Méthodologie de Systèmes d'Information: une perspective orientée-objet

La notation unifiée UML et Processus Unifié

Auteur: S. SI-SAID CHERFI

Maître de conférences – CNAM - PARIS

- **U**nified **M**odeling **L**anguage

Langage unifié pour la modélisation objet

J. Rumbaugh, G. Booch, I. Jacobson

Objectifs d'UML

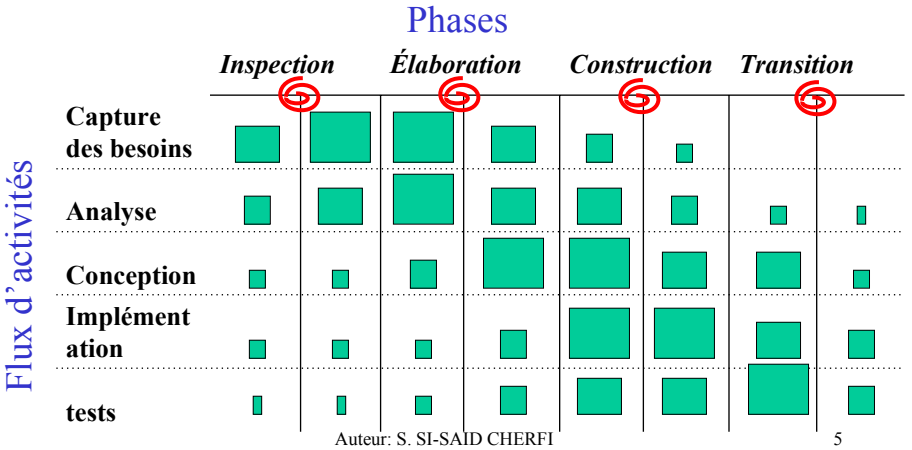
- Proposer un langage visuel de modélisation
  - Utilisable par toutes les méthodes
  - Adapté à toutes les phases du développement
  - Compatible avec toutes les techniques de réalisation
- Proposer des mécanismes d'extension et de spécialisation pour pouvoir étendre les concepts de base
- Être indépendant des langages particuliers de programmation
- Proposer une base formelle pour comprendre le langage de modélisation
- Encourager l'application des outils OO

Processus de développement: principes

- Spécifier ce qui doit être fait, quand le faire, comment le faire et par qui pour atteindre l'objectif fixé.
- Le processus unifié (**USDP Jacobson et al;99**) est censé regrouper les pratiques de conception de SI les plus reconnus impliquant un développement:
  - interactif et incrémental;
  - fondé sur les composants;
  - orienté par les besoins;
  - Configurable;
  - Centré sur l'architecture;
  - Utilisant des techniques visuelles de modélisation.

Processus de développement: activités

- Contrairement au model en cascade, on distingue les phases du projet des activités de développement.



Processus de développement: activités

Activité	Techniques	Résultats (livrables)
Capure et modélisation des besoins	Acquisition des besoins Modélisation des cas d'utilisation Prototypage	Modèle de cas d'utilisation Liste des besoins Prototypes

Processus de développement: activités

Activité	Techniques	Résultats (livrables)
Analyse des besoins	Modélisation des interactions (collaborations) Modélisation de la structure (classes et objets)	Modèles d'analyse

Processus de développement: activités

Activité	Techniques	Résultats (livrables)
Conception du système	Modèle de déploiement Modèle de composants Modèles de paquetages Modèle d'architecture	Ébauche du modèle de conception et de l'architecture de l'implémentation

Processus de développement: activités

Activité	Techniques	Résultats (livrables)
Conception détaillée des classes	Modélisation des objets et des classes Modélisation des interactions Modélisation des états	Modèles de conception

Processus de développement: activités

Activité	Techniques	Résultats (livrables)
Conception des interfaces utilisateur	Modélisation des objets et des classes Modélisation des interactions Modélisation des états Modélisation des paquetages Prototypage	Modèles de conception avec spécification des interfaces

Processus de développement: activités

Activité	Techniques	Résultats (livrables)
Conception de la gestion des données	Modélisation des objets et des classes Modélisation des interactions Modélisation des états Modélisation des paquetages	Modèles de conception avec spécification de la base de données

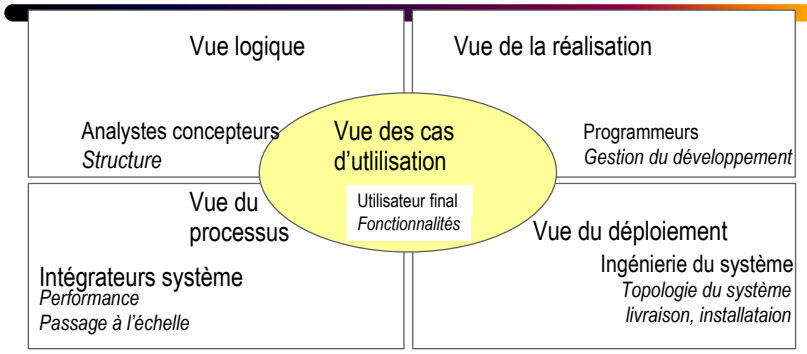
Processus de développement: activités

Activité	Techniques	Résultats (livrables)
Implémentation	Programmation Réutilisation des composant Spécification de la base de données	Modèle de réalisation Documentation

Processus de développement: activités

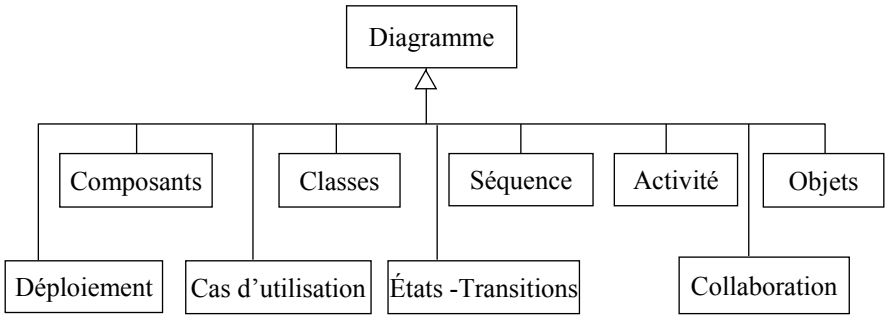
Activité	Techniques	Résultats (livrables)
Tests	Procédures et scénarios de test	Système testé

Les cinq vues de Ph. Kruchten



- Ces cinq vues sont indépendantes les unes des autres, ce qui permet aux différents intervenants de se concentrer sur les problèmes de l'architecture du système qui les concernent le plus.
- Elles interagissent également entre elles— les nœuds de la vue de déploiement comprennent des composants de la vue d'implémentation, qui, à leur tour, correspondent à la réalisation physique de classes, d'interfaces, de collaborations et de classes actives dans les vues de conception et de processus.
- UML permet de représenter chacune de ces cinq vues ainsi que leurs interactions.

Digrammes d'UML



Digrammes d'UML

- ***Diagramme de cas d'utilisation:*** fonctions du système du point de vue de l'utilisateur
- ***Diagramme de classes:*** structure statique en termes de classes et de relations
- ***Diagramme de séquence:*** représentation temporelle des objets et de leurs interactions
- ***Diagramme de collaboration:*** représentation spatiale des objets, des liens et des interactions
- ***Diagramme d'objets :*** Objets et leurs relations. Diagrammes de collaboration simplifiés, sans représentation des envois de message
- ***Diagramme d'états-transitions:*** comportement d'une classe en terme d'états (Statecharts) [Harel, D. 1987. Statecharts: A Visual Formalism for Complex Systems. Science of Computer Programming vol. 8.]
- ***Diagramme d'activités:*** comportement d'une opération en termes d'actions
- **Diagramme de déploiement**: déploiement des composants sur les dispositifs matériels
- **Diagrammes de composants**: composants physiques d'une application.

La capture des besoins: l'acquisition

- Un SI doit répondre aux besoins des utilisateurs. Leurs capture nécessite:
  - L'étude du système existant
  - L'acquisition des nouveaux besoins
    - Fonctionnels : fonctionnalités du futur système
    - Non fonctionnels: performances, sécurité etc.
    - D'utilisation: critères d'acceptation par les utilisateurs, facteurs situationnels etc.
  - L'utilisation de techniques d'acquisition:
    - Analyser la documentation existante, interviews, observation des utilisateurs, questionnaires etc.

La capture des besoins: la spécification

- UML propose les diagrammes de cas d'utilisation pour modéliser et documenter les besoins
  - Un cas d'utilisation n'est pas un besoins mais une fonctionnalité du système devant répondre à un besoin
  - Les diagrammes de cas d'utilisation ne permettent pas la spécification des besoins non-fonctionnels
  - Il est nécessaire d'établir une liste des besoins non-fonctionnels pour compléter la spécification des besoins.

Diagramme de cas d'utilisation

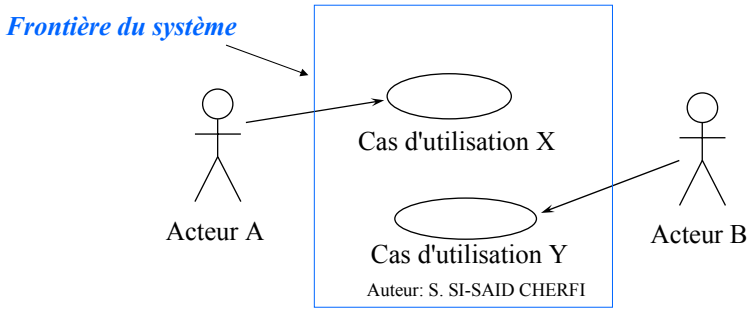
- Formalisés par **Ivar Jacobson.**
- Décrivent sous la forme d'actions et de réactions, le comportement d'un système du point de vue de l'utilisateur.
- Définissent les limites du système et les relations entre le système et l'environnement.
- Manière spécifique d'utiliser un système. C'est l'image d'une fonctionnalité du système, déclenchée en réponse à la stimulation d'un acteur externe.

Exemple de besoins

- Gestion des malades dans un cabinet médical
  - Un médecin doit pouvoir
    - Créer des dossiers médicaux de patients, les modifier, les consulter et les supprimer
    - Créer des ordonnances, les modifier, les imprimer et les supprimer
    - Créer et modifier des instructions à l'attention du secrétariat
    - Créer et modifier les informations sur le patient (informations non médicales)
  - Un secrétaire doit pouvoir
    - Créer et modifier les informations sur le patient (informations non médicales)
    - Éditer une feuille de soin
    - Imprimer un récépissé lors de l'encaissement d'un paiement
    - Etc.

Diagrammes de cas d'utilisation : les concepts

- Il comprend les acteurs, le système et les cas d'utilisation eux-mêmes.
- Les acteurs se représentent sous la forme de petits personnages qui déclenchent des cas d'utilisation; ces derniers sont représentés par des ellipses contenues par le système.

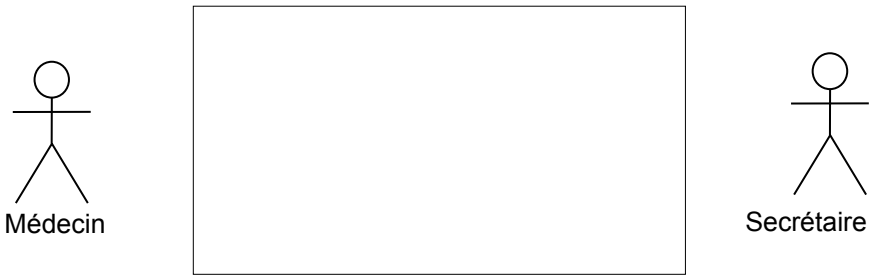


Diagrammes de cas d'utilisation : l'acteur

- Un acteur représente tout ce qui est *externe au système*, *humain on non*, qui *interagit* avec le système et qui correspond à une catégorie d'utilisateurs (plus précisément à *un rôle*).
- Les acteurs se déterminent en observant les utilisateurs directs du système, ainsi que les autres systèmes qui interagissent avec le système en question.
- La même personne physique peut jouer le rôle de plusieurs acteurs (vendeur, client). Plusieurs personnes peuvent jouer le même rôle, et donc agir comme le même acteur (tous les clients). Le nom de l'acteur décrit le rôle joué par l'acteur.

