Les différentes variantes du *Rational Unified Process*

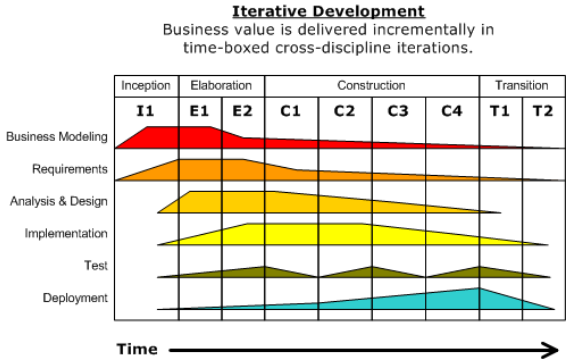
- Le *Unified Process* (UP) est la version la plus documentée du RUP.

⇒ C'est une version générique adaptable aux besoins particuliers.

- Le *Agile Unified Process* (AUP) ajoute un caractère *évolutif* à RUP.

⇒ On s'appuie sur une haute qualification des développeurs pour limiter le plus possible la production de documents préliminaires au développement. Les cas d'utilisation sont représentés par des tests exécutables. Un prototype est développé très rapidement et dirigés par la validation de ces tests. La spécification est construite en même temps que le logiciel, une fois que celui-ci est confronté, sous une forme exécutable, aux utilisations des clients.

Les phases du *Rational Unified Process*



- Dans ce diagramme, les lignes correspondent aux itérations.
- Les colonnes correspondent aux phases.

Phase « initialisation » (*inception*)

- Cette phase phase correspond à l'*étude de faisabilité* dont nous avons parlée dans le cours précédent.

Phase « élaboration »

- Il s'agit de l'*analyse des besoins*.
- Celle-ci fait un usage intensif des cas d'utilisation pour raffiner la compréhension du problème posé et expliciter les spécificités du domaine.
- Des prototypes (parmis lesquels on trouve l'architecture exécutable) sont développés pour évaluer concrètement des points techniques risqués.

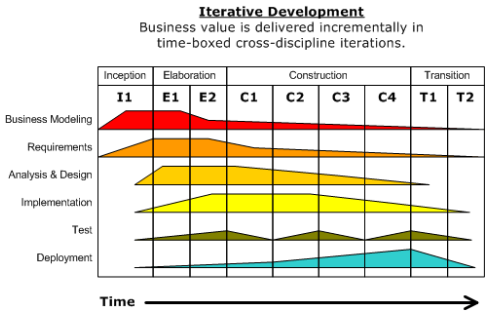
Phase « construction »

- Cette phase correspond à la *conception* et au *développement*.
- Elle suit en général une progression incrémentale aboutissant à diverses versions du système, résolvant en priorité les problèmes techniques à hauts risques.
- Dans une conception orienté objet, il est parfois difficile de bien distinguer la spécification de l'implémentation, certains experts préconisent la définition de deux modèles disjoints : un modèle logique et un modèle d'implémentation.
- Cette distinction est importante car il doit toujours exister une spécification servant de référence aux développements et sur laquelle appuyer le cahier des charges.

Phase « transition »

- La phase de transition de ce processus correspond à l'activité de maintenance et de déploiement.
- Il s'agit de vérifier la mise en place du système auprès des utilisateurs (production de manuel d'utilisation, formation, ...) et de préparer ses futures évolutions.

Cours d'aujourd'hui



- Nous allons nous intéresser aux diagrammes d'UML utilisés dans ces différentes phases.

Spécification à l'aide d'UML

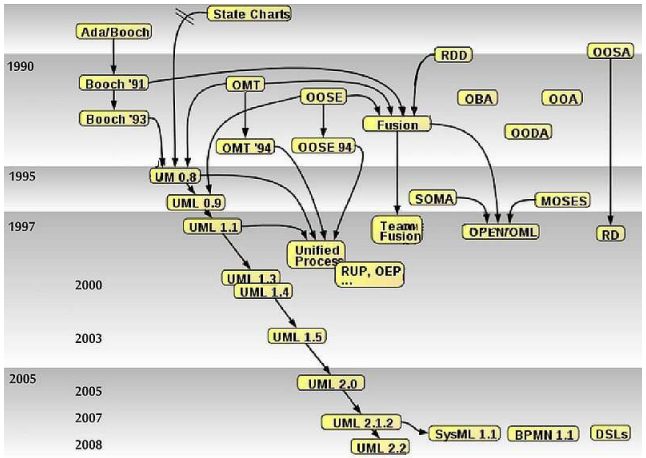
Présentation

- UML est l'acronyme de *Unified Modeling Language*.
- UML est un ensemble de notations.
- Ces notations sont en majorité des formats de diagrammes.
- UML est standardisé par l'*Object Management Group* (OMG).
- UML est la notation la plus utilisée par l'industrie logicielle.
- La spécification d'UML 2.2 se trouve ici :

  `http://www.omg.org/spec/UML/2.2/`
- Nous ne pourrons pas l'étudier dans ses recoins.

⇒ Vous devez vous y référer pour écrire vos spécifications.

Histoire



- Résolution des conflits par union plutôt que par intersection.

