UML

Nuwan Herath

2022-2023

Cours est basé sur ceux de Laurent Audibert et de Myriam Servières.

Introduction

Des méthodes orientées objet

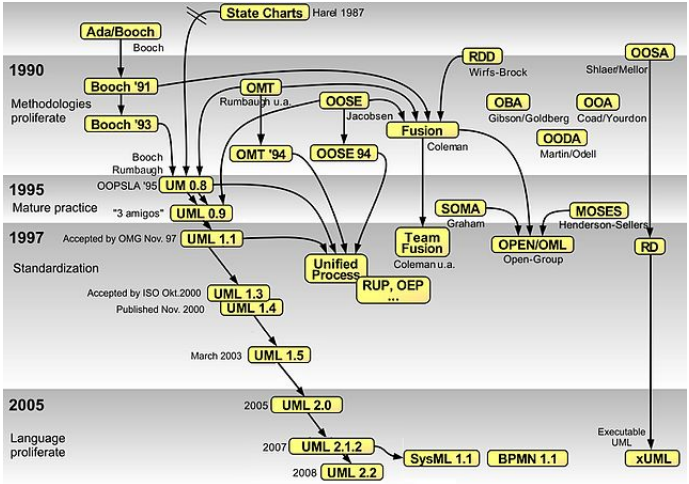
- Les méthodes orientées objet ont été influencées par le développement d'Ada et des langages de programmation basés sur les objets comme C++
- Elles abordent l'étude d'un problèmes suivant :
  - l'aspect statique : identifie les propriétés des objets et leurs liaisons avec les autres objets
  - l'aspect dynamique : définit le cycle de vie des objets (comportement des objets, les différents états par lesquels ils passent, les évènements déclenchant ces changements d'états)
  - l'aspect fonctionnel : précise les fonctions réalisées par les objets par l'intermédiaire des méthodes)

De l'unification des méthodes à UML

- En 1994, Grady Booch (Booch) et James Rumbaugh (OMT) unifient leur travaux pour proposer la méthode unifiée (unified method)
- En 1995, Ivar Jacobson (OOSE, use cases) les rejoint
- Ils se fixent quatre objectifs :
  - représenter des systèmes entiers par des concepts objets
  - établir un couplage explicite entre les concepts et les artéfacts exécutables
  - prendre en compte les facteurs d'échelle inhérents aux systèmes complexes et critiques
  - créer un langage de modélisation utilisable à la fois par les humains et les machines

$\longrightarrow$ Unified Modeling Language (UML)

Aperçu des langages de modélisation



Le formalisme d'UML

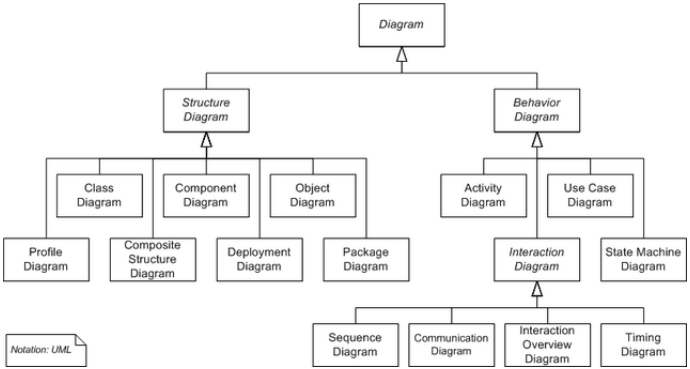
- UML est un langage de modélisation visuel ayant un ensemble de notations
- UML n'est pas une méthode d'analyse ou de conception

Trois catégories principales de diagrammes

Des diagrammes dépendant hiérachiquement et se complétant

- Diagrammes Structurels ou Diagrammes statiques (Structure Diagram)
- Diagrammes Comportementaux ou Diagrammes dynamiques (Behavior Diagram)
- Diagramme d'Interactions (Interaction Diagram)