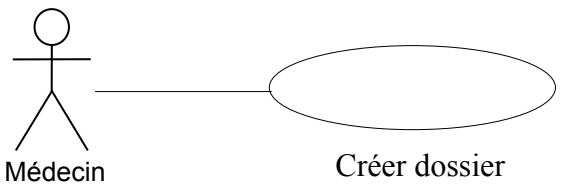
Diagrammes de cas d'utilisation : l'acteur

- Dans le cas du cabinet médical, nous avons deux acteurs:
  - Le médecin
  - Le secrétaire



Diagrammes de cas d'utilisation : les cas d'utilisation

- Un cas d'utilisation
  - Est représenté graphiquement par une ellipse avec le nom du cas en dessous
  - Décrit une séquence d'action effectuée par le système pour livrer un résulat à l'acteur



Diagrammes de cas d'utilisation : les cas d'utilisation

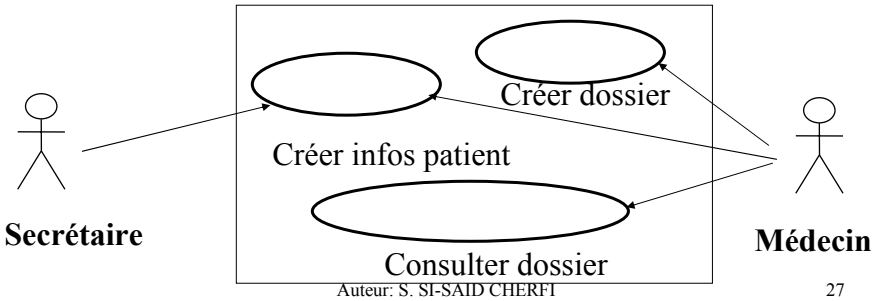
- Les cas d'utilisation se déterminent en observant et en précisant, acteur par acteur, les séquences d'interaction - les scénarios - du point de vue de l'utilisateur.
- La participation de l'acteur est signalée par un lien de communication entre l'acteur et le cas d'utilisation. Ce lien peut être orienté pour indiquer l'initiateur de l'interaction.

Diagrammes de cas d'utilisation : Les cas d'utilisation

- Les cas d'utilisation doivent être vus comme des classes dont les instances sont les scénarios. Chaque fois qu'un acteur interagit avec le système, la cas d'utilisation instancie un scénario; ce scénario correspond au flot de messages échangés par les objets durant l'interaction particulière qui correspond au scénario.
- Les cas d'utilisation servent de fil conducteur pour l'ensemble du projet.

Le modèle de cas d'utilisation

- Le modèle de cas d'utilisation
  - comprend une collection de cas d'utilisation
  - Caractérise le comportement de l'ensemble du système et des acteurs externes (dans leur interaction).



Raffinement des cas d'utilisation

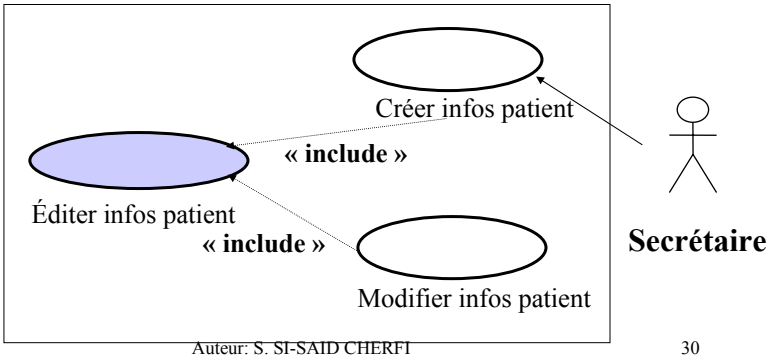
- Relations entre cas d'utilisation
  - Deux relations Extend et Include entre les cas d'utilisation
  - Apparaissent comme des relations stéréotypées
  - Les stéréotypes sont écrits sous forme de texte entre guillemets: «extend» et «include»

Raffinement des cas d'utilisation : la relation « include"

- Une relation d'inclusion entre cas d'utilisation signifie **que toute instance du cas d'utilisation source comprend nécessairement** le comportement décrit dans le cas d'utilisation destination.
- Quand l'utiliser:
  - lorsqu'un ensemble d'actions peut être utilisé dans plusieurs cas d'utilisation et que l'on ne souhaite pas répéter cet ensemble,
  - Un tel ensemble est alors décrit dans un cas d'utilisation séparé et est lié au cas d'utilisation qui l'utilise par un lien « include »
  - L'avantage d'une telle démarche est la réutilisation

Raffinement des cas d'utilisation : la relation « include »

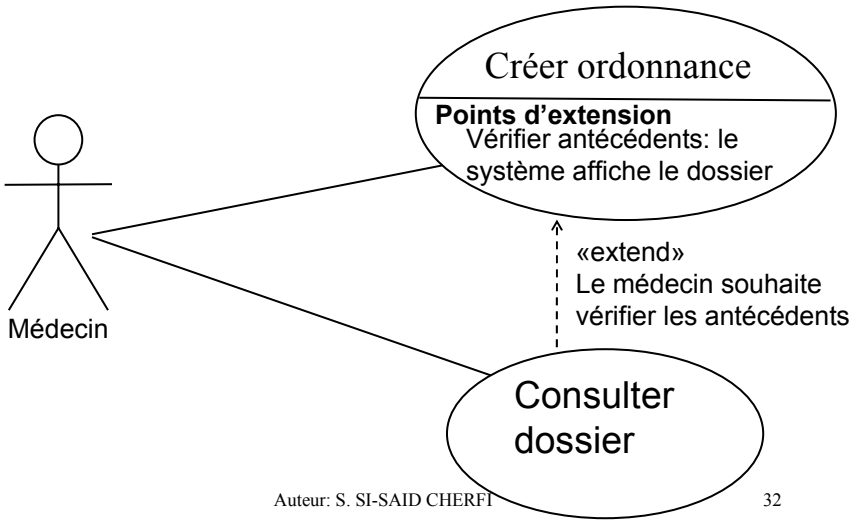
Exemple du cabinet médical



Raffinement des cas d'utilisation : la relation « extend »

- Une relation d'extension entre cas d'utilisation signifie que le cas d'utilisation source étend le comportement du cas d'utilisation destination.
- Quand l'utiliser:
  - lorsqu'un cas d'utilisation est similaire à un autre cas d'utilisation à l'exception d'une petite variation,
  - une telle variation est décrite dans un cas d'utilisation à part,
  - Les deux cas d'utilisation sont ensuite liés par une relation d'extension.
- Les points d'extension montrent à quel moment survient l'extension
- une condition peut être rajoutée à la relation d'extension pour la préciser.

Raffinement des cas d'utilisation : la relation « extend »



Diagrammes de cas d'utilisation : Les cas d'utilisation

- **Un cas d'utilisation :**
  - est un ensemble complet d'actions (avec des événements de début et de fin, les acteurs impliqués dans les actions, et les objets utilisées dans les actions) et de règles qui régissent l'enchaînement des actions.
  - Il définit les interactions entre le système et l'acteur
  - inclut le déroulement normal et tous les déroulements alternatifs.
- **Un scénario :**
  - est un déroulement spécifique des événements, ce déroulement dépend des événements à l'origine et à l'issue de chaque action spécifié dans le cas d'utilisation et dont dépend l'enchaînement des actions.
  - Il est possible de dériver plusieurs scénario à partir d'un cas d'utilisation. Il suffit pour cela que l'enchaînement des actions ne soit une simple séquence.

Description des cas d'utilisation

- La description peut être sous une forme textuelle simple et peu structurée.
- Exemple
  - « le médecin cherche le patient dans la liste des patients. Si celui-ci n'existe pas il le crée, sinon il crée son dossier. Le médecin introduit les informations sur les antécédents du patient, ses allergies, la liste des substances auxquelles il est allergique, la liste des traitements qu'il suit et la durée de ces traitements etc. »

Description des cas d'utilisation

- Elle peut également être décomposée en précisant les interactions entre le système et l'acteur en distinguant le déroulement de base des déroulements alternatifs

Créer ordonnance

Acteur	**Réponse du systeme**
1- Le médecin demande à créer une ordonnance	**2**- Le système lui demande de choisir un patient
3- Le médecin choisi le patient souhaité	**4**- Le système édite une ordonnance vierge
5- le médecin sélectionne un médicament dans une liste	**6**- le système demande à saisir les doses et la fréquence des prises
…	…

Déroulement alternatif 5-6

- Le médecin entre le nom du médicament ainsi que les doses et les fréquences des prises

Description des cas d'utilisation

- La description d'un cas d'utilisation comprend les éléments suivants:
  - *le début* :"Le cas d'utilisation débute quand X se produit.";
  - *la fin*:"Quand X se produit, le cas d'utilisation est terminé.";
  - *l'interaction entre le cas d'utilisation et les acteurs*;
  - *les échanges d'informations*: par exemple, "L'utilisateur se connecte au système et donne son nom et son mot de passe.";
  - *La chronologie et l'origine des informations*;
  - *Les répétitions de comportement* qui peuvent être décrites au moyen de pseudo-code, avec des constructions du type:

boucle	ou	Tant que
-- quelque chose		-- autre chose
fin de boucle		fin tant que

Règles de mise en œuvre des cas d'utilisation

- Un cas d'utilisation décrit une fonctionnalité ou une motivation et aussi **une interaction** entre un acteur et un système sous la forme d'un flot d'évènements.
- La description de l'interaction se concentre sur ce qui doit être fait.
- Un cas d'utilisation doit être simple.
- Il ne faut pas mélanger les cas d'utilisation.
- Un cas d'utilisation doit éviter d'employer des expressions floues et imprécises.

Règles de mise en œuvre des cas d'utilisation

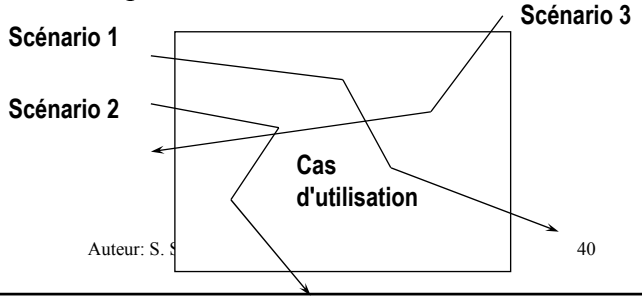
- *les situations optionnelles*: "L'acteur choisit l'un des éléments suivants, éventuellement plusieurs fois de suite:
  a) choix X, b) choix Y, c) choix Z
  puis l'acteur continue en...";
- Il est également primordial de trouver le bon niveau d'abstraction.
- Les réponses apportées aux deux interrogations suivantes peuvent servir de gabarit:
  - est-il possible d'exécuter une activité donnée indépendamment des autres, ou faut-il toujours l'enchaîner avec une autre activité?
  - Est-il judicieux de regrouper certaines activités en vue de les documenter, de les tester ou de les modifier?

Construction des cas d'utilisation

- En règle générale, il n'y a qu'un seul acteur par cas d'utilisation.
- Lors de la construction, il faut se demander:
  - Quelles sont les tâches de l'acteur ?
  - Quelles informations l'acteur doit-il créer, sauvegarder, modifier, détruire ou simplement lire ?
  - L'acteur devra-t-il informer le système des changements externes ?
  - Le système devra- t-il informer l'acteur des conditions internes ?
- Cas d'utilisation peuvent :
  - être présentés aux travers de vues multiples.
  - être groupés selon leurs séquences de déclenchement types ou en fonction des différents points de vue.

Processus d'élaboration des cas d'utilisation

- Définir un guide de style pour la rédaction.
- Définir grossièrement les cas d'utilisation.
- Approfondir la compréhension et la description d'un cas d'utilisation particulier. Identifier les scénarios.
- Un scénario est un chemin particulier au travers de la description abstraite et générale fournie par le cas d'utilisation.