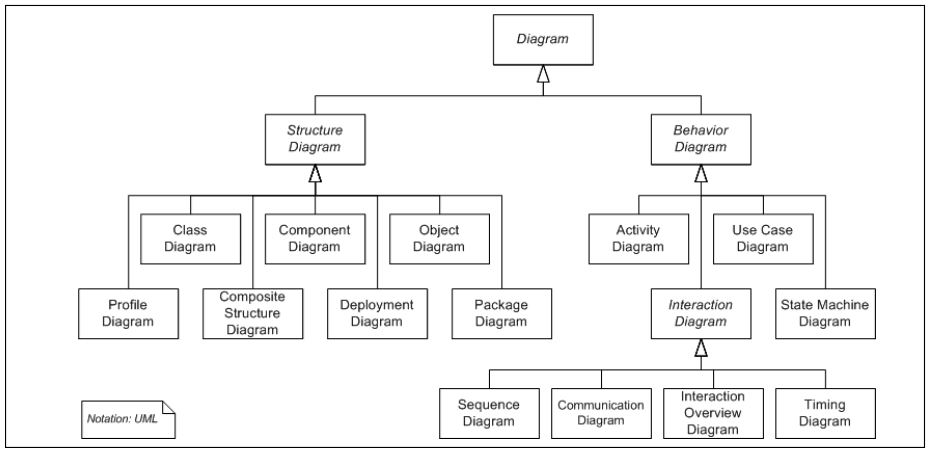
Critiques d'UML

- Avantages :
  - Plusieurs modèles sont réunis : objet, orienté donnée, flot de données.
  - Il existe de nombreux outils pour produire des diagrammes UML.
  - C'est le résultat d'un consensus entre plusieurs « écoles » de modélisation.
- Inconvénients :
  - La sémantique d'UML n'est pas encore fixée. (Toutefois, des experts essaient de définir *Precise UML*, un sous-ensemble formalisé d'UML.)
  - C'est seulement depuis la version 2.0 que la syntaxe est standardisée.
  - Les notations sont parfois redondantes.

Différentes vues sur un système

- UML fournit des diagrammes pour 4 types de vues :
  - les vues de cas d'utilisation ;
  - les vues d'architecture ;
  - les vues dynamiques ;
  - les vues statiques.

Une abondance de diagrammes . . .



Les diagrammes d'UML 2.2.

Vues de cas d'utilisation

Les cas d'utilisation

- Rappel de la définition :

  Un cas d'utilisation est la représentation d'une interaction entre le système et un acteur en vue de la réalisation d'un objectif.

- On applique ici le principe de séparation des problèmes en sous-problèmes : on se focalise sur une certaine utilisation du système en oubliant le reste.
- En plus de réduire temporairement la complexité du système, cette unité de description est intéressante car elle est accessible aux clients non-experts.
- Lorsque l'on suit RUP, les cas d'utilisation sont décrits par deux notations :
  - les diagrammes de cas d'utilisation d'UML ;
  - les spécifications en langage structuré (vues la dernière fois).
- La méthode AGILE représente les cas d'utilisation par des **programmes exécutables** de façon à pouvoir vérifier leur satisfiabilité de façon automatique et quantifier l'avancée du développement.

Retour sur l'exemple en langage structuré

Contexte	Une partie est en cours. Le joueur a formulé une requête d'action.
Flot normal	La requête est excaucée. L'état de la partie est modifiée en accord avec le scénario et l'interface graphique est mise-à-jour. Un message textuel informe le joueur du changement.
Cas problématique	L'action n'est pas applicable. Le joueur est informé des causes de l'erreur. Il peut formuler une autre action.
Activités concurrentes	Les animations de la scène se poursuivent tout au long de la résolution de la requête d'action.

Scénario « résolution d'une action »

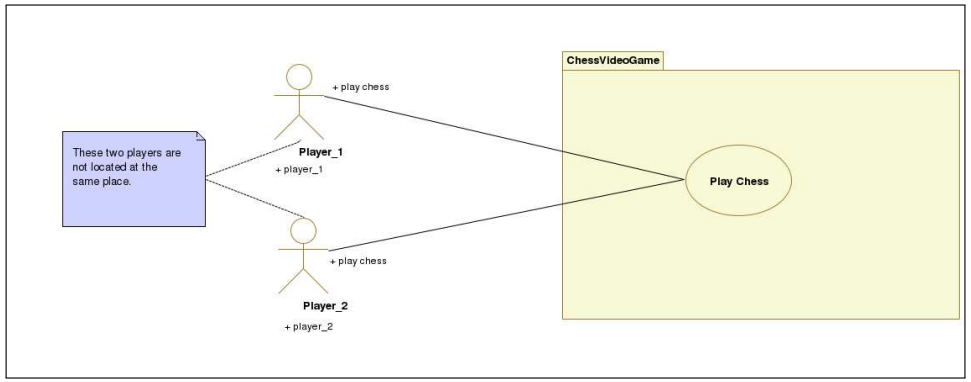
Grammaire visuelle des diagrammes de cas d'utilisation d'UML

- On représente un acteur par un personnage schématisé.

⇒ Attention, cependant, un acteur n'est pas forcément un utilisateur !

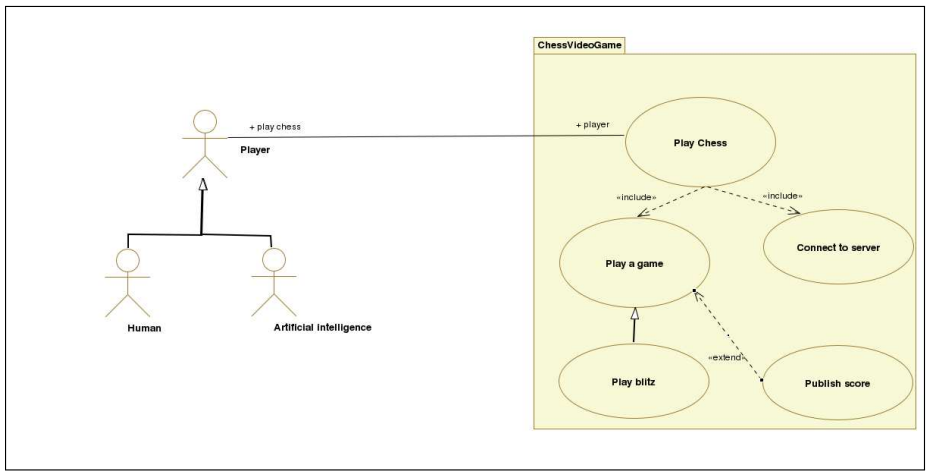
- On représente des relations logiques entre les acteurs (voir plus loin).
- Le système est inclus dans un rectangle, éventuellement étiqueté.
- Les interactions entre le système et les acteurs sont représentées par des flèches pointillées.
- Les cas d'utilisation sont des verbes à l'infinitif entourés par des ellipses.
- On représente des relations logiques entre les cas d'utilisation (voir plus loin).