Hiérarchie des diagrammes



Notations

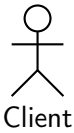
Notations

Diagrammes de cas d'utilisation

Les éléments des diagrammes de cas d'utilisation

Acteur

Ce qui existe en dehors du système
Tout ce qui doit échanger de l'information avec le système : personne, machine, organisation, autre ordinateur, autre système
Correspond à un rôle générique que l'utilisateur joue (une manière d'utiliser le système
La même personne (machine, ...) peut jouer plusieurs rôles
Généralement le bonhomme sert à représenter un acteur humain

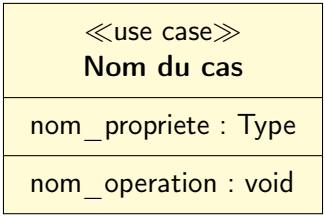


Les éléments des diagrammes de cas d'utilisation

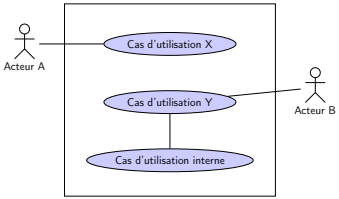
Cas d'utilisation

Description d'un ensemble de séquences d'actions, incluant des variantes, qu'un système effectue pour fournir un résultat observable et ayant une valeur pour un acteur

Si on souhaite représenter des attributs ou des opérations :



Vue d'ensemble

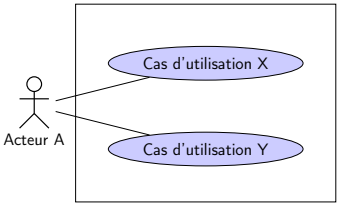


- Représentent les cas d'utilisation, les acteurs et les relations entre les cas d'utilisation et les acteurs
- Décrivent, sous la forme d'actions et de réactions, le comportement d'un système du point de vue d'un utilisateur
- Permettent de définir les limites du système et les relations entre un système et l'environnement

Intérêt

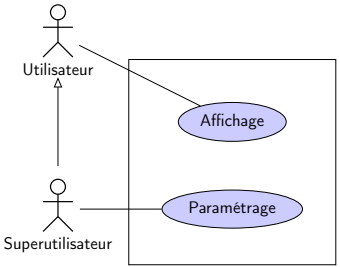
- Constituent un moyen d'exprimer les besoins d'un système
- Utilisés par les utilisateurs finaux pour exprimer leurs attentes et leurs besoins
- Permettent d'impliquer les utilisateurs dès les premiers stades du développement
- Constituent une base pour les tests fonctionnels

Acteurs



- Ce qui existe en dehors du système
- Tout ce qui doit échanger de l'information avec le système : personne, machine, organisation, autre ordinateur, autre système
- Correspond à un rôle générique que l'utilisateur joue (une manière d'utiliser le système)
- La même personne (machine, ...) peut jouer plusieurs rôles

La généralisation d'un acteur

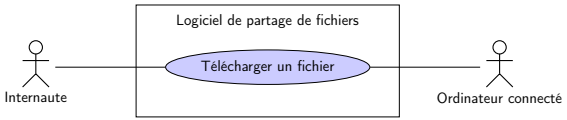


- Un acteur peut également participer à des relations de généralisation avec d'autres acteurs
- Les acteurs « fils » seront alors capables de communiquer avec les mêmes cas d'utilisation que le ou les acteurs parents

Quatre types d'acteurs

- Les acteurs principaux : les personnes qui utilisent les fonctions principales du système
- Les acteurs secondaires : les personnes qui effectuent les tâches administratives ou de maintenance
- Le matériel externe : les dispositifs matériels incontournables qui font partie du domaine d'application et qui doivent être utilisés
- Les autres systèmes : les systèmes avec lesquels le système doit interagir

Les acteurs principaux et secondaires

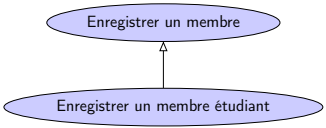


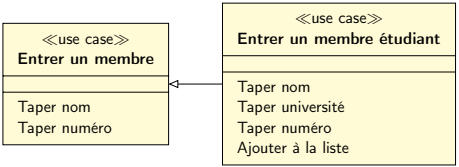
Un acteur principal obtient un résultat observable du système tandis qu'un acteur secondaire est sollicité pour des informations complémentaires