Résumé

Dans ce cours nous avons vu:

- L'objectif des diagrammes de cas d'utilisation
- La notation des des diagrammes de cas d'utilisation
- Comment les dessiner
- Comment les décrire
- Comment les construire.

De l'analyse des besoins à la conception

Pourquoi analyser les besoins ?

- Le modèle de cas d'utilisation ne suffit pas à lui seul
  - Il peut comporter des répétitions
  - Certaines parties peuvent être déjà disponibles sous forme de composants.
- L'analyse sert à identifier:
  - Les éléments communs
  - Les éléments existant déjà
  - L'interaction entre les différents éléments

Finalité d'un modèle d'analyse

- Confirmer les besoins des utilisateurs (préciser)
  - Compréhensible et complet
- Spécifier ce que les concepteurs doivent concevoir
  - Non ambigu
- Décrire ce que le système doit faire
- Montrer les interactions entre les concepts
- Servir de base à la future implémentation

Comment construire un modèle d'analyse

- L'outil essentiel est le diagramme de classes
- Il peut être obtenu
  - Directement en utilisant des connaissances du domaine (modèle de domaine)
  - En construisant un diagramme de classes pour chaque cas d'utilisation. Ces cas sont ensuite assemblés en un seul modèle. (modèle de classes d'analyse)

Stéréotypes des classes d'analyse

- Les stéréotypes servent à la conception
- Les stéréotypes précisent le rôle joué par chaque objet
  - **« frontière»** modéliser les frontières entre le système et ses acteurs (et les autres systèmes) (« boundary »)
  - **« entité »** informations durables et persistantes. Elles représentent des informations et des comportements du domaine d'application (« entity »)
  - **« contrôle »** contrôler et coordonner les autres objets (« control »)

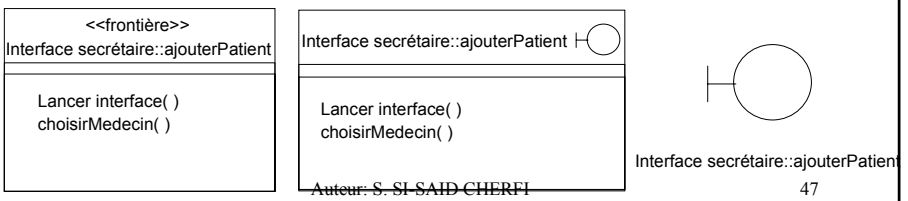
Stéréotypes des classes d'analyse

•Une classe interface utilise un type pour décrire le comportement visible d'une classe, d'un composant, ou d'un paquetage.

•Peut être un dispositif matériel (écran, imprimante) ou un agent humain jouant le rôle d'acteur interne.

•La classe interface doit être capable d'interagir avec les acteurs externes (interroger, recueillir des informations ou requêtes, fournir des informations).

- Notations pour les classes « frontière »



Stéréotypes des classes d'analyse

•Permet de :

– stocker l'information.

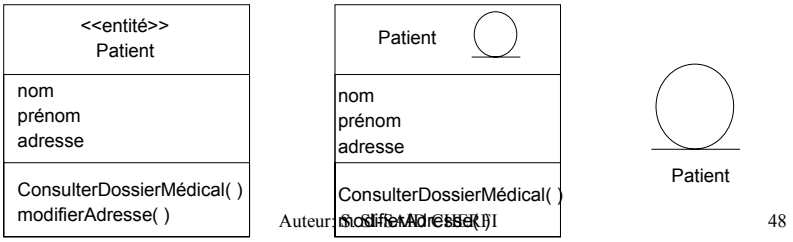
–définir le comportement.

–contient des opérations pour :

•entrer et stocker l'information

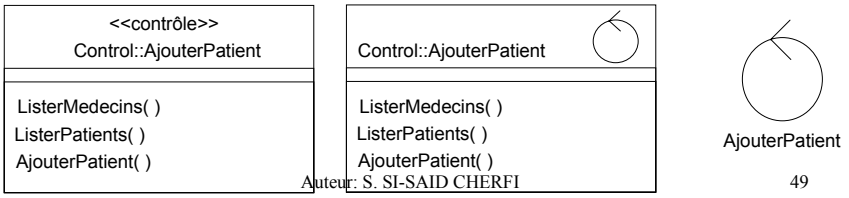
•créer et supprimer les objets de la classe.

•Notations pour les classes « entité »



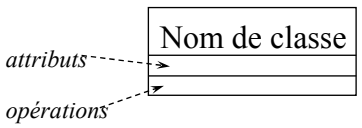
Stéréotypes des classes d'analyse

- Jouent le rôle de connecteur entre les classes « frontère » et les classes « entité »
- Représentent le comportement lié aux transactions, séquences etc.. Ces comportements sont généralement transverses à différentes fonctions du système.
- Sont sujettes à des modifications fréquentes d'où l'intérêt de les dissocier des autres classes.
- Notations pour les classes « contrôle »



Classe

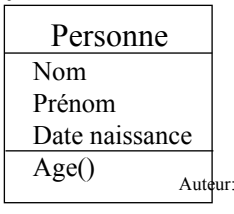
- Une classe est une description abstraite d'un ensemble d'objets ayant des propriétés similaires, un comportement commun, des relations communes avec d'autres objets et des sémantiques communes.



Nom de classe

Exemples:

Selon le niveau de détails souhaité

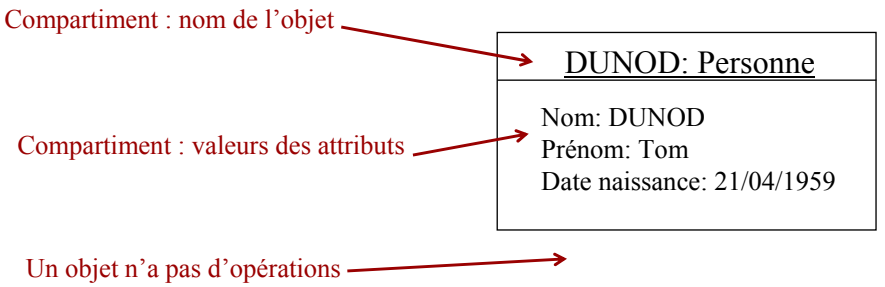


Personne

Poste de travail

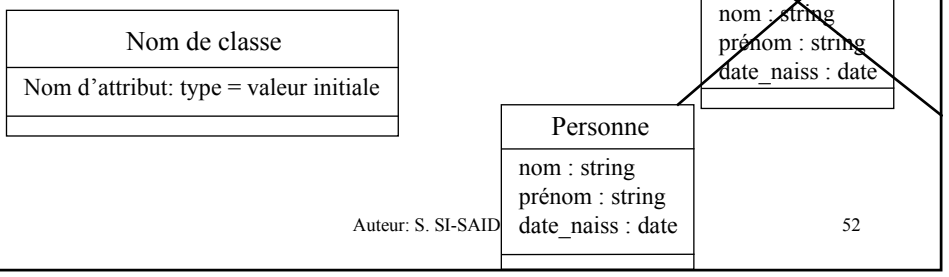
Département

Instance d'une classe: l'objet

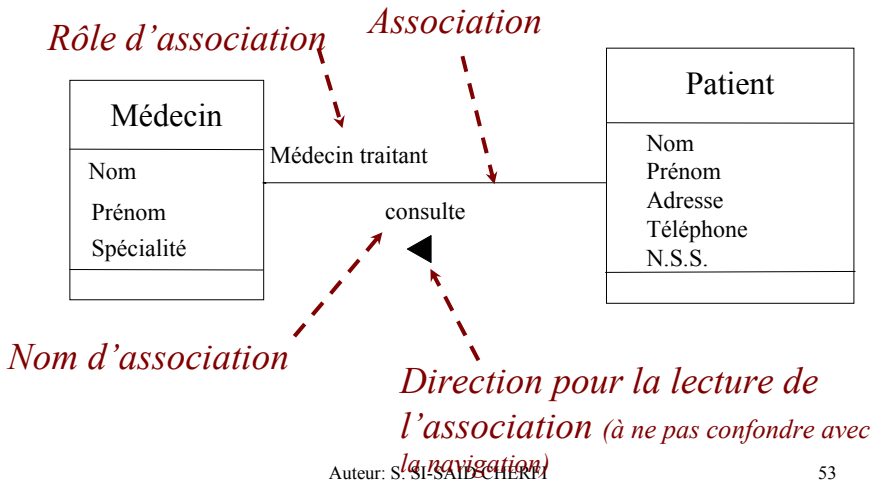


La classe: les attributs

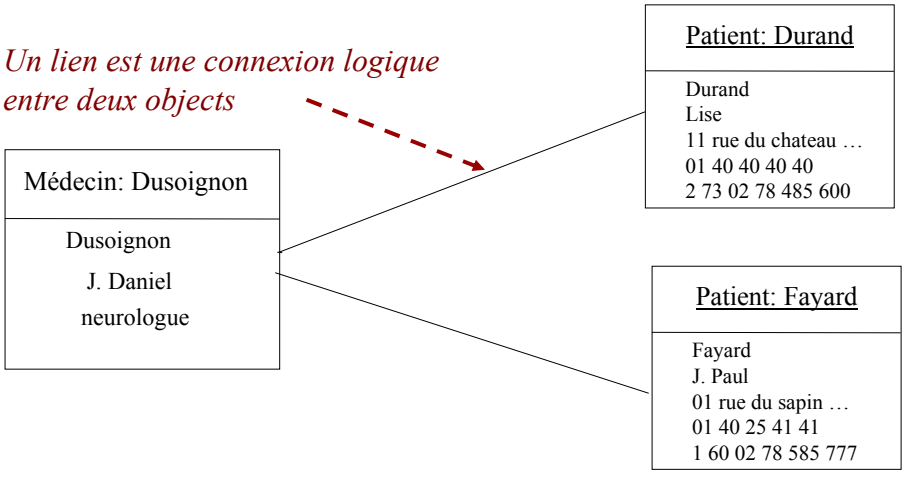
- Définissent les propriétés des objets d'une classe
- Chaque nom d'attribut est unique dans une classe.
- L' identification est fournie par le système, elle n'est pas à considérer dans le modèle objet
- Chaque objet a ses propres valeurs pour chacun des attributs de sa classe.



Les associations dans un diagramme de classes

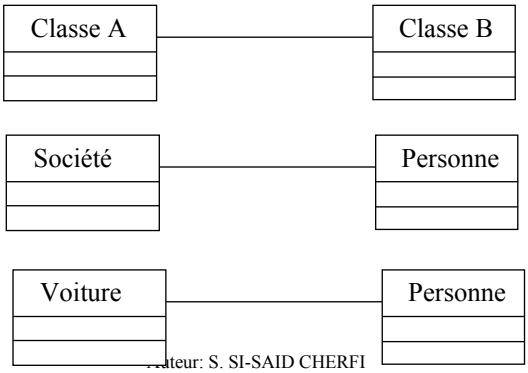


Les liens entre les objets des classes

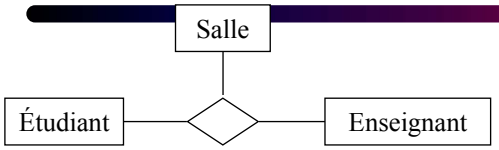


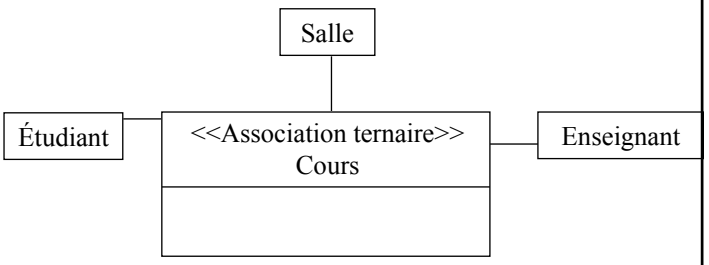
Association

- Représente une relation structurelle entre classes d'objets.
- Une association symbolise une information dont la durée de vie n'est pas négligeable par rapport à la dynamique générale des objets instances des classes associées.