Exemple de diagramme



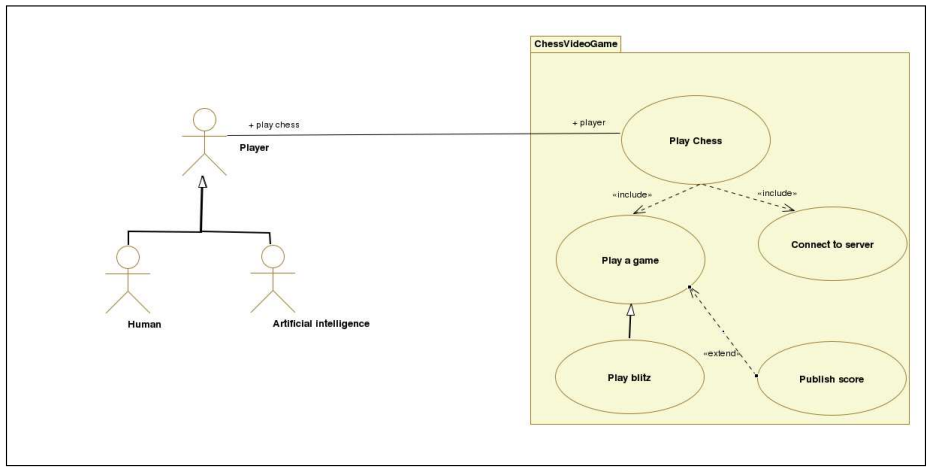
- En général, tout diagramme UML peut être annoté par un complément textuel d'information attaché à ces entités visuelles.

Exemple de diagramme



- Une relation d'héritage permet de classifier les acteurs.
- Si l'acteur $\mathcal{A}_2$ hérite de l'acteur $\mathcal{A}_1$ alors tous les scénarii de $\mathcal{A}_1$ sont accessibles à $\mathcal{A}_2$.

Exemple de diagramme



- On retrouve ici les relations « étend » et « inclus » définies la dernière fois.

Étude de cas : Diagramme de cas d'utilisation

Exercice

Traduisez la description textuelle du cas d'utilisation en un diagramme.

Rôle des cas d'utilisation dans RUP

- Ils jouent un **rôle central**.
- Ils servent de matériau de base à la phase de conception.
- Classifier les cas d'utilisation, en termes logiques, de priorité ou de risque, permet d'organiser l'analyse qui suit.
- De nombreuses vues dynamiques sont des **raffinements** des cas d'utilisation.
- Ces raffinements précisent le **vocabulaire** et les mécanismes mis-en-jeu.
- Si une nouvelle utilisation du système apparaît pendant l'analyse des besoins, il est systématiquement inséré dans la base des cas d'utilisation.
- La validation consiste souvent à formuler une version **vérifiable/exécutable** des cas d'utilisation et à y confronter le système.
- Enfin, le manuel d'utilisation du système s'appuie très largement sur cette base de connaissance.

Activités liées à l'explicitation des cas d'utilisation

- À titre indicatif, voici une succession d'activités pouvant mener à l'obtention des cas d'utilisation :

1. Identification des acteurs principaux.
   - Les acteurs à satisfaire en priorité.
   - Les entités externes vitales au système.
2. Identification des cas d'utilisation principaux.
   - On omet les situations exceptionnelles.
   - On obtient une description intentionnelle (centrée sur les objectifs).
   - On met à jour les termes et concepts incontournables du système.
3. Identification des acteurs secondaires.
   - Des acteurs qui interviennent dans les cas d'utilisation découverts.
4. Identification des cas d'utilisations secondaires.
   - Par raffinement des cas d'utilisation principaux.
5. Factorisation des redondances.
6. Définition du vocabulaire du domaine.
   - Les cas d'utilisation soulèvent des questions sur le sens des termes employés par les acteurs.

Étude de cas : Extraction des cas d'utilisation

Exercice

Définissez les acteurs et cas d'utilisation principaux du moteur générique de jeu d'aventure.

Critique des cas d'utilisation

- Malgré les qualités citées plus tôt, la centralisation autour des cas d'utilisation peut avoir des faiblesses :
- Coût : l'énumération des cas d'utilisation et de leurs variations peut induire une combinatoire assez importante.
- Conception : le point de vue « utilisateur » n'est pas forcément la bonne façon d'aborder un problème. Par exemple, les utilisateurs peuvent avoir une vue incomplète du problème ou être des instances (inconscientes) de problèmes plus généraux.
- Imprécision : il est très difficile d'avoir un discours précis en s'exprimant seulement à l'aide de cas d'utilisation. Formaliser rapidement les concepts ou processus primordiaux permet d'en saisir les subtilités.

Vues d'architecture

Vues d'architecture

- L'architecture est une vue d'ensemble du système.
- C'est un point de conception à haut risque.
- Il s'agit de partitionner le système en sous-systèmes.
- Un bon partitionnement établit :
  - une faible dépendance entre les sous-systèmes ;
  - affecte un rôle clair et distinct à chaque sous-système ;
  - permet de couvrir l'ensemble des les cas d'utilisation.