En général, l'acteur principal initie le cas d'utilisation par ses sollicitations

On peut retrouver les annotations « primary » et « secondary »

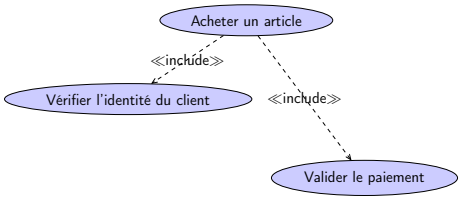
Généralisation d'un cas d'utilisation





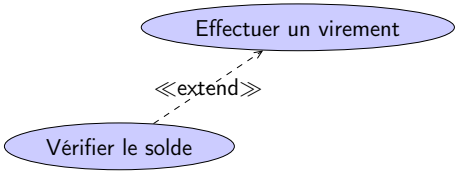
- Extraire le comportement commun
- Le comportement fils doit inclure le comportement parent

Inclusion



- Inclure le comportement d'un autre cas d'utilisation
- Décomposer les comportements complexes
- Obligatoire

Extension



- Chemins alternatifs complexes ajoutés à certains points
- Optionnelle
- Peut être soumis à une condition

Scénario

- Le diagramme de cas d'utilisation décrit les grandes fonctions d'un système du point de vue des acteurs
- Une famille de scénarios est regroupée dans un cas d'utilisation selon un critère fonctionnel
- Un scénario est une instance d'un cas d'utilisation
- Décrit le flux des évènements
- Utilise un langage naturel
- Il peut y avoir un scénario principal et des scénarios alternatifs

Exemple de scénarios

Cas d'utilisation : Localiser un élément dans une bibliothèque

- Flux d'évènements principal : (élément localisé) Le bibliothécaire et l'emprunteur vont tous les deux chercher l'élément dans la bibliothèque à l'aide d'un ordinateur. L'élément peut être localisé par son type, son titre, son auteur, ses mots-clés dans sa description et aussi par une sélection de media spécifiques (ex. format de logiciel). Le résultat de la recherche est fourni dans un format résumé. La description détaillée de chaque instance de l'élément peut être montrée et imprimée
- Flux d'évènements alternatif : (élément non localisé) Le système ne localise pas l'élément. L'emprunteur est informé de ce résultat et peut éditer la dernière recherche ou annuler la demande

Description des scénarios

La description doit contenir :

- Une identification
  - Le nom du cas
  - Le résumé de son objectif
  - Les acteurs impliqués (principaux et secondaires)
  - Les dates de création et de mise à jour de la description courante
  - Le nom des responsables
  - Le numéro de version
- Un séquencement
  - Description de l'événement déclencheur
  - Description des pré-conditions
  - Description de l'enchainement nominal (auquel peuvent s'ajouter des séquences alternatives et des séquences d'exception)
  - Description des post-conditions

Exemple de description textuelle (1/2)

Retirer des billets de banque

- Identification
  - Nom du cas : retrait d'espèces en euros
  - But : détaille les étapes permettant à un guichetier d'effectuer l'opération de retrait d'euros demandé par un client
  - Acteur principal : Guichetier
  - Acteur secondaire : Système central
  - Date : le 27/04/09
  - Responsable : M. Dupont
  - Version : 1.0
- Séquencement
  - Le cas d'utilisation commence lorsqu'un client demande le retrait d'espèces en euros.
  - Pré-conditions : le client possède un compte (donne son numéro de compte)

Exemple de description textuelle (1/2)

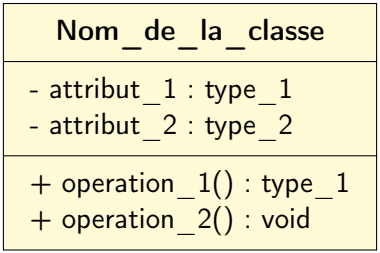
Retirer des billets de banque

- Séquencement (suite)
  - Enchaînement nominal
    - Le guichetier saisit le numéro de compte client.
    - L'application valide le compte auprès du système central.
    - L'application demande le type d'opération au guichetier.
    - Le guichetier sélectionne un retrait d'espèce de X euros.
    - L'application demande au système central de débiter le compte.
    - Le système notifie au guichetier qu'il peut délivrer le montant demandé.
  - Post-condition : le guichetier ferme le compte et le client récupère l'argent