Association n-aire

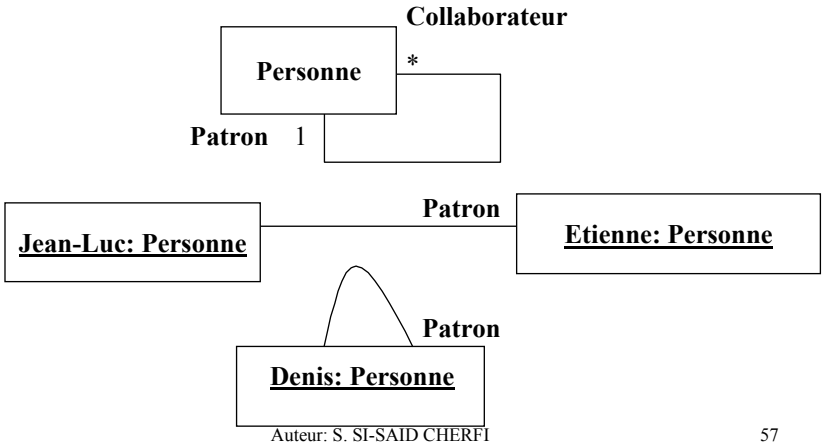




Représentation d'une association ternaire au moyen d'une classe avec stéréotype.

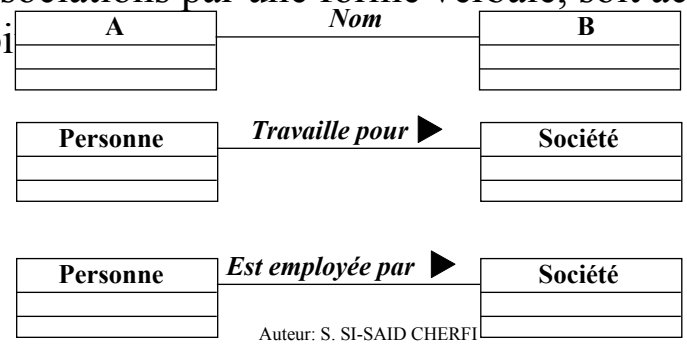
Liens entre les objets

• Les liens instances des associations réflexives peuvent relier un objet à lui-même.



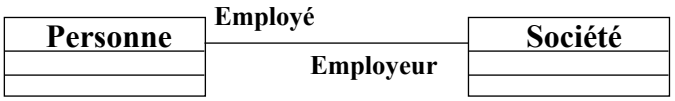
Nommage des associations

- Le nom d'une association apparaît en italique, au milieu de la ligne qui symbolise l'association.
- L'usage recommande de nommer les associations par une forme verbale, soit active, soi



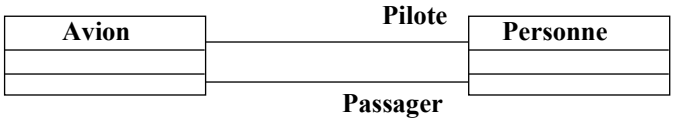
Nommage de rôles

- Chaque association binaire possède 2 rôles.
- Le rôle décrit comment une classe voit une autre classe au travers d'une association.
- Le nommage des associations et le nommage des rôles ne sont pas exclusifs l'un de l'autre.



Nommage de rôles

- Le nommage des rôles prend tout son intérêt lorsque plusieurs associations relient 2 classes. Chaque association exprime un concept distinct.



Multiplicité des associations

- La multiplicité est une information portée par le rôle, sous la forme d'une expression entière bornée.

1	**Un et un seul**
0 .. 1	**zéro ou un**
M .. N	**De M à N (entiers naturels)**
*****	**De zéro à plusieurs**
0 .. *	**De zéro à plusieurs**
1 .. *	**De un à plusieurs**

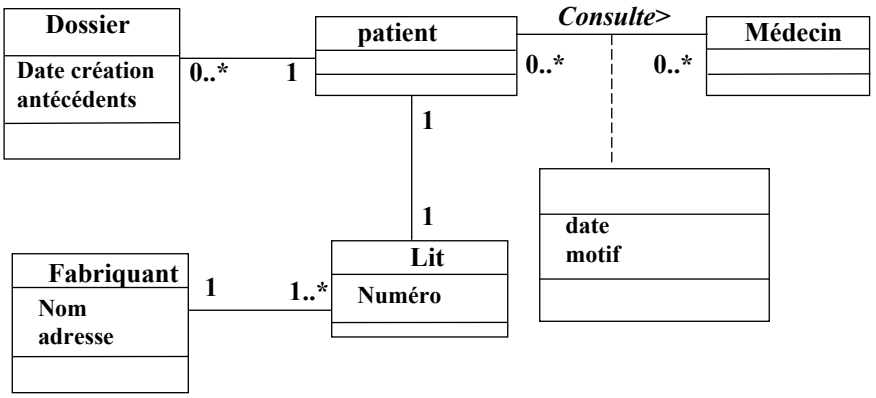
Multiplicité des associations

- Une valeur de multiplicité supérieure à **1** implique une collection d'objets.

- Les valeurs de multiplicité expriment des contraintes liées au domaine de l'application, valables durant l'existence des objets. Ces valeurs ne doivent pas être considérées durant les régimes transitoires.

Exemple

Exemple



La classe: les opérations

- Définissent les fonctions appliquées à des objets d'une classe.

Personne
nom : string prénom : string date_naiss : date
age () change_adresse ()

Nom de classe
Nom d'opération(liste_arguments) : type_retour

La syntaxe pour la description des opérations :

```
Nom_opération (Nom_Argument : Type_Argument =
Valeur_Par_Défaut, …): Type_Retourné
```

La classe: les opérations

- Une opération peut:
  - fixer, modifier ou révéler la valeur d'un attribut
  - effectuer des calculs
  - envoyer des messages à d'autres objets
  - créer ou détruire des liens avec d'autres objets

Personne
nom : string prénom : string date_naiss : date Adresse: [num, rue, ville] Supérieur: Personne
age () changer_adresse () Modifier_supérieur()

La classe: attributs dérivés

• Les attributs dérivés offrent une solution pour allouer des propriétés à des classes, tout en indiquant clairement que ces propriétés sont dérivées d'autres propriétés déjà allouées.

Exemple:

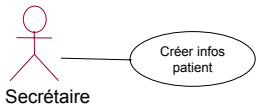
Rectangle
Longueur Largeur /Surface

En conception ⇢

Rectangle
Longueur Largeur
Surface( )

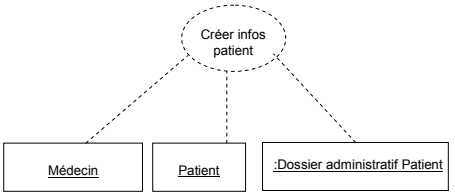
Des besoins aux classes

1

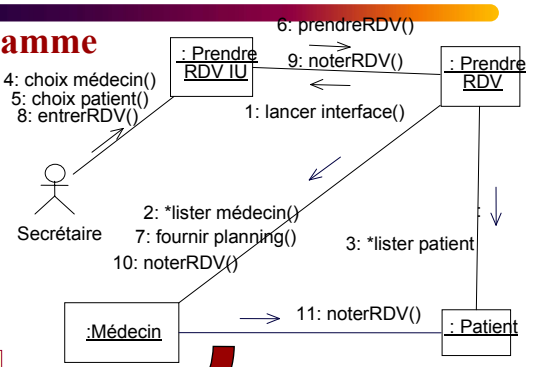


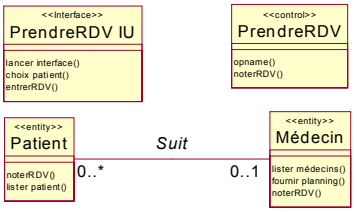
1: Point de départ un cas d'utilisation

2: Réalisation du cas d'utilisation



3: Construire le diagramme de collaboration





4: Identifier les classes candidates

Des besoins aux classes

- Commencer par un cas d'utilisation
- Identifier les classes vraisemblables (associer une collaboration à chaque cas)
- Construire le diagramme de collaboration qui permet d'atteindre les besoins décrits dans le cas d'utilisation
- Traduire ce diagramme de collaboration en un diagramme de classe
- Répéter le processus pour les autres cas d'utilisation

Des besoins aux classes

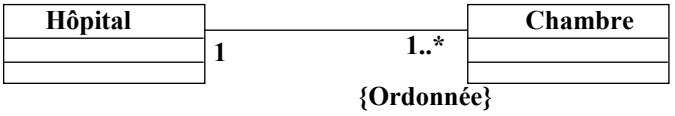
- Vérifier la pertinence des classes identifiées
  - Est ce qu'elle est en dehors de la problématique du système?
  - Est ce qu'elle répète ce qui est déjà représenté dans d'autres classes?
  - Est ce qu'elle est vague?
  - Est ce qu'elle est trop liée à des choix techniques ou d'implantation?
  - Est ce qu'elle ne serait pas un attribut, une opération ou une association?
- Si la réponse à l'une des questions est « oui » alors reconsidérer la classe.

Contraintes sur les associations

- Toutes sortes de contraintes peuvent être définies sur une relation ou sur un groupe de relations. La multiplicité, par exemple, en est une.
- Les contraintes se représentent dans les diagrammes par des expressions placées entre accolades.

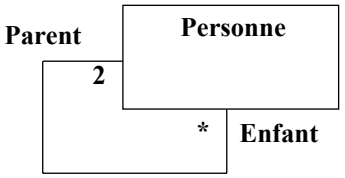
Par exemple

- La **contrainte {ordonnée}** peut être placée sur le rôle pour spécifier qu'une relation d'ordre décrit les objets placés dans la collection.



D'autres contraintes sont définies

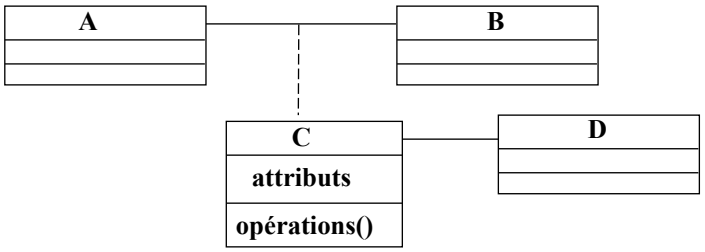
Association réflexive



Le nommage des rôles est essentiel à la clarté du diagramme.

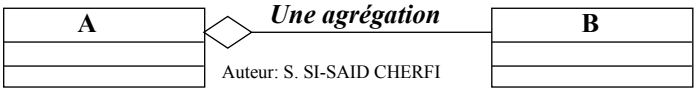
Classe association

- Une association peut être représentée par une classe pour ajouter, par exemple, des attributs et des opérations dans l'association. Nommée, classe associative ou classe-association, c'est une classe comme les autres. La notation utilise une ligne pointillée pour attacher une classe à une association.



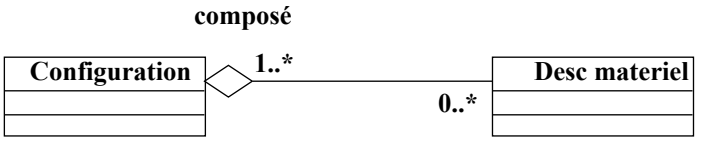
Agrégation

- Représente une association non symétrique dans laquelle une des extrémités joue un rôle prédominant par rapport à l'autre extrémité.
- Quelle que soit l'arité, l'agrégation ne concerne qu'un seul rôle d'une association.
- Les critères suivants impliquent une agrégation:
  - une classe fait partie d'une autre classe;
  - les valeurs d'attributs d'une classe se propagent dans les valeurs d'attributs d'une autre classe;
  - une action sur une classe implique une action sur une autre classe;
  - les objets d'une classe sont subordonnés aux objets d'une autre classe;



Agrégation

- L'agrégation peut être multiple, comme l'association

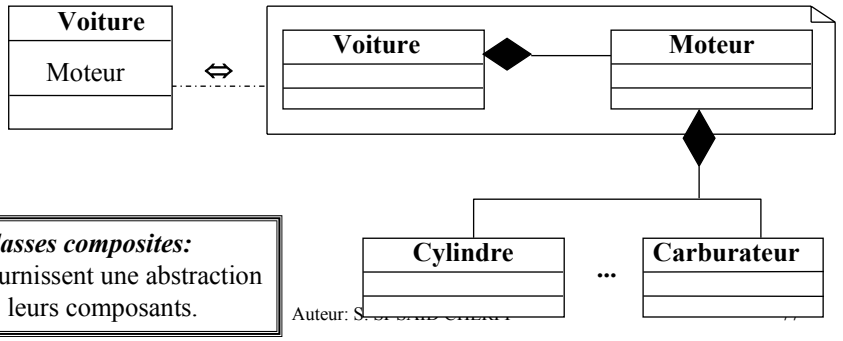


Composition

- Agrégation réalisée par valeur sur des attributs: composition.
- Ils sont physiquement contenus dans l'agrégat.
- La composition implique une contrainte sur la valeur de la multiplicité du côté de l'agrégat: elle ne peut prendre que les valeurs 0 ou 1.
- La valeur 0 du côté du composant correspond à un attribut non renseigné.