Exemple de diagramme

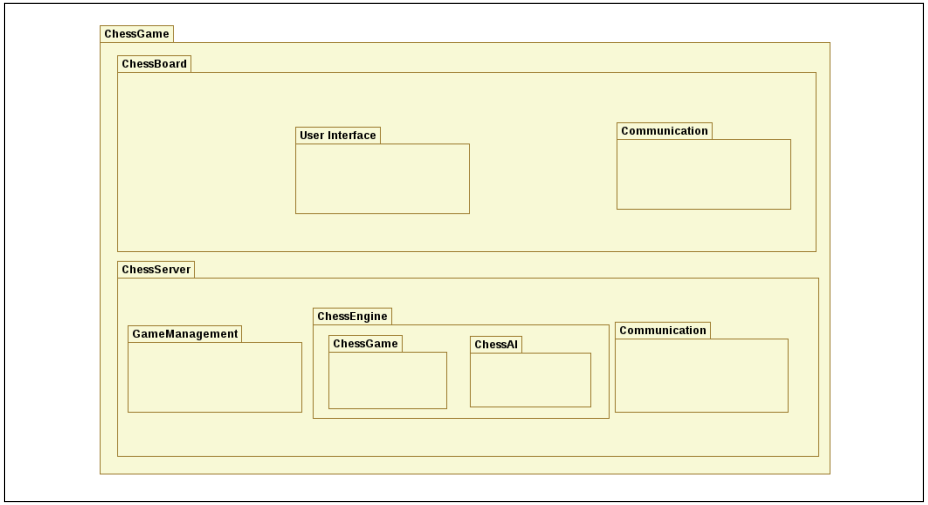
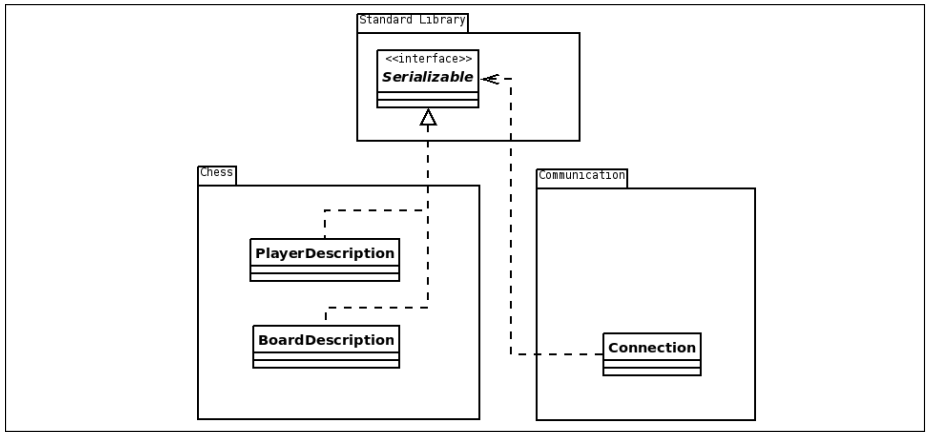


Diagramme d'architecture ou de paquets.

Exemple de diagramme

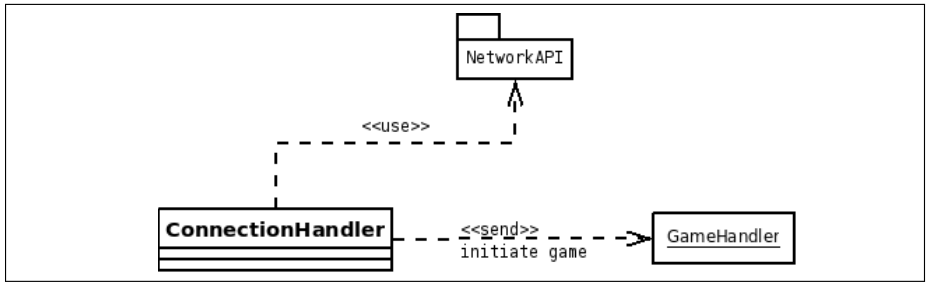


- Les « boîtes » qui apparaissent ici sont des classes importantes du système que l'on commence à classifier en termes de leur appartenance à un sous-système donné.
- Les relations entre ces classes induisent des dépendances entre les sous-systèmes.

Dépendances entre sous-systèmes

- Lorsqu'un sous-système dépend d'un autre, on doit commencer par établir l'interface de ce dernier.

Exemple de diagramme



- On peut préciser les relations de dépendances à l'aide d'annotations.
  (Voir la spécification d'UML pour connaître les annotations standards.)

Exemple de diagramme

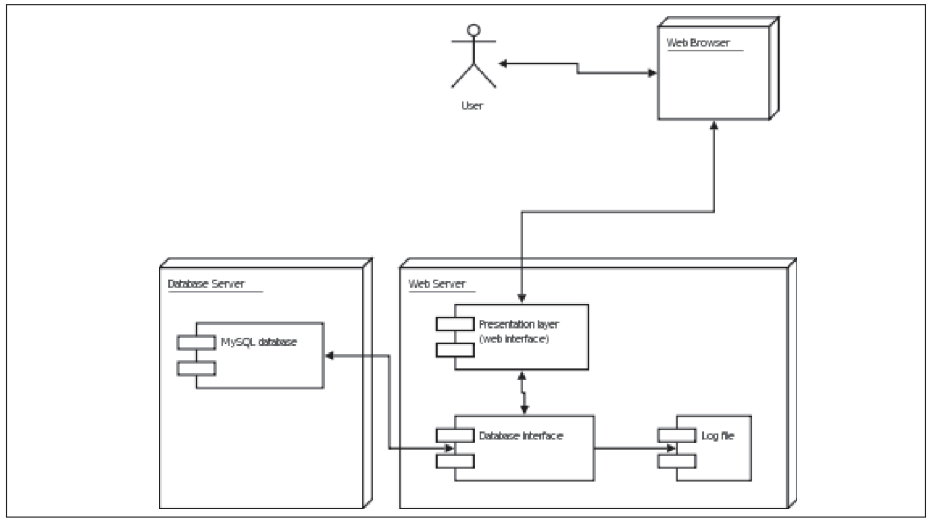


Diagramme de déploiement.

Vues dynamiques

Vues dynamiques

- Les vues dynamiques décrivent le comportement du système.
- Elles permettent de préciser les cas d'utilisation sous la forme d'**interaction** entre objets ou de décrire l'**évolution de l'état** des objets de façon abstraite en termes de réactions vis-à-vis de leur environnement et des messages qui leur sont envoyés.

Exemple de diagramme

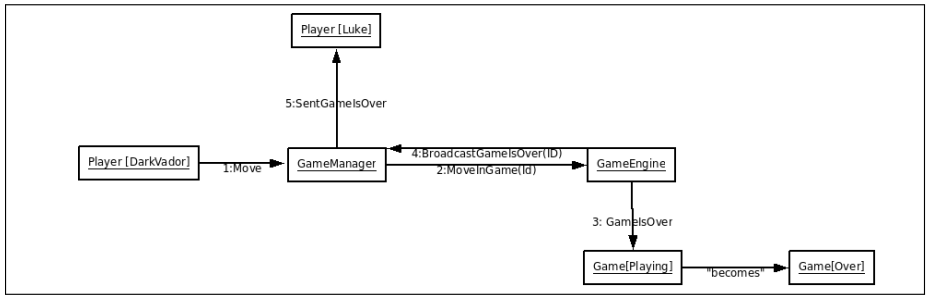


Diagramme de collaboration ou de communication.