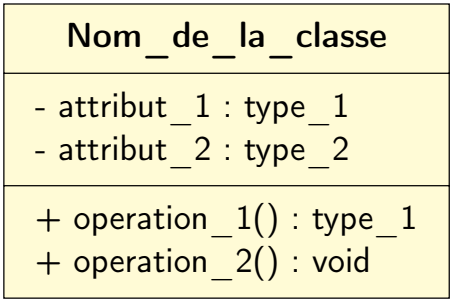
Notations

Diagrammes de classes

Représentation graphique



Attributs et opérations

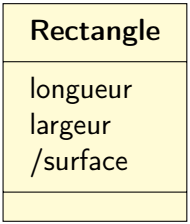


- Attributs : correspondent aux propriétés de la classe
  - Nom
  - Type
  - Valeur initiale (facultatif)
- Opérations : spécifient le comportement de l'objet
  - Les constructeurs
  - Les destructeurs
  - Les sélecteurs
  - Les modificateurs
  - Les itérateurs

Encapsulation et visibilité

- L'encapsulation est un mécanisme consistant à rassembler les données et les méthodes au sein d'une structure en cachant l'implémentation de l'objet
- Règles de visibilité : privé (-), protégé (#), public (+)

Attributs



- Le type des attributs peut être une classe, un type primitif ou une expression complexe
- UML permet de préciser la mutabilité des attributs (utilisation de la propriété de modification) :
  - gelé : attribut non modifiable
  - variable : propriété par défaut qui définit un attribut modifiable
  - ajoutUniquement : seul l'ajout est possible (pour des attributs dont la multiplicité en supérieure à 1)
- Attributs dérivés : sont indiqués en faisant précéder leur nom d'un /

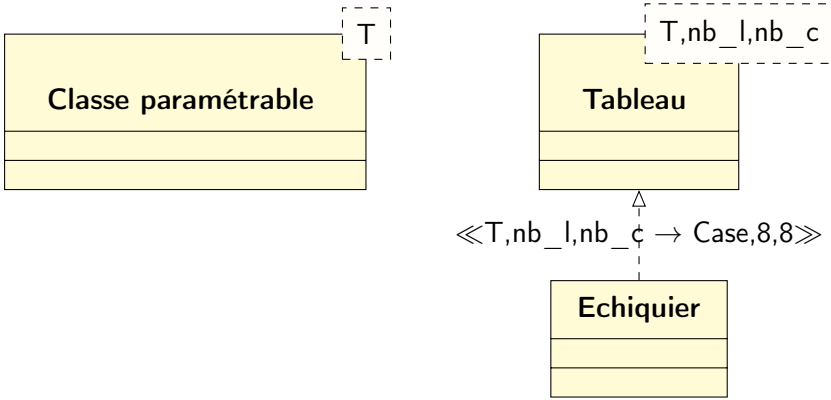
Mots-clés

- Utilisé pour renforcer la description d'une classe
- Mots-clés courants :
  - « acteur » : la classe modélise un ensemble de rôles joués par un acteur intervenant dans le système
  - « interface » : la classe contient uniquement une description des opérations visibles
  - « signal » ou « exception » : la classe modélise des éléments de type signal

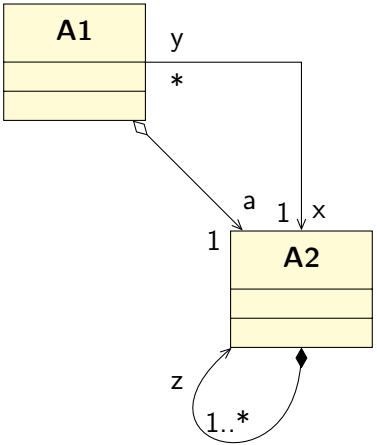
Stéréotypes

- Principal mécanisme d'extension d'UML
- Permet de construire de nouveaux types
- Exemples : « énumération », « métaclasse », « powertype », « processus », « thread », « type », « utilitaire »