Composition

- La composition et les attributs sont sémantiquement équivalents.
- La notation par composition s'emploie dans un diagramme de classes lorsqu'un attribut participe à d'autres relations dans le modèle.



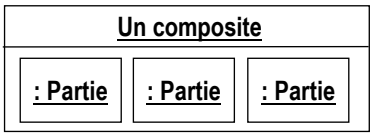
Classes composites: fournissent une abstraction de leurs composants.

agrégation vs. composition

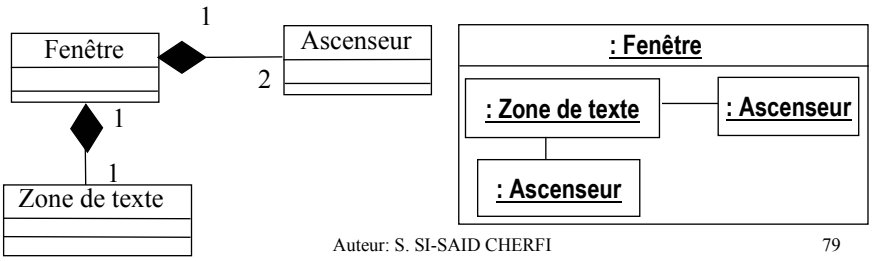
- L'agrégation représente une relation de partie-de
- La sémantique peut être imprécise
- Composition est une relation plus « forte »:
  - Chaque partie peut appartenir à un et un seul tout à un instant donné.
  - Lorsque le tout est détruit (n'a plus d'existence) il en est de même pour toutes les parties qui le composent.

Objet composite

• Les objets composés de sous-objets peuvent être représentés au moyen d'un objet composite.

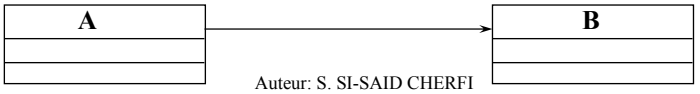


• Les objets composites sont instances de classes composites:



Navigation

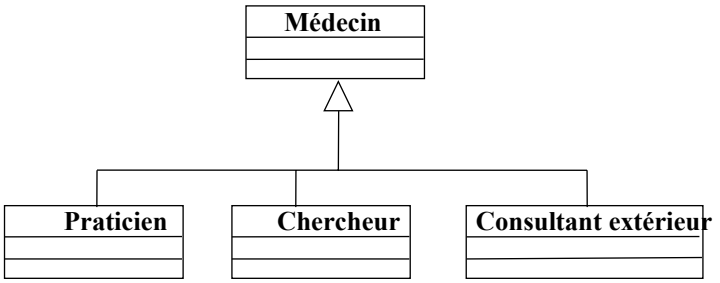
- Les associations décrivent le réseau de relations structurelles entre les classes. Les liens, instances de ces associations, peuvent être vus comme des canaux de navigation entre les objets, instances des classes.
- Ces canaux permettent de se déplacer dans le modèle et de réaliser des scénarios.
- Par défaut, les associations sont navigables dans les 2 directions.
- Si seule une direction est utile, ceci se représente par une flèche portée par le rôle vers lequel la navigation est possible.



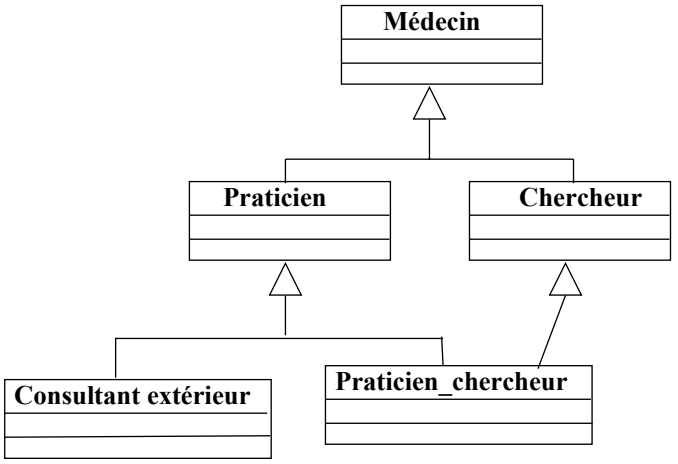
Généralisation

- La généralisation s'applique principalement aux classes, aux paquetages et aux cas d'utilisation.
- Dans le cas des classes, la relation de généralisation signifie ***est un*** ou ***est une sorte de.***

Généralisation

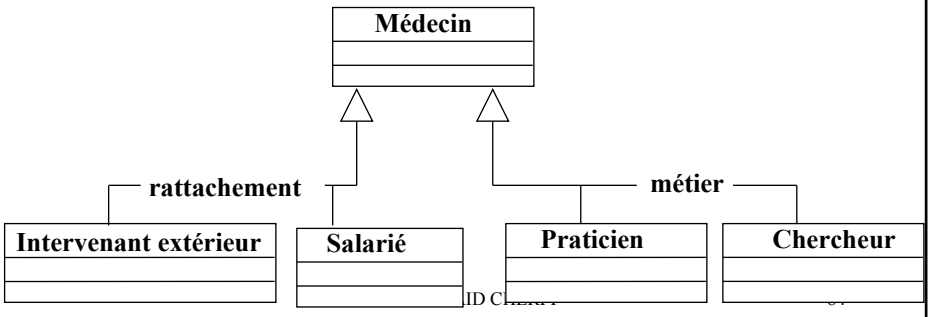


Généralisation multiple



Généralisation

- Une classe peut être spécialisée selon plusieurs critères simultanément.
- Différentes contraintes peuvent être appliquées aux relations de généralisation.



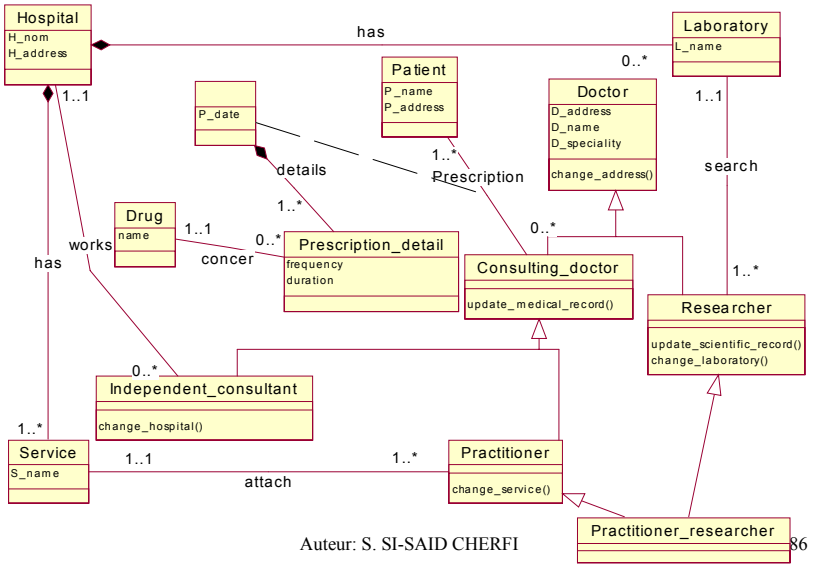
Classe abstraite

- Une classe abstraite n'est pas instanciable directement.
- Spécification plus abstraite pour des objets instances de ses sous-classes.
- Elles forment une base pour les logiciels extensibles.
- Une classe est désignée comme abstraite au moyen de la propriété booléenne **Abstraite** définie pour tous les éléments généralisables: les types, les paquetages et les stéréotypes.
- La propriété abstraite peut également être appliquée à une opération afin d'indiquer que le corps de l'opération doit être défini explicitement dans les sous-classes.

Classe Abstraite

Au

Exemple de diagramme de classes



Résumé

Nous avons vu:

- A quoi sert l'analyse des besoins
- Les concepts du diagramme de classes
- Comment passer des besoins au diagramme de classes
- Comment réaliser un cas d'utilisation par un diagramme de collaboration et un diagramme de classes

Méthodologie de systèmes d'information: une perspective orientée-objet

Le processus unifié et la notation UML

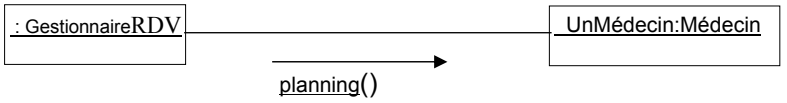
Les interactions entre objets

Auteur : S. SI-SAID CHERFI

L'envoi de messages

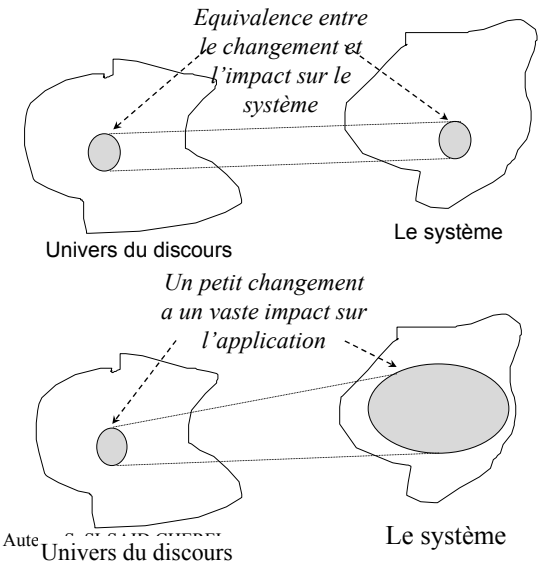
- Les objets communiquent par envoi de messages
- Envoyer le message fournir_planning() à un objet UnMédecin, doit utiliser la syntaxe suivante:

  planning = UnMédecin.planning()



L'envoi de messages

• Les objets communiquent à travers l'envoi de messages
•L'objectif des diagrammes d'interaction et de préciser les responsabilités des objets pour minimiser l'effet de bord résultant des changements des besoins (la résistance au changement)



Interaction et collaboration

• On appelle *collaboration* l'ensemble formée par la structure des instances participant à un comportement et par les liens qui les lient.

•Une *interaction* est définie dans le contexte d'une collaboration.

• Elle spécifie le canevas de communication entre les rôles dans la collaboration. Plus précisément, elle contient un ensemble de *messages* partiellement ordonné. Chaque message spécifie une communication comme par exemple l'opération à invoquer, ainsi que les rôles joués par la source et la cible du message.