- Les objets soulignés correspondent à des **instances**.
- On représente ici un scénario comme un enchaînement d'envoi de messages entre objets
- Le nom entre crochets symbolise l'état de l'objet à chaque étape du scénario, lorsqu'il est informatif.
- Cette notation met l'accent sur les objets nécessaires à la réalisation d'un cas d'utilisation.

Exemple de diagramme

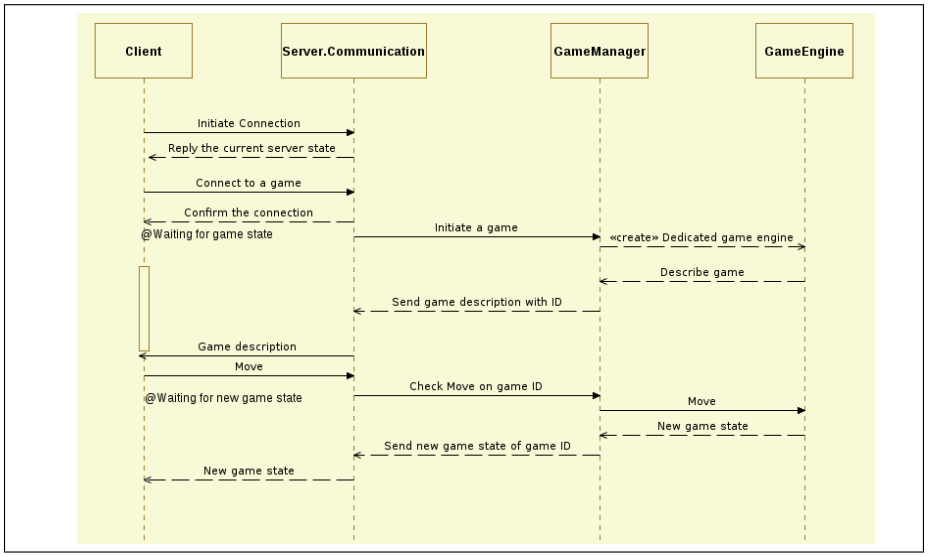


Diagramme de séquence

Diagramme de séquence

- Un diagramme de séquence présente les interactions entre les objets comme une succession de couples **message/réponse**.
- On peut y dénoter des contraintes de réponses synchrones ou asynchrones, des états bloquants, ...
- Cette notation met l'accent sur le protocole de communication entre les objets.

Exemple de diagramme

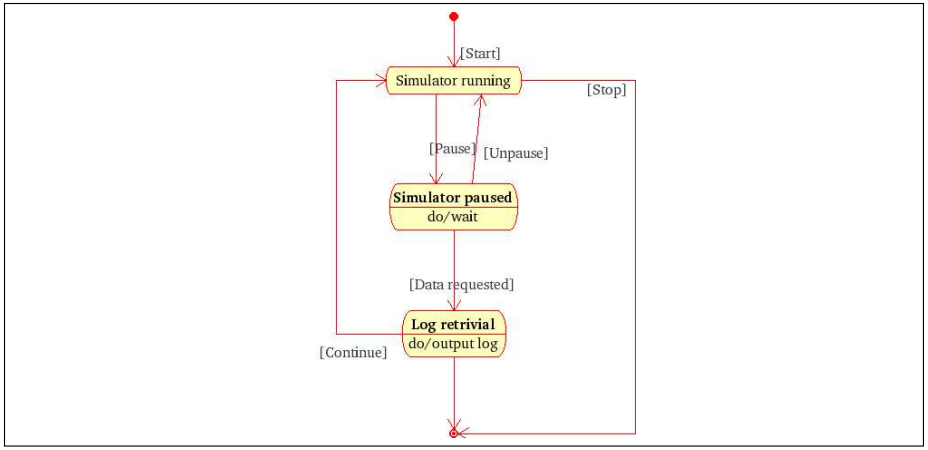


Diagramme d'état

Exemple de diagramme

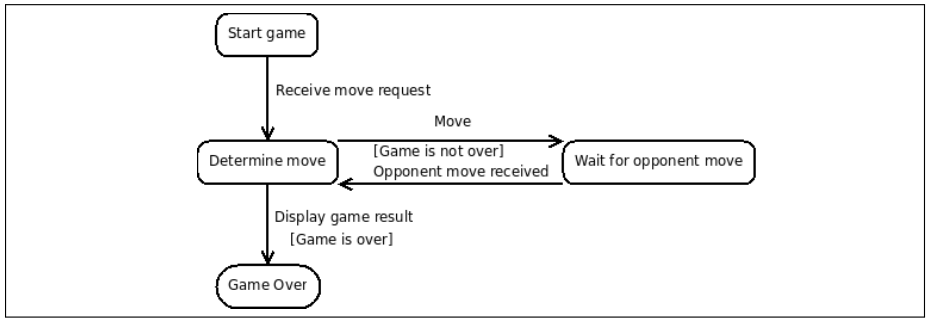


Diagramme d'état

Diagramme d'état

- Les diagrammes d'état représentent l'évolution de l'état du système (ou d'un sous-système) sous la forme d'un automate.
- Une transition de cet automate est suivie en réaction à un événement.
- Elle peut être conditionnée par des contraintes exprimées sur le système.