Classes paramétrables

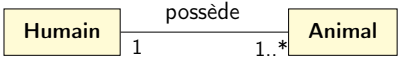


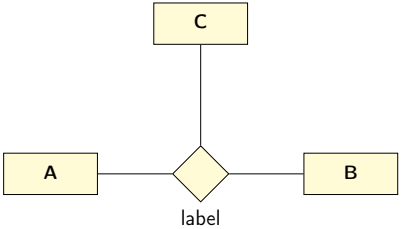
Associations



Associations binaires ou n-aires

Relations structurelles qui existent entre objets de différentes classes



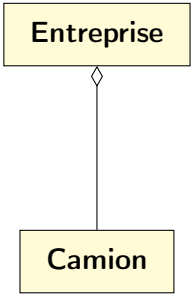


Le nom de l'association peut être suivi d'une flèche de direction (triangle plein) qui précise le sens de lecture

Multiplicité ou cardinalité

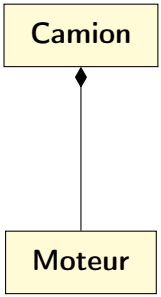
- 1 : un et un seul
- 0..1 : zéro ou un
- M..N : de M à N
- * ou 0..* : de zéro à plusieurs
- 1..* : de un à plusieurs

Aggrégation



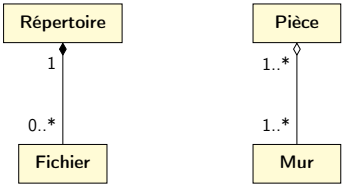
Une agrégation est une association qui représente une relation d'inclusion structurelle ou comportementale d'un élément dans un ensemble

Composition



- La composition, également appelée agrégation composite, décrit une contenance structurelle entre instances
- Ainsi, la destruction de l'objet composite implique la destruction de ses composants
- Une instance de la partie appartient toujours à au plus une instance de l'élément composite : la multiplicité du côté composite ne doit pas être supérieure à 1 (i.e. 1 ou 0..1)

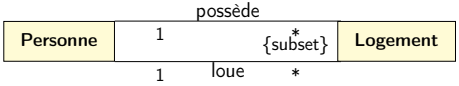
Aggrégation / composition



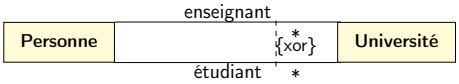
- La destruction de l'objet composite implique la destruction de ses composants
- Une instance de la partie appartient toujours à au plus une instance de l'élément composite : la multiplicité du côté composite ne doit pas être supérieure à 1 (i.e. 1 ou 0..1)

Contraintes

Contrainte ordonnée : une relation d'ordre décrit les objets placés dans la collection, l'ordre doit être maintenu

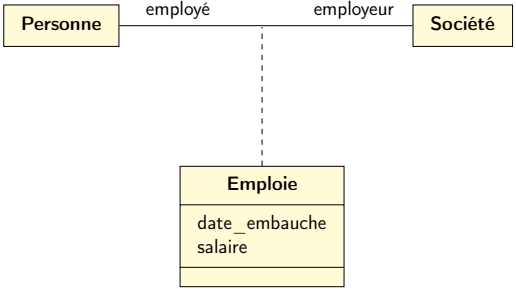


Contrainte sous-ensemble : une collection est incluse dans une autre collection



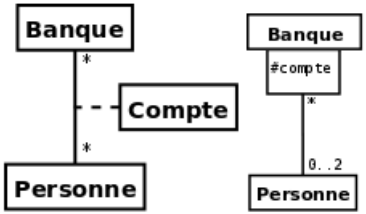
Contrainte ou-exclusif : pour un objet donné, une seule association est valide

Classe association



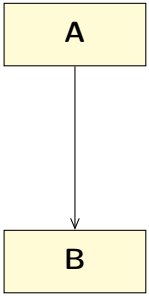
Parfois, une association doit posséder des propriétés