« OMG 2001 »

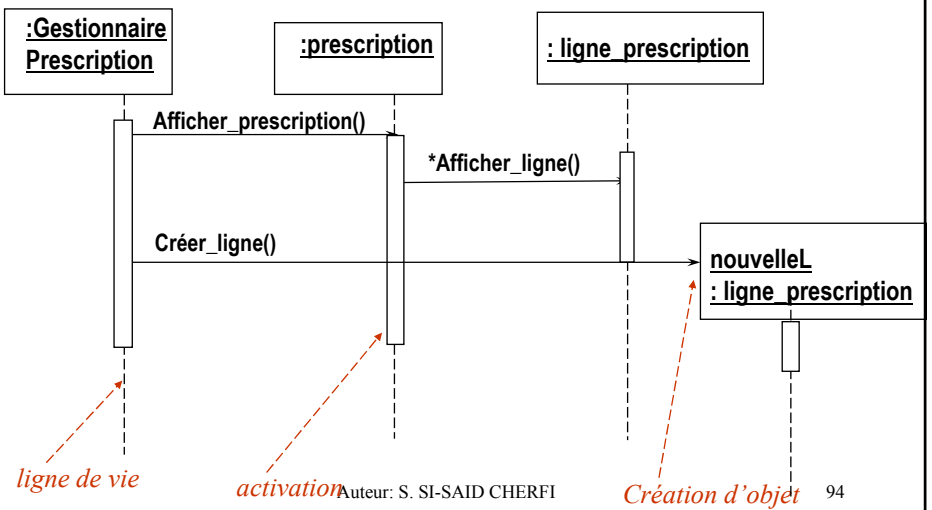
Diagramme de séquence

- Les interactions entre objets selon un point de vue temporel.
- Le contexte des objets n'est pas représenté de manière explicite.
- La représentation se concentre sur l'expression des interactions.
- Peuvent être utilisés à différentes phases du cycle de vie avec plus ou moins de détails.
- Peuvent être utilisés de façon:
  - informelle: documentation des cas d'utilisation.
  - formelle: pour une représentation précise des interactions entre objets. Le concept de message unifie toutes les formes de communication entre objets (appel de procédure, événement discret, …).

Les diagrammes de séquence

- La dimension verticale représente *le temps*
- Les *objets* impliqués dans l'interaction sont disposés horizontalement. Il sont représentés par des *lignes de vie* verticales
- Les *messages* sont représentés par des flèches pleines horizontales
- L'exécution d'une opération est matérialisée par une *activation*

Diagramme de séquence : notation

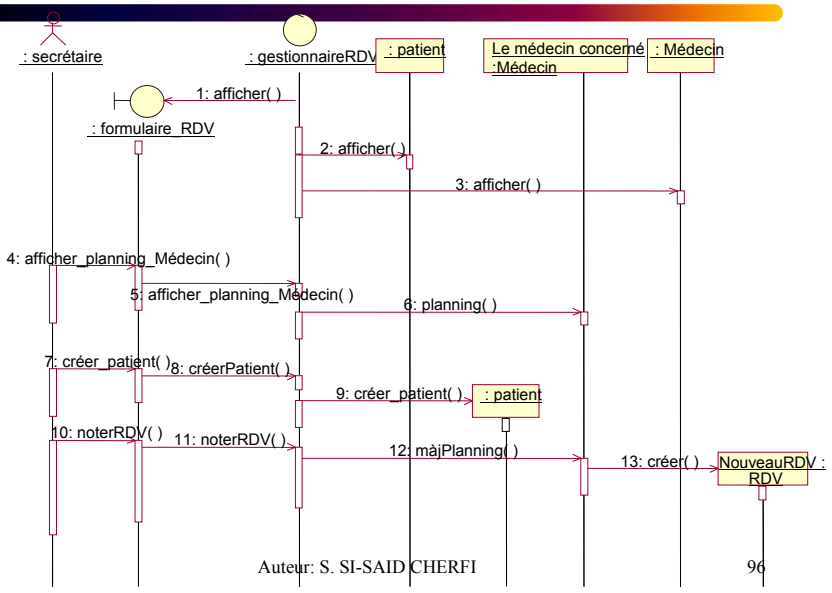


Classes contrôle et interface

- La majorité des cas d'utilisation nécessitent une classe interface qui gère la communication entre le système et les acteurs externes. Une classe de type « contrôle» gère l'ensemble de la communication entre objets.
- Certaines méthodes se limitent aux classes « entité » durant la phase d'analyse
- D'autres comme (Larman, 1998)* propose de représenter uniquement les classes « interface » (ou le système) et de se limiter aux échanges acteur-système
- Le processus unifié propose une solution intermédiaire.

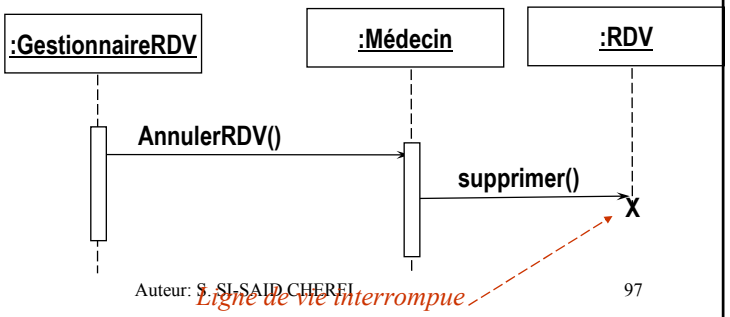
**** Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process (2nd Edition)***

Classes contrôle et interface



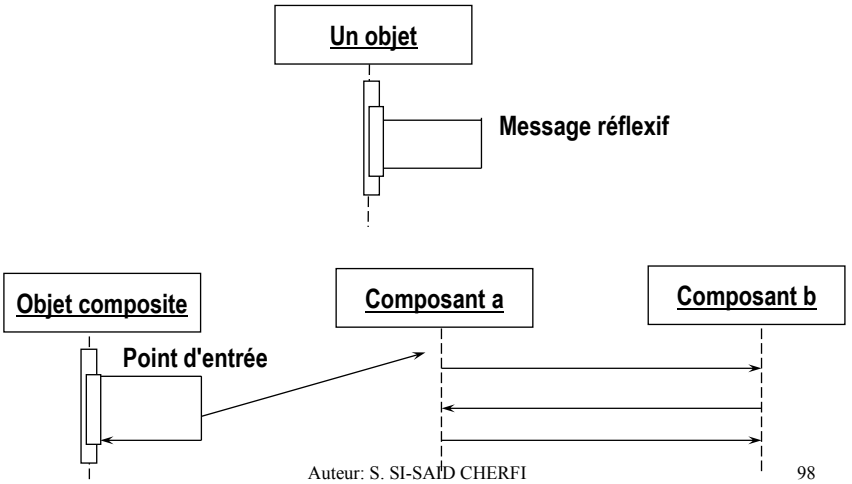
Création et destruction des objets dans un diagramme de séquence

• La création des objets se représente en faisant pointer le message de création sur le rectangle qui symbolise l'objet crée.
La destruction est indiquée par la fin de la ligne de vie et par une lettre X.

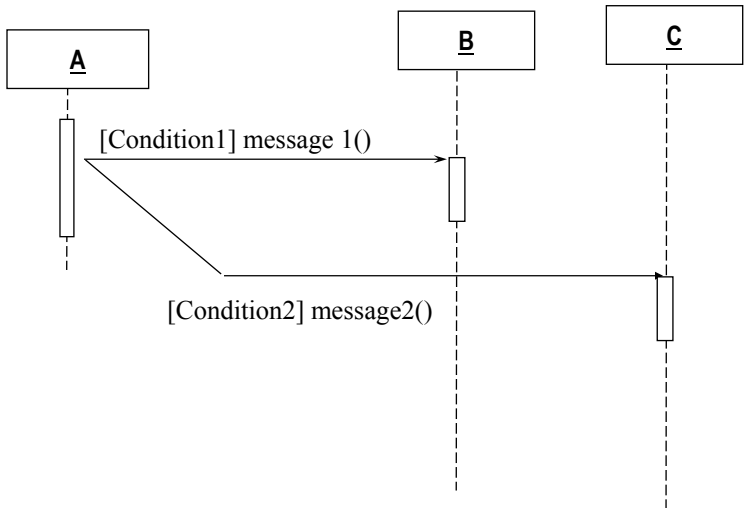


Messages réflexifs

• Un objet peut s'envoyer un message.

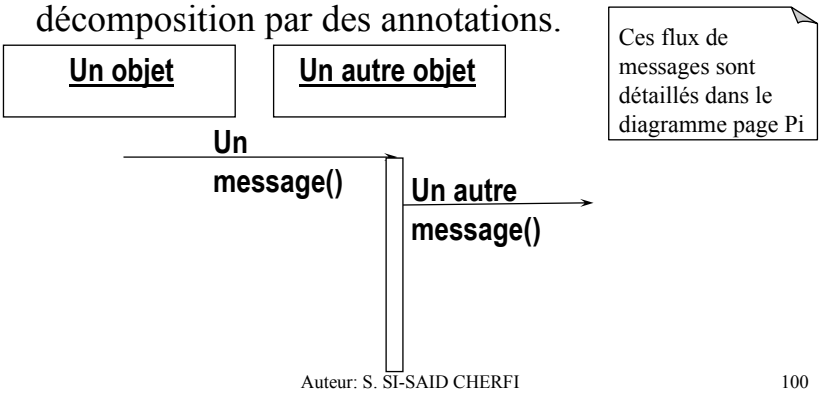


Les branchements conditionnels



Gérer la complexité

- Les diagrammes complexes peuvent être décomposés en plusieurs diagrammes plus simples en précisément convenablement cette décomposition par des annotations.



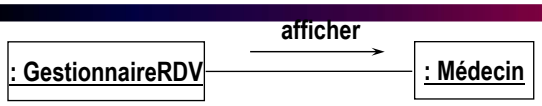
Diagrammes de collaboration

- Montrent des interactions entre objets, en insistant sur la structure spatiale statique qui permet la mise en collaboration d'un groupe d'objets.
- Extension des diagrammes d'objets.

Représentation des interactions

- Le contexte d'une interaction comprend les arguments, les variables locales créées pendant l'exécution, ainsi que les liens entre les objets qui participent à l'interaction.
- Une interaction est réalisée par un groupe d'objets qui collaborent en échangeant des messages.

Diagramme de collaboration



• Le temps n'est pas représenté de manière implicite: les messages sont numérotés pour indiquer l'ordre des envois.

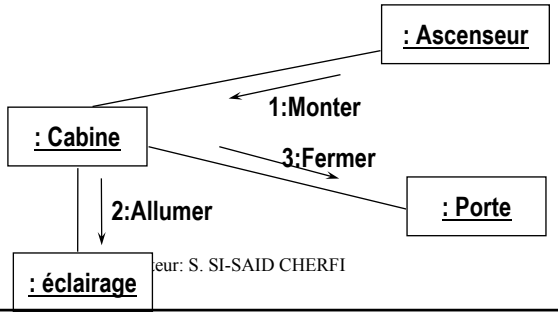
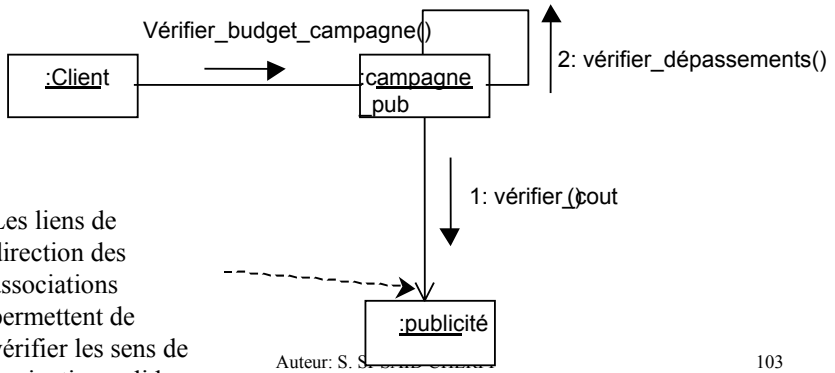


Diagramme de collaboration

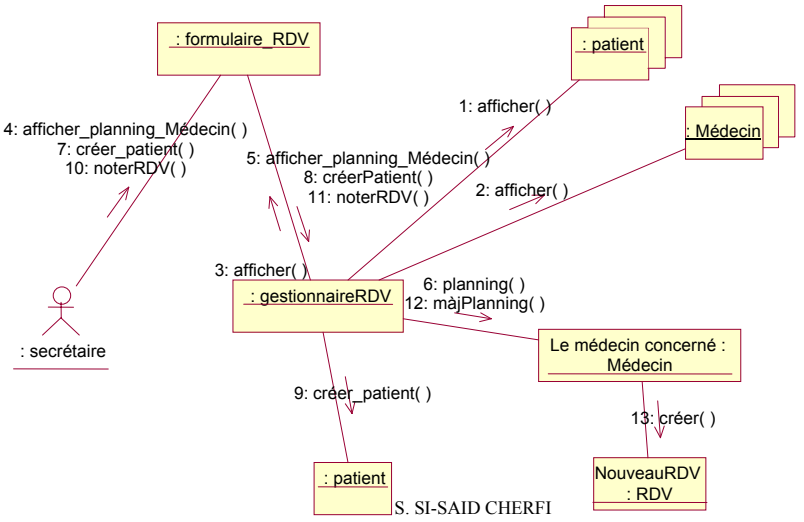
• Les diagrammes de collaboration montrent simultanément les interactions entre les objets et les relations structurelles qui permettent ces interactions.



