Diginamic Formation

« Dynamisez votre carrière digitale ! »

UML – les bases

- Un chirurgien, un ingénieur des Ponts et Chaussées et un informaticien débattent sur la question : « quel est le plus vieux métier du monde ? ». Le chirurgien fait remarquer : « Et bien, il est dit dans la Bible que Dieu a créé Ève en retirant une côte à Adam. Ceci nécessite évidemment une opération chirurgicale, je peux donc revendiquer d'exercer le plus vieux métier du monde.» L'ingénieur des Ponts et Chaussées l'arrête et dit : «Mais bien avant, dans le livre de la Genèse, il est dit que Dieu a créé l'ordre à partir du Ciel et la Terre à partir du Chaos. Ceci est certainement la plus grandiose et la plus ancienne application du génie civil. Donc mon cher chirurgien, vous avez tort : ma profession est la plus ancienne.» Alors l'informaticien se met à rigoler doucement et leur demande : « OK, mais qui croyez vous, a créé le Chaos ?»

- La différence entre la théorie et la pratique, c'est qu'en théorie il n'y a pas de différence.

Diginamic Formation
« Dynamisez votre carrière digitale ! »

Les classiques 1/2

- Loi n° 1 :
  - *« Aucun grand projet informatique n'est jamais mis en place dans les délais, dans les limites du budget, avec le même personnel qu'au départ, et le projet ne fait pas ce qu'il est censé faire non plus... Il est fort improbable que le votre soit le premier! »*
    - *Corollaire :*
      - *Les bénéfices seront inférieurs aux estimations (si on a pensé à en faire),*
      - *le système finalement en place le sera avec retard,*
      - *il coûtera plus cher que prévu, mais ce sera une réussite technique. »*
- Loi n° 2 :
  - *« L'un des avantages de fixer des objectifs vagues à un projet repose sur le fait que vous n'aurez pas de difficulté à en estimer les dépenses correspondantes. »*

- Loi n° 3 :
  - *« L'effort pour redresser le cap croît exponentiellement avec le temps. »*
- Loi n° 4 :
  - *« Les buts, tels que les entend celui qui en décide, seront compris différemment par chacun des autres. »*

Diginamic Formation

« Dynamisez votre carrière digitale ! »

Les classiques 2/2

- Loi n° 5 :
  - *« Plus la complexité technique d'un projet est importante, moins vous aurez besoin d'un technicien pour le diriger. »*
    - *Corollaire :*
      - *Trouvez le meilleur manager possible, il trouvera le technicien,*
      - *le contraire n'est jamais vrai!... »*
- Loi n° 6 :
  - *« S'il y a un risque que quelque chose marche mal, ça marchera mal. Quand les choses semblent aller bien, c'est que quelque chose ira mal prochainement. »*
    - *Corollaire :*
      - *Si les choses semblent aller mieux, c'est que vous avez oublié quelque chose! »*

- Loi n° 7 :
  - *« Les projets progressent en général rapidement jusqu'à 90% de leur achèvement, puis ils restent inachevés à 90% pour toujours. »*
- Loi n° 8 :
  - *« Si on laisse le contenu d'un projet changer librement, le taux de changement dépassera le taux d'avancement très rapidement. »*
- Loi n° 9 :
  - « Quelque soit la liste de 10 Lois, on n'arrive jamais à en trouver la dixième. »

Diginamic Formation
« Dynamisez votre carrière digitale ! »

Les 6 meilleurs pratiques

- Développer le logiciel de façons itérative;
- Gérer les exigences;
- Utiliser des architectures à base de composants;
- Modéliser graphiquement le logiciel;
- Vérifier la qualité du logiciel;
- Contrôler les changements apportés au logiciel.

Diginamic Formation
« Dynamisez votre carrière digitale ! »

Les origines d'UML : The Three Amigos

- Différentes méthodes de modélisation basée sur les objets ont été développées durant les années 80. Il y en avait plus de 50 au début des années 90.

  Chacune avait des avantages et des inconvénients,

  aucune ne faisait l'unanimité. Parmi les trois plus populaires, on pouvait trouver :

la méthode de Grady Booch, appelée parfois OOD

OMT de James Rumbaugh

OOSE et Objectory de Ivar Jacobson.

Diginamic Formation

« Dynamisez votre carrière digitale ! »

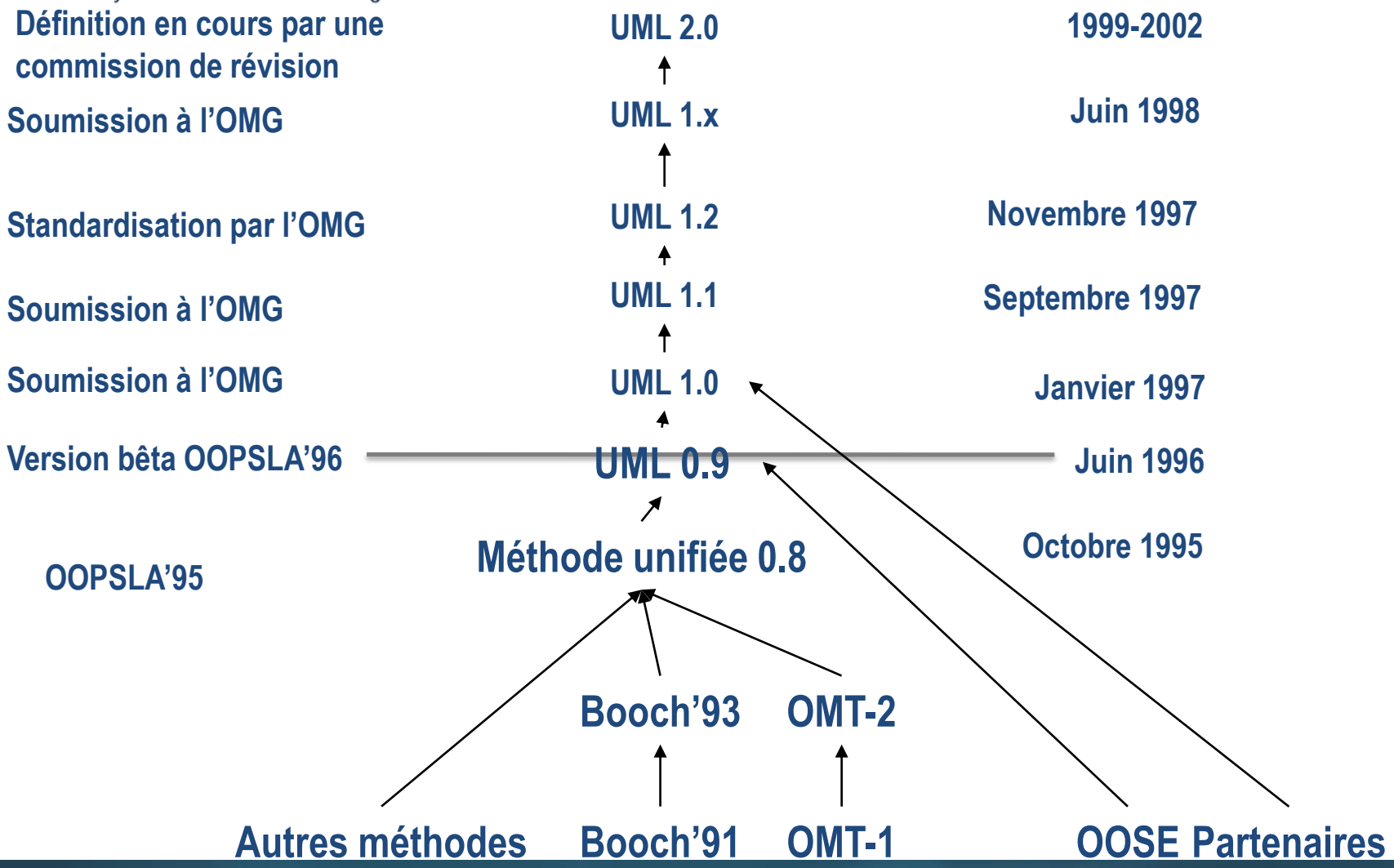
La guerre des méthodes n'aura pas lieu

- On aurait pu s'attendre à une guerre des méthodes (mais non).
  - Booch était chez Rational Software (les développeurs de Rose). En 1994, Rumbaugh a rejoint Rational également.
  - Booch et Rumbaugh ont annoncé leur intention de fusionner leurs méthodes.
    - Les industriels n'étaient pas trop contents de voir une entreprise prendre le marché de cette manière (« à la Microsoft !»).
- Mais toute résistance était inutile : en 1995, Rational achète l'entreprise de Jacobson, Objectory.
  - Un premier brouillon de la méthode unifiée est soumis à approbation de l'OMG (Object Management Group).
- Plus tard, l'objectif a été réajusté pour développer un langage de modélisation plutôt qu'une méthode, ce qui nous a apporté UML.

Diginamic Formation

« Dynamisez votre carrière digitale ! »

Un peu d'histoire



Diginamic Formation
« Dynamisez votre carrière digitale ! »

Présentation générale d'UML 1/2

UML signifie Unified Modeling Language.

- Unified : historique des méthodes de conception (voir précédant slide)
- Modeling (analyse et conception) : Encore une fois, UML n'offre pas une méthodologie pour l'analyse et la conception, mais un langage.
  - Analyse :
    - répond à la question « que faut-il faire ? »,
    - a pour but de se doter d'une vision claire et rigoureuse du problème posé et du système à réaliser.
    - Son résultat est un modèle représentant le problème et les principaux éléments du système à réaliser
    - Elle regroupe les phases d'expressions des besoins et de spécification des méthodologies classiques.
    - Il est important d'insister sur les caractéristiques suivantes de l'analyse :
      - elle nécessite de comprendre en détail les problèmes et concepts de l'utilisateur,
      - elle doit être (en principe) indépendante des aspects techniques,
    - Les éléments du système complet peuvent alors être décrits selon trois points :
      - la vue fonctionnelle, qui indique ce que doivent faire les objets du programme,
      - la vue structurelle (ou statique), qui fait apparaître la composition hiérarchique des objets et leur agencement au travers des diagrammes de classes,
      - la vue dynamique, qui prend en compte le déroulement de l'application sous forme d'événements, d'émission de messages, et d'évolution des états.

Diginamic Formation

« Dynamisez votre carrière digitale ! »

La suite 2/2

- Conception : elle est menée à la suite de l'analyse,
  - répond à la question « comment faire ce qu'il faut faire » ?
  - Le processus de conception a pour but
    - de figer les choix techniques (un ou plusieurs langages de programmation, bibliothèques, etc.).
    - Il fixe les performances (si elles ne font pas déjà partie du cahier des charges, par exemple quand le contexte opérationnel l'impose),
    - définit la stratégie de programmation :
      - choix des structures de données (et de leur persistance),
      - choix des algorithmes à utiliser.
  - Le résultat de la conception est une ébauche d'implémentation, décrite par un modèle précis du système à réaliser.