Qualification des associations



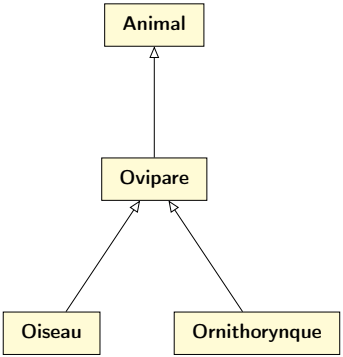
- La qualification (ou restriction) consiste à sélectionner un sous-ensemble d'objets parmi l'ensemble des objets qui participent à une association
- Est réalisé par un attribut spécifique appelé « clé » :
  - Représentée sur le rôle de la classe de départ
  - Appartient pleinement à l'association et non aux classes associées
- Une association qualifiée est plus adaptée à modéliser un identifiant qu'un attribut de classe

Navigabilité des associations



- Par défaut les association sont navigables dans les deux sens
- Dans certains cas, elles ne sont navigables que dans le sens de la flèche

Généralisation



- Relation de classification transitive entre une classe plus générale (super-classe) et une ou plusieurs autres classes plus spécifiques (sous-classes)
- Les attributs, opérations, relations et contraintes définies dans les super-classes sont hérités intégralement dans les sous-classes

Interfaces et classes abstraites

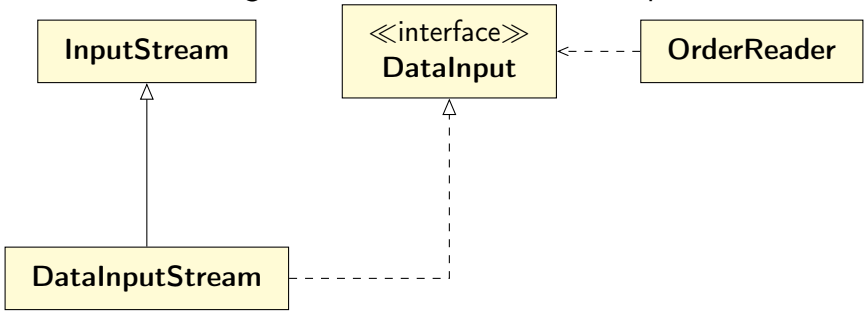
Définition

- Classe abstraite :
  - Son interface est la signature de toutes ses propriétés non privées
  - Notation : nom en italique ou la contrainte {abstract}
- Interface :
  - Utilise un type pour décrire le comportement visible d'une classe, sans implémentation
  - Peut ainsi fournir une vue partielle ou globale d'un ensemble de services
  - Est un stéréotype
- Une classe peut :
  - Spécialiser une classe abstraite
  - Réaliser une interface

Interfaces et classes abstraites

Représentation

Flèches de généralisation, réalisation et dépendance



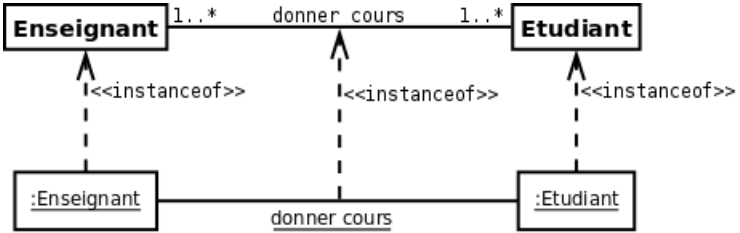
Notations

Diagrammes d'objets

Definition

- Les diagrammes de classes et d'objets sont deux vues complémentaires du modèle
- Diagramme de classes : montre une abstraction de la réalité, concentrée sur l'expression de la structure statique d'un point de vue général
- Diagramme d'objet : représente plutôt un cas particulier, une situation concrète à un instant donné, il exprime à la fois la structure statique et un comportement