Les liens de direction des associations permettent de vérifier les sens de navigation valides.

Vérifier la cohérence des modèles

- La définition des opérations dans les classes doit être cohérente avec les massages et les signatures de ces opérations avec les arguments des messages.
- Tout objet à l'origine d'un message doit avoir la référence de l'objet destination du message
  - à travers une association, ou un chemin d'associations
  - cette étape est importante pour la définition des associations dans le diagramme de classe (essentiellement celui de la conception)
  - Il est nécessaire de bien vérifier tous les chemins des messages.
- Les diagrammes de séquence et de collaboration doivent être cohérents entre eux.
- Les Les diagrammes de séquence et de collaboration doivent être cohérents avec les diagrammes d'état transition des objets impliqués.



Résumé

- Comment développer des collaborations à partir des cas d'utilisation
- Comment modéliser des collaborations avec des diagrammes de séquence
- Comment développer des collaborations avec des diagrammes de collaboration
- Comment vérifier la cohérence des diagrammes.

Spécifier les opérations et les contrôles:

Diagrammes d'activités et diagrammes d'états-transitions

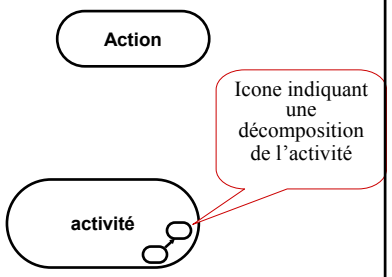
Auteur : S. SI-SAID CHERFI

Diagrammes d'activités

- Variante des diagrammes d'états-transitions, organisés par rapport aux actions et destiné à représenter le comportement interne d'un processus, d'une méthode ou d'un cas d'utilisation.
- Un diagramme d'activités représente l'état de l'exécution d'un mécanisme, sous la forme d'un déroulement d'étapes regroupées séquentiellement dans des branches parallèles de flot de contrôle.
- Chaque activité représente une étape particulière dans l'exécution de la méthode qui l'englobe.
- Les activités sont reliées par des transitions automatiques. Lorsqu'une activité se termine, la transition est déclenchée et l'activité suivante démarre.
- Les activités ne possèdent ni transitions internes, ni transitions déclenchées par des événements.

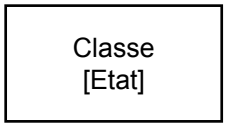
Les étapes dans un diagramme d'activités

- Action (état)
- activité (état)
- Similaires aux états (état simple et composé) à la différence que ...
- ... Les transitions sont faites lorsque l'étape se termine, au lieu d'être déclenchée par événement externe, …



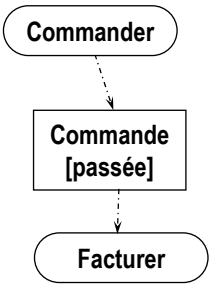
Flots d'objets

- Un concept qui permet de représenter la disponibilité d'un objet dans un état donné.
- Ne correspond pas à l'invocation d'une action et le contrôle passe à l'étape suivante
- Il permet de représenter des contraintes sur les entrée/sorties des activités.

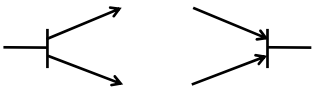


Flots d'objets

- Facturer nécessite en entrée l'existence d'une commande dans l'état passée.
- Les lignes en pointillées ont la même sémantique qu'une transition.

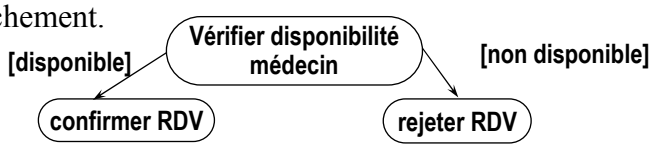


Ordonnancement des étapes

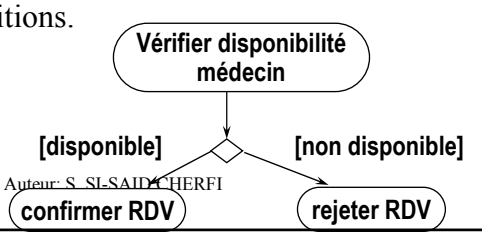
- Hérité des machines à états
- Etat initial 
- État final 
- Fork et join 

Ordonnancement des étapes

- Les transitions entre activités peuvent être gardées par des conditions booléennes, mutuellement exclusives. Les gardes sont à proximité des transitions dont elles valident le déclenchement.

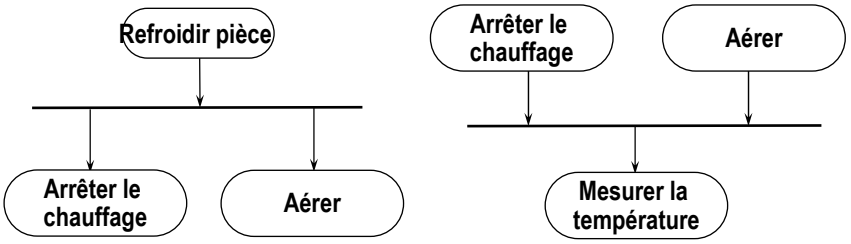


- UML définit un stéréotype optionnel pour la visualisation des conditions. Une condition est matérialisée par un losange d'où sortent plusieurs transitions.



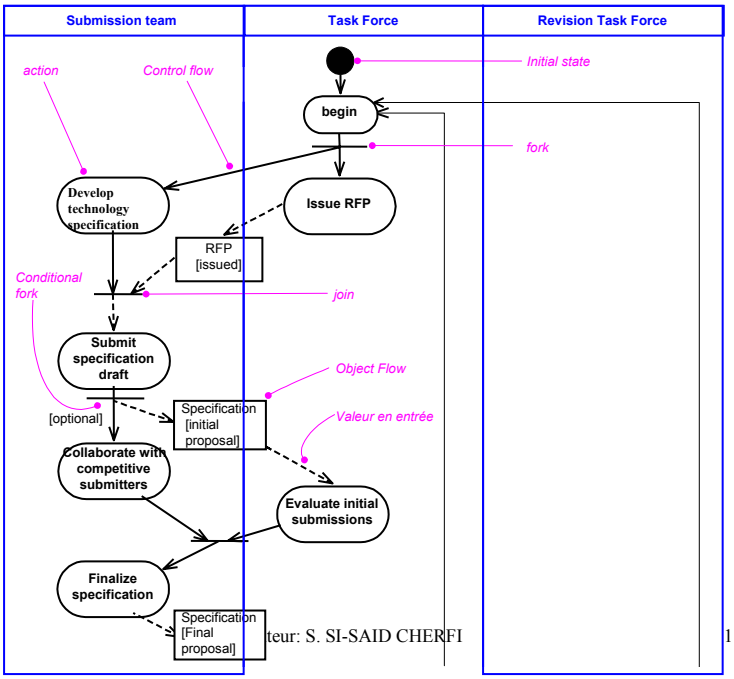
Ordonnancement des étapes

- Les diagrammes d'activités représentent les synchronisations entre flots de contrôle, au moyen de barres de synchronisation.

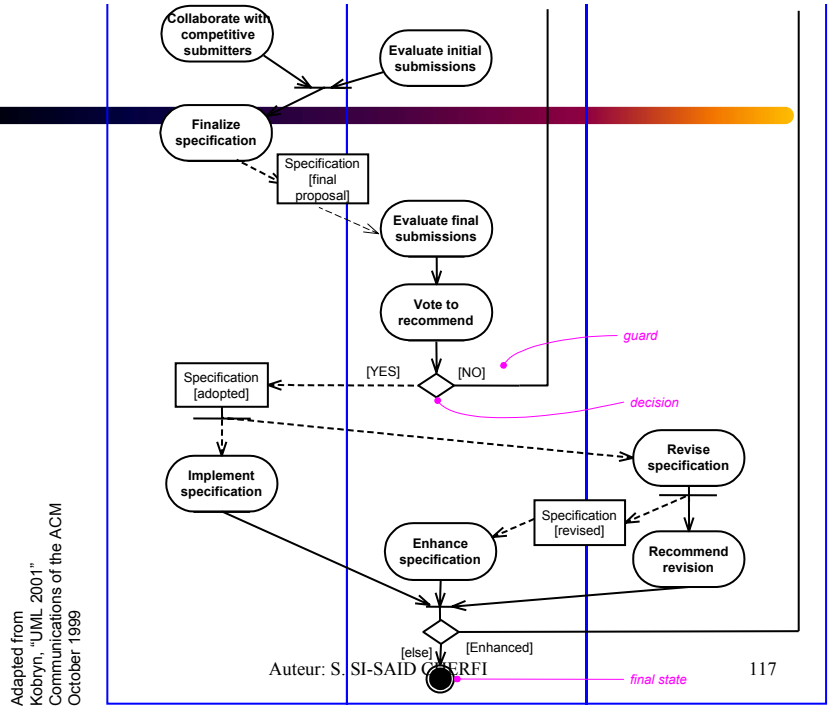


Ordonnancement des étapes

- Les objets peuvent être utilisés comme moyen de synchronisation
- Les diagrammes d'activités peuvent être découpés en **couloirs d'activités** pour montrer les différentes responsabilités au sein d'un mécanisme ou d'une organisation. Chaque responsabilité est assurée par un ou plusieurs objets et chaque activité est allouée à un couloir donné.



Exemple



Adapted from Kobryn, "UML 2001" Communications of the ACM October 1999

Diagramme d'état transition

- Modéliser le comportement d'un système face à la survenance d'un événement:
  - Exemple
    - Si l'on introduit de la monnaie dans un distributeur de boissons, la réaction de la machine peut être:
      - servir le café,ou
      - ne rien faire,
    - La deuxième réaction peut signifier que:
      - la somme introduite est insuffisante, ou
      - il n'y a plus de café dans la machine, ou
      - elle est en panne.
    - La réaction face à un événement dépend de l 'état de l'objet qui **évolue dans le temps**.

L'état

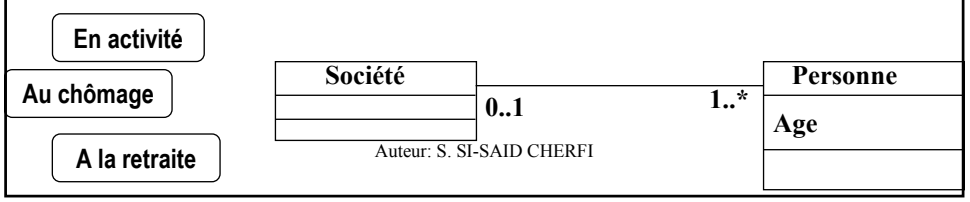
'Un état est une condition durant la vie d'un objet ou d'une interaction pendant laquelle il remplie une certaine condition, accomplie une action ou attend un évènement…

Conceptuellement, un objet reste dans un état durant un intervalle de temps. …'

OMG, 2001

État

- Chaque objet est à un moment donné dans un état particulier. Chaque état possède un nom qui l'identifie.
- Les états se caractérisent par la notion de durée et de stabilité.
- Un état est toujours l'image de la conjonction instantanée des valeurs contenues par les attributs de l'objet, et de la présence ou non de liens, de l'objet considéré vers d'autres objets.