- Language : méthodologie ou langage de modélisation
  - UML n'est pas une méthode, mais un langage.
    - Il peut donc être utilisé sans remettre en cause les procédés habituels l'entreprise.
    - UML facilite la communication
      - entre clients et concepteurs,
      - entre les équipes de concepteurs.

Diginamic Formation
« Dynamisez votre carrière digitale ! »

Vue d'ensemble d'UML

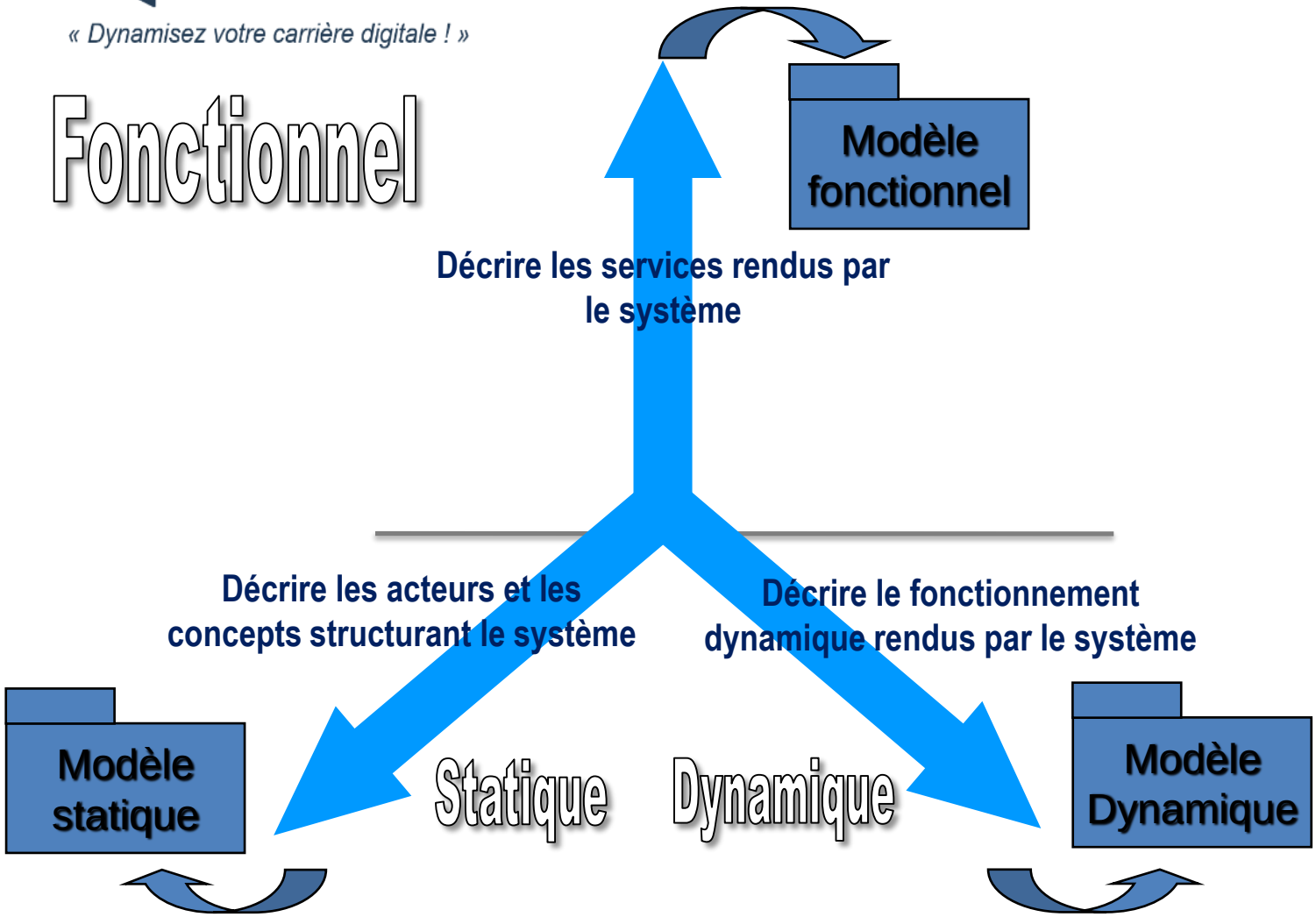
- *UML est un langage conçu pour:*
  - *visualiser*
  - *spécifier*
  - *construire*
  - *Documenter*

- *Un processus correctement défini aide à choisir:*
  - *les artefacts à produire*
  - *les procédures à développer*
  - *les intervenants chargés de leur création et de leur gestion*
  - *la manière d'employer ces artefacts pour évaluer et diriger le projet dans son ensemble*

Diginamic Formation

« Dynamisez votre carrière digitale ! »

Les 3 axes de modélisations



Diginamic Formation

« Dynamisez votre carrière digitale ! »

Différentes vues et diagrammes

- Toutes les vues proposées par UML sont complémentaires les unes des autres,

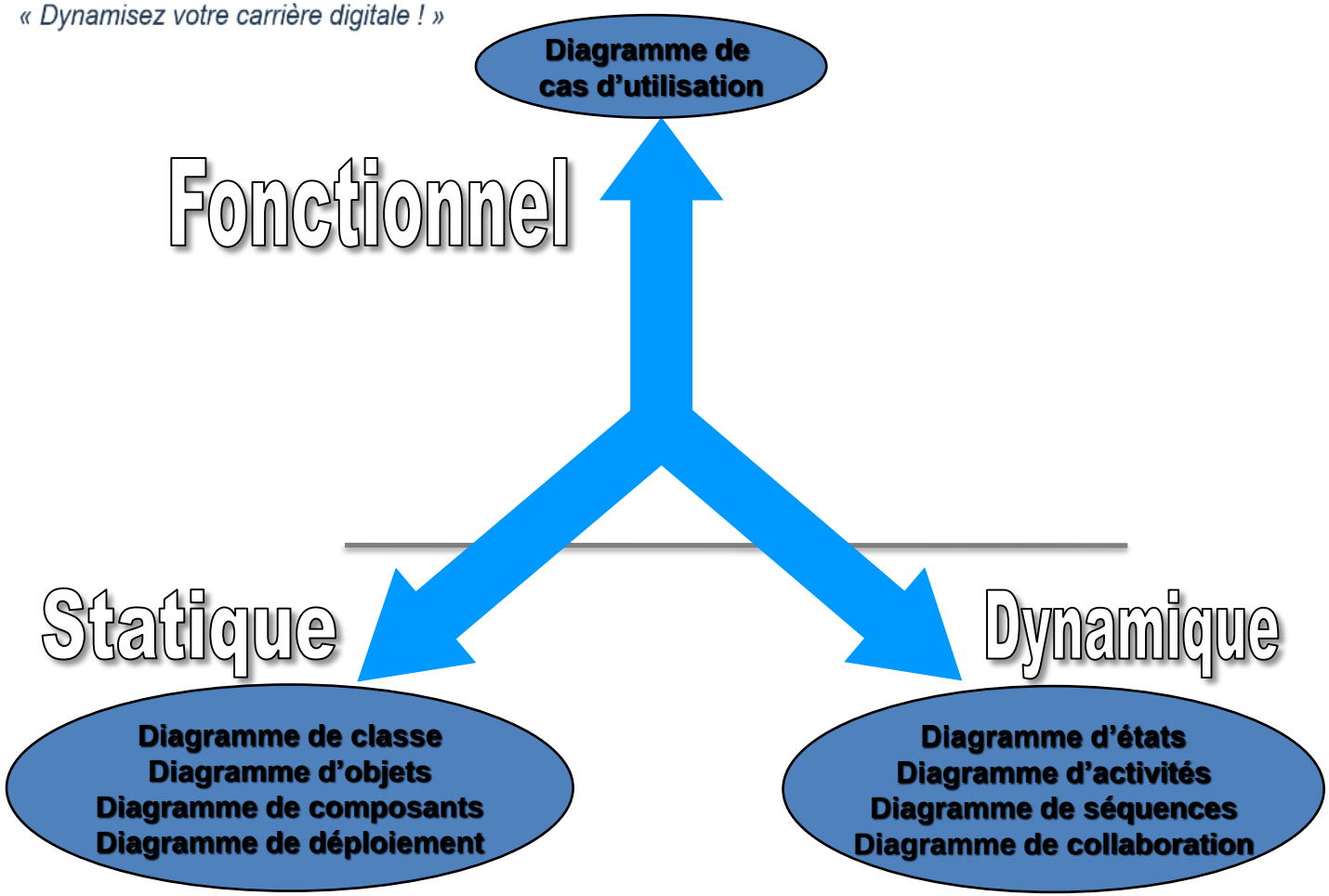
---

- On peut organiser une présentation d'UML autour d'un découpage
  - en vues,
  - ou en différents diagrammes, selon qu'on sépare plutôt
    - les aspects fonctionnels
    - des aspects architecturaux,
    - ou les aspects statiques
    - des aspects dynamiques.

Diginamic Formation

« Dynamisez votre carrière digitale ! »

Les diagrammes UML par axe

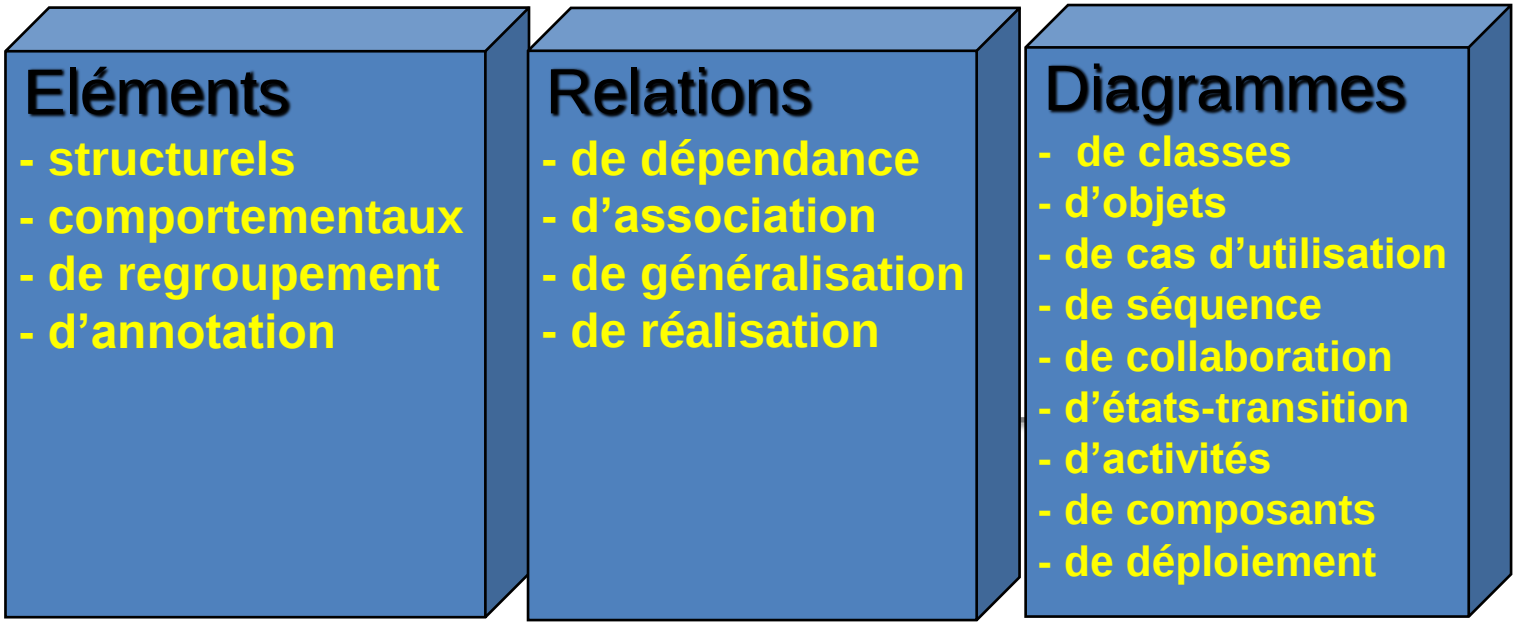


Diginamic Formation

« Dynamisez votre carrière digitale ! »