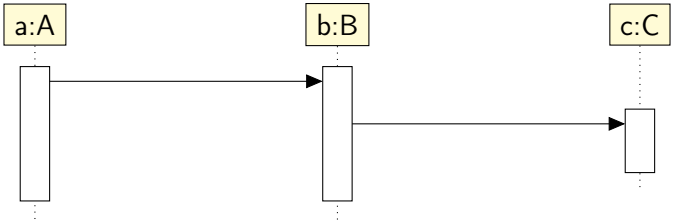
Représentation



Notations

Diagrammes de séquence

Représentation



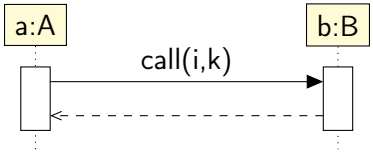
Définition

- Cas particulier d'un diagramme d'interaction
- Décrit un scénario d'un cas d'utilisation selon un aspect temporel :
  - Étapes
  - Enchaînements
  - Exécutions en parallèle
- Décrit comment les objets interagissent au sein du système au vu des fonctionnalités
- Deux dimensions :
  - Verticale : représente le temps
  - Horizontale : représente les différents objets

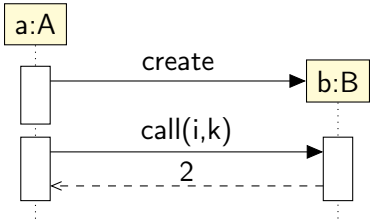
Message synchrone

Message asynchrone

Réponse à un message

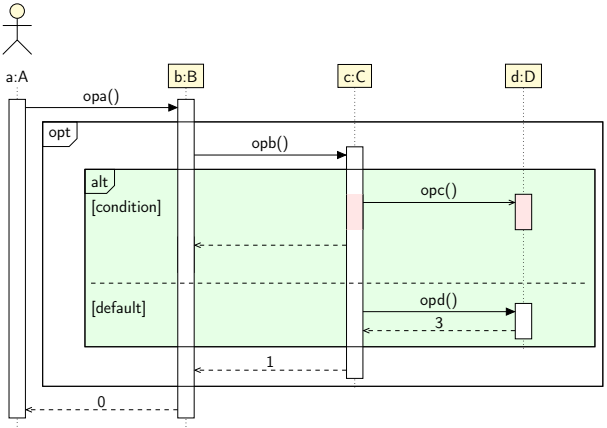


Création d'un objet



Fragments combinés

Représentation



Fragments combinés

Quelques exemples

- Fragment alternatif (alt) : modélise if...then...else
- Fragment optionnel (opt) : modélise switch
- Fragment boucle (loop) : modélise while
- Fragment break (break) modélise une séquence alternative d'évènements qui est exécuté à la place du reste du diagramme / du fragment qui l'encapsule
- Fragment parallèle (par) : modélise des processus concurrents
- Fragment de référence (ref) : encapsule une séquence

Notations

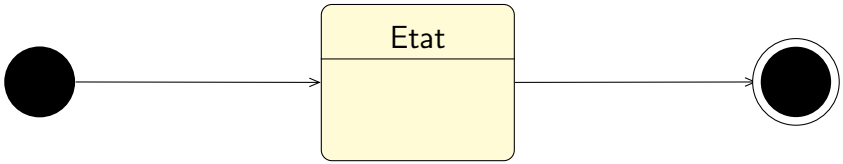
Diagrammes d'états-transitions

Définition

- Automates à états finis
- Décrivent :
  - Le comportement et tous les états possibles des instances d'une classe tout au long de leur vie
  - Les transitions d'un état à un autre
  - Les diagrammes rattachés à cette classe
- Utiles dans la modélisation de systèmes réactifs
- Nécessaires pour documenter les classes qui ont :
  - Des états clairement identifiables
  - Un comportement complexe
  - Beaucoup d'évènements

Représentation

Un état initial, un état simple et un état final



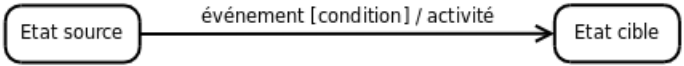
Une transition est une connexion unidirectionnelle
Elle est déclenchée par un événement

Événement

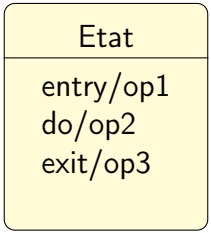
Un événement est une information instantanée qui doit être traitée sans délai
Ce peut être