Exemple de diagramme

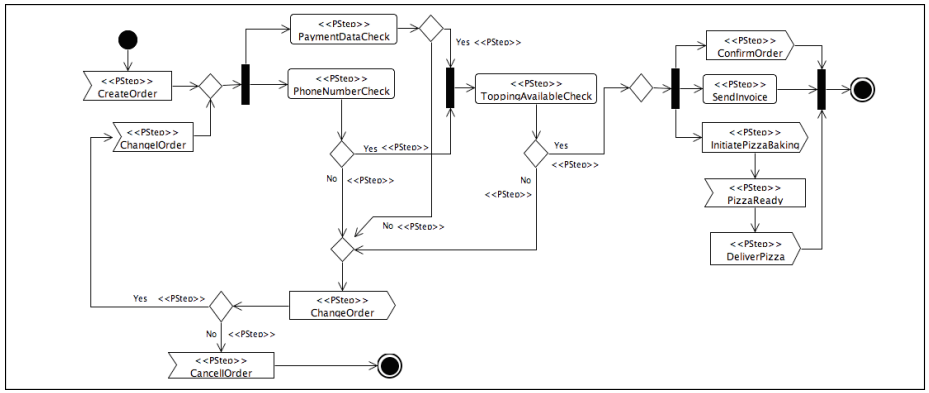


Diagramme d'activité.

Exemple de diagramme

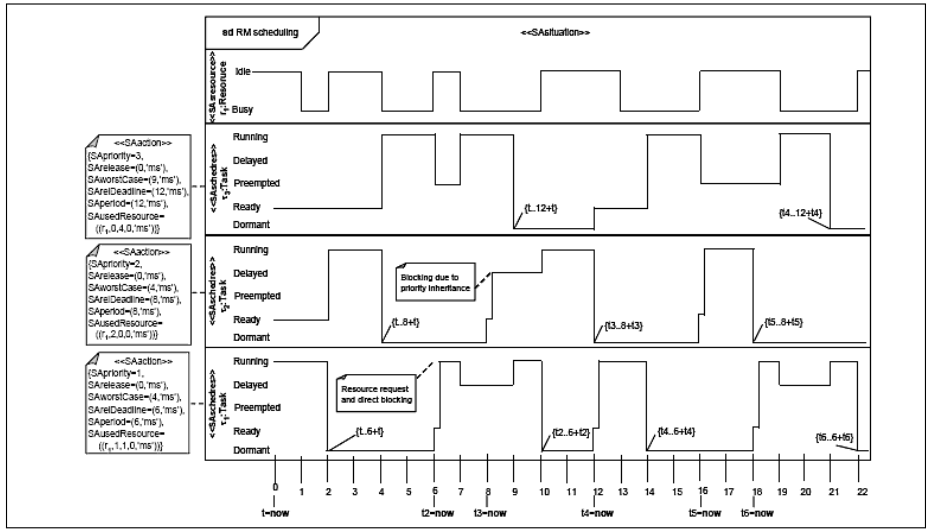


Diagramme de mesure temporelle.

Vues statiques

Vues statiques

- Les vues statiques établissent la **structure** du système.
- Il s'agit d'énumérer les différentes **classes** d'objets et leur relation.

Exemple de diagramme

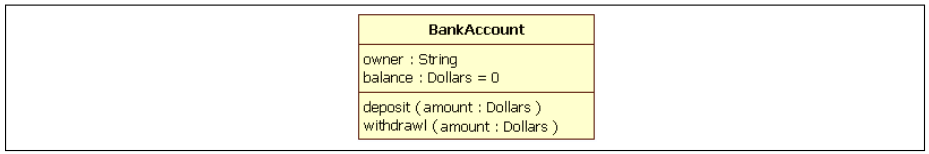


Diagramme de classe (réduit à une classe)

Relation entre classes

- Les classes peuvent être mise en relation.
- UML propose les relations suivantes :
- Association : un lien sémantique entre deux classes.
- Agrégation/composition : une relation d'appartenance.
- Généralisation/spécialisation : une relation d'abstraction.
- Instanciation : une relation d'affectation de paramètres.
- Réalisation : une relation de conformité entre une interface et une implémentation.

Relation d'association

- Il s'agit de la notion mathématique de relation.
- Une relation a une arité à gauche et à droite.
- Chaque objet impliqué a un rôle dans la relation.
- Exemples :
  - Un scénario est joué par un joueur dans un partie.
  - Une action est appliquable sur plusieurs objets d'une scène.
  - Des objets sont nécessaires pour autoriser une action.

Exemple de diagramme

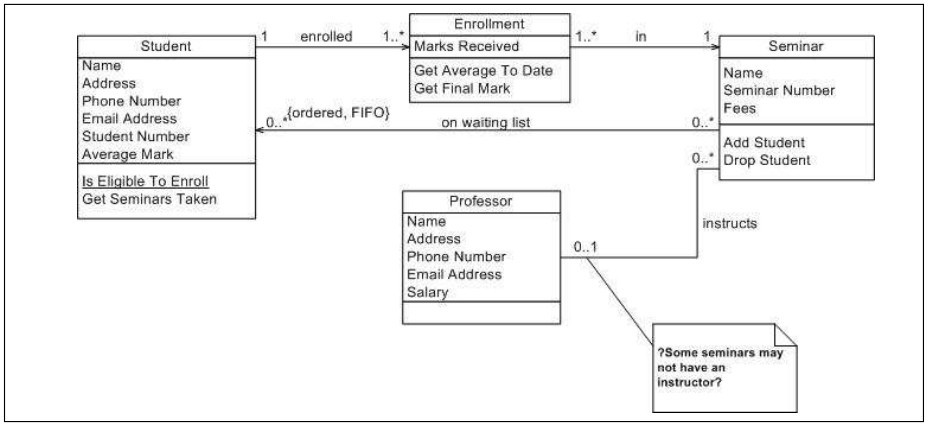


Diagramme de classe (avec associations)

- Un association est formée par :
  - un nom ;
  - des multiplicités à gauche et à droite ;
  - des rôles affectés à chaque objet.

Syntaxe des multiplicités

- Un entier n : n objets interviennent dans la relation.
- L'étoile $*$: plusieurs objets interviennent.
- Le segment $n \ldots *$: au moins n objets interviennent.
- Le segment $n \ldots m$: au moins n et au plus m objets interviennent.