État

Exemple

Pour connaître la situation d'une personne en particulier, il faut étudier la conjonction suivante:

- âge de la personne,
- présence d'un lien vers une société.

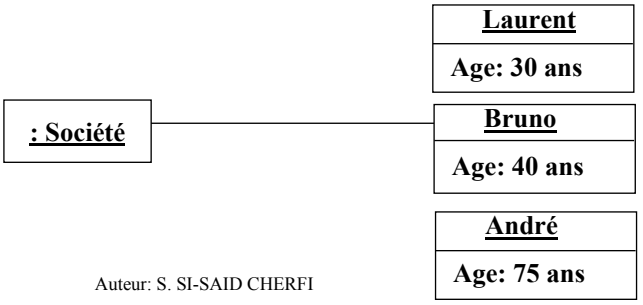


Diagramme d'états-transitions

- Visualise un automate d'états finis, du point de vue des états et des transitions.

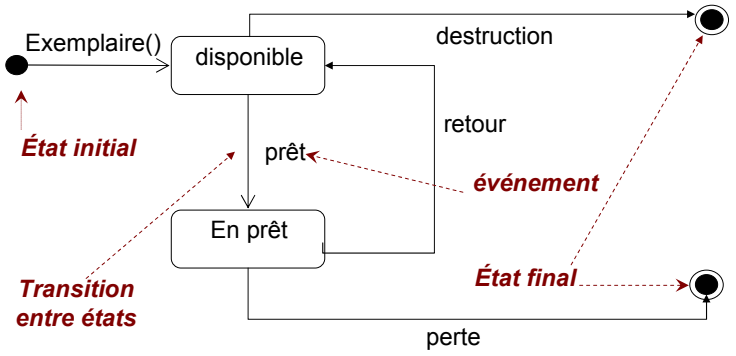
Les automates

- Le comportement des objets d'une classe peut être décrit de manière formelle en termes d'états et d'événements, au moyen d'un automate relié à la classe considérée.

Diagramme d'états-transitions

- Les objets qui ne présentent pas un comportement réactif très marqué peuvent être considérés comme des objets qui restent toujours dans le même état: leurs classes ne possèdent alors pas d'automate.
- Formalisme des automates: s'inspire des *Statecharts*. (Harel, D. 1987)
- Un automate est une abstraction des comportements possibles.

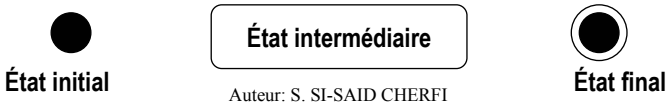
Diagramme d'état transition



- La transition d'un état vers l'autre dépend de l'événement qui survient et de l'état de l'objet.

Diagramme d'état transition

- Les automates retenus par UML sont déterministes. Il faut toujours décrire ***l'état initial*** du système. Pour un niveau hiérarchique donné, il y a toujours ***un et un seul état initial***. En revanche, il est possible d'avoir ***plusieurs états finaux*** qui correspondent chacun à une condition de fin différente. Il est également possible de n'avoir ***aucun état final***, dans le cas par exemple d'un système qui ne s'arrête jamais.

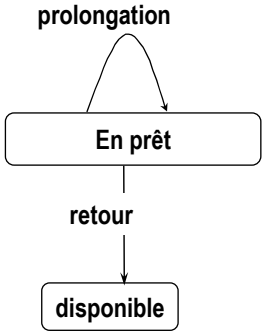


Transition

- Lorsque les conditions dynamiques évoluent, le objets changent d'état en suivant les règles décrites dans l'automate associé à leurs classes.
- Les diagrammes d'états-transitions sont des graphes dirigés.
- Les états sont reliés par des connexions unidirectionnelles, appelées ***transition.***

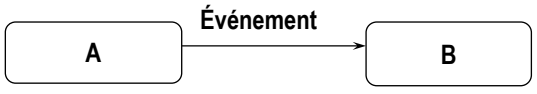
Transition

- Les transitions ne relient pas nécessairement des états distincts.



Événement

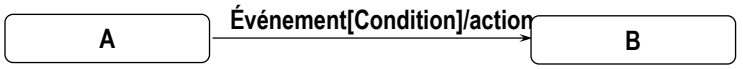
- Un événement correspond à l'occurrence d'une situation donnée dans le domaine du problème.
- Un événement est par nature une information instantanée qui doit être traitée sans plus attendre. C'est un déclencheur pour passer d'un état à un autre.
- Les événements déterminent quels chemins doivent être suivis dans les graphes d'états. Un objet, placé dans un état donné, attend l'occurrence d'un événement pour passer dans un autre état.



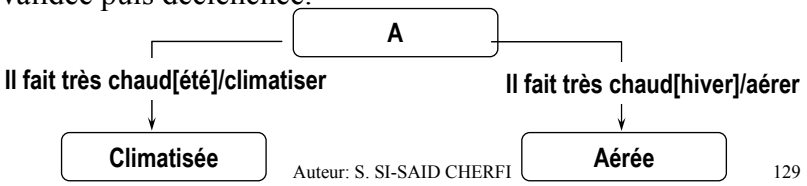
Un événement déclenche la transition qui lui est associée.

Garde

• Une garde est une condition booléenne qui valide ou non le déclenchement d'une transition lors de l'occurrence d'un événement.



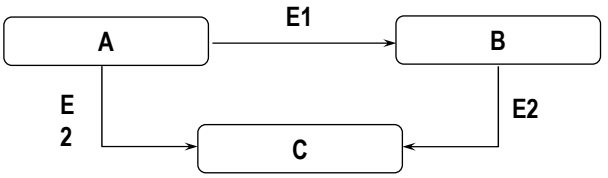
• Les gardes permettent de maintenir l'aspect déterministe d'un automate d'états finis. Lorsque l'événement a lieu, les gardes, qui doivent être mutuellement exclusives, sont évaluées et une transition est validée puis déclenchée.



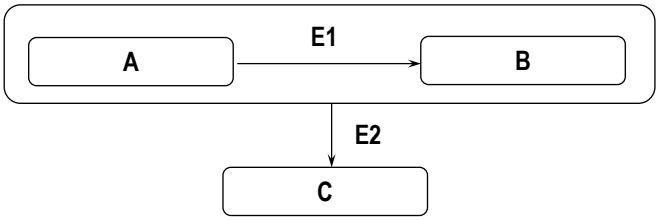
Généralisation d'états

- Principe de généralisation d'états: les états plus généraux sont appelés ***super-états***, les états plus spécifiques ***sous-états.***
- Un état peut être décomposé en plusieurs sous-états disjoints; les sous-états héritent des caractéristiques de leur super-état.
- *Décomposition disjonctive:* l'objet doit être dans un seul et un seul sous-état à la fois.

Généralisation d'états

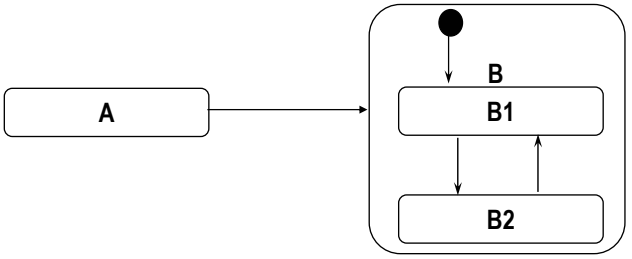


E2 peut être factorisée.



Généralisation d'états

• Il est préférable de limiter les liens entre niveaux hiérarchiques d'un automate en définissant systématiquement en état initial pour chaque niveau.



Construire les diagrammes d'états-transitions

- Deux approches sont possibles:
  - L'approche comportementale
  - L'approche par cycle de vie

Allen and Frost (1998)

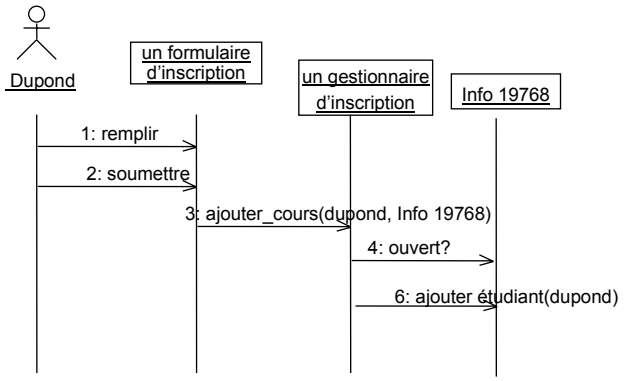
L'approche comportementale

1. Examiner toutes les diagrammes d'interaction qui implique un fort échange de messages.
2. Identifier les messages entrants pouvant correspondre à des événements pour chaque diagramme d'interaction. Identifier également les éventuels états résultants.
3. Documenter ces états et ces événements par un diagramme d'états transitions.
4. Compléter le diagramme obtenu.

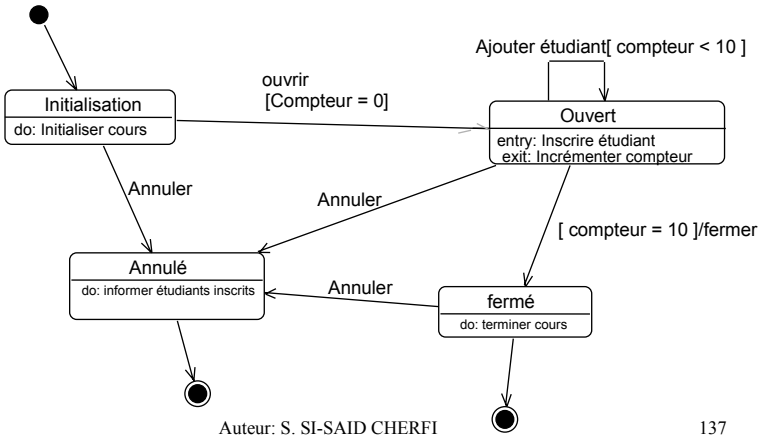
L'approche comportementale

5. Développer les états complexes.
6. Contrôler la cohérence du diagramme obtenu.
7. Répéter les étapes 4 et 5 jusqu'à couvertures de tous les comportements possibles.
8. Vérifier la cohérence du diagramme obtenu en le confrontant aux autres diagrammes (classes, interactions).

Approche comportementale



Approche comportementale



Approche par cycle de vie

- Considérer le cycle de vie des objets pour chaque classe
- Les événements et les états sont identifiés à partir des cas d'utilisation et à partir de toute autre spécification ou documentation des besoins
- On commence par recenser les événements majeurs du système
- Chacun des événements est ensuite examiné pour identifier les objets susceptibles d'avoir un comportement dépendant de cet événement et pouvant y répondre.

Approche par cycle de vie

1. Identifier les événements majeurs du système.
2. Identifier les classes susceptibles de répondre à cet événement.
3. Pour chacune de ces classe construire un diagramme d'états initial retraçant un cycle de vie type.
4. Étudier le diagramme résultant et y ajouter les détails nécessaires au comportement décrit

Approche par cycle de vie

5. Inclure les comportements alternatifs.
6. Réviser le diagramme pour s'assurer de sa cohérence avec les cas d'utilisation. En particulier, vérifier que les contraintes imposées par le diagramme d'états sont valides (vis à vis des besoins).
7. Répéter les étapes 4 et 5 jusqu'à couvertures de tous les comportements possibles.
8. Vérifier la cohérence du diagramme obtenu en le confrontant aux autres diagrammes (classes, interactions).

Approche par cycle de vie

- Est moins formelle que l'approche comportementale dans la manière d'identifier les événements et les les états de départ
- Il est courant de combiner les deux approches

Vérifier la cohérence

- Chaque événement doit apparaître comme un événement entrant sur l'objet approprié dans les diagrammes d'interaction
- Chaque action doit correspondre à l'exécution d'une opération de la classe appropriée, et éventuellement à l'envoi de message vers d'autres objets
- Chaque événement doit correspondre à une opération mais l'inverse n'est pas toujours vrai.
- Chaque événement sortant du diagramme d'état doit correspondre à un message envoyé vers un autre objet.