- un événement de type signal (signal)
- un événement d'appel (call)
- un événement de changement (change)
- un événement temporel (after ou when)

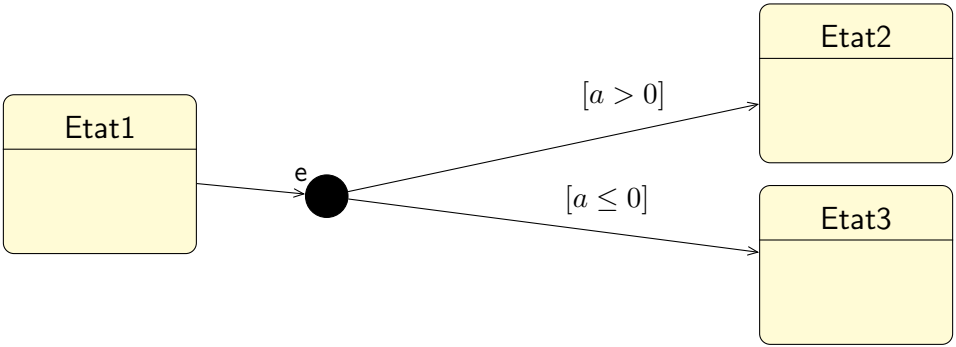
Transition



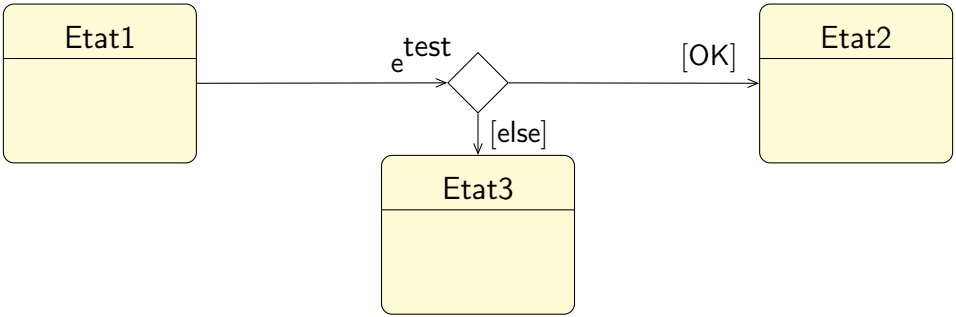
Actions internes



Point de jonction



Point de décision



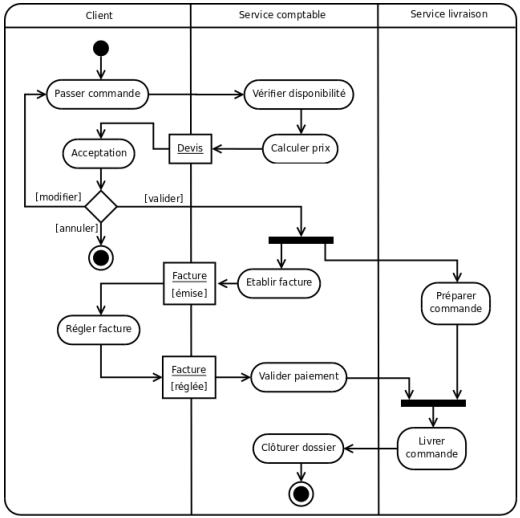
Notations

Diagrammes d'activité

Définition

- Variante des diagrammes d'états-transitions
- Décrit le déroulement du système
- Organisé par rapport aux actions
- Destiné à représenter le comportement interne d'une méthode ou d'un cas d'utilisation
- Formalisme graphique identique à celui du diagramme états-transitions :
  - une activité comme un état
  - chaque activité est une étape particulière d'exécution
  - transitions instantanées

Exemple


Client
Service comptable
Service livraison
Passer commande
Vérifier disponibilité
Acceptation
Devis
Calculer prix
[modifier]
[valider]
[annuler]
Facture
[émise]
Etablir facture
Préparer
commande
Régler facture
Facture
[réglée]
Valider paiement
Clôturer dossier
Livrer
commande

Mise en œuvre

UML dans un processus de développement

Identification des besoins - Analyse - Conception