Agrégation/Composition

- L'agrégation est une relation d'appartenance.
- Exemples :
  - Les pièces d'un échiquier lui appartiennent.
  - Les joueurs d'une partie appartiennent à la partie.
- La composition est une relation d'agrégation qui établit une relation de vie ou de mort d'un objet sur un autre.
- Exemples :
  - Si l'échiquier est détruit alors ses pièces aussi.
  - Si une partie est terminée, les joueurs peuvent en jouer une autre. (Ils survivent à la partie.)

$\Rightarrow$ Cette dernière relation est assez subtile et souvent tributaire de certains choix d'implémentation. Il est préférable de ne pas l'utiliser (sauf en C++ car la gestion explicite de la mémoire nécessite une réflexion précise sur la notion de durée de vie qu'il faut considérer dès la phase de spécification).

Exemple de diagramme

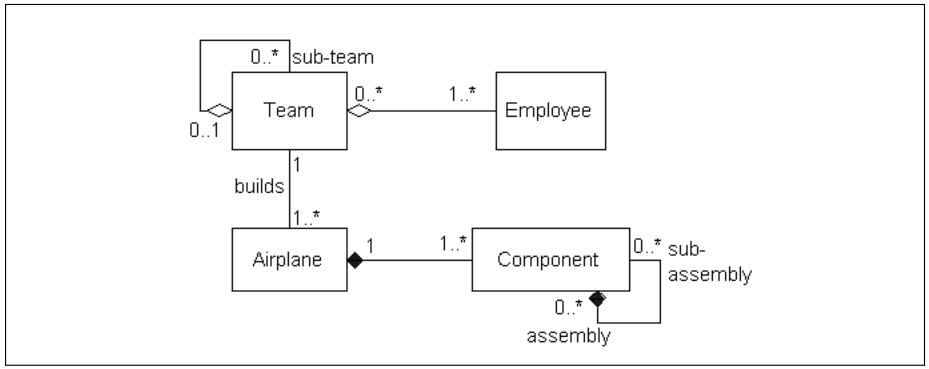


Diagramme de classe (avec compositions et agrégations)

- Le losange vide signifie « est agrégé à ».
- Le losange plein signifie « est composé de ».

Objectifs de la généralisation/spécialisation

- Spécialisation :
  - Ajout d'une fonctionnalité.
  - Focalisation sur un aspect spécifique à une classe.
- Généralisation :
  - Factorisation de critères communs.
  - Abstraction des détails.

$\Rightarrow$ Analogie avec la relation d'inclusion ensembliste.

- Une classe est abstraite si elle n'est jamais vouée à être instanciée.

$\Rightarrow$ Cette caractéristique capture la notion de concept.

Exemple de diagramme

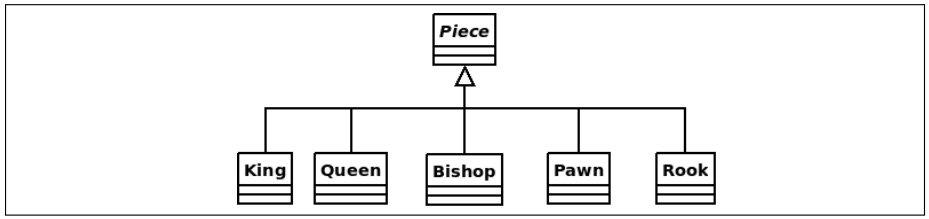


Diagramme de classe (avec relation de généralisation)

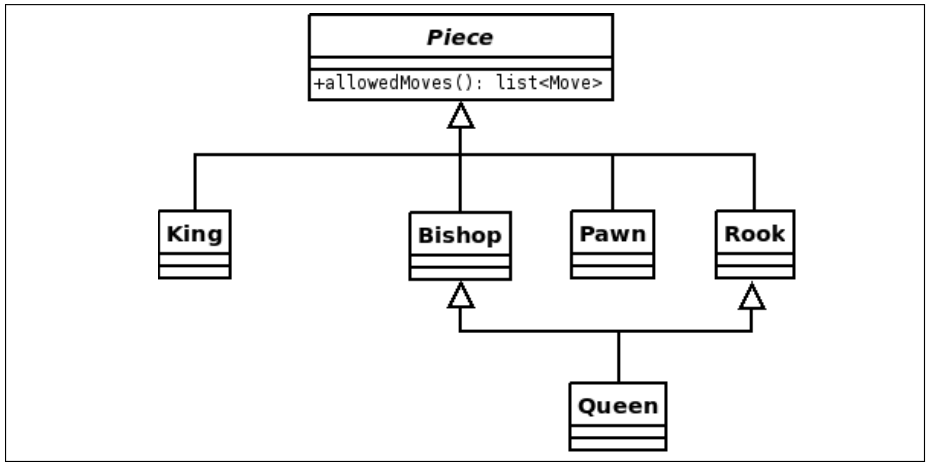
- « Piece » est une super-classe de « Queen », elle généralise cette dernière.
- « Queen » est une sous-classe de « Piece », elle spécialise cette derniere.

⇒ On exprime ici une relation d'abstraction entre composants.

Annotations de la relation de généralisation

- On peut annoter la relation de généralisation par :
- « incomplete » : on pourra rajouter une nouvelle sous-classe dans le futur.
- « complete » : on ne peut plus rajouter une nouvelle sous-classe.
- « disjoint » : les sous-classes ne pourront pas être les parents d'une future sous-classes.
- « overlap » : les sous-classes pourront être utilisées comme super-classes d'une même sous-classe dans le futur.

Exemple de diagramme



- La relation suivante n'est pas correcte puisqu'une règle n'est pas un cas particulier de tour et de fou.
- Il **ne faut pas confondre** factorisation de code et généralisation.

Exemple de diagramme

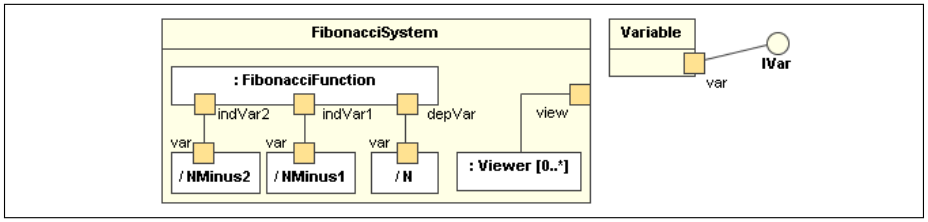


Diagramme composite.