Les briques de base d'UML

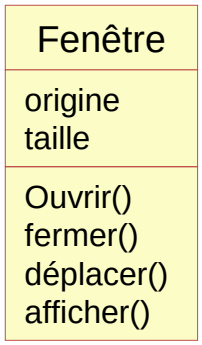
Modèles conceptuel d'UML



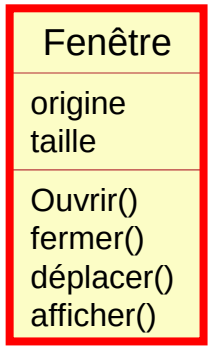
Eléments structurels

Conceptuel (représentent des concepts intellectuels)

Les classes



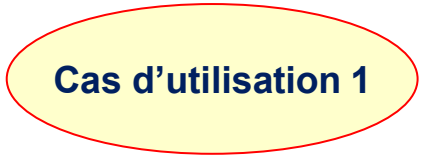
Les classes actives



Les interfaces

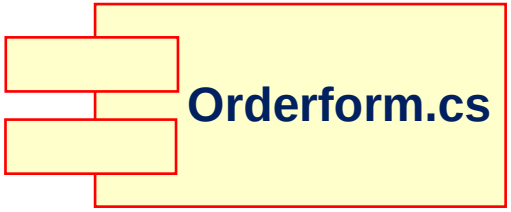
Les collaborations

Les cas d'utilisations

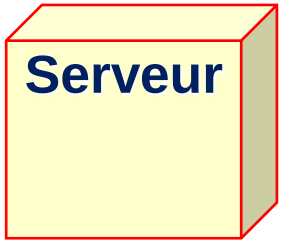


Physiques (réalisations concrètes)

Composant



Noeud



Diginamic Formation

« Dynamisez votre carrière digitale ! »

Les éléments (la suite)

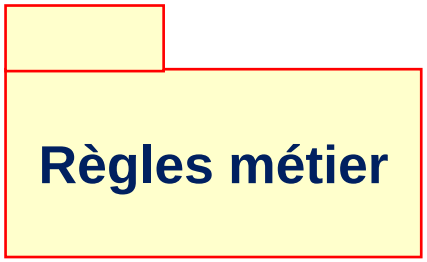
Les éléments comportementaux

Les interactions

Les automate à états finis

Les éléments de regroupement, d'annotation

Les paquetages



Les annotations

Diginamic Formation

Dépendances

Les relations

Cible

Relation sémantique entre deux éléments selon laquelle un changement apporté à l'un peut affecter la sémantique de l'autre

Associations

Relation structure qui décrit un ensemble de liens, un liens constituant une relation entre 2 objets.

Généralisations

Parent

Relation de spécialisation/généralisation selon laquelle les attributs de l'élément spécialisé (l'enfant) peuvent substituer aux attributs de l'élément généralisé (le parent)

Réalisations

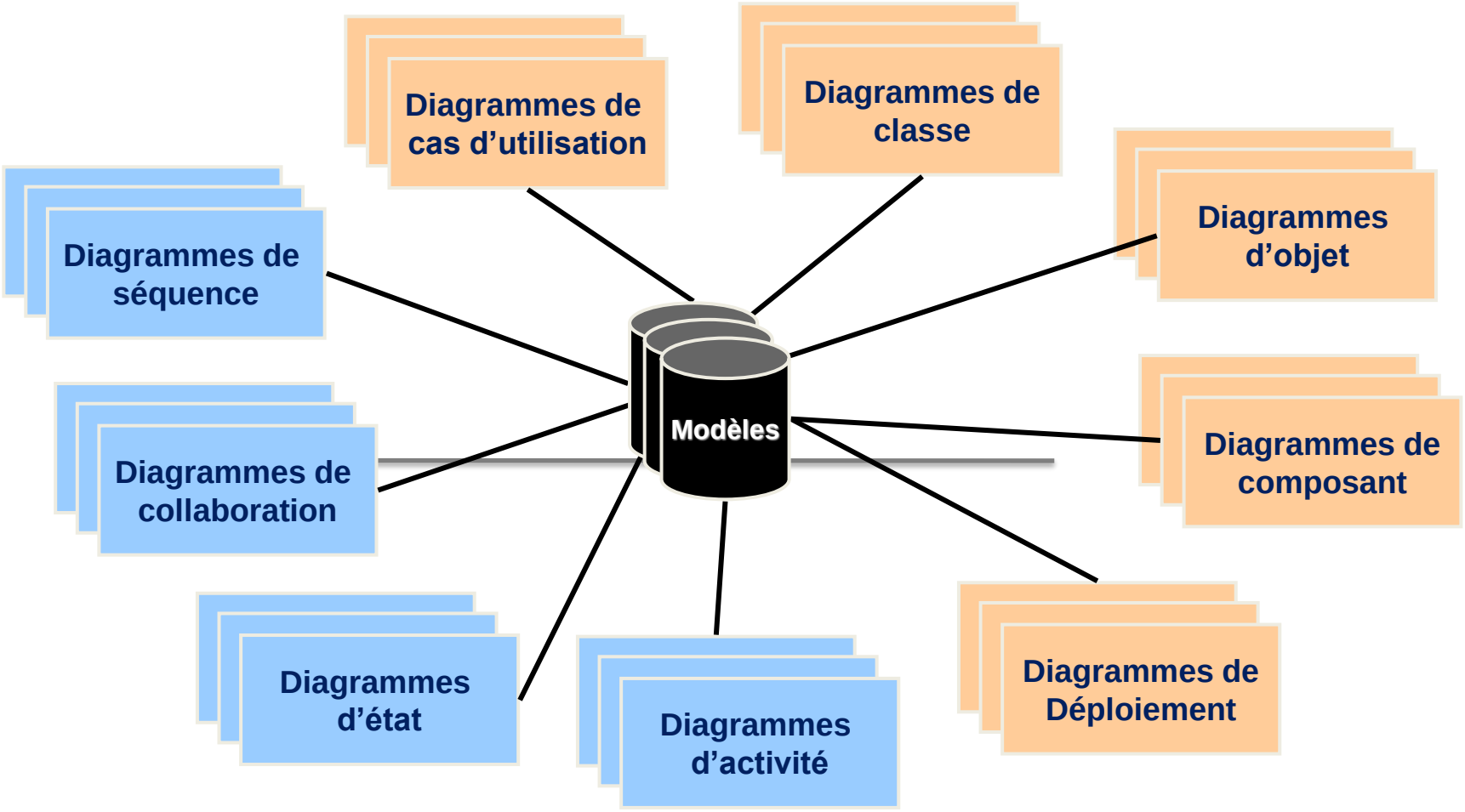
Contrat

Relation sémantique entre classificateurs, selon laquelle un classificateur spécifie un contrat dont l'exécution est garantie par un autre classificateur.

Diginamic Formation

« Dynamisez votre carrière digitale ! »

Un problème - Un diagramme



Diginamic Formation
« Dynamisez votre carrière digitale ! »

Les règles d'UML

- UML possède des règles sémantiques pour :
  - Les noms : la manière de désigner les éléments, les relations et les diagrammes;
  - le contexte : l'environnement qui donne une signification bien précise à un nom;
  - la visibilité : la manière dont ces noms peuvent être vus et utilisés par d'autres;
  - l'intégrité : la manière dont les objets établissent des relations correctes et cohérentes entre eux;
  - l'exécution les conséquences de l'exécution ou de la simulation d'un modèle dynamique.

- Les modèles construits au cours du développement d'un système à forte composante logicielle ont tendance évoluer. Il est courant que l'équipe de développement construise parfois des modèles:
  - partiels afin de simplifier la représentation graphique, certains éléments sont cachés;
  - incomplets il manque certains éléments;
  - Incohérents l'intégrité du modèle n'est pas garantie.

- Ces modèles imparfaits sont inévitables car les détails d'un système se précisent et se combinent tout au long du cycle de développement logiciel.
- Les règles d'UML encouragent, mais n'oblige pas à répondre aux principales questions d'analyse, de conception et d'implémentation qui permettent à ces modèles d'acquérir peu à peu une forme correcte.

Diginamic Formation

« Dynamisez votre carrière digitale ! »

Modélisation des structures élémentaires

Diginamic Formation
« Dynamisez votre carrière digitale ! »

Classes

- Les classes constituent les briques de base les plus importantes d'un système orienté objet.
- Une classe est la description d'un ensemble d'objets qui partagent les mêmes attributs, les mêmes opérations, les mêmes relations et la même sémantique.
- Une classe implémente une ou plusieurs interfaces.
- Les classes sont utilisées pour capturer le vocabulaire du système que l'on développe.

- Elles peuvent contenir des abstractions du domaine étudié ainsi que des classes pour la réalisation.
- On peut utiliser les classes pour représenter
  - des éléments logiciels,
  - des éléments matériels
  - et même des objets purement conceptuels.
- Les classes bien structurées ont des frontières clairement délimitées et forment une répartition équilibrée des responsabilités au sein du système.

Diginamic Formation
« Dynamisez votre carrière digitale ! »

Allouer des responsabilités aux classes

- Allouer des responsabilités aux classes
  - Une responsabilité est un contrat ou une obligation qu'une classe doit respecter.
  - Chacune des exigences fonctionnelles doivent être attribuées à l'une des classes
    - Toutes les responsabilités d'une classe doivent être reliées entre elles.
    - Si une classe a trop de responsabilités, elle devrait être scindée en plusieurs classes
    - Si une classe a aucune responsabilité, alors celle-ci est probablement inutile
    - Lorsqu'une responsabilité ne peut être attribués à aucune classe, alors c'est qu'une nouvelle classe devrait être introduite
  - La meilleur façon de modéliser une classe est de définir les responsabilités des éléments dans le vocabulaire. Les techniques comme
    - les cartes CRC
    - Les cas d'utilisation sont très utiles.

Diginamic Formation

« Dynamisez votre carrière digitale ! »

Catégories de responsabilités

- Catégories de responsabilités
  - Fixer et obtenir la valeur d'un attribut
  - Créer et initialiser de nouvelle instances
  - Sauvegarder et récupérer de l'information persistante
  - Détruire des instances
  - Ajouter et détruire des liens
  - Copier, convertir, transformer, transmettre, afficher
  - Calculer des résultats numériques
  - Naviguer et rechercher
  - Tout autre tâche…

Diginamic Formation

« Dynamisez votre carrière digitale ! »

Classe : Trucs et astuces

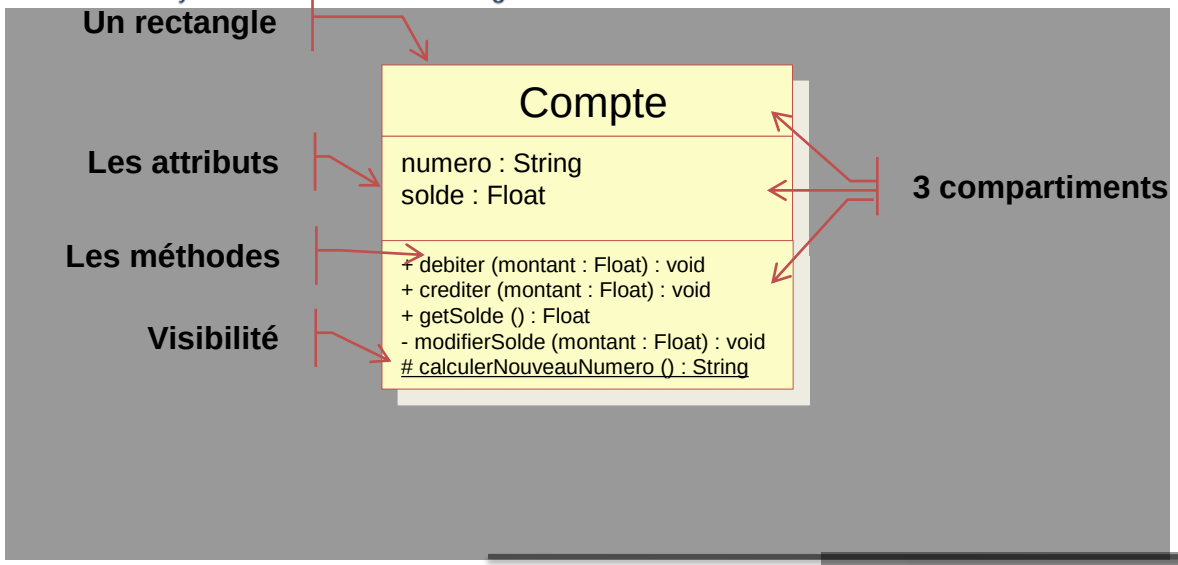
- Une classe bien structurée:
  - fournit une abstraction claire de quelque chose tiré du vocabulaire du domaine du problème ou de celui de la solution;
  - comprend un petit ensemble de responsabilités bien définies et les réalise parfaitement;
  - fournit une séparation nette entre les spécifications de l'abstraction et son implémentation;
  - est compréhensible et simple tout en étant extensible et adaptable.
  - Lorsqu'on représente une /classe avec UML:

  - seules les propriétés de la classe essentielles à la compréhension de l' abstraction dans son contexte sont représentées;
  - de longues listes d'attributs et d'opérations sont organisées en fonction de leurs catégories;
  - les classes en relation sont représentées sur les mêmes diagrammes de classes.

Diginamic Formation

« Dynamisez votre carrière digitale ! »

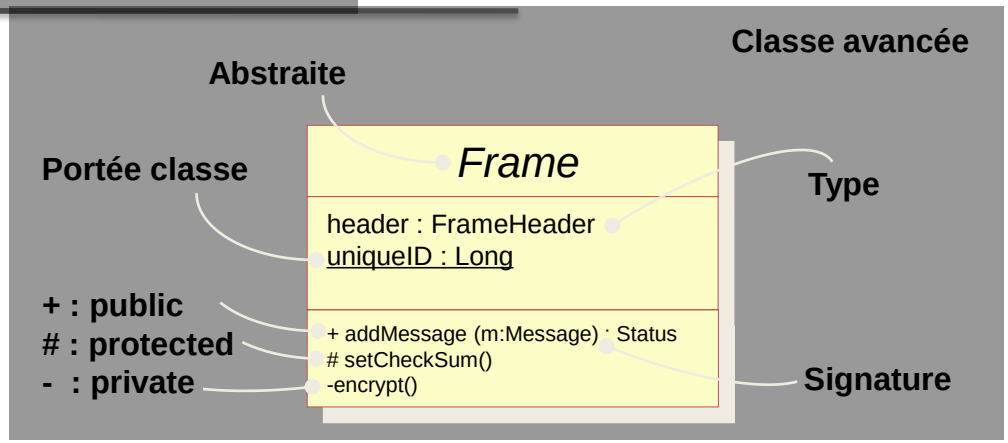
Classe : représentation UML



+ public : tout classificateur extérieur ayant une visibilité sur le classificateur donné peut utiliser cette caractéristique indiquée par le symbole +.

 # protected : seuls les descendants du classificateur peuvent utiliser cette caractéristique indiquée par le symbole #.

- private : le classificateur, et lui seul, peut utiliser cette caractéristique indiquée par le symbole -



Diginamic Formation
« Dynamisez votre carrière digitale ! »

Les relations

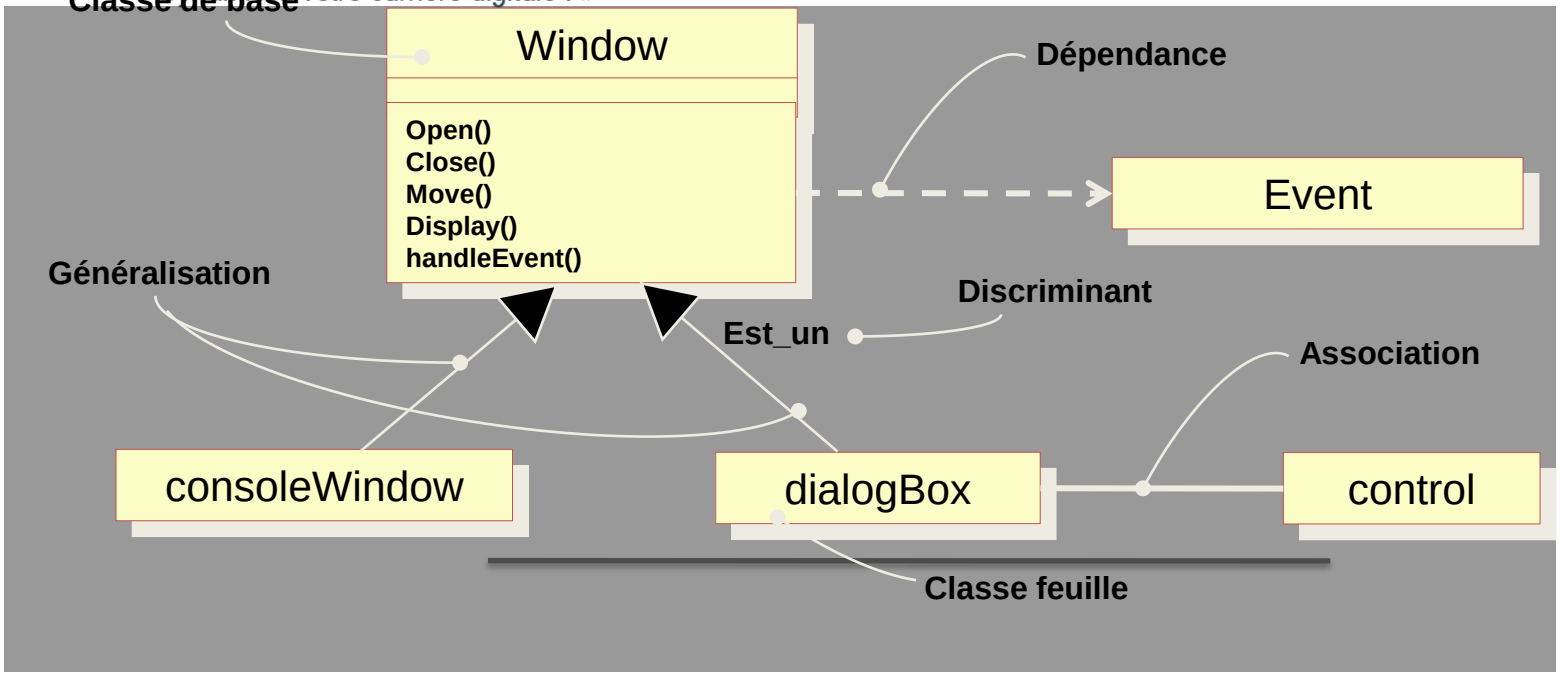
- La construction d'abstractions enseigne que très peu de classes sont isolées mais qu'au contraire la plupart d'entre elles collaborent avec d'autres classes de différentes manière.
- Modéliser un système nécessite
  - identification des éléments qui composent son vocabulaire,
  - mais également la compréhension des relations qui s'établissent entre eux.
- En modélisation orientée objet, trois sortes de relations sont particulièrement importantes:

  - les *dépendances,* c'est-à-dire les relations d'utilisation entre les classes (comme les relations de raffinement, de trace et de liaison),
  - les *généralisations,* qui relient les classes générales à leurs spécialisations,
  - et les *associations,* qui décrivent les relations structurelles entre objets. Toutes ces relations proposent une manière différente de combiner des abstractions.
- Si elle est trop élaborée, l'écheveau des relations rend le modèle incompréhensible.
- À l'inverse, si elle est trop laxiste, une bonne partie de la richesse du système, à savoir la manière dont les éléments collaborent, n'est pas exploitée.

Diginamic Formation

« Dynamisez votre carrière digitale ! »

Représentation



- Une relation est une connexion sémantique entre des éléments.
- Une relation est représentée par une ligne.
- Chaque sorte de relation est représentée par un type de ligne différent.

Diginamic Formation

« Dynamisez votre carrière digitale ! »

Dépendance et généralisation

- Dépendance
  - Relation d'utilisation qui établit qu'un changement de spécification d'un élément (par exemple, la classe Event) peut en affecter un autre qui l'utilise (par exemple, la classe Window), mais que l'inverse n'est pas nécessairement vrai.

- Généralisation
  - relation entre un élément général (appelé super-classe ou mère) et un élément dérivé de celui-ci, mais plus spécifique (désigné par le terme sous-classe ou fille).
  - La généralisation est parfois qualifiée de relation "est_une_sorte_de" :
  - La généralisa!ion implique que des objets de l'enfant puissent être utilisés partout où le parent apparaît, mais pas l'inverse.
  - En d'autres termes, elle indique que l'enfant peut remplacer le parent. L'enfant hérite des propriétés du parent, en particulier de ses attributs et de ses opérations mais, le plus souvent, ceux-ci s'ajoutent à ses propres attributs et à ses propres opérations. Lorsqu'un enfant a une opération dont la signature est la même que celle d'une opération du parent, l'opération de l'enfant prévaut sur celle du parent.
  - Ce phénomène est connu sous le nom de polymorphisme. Comme le

Diginamic Formation

« Dynamisez votre carrière digitale ! »

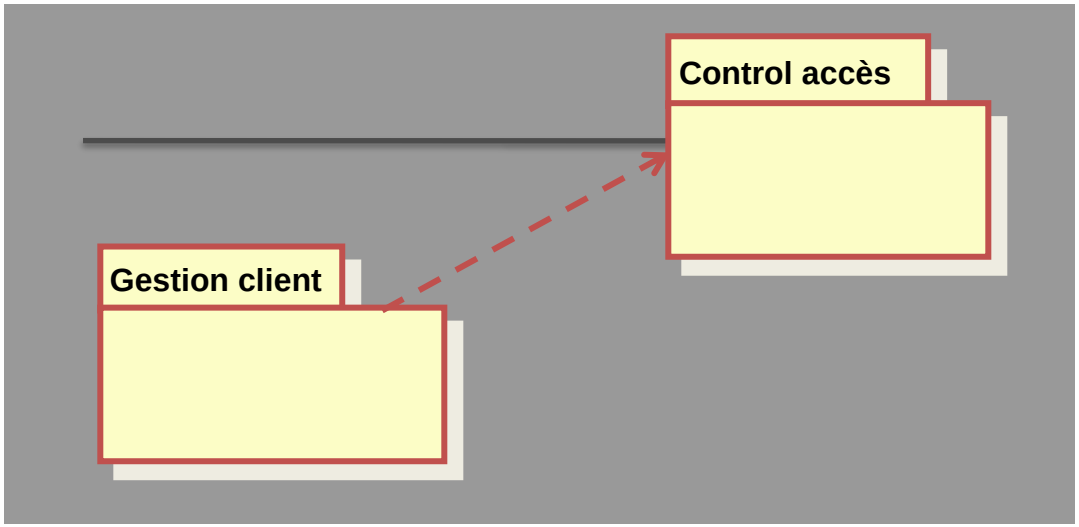
Association

- Une *association* est une relation structurelle qui précise que les objets d'un élément sont reliés aux objets d'un autre élément.
- En reliant deux classes, elle autorise la navigation d'un objet de l'une d'elles à un objet de l'autre, et vice versa.
- Il n'est pas interdit que les deux extrémités de ce genre de relation forment une boucle et se rattachent à la même classe. Cela signifie qu'un objet qui appartient à cette classe peut être connecté à d'autres objets de cette même classe.
- On appelle "association binaire" une association qui relie seulement deux classes entre elles.
- Une association qui relie plus de deux classes, ce qui est moins fréquent, est appelée "association n-aire".
- Une association est représentée par une ligne pleine qui relie une classe à d'autres classes ou à elle-même.
- On utilise les associations pour montrer les relations structurelles.
- 5 décorations s'appliquent aux associations.
    1. Le nom
    2. Le rôle
    3. La multiplicité
    4. La navigabilité
    5. Agrégation

Diginamic Formation
« Dynamisez votre carrière digitale ! »

Analyser et valider les dépendances

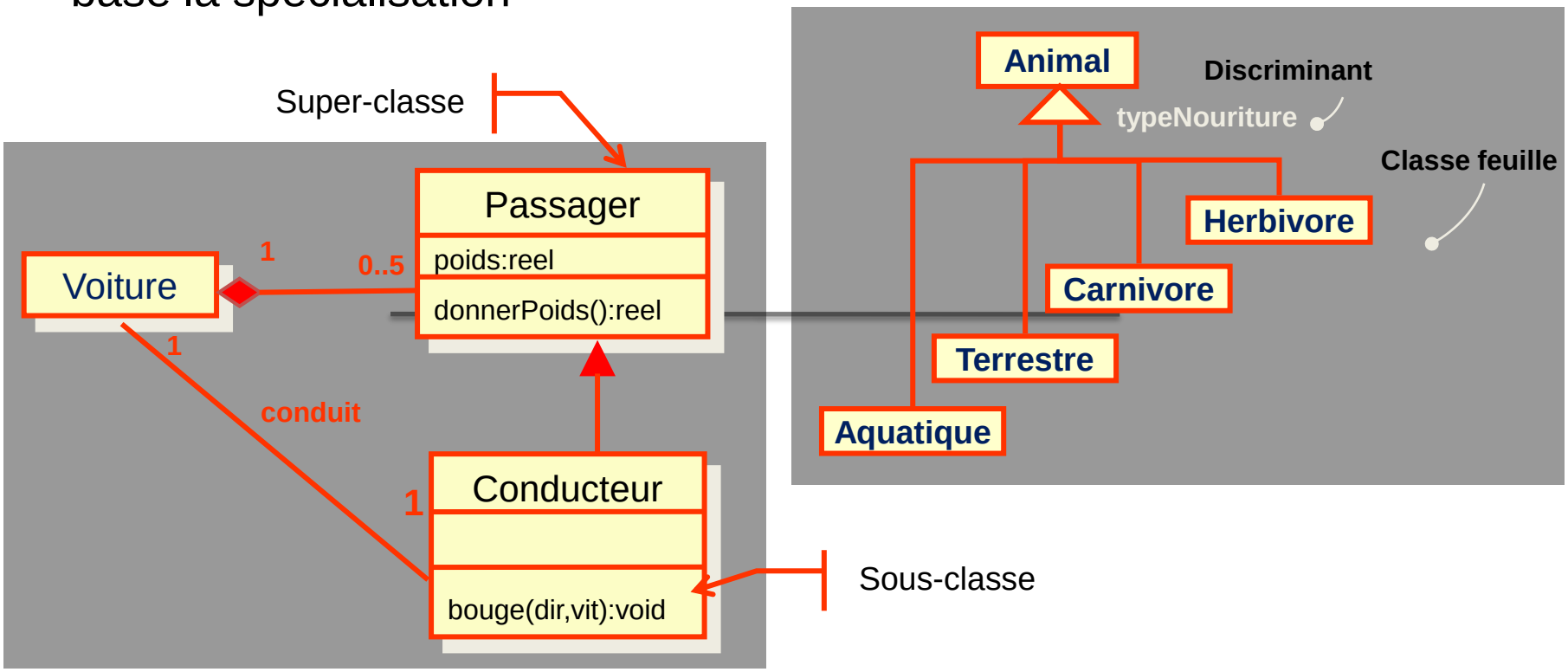
- Relation client - serveur
- Exprime le besoin d'un service
- Sans savoir si le serveur existe et surtout quelle forme il a



Diginamic Formation
« Dynamisez votre carrière digitale ! »

Analyser et valider généralisation

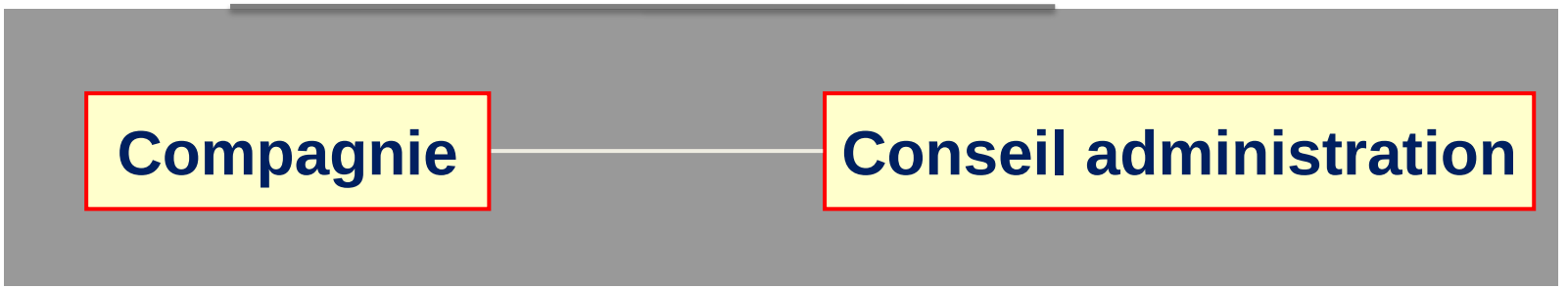
- Une super-classe se spécialise en sous-classes
  - Le *discriminant* est une étiquette décrivant le critère suivant lequel se base la spécialisation



Diginamic Formation
« Dynamisez votre carrière digitale ! »

Analyser et valider les associations (1..1)

- Une à une
  - A chaque compagnie est associé un conseil d'administration
  - Un conseil d'administration gère une seule compagnie
  - Une compagnie doit avoir un conseil d'administration
  - Un conseil d'administration est toujours attaché à une et une seule compagnie



Diginamic Formation
« Dynamisez votre carrière digitale ! »

Analyser et valider les associations (1..*)

- Une à plusieurs
  - Une compagnie a plusieurs employés
  - Un employé ne peut travailler que pour une seule compagnie
    - Qu'en est-il des employés occupant un double emploi!
  - Une compagnie peut n'avoir aucun employé
  - Un employé associé à une compagnie travaille pour cette compagnie

Diginamic Formation
« Dynamisez votre carrière digitale ! »

Analyser et valider les associations (*..*)

- Plusieurs à plusieurs
  - Un(e) secrétaire peut travailler pour plusieurs superviseurs
  - Un superviseur peut avoir plusieurs secrétaires
  - Les secrétaires peuvent travailler en équipes
  - Les superviseurs peuvent avoir recours à un groupe de secrétaires
  - Certains superviseurs peuvent n'avoir aucun(e) secrétaires

---

- Est-il possible qu'un(e) secrétaire puisse se retrouver, ne serait-ce que temporairement, sans superviseur?

Diginamic Formation

« Dynamisez votre carrière digitale ! »

Associations (exemple)

Diginamic Formation
« Dynamisez votre carrière digitale ! »

Quelques trucs

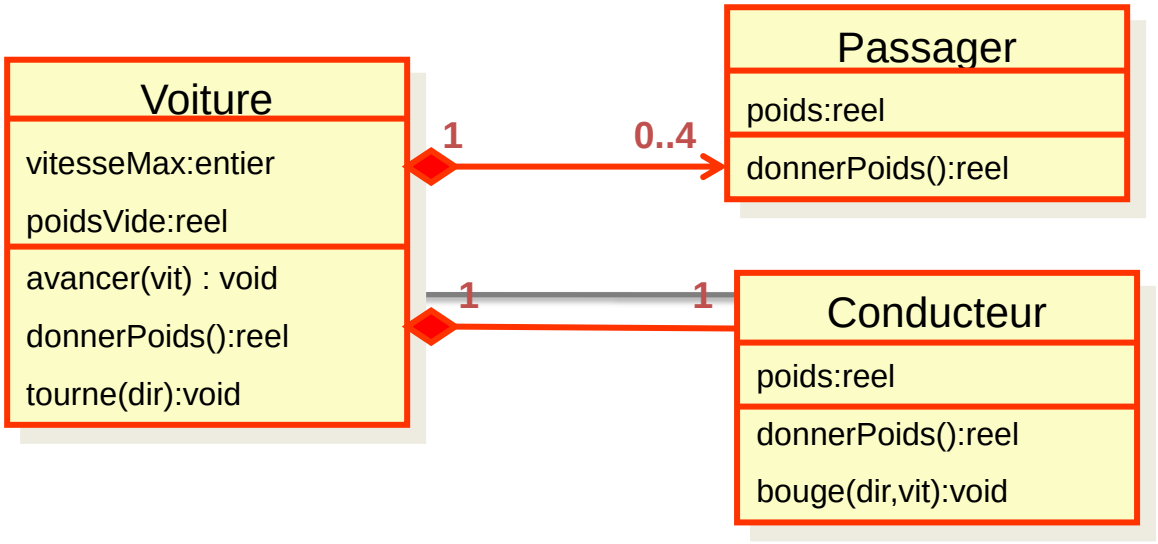
- Une association devrait exister si une classe
  - *possède*
  - *contrôle*
  - *est connecté à*
  - *est relié à*
  - *est une partie de*
  - *est fait de parties de*
  - *est membre de*
  - *a comme membres*

  une autre classe dans le modèle
- Spécifier ensuite la multiplicité à chaque extrémité de l'association
- Étiqueter clairement cette association

Diginamic Formation
« Dynamisez votre carrière digitale ! »

La navigabilité

- Une flèche qui restreint la navigation dans un sens



Diginamic
Formation

Association

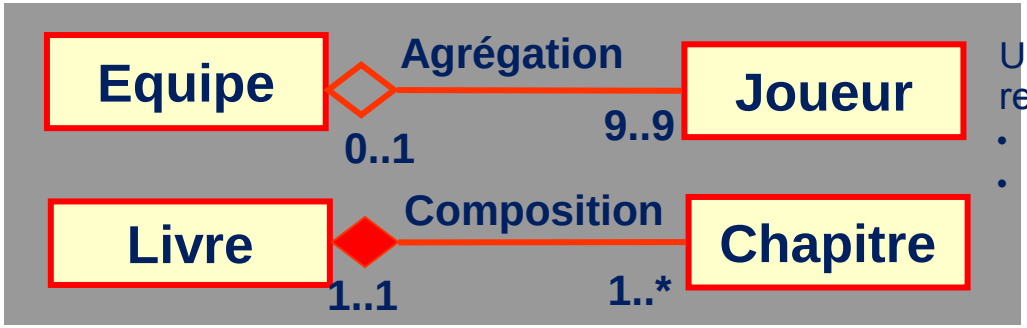
Les objets se connaissent et peuvent travailler ensemble

Agrégation

1. **Protège l'intégrité de la configuration**
2. **Fonctionne comme une entité unique**
3. **Contrôle par un objet – propagation vers le bas**

Composition

Chaque partie ne peut être membre que d'un seul agrégat.



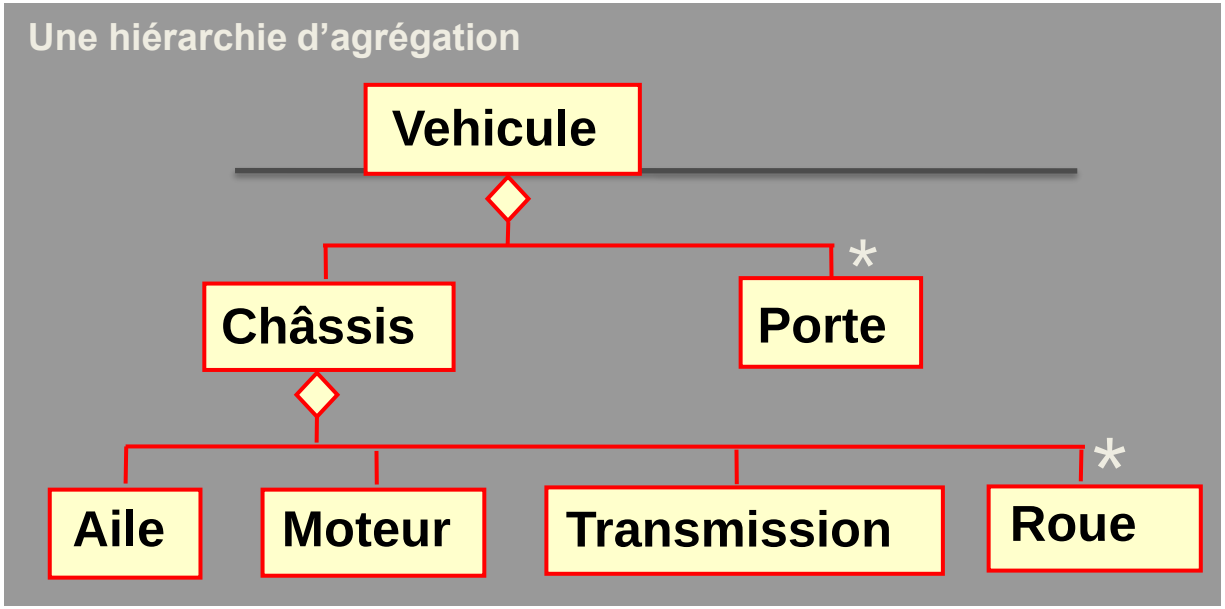
Une agrégation est une forme spéciale représentant une relation 'partie-tout'.

- Le 'tout' est souvent appelé l'ensemble ou l'agrégat
- Le symbole désignant l'agrégation se place du côté de la parti

Diginamic Formation
« Dynamisez votre carrière digitale ! »

Quand faut-il utiliser l'agrégation?

- En général, une association peut être représentée comme une agrégation si:
  - il y a une relation de type *est-une-partie-de*
  - il y a une relation de type *est-composé-de*
- Lorsque quelque chose contrôle l'agrégat, il contrôle aussi ses parties



Diginamic Formation
« Dynamisez votre carrière digitale ! »

Propagation

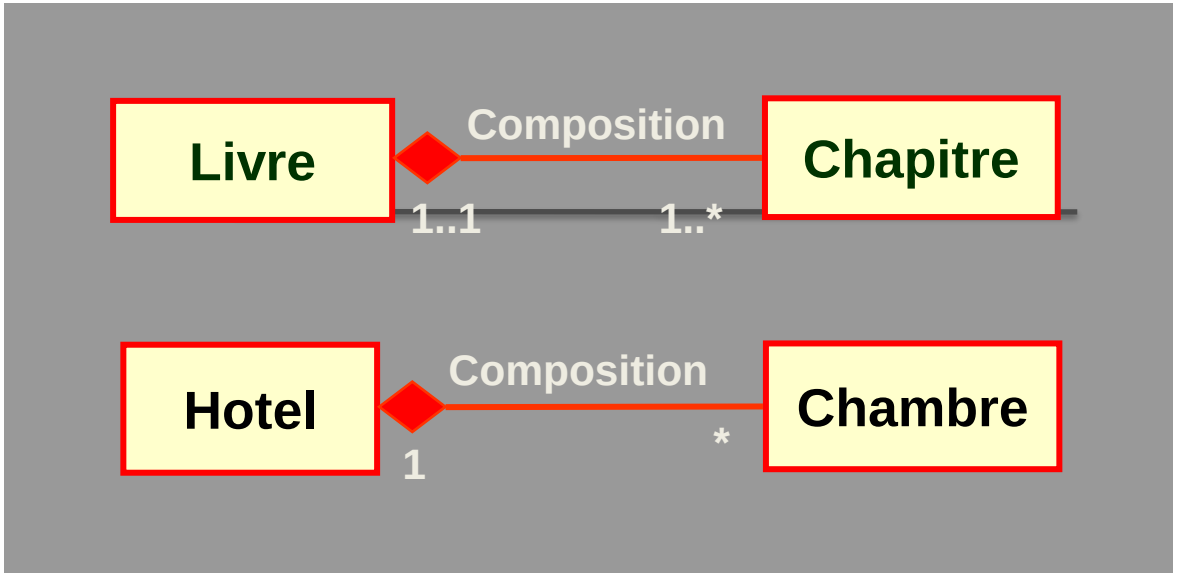
- La propagation est un mécanisme statuant que lorsqu'une opération est effectuée sur le tout elle doit aussi s'appliquer à ses parties
- La propagation permet aussi aux propriétés des parties de se propager vers le tout
- La propagation est à l'agrégation ce que l'héritage est à la généralisation.
  - La différence majeure est que:

  - L'héritage est un mécanisme implicite
  - La propagation doit être programmée

Diginamic Formation
« Dynamisez votre carrière digitale ! »

Composition

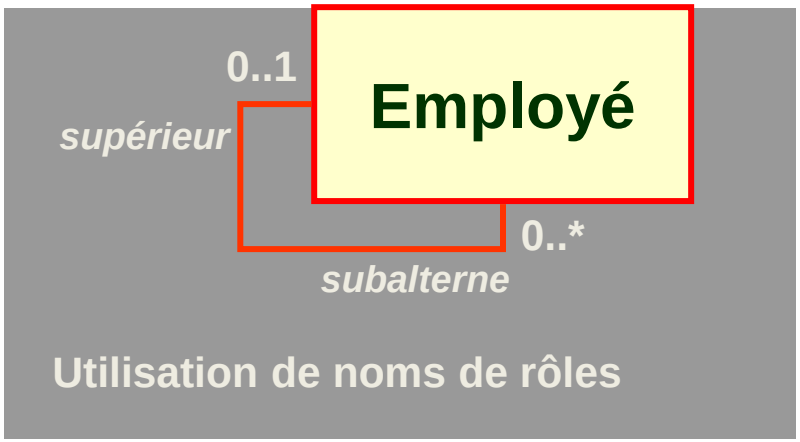
- **Une *composition* est une forme forte d'agrégation**
  - **Si l'agrégat est détruit, alors ses parties le sont aussi.**

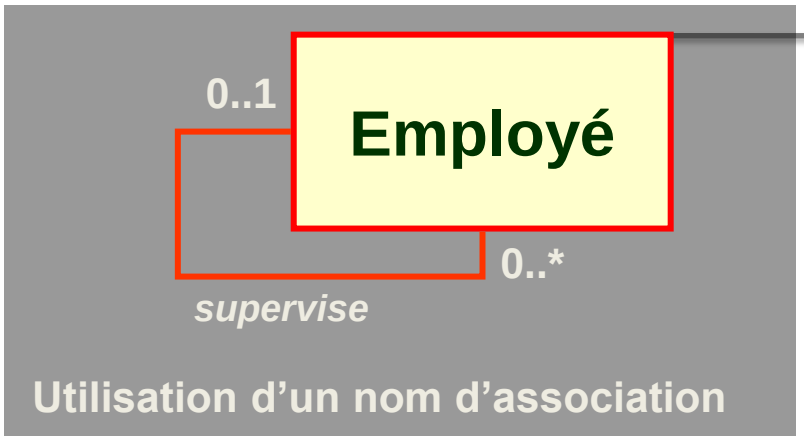


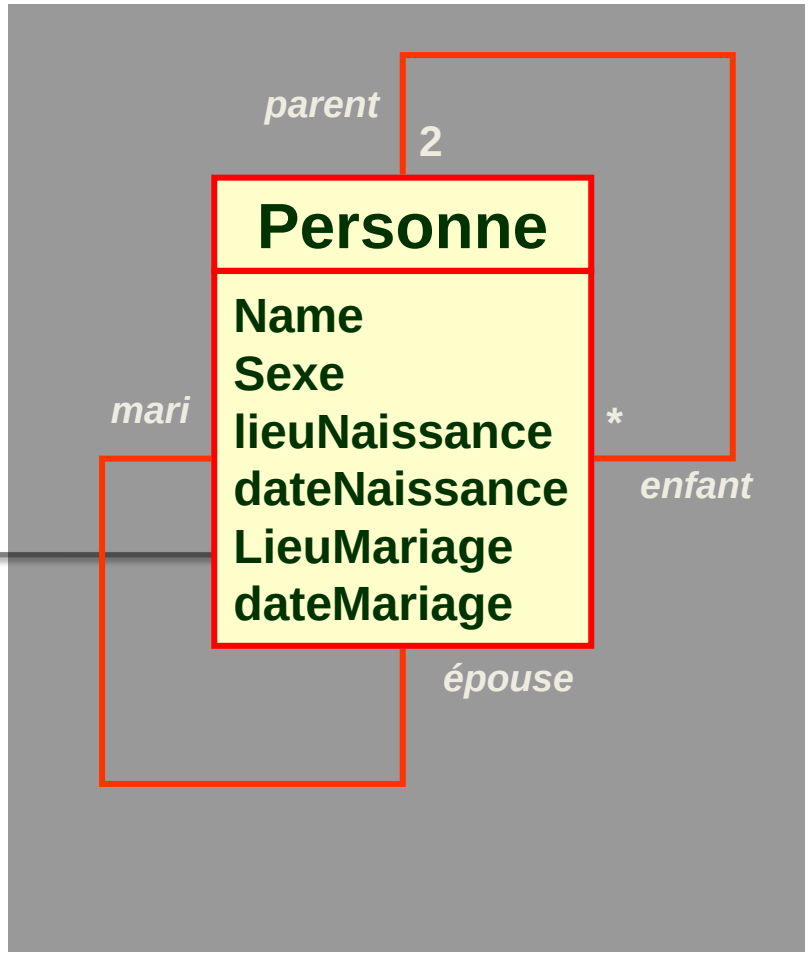
Diginamic Formation

« Dynamisez votre carrière digitale ! »

Associations réflexives

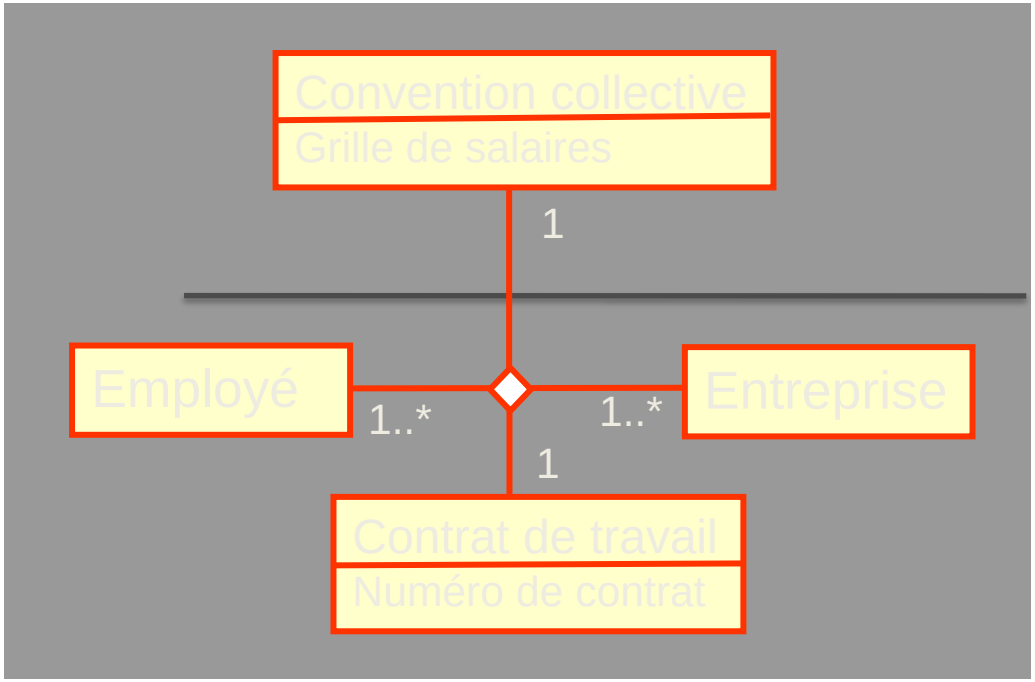






Associations n-aires

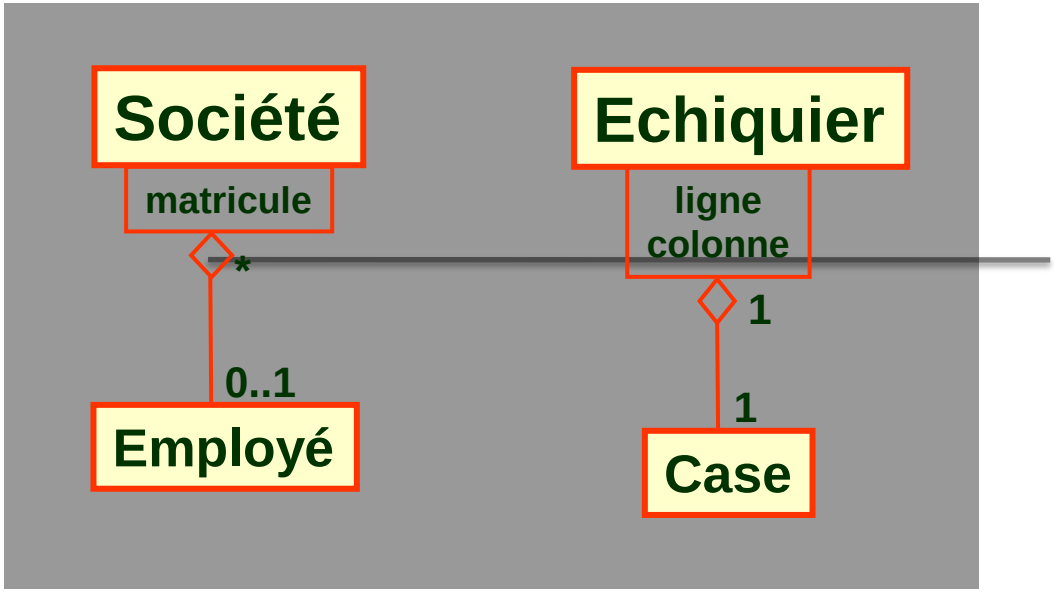
- Permet de représenter des associations multi-membres
- A n'utiliser qu'en analyse
- Pose un problème de représentation pour la conception



Diginamic Formation
« Dynamisez votre carrière digitale ! »

Association qualifiée

- Attribut d'association dont les valeurs partitionnent l'ensemble des objets reliés à un objet à travers une association.
  - Exemple : le matricule permet d'identifier une personne dans une société.
  - Exemple : ligne et colonne permettent d'identifier une case sur un échiquier

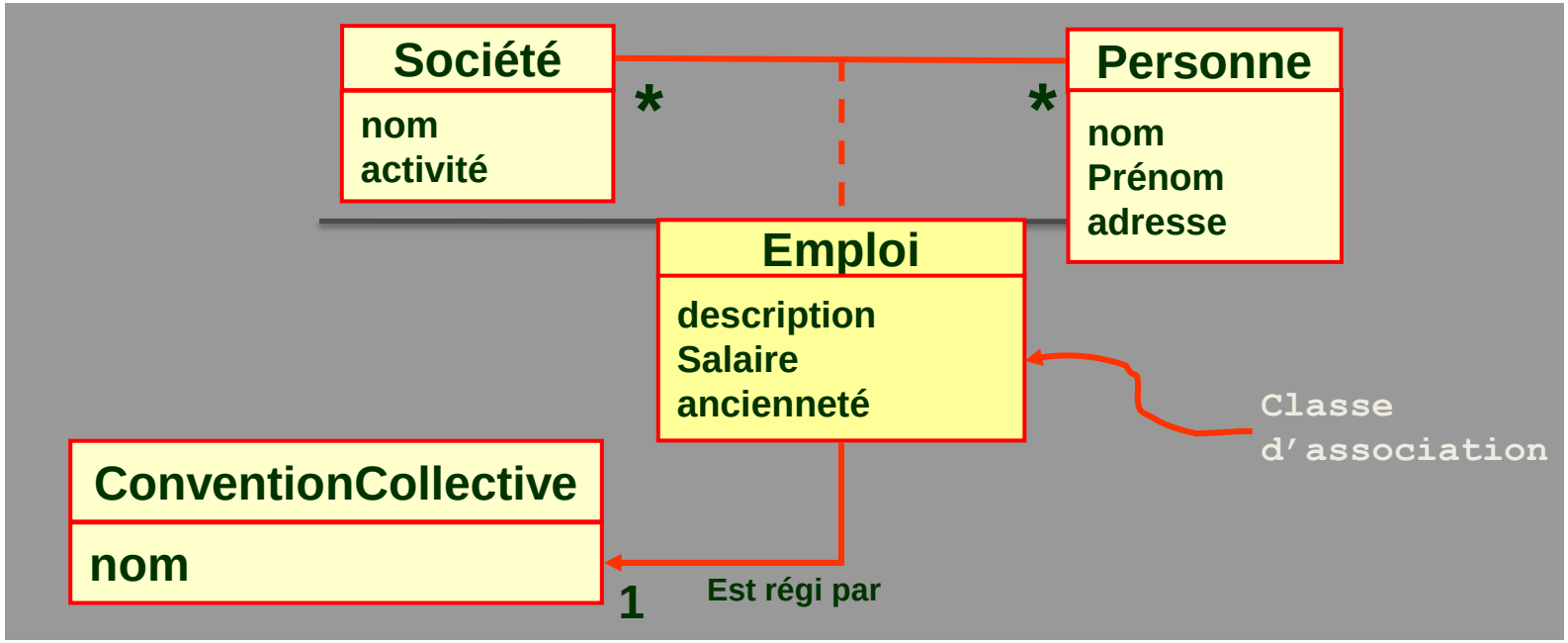


Diginamic Formation

« Dynamisez votre carrière digitale ! »

Classes-Associations

- Une classe-association est une association qui est aussi une classe.
- Les classes-associations sont utilisées lorsque les associations doivent porter des informations
- Il est toujours possible de se passer des classes-associations.

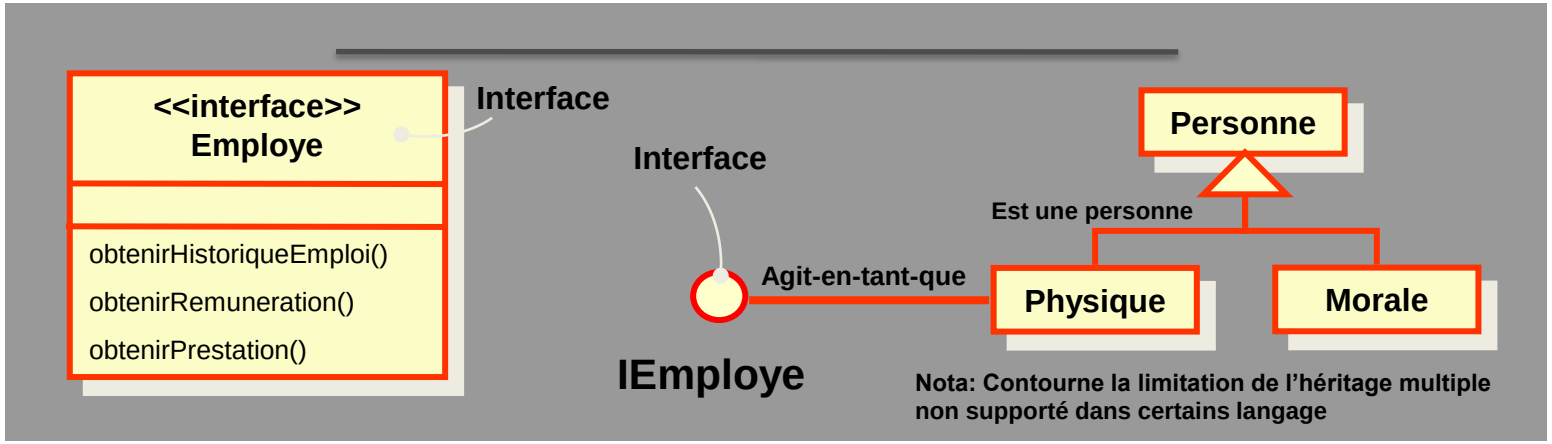


Diginamic Formation

« Dynamisez votre carrière digitale ! »

Les interfaces

- Une interface définit une limite entre la spécification des actions d'une abstraction et l'implémentation de la manière dont cette abstraction les exécute.
- Une interface est un ensemble d'opérations utilisées pour décrire un service d'une classe ou d'un composant.
- On utilise les interfaces pour :
  - visualiser,
  - spécifier, construire
  - documenter les points de soudure d'un système.
- Une interface bien structurée apporte une séparation nette entre les vues externes et internes d'une abstraction.



Diginamic Formation
« Dynamisez votre carrière digitale ! »

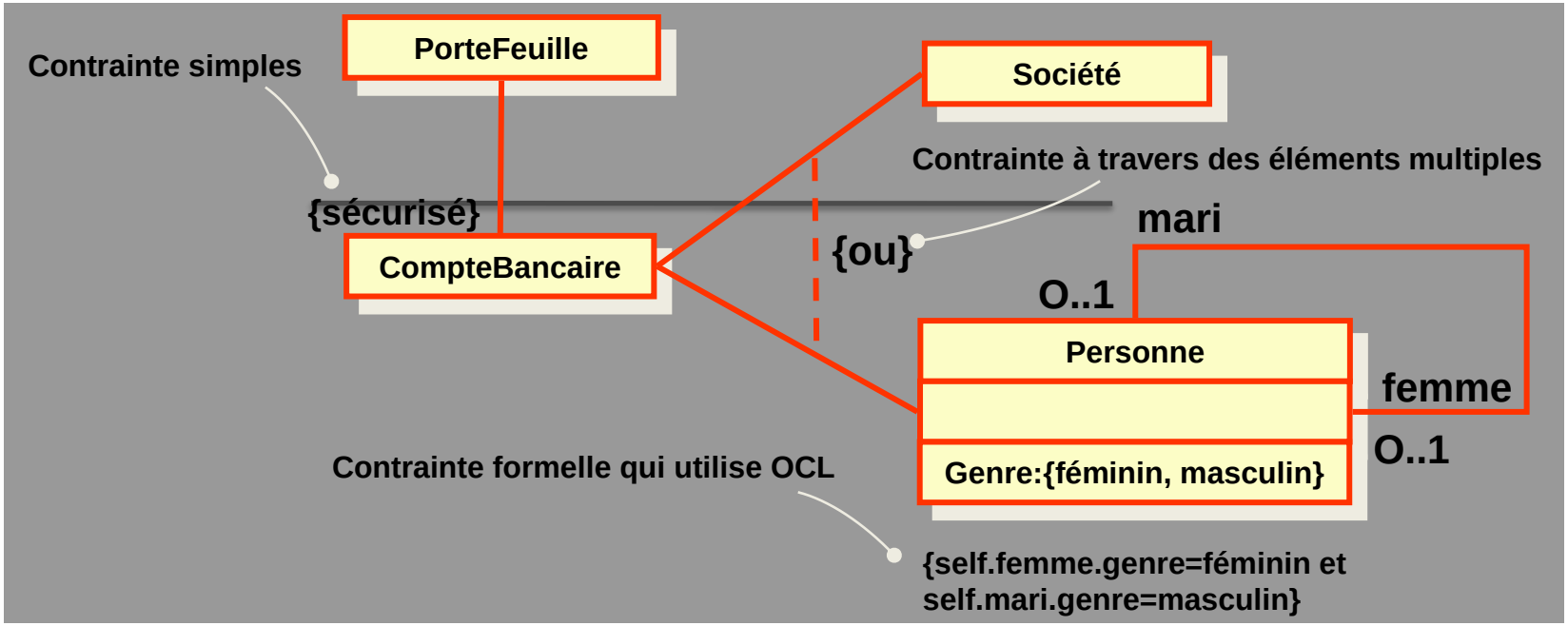
Notes

- Les notes
  - Les notes constituent les décorations indépendantes les plus importantes.
  - Ce sont des symboles graphiques utilisés pour représenter les contraintes et les commentaires rattachés à un élément ou à un ensemble d'éléments.
  - Elles sont utilisées pour ajouter des informations à un modèle, comme des exigences, des observations, des révisions ou des explications.

Diginamic Formation
« Dynamisez votre carrière digitale ! »

Contraintes

- Contrainte
  - En UML, chaque élément possède sa propre sémantique.
  - Les contraintes permettent d'ajouter de nouvelles règles sémantiques ou de changer celles qui existent. Elles précisent des conditions indispensables pour que le modèle soit correctement formé.



Diginamic Formation

« Dynamisez votre carrière digitale ! »

Paquetages

- La visualisation, la spécification, la construction et la documentation de grands systèmes impliquent la manipulation d'un nombre potentiellement élevé
  - de classes,
  - d'interfaces,
  - de composants,
  - de nœuds,
  - de diagrammes et autres éléments.

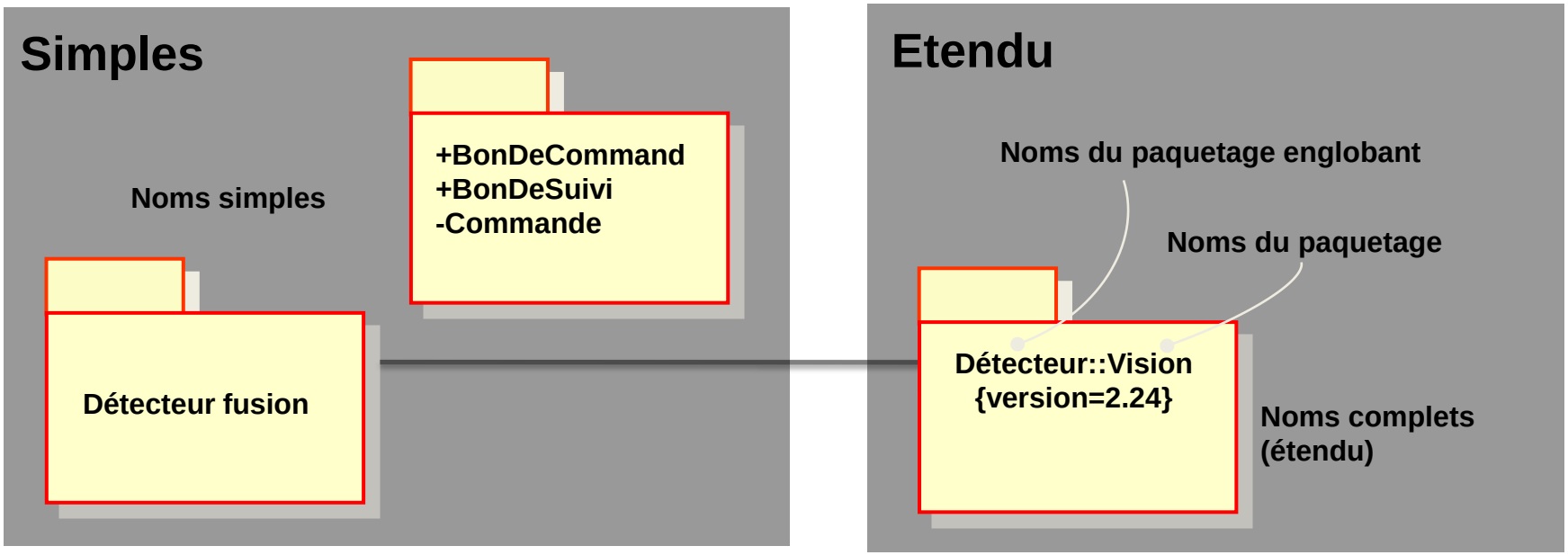
- En UML, un paquetage est un mécanisme générique pour organiser des éléments de modélisation en groupes.

- Grâce aux paquetages, il est possible
  - de manipuler un ensemble d'éléments comme un groupe.
  - On peut contrôler la visibilité des éléments au sein du paquetage,
  - On peut également utiliser les paquetages pour présenter différentes vues de l'architecture d'un système.
- Les paquetages bien conçus regroupent des éléments proches sur le plan sémantique et qui ont tendance à évoluer ensemble.
- Un paquetage est un mécanisme d'intérêt général permettant d'organiser les éléments en groupes.
  - Ils permettent de disposer les éléments des modèles de manière à en faciliter la compréhension
  - de contrôler l'accès aux contenus afin de pouvoir surveiller les points de soudure dans l'architecture du système.

Diginamic Formation

« Dynamisez votre carrière digitale ! »

Paquetages simples et étendