il y a lieu émet un message à destination de la personne responsable du service de gestion des réparations qui se chargera de la procédure à réaliser en cas de dommages aux véhicules (contact et suivi des assureurs, contact et suivi garages, contact des clients auxquels avaient été promis ce véhicules)². Il remet au client un formulaire stipulant les caractéristiques nécessaires à la clôture de la réservation (identification du véhicule et kilométrage courant).

Après avoir restitué le véhicule, le client se présente, avec ce formulaire, au service des paiements. Ce service élaboré la facture reproduite en deux exemplaires. Un exemplaire est remis au client tandis que l'autre est conservé dans le service. Les factures, non payées immédiatement, sont payables dans les 15 jours. La facturation doit

¹ Ce prix indicatif est donc équivalent au montant forfaitaire correspondant à la formule de location choisie et à la classe de tarification à laquelle appartient le véhicule.

² Le SI ne doit actuellement pas supporter la réalisation de cette procédure.

respecter les règles stipulées dans l'énoncé. Elle doit tenir compte du paiement ou non de la caution, du montant forfaitaire correspondant à la réservation, du dédommagement éventuel (cas où la voiture réservée n'est pas disponible) et s'il y a lieu, des suppléments résultants d'une part du dépassement du kilométrage maximum couvert par le montant forfaitaire et d'autre part, de l'omission de restitution du véhicule à la date prévue. Il n'y a pas de réduction si le véhicule est :

- restitué avant la date prévue
- retiré entre après la date de retrait prévue et la date de restitution prévue

Dans le cas où la société ne peut mettre un véhicule réservé à disposition du client et qu'il n'y a pas de nouvelle réservation, le dédommagement est tout de même dû au client. Le paiement de ce dédommagement n'est pas immédiat, il est effectué par le service de paiement en fin de semaine.

Toute réservation peut être annulée par le client. L'annulation d'une réservation est prise en charge par le service de réservation. Elle ne donne pas lieu à une pénalisation financière si elle est réalisée au moins deux jours avant la date du retrait prévue. Sinon la pénalisation financière correspond au montant de la caution qui aurait été demandée. Le traitement des pénalisations d'annulation s'effectue également par le service de paiement en fin de semaine.

On désire gérer les clients ainsi qu'un certain "historique" des réservations. Ainsi :

- une réservation annulée par la volonté du client ne donne pas lieu à la suppression de la réservation dans le système d'information : on enregistre le fait qu'elle se trouve dans un état "supprimé" et sa date d'annulation.
- une réservation annulée pour des raisons d'indisponibilité du véhicule (véhicule non remis ou en réparation) est dans un état "annulé". Dans ce cas on désire pouvoir retrouver l'éventuelle réservation qui a été faite en remplacement de la réservation annulée.
- une réservation qui a donné lieu à la location d'un véhicule, est considérée comme "terminée" lors de la restitution du véhicule.
- toute réservation non supprimée, non annulée ou non terminée est dans un état "effectif".

Chaque fin mois, les réservations qui ont plus de 6 mois dans le S.I. sont réellement supprimées. On ne gère pas les demandes de réservation en attente (le client est invité à recontacter la société, s'il veut profiter de disponibilités nouvelles).