- Les diagrammes composites servent à donner une vision abstraite de l'état interne d'un objet.
- On brise ici le principe d'encapsulation : à n'utiliser qu'en cas de stricte nécessité.

Exemple de diagramme

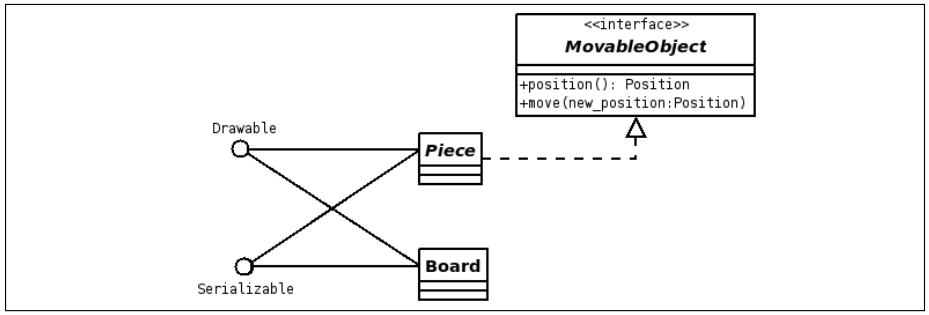


Diagramme de réalisation.

- Un diagramme de réalisation illustre la compatibilité entre un objet et une interface.
- Il y a deux notations possibles pour cela en UML :
  - Un lien vers un cercle faisant référence au nom de l'interface.
  - Une relation de généralisation en pointillés vers une description précise de l'interface.

Exemple de diagramme

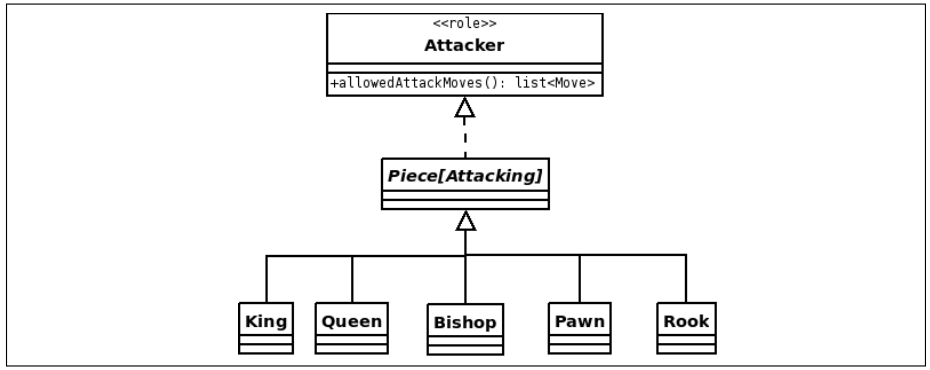
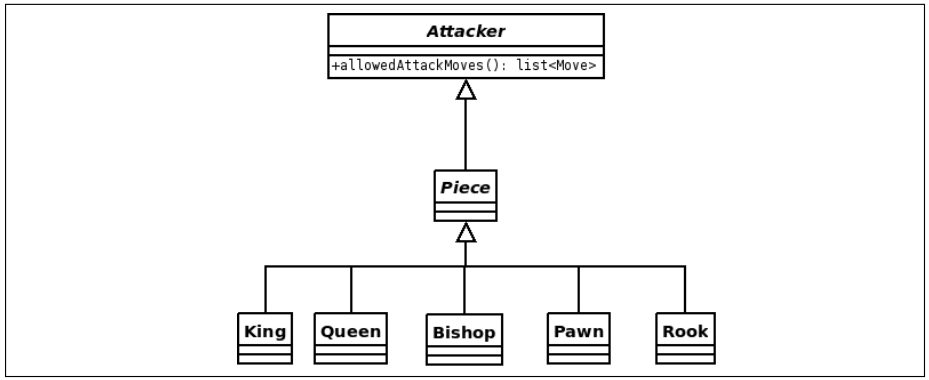


Diagramme de rôle ou de profil.

- Un rôle peut être joué par un objet dans une situation particulière ou en fonction de son état.

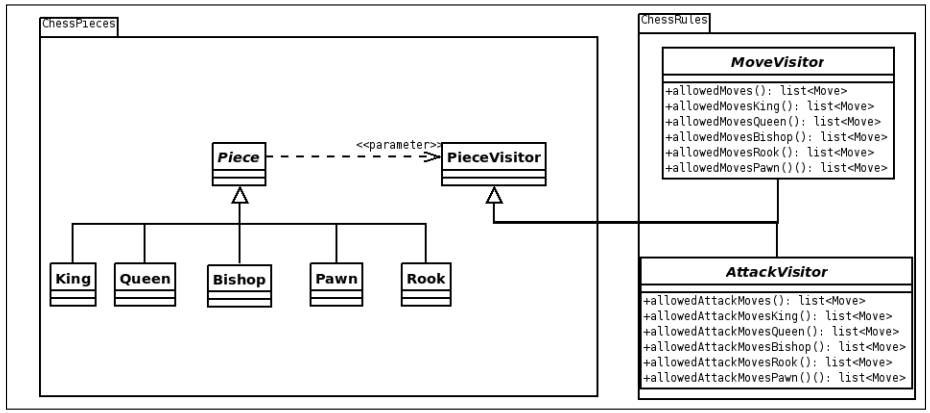
Exemple de diagramme



Point de vue d'implémentation maladroit sur un rôle.

- Il faut distinguer généralisation et prise temporaire d'un rôle.

Exemple de diagramme



Une implémentation correcte d'un rôle.

Synthèse

Résumé

- Nous avons brièvement présenté RUP.

⇒ Nous en étudierons les principes dans le cours de conception orientée objet des systèmes.

- Nous avons survolé UML.

⇒ Ce sera un outil que nous appliquerons et approfondirons par la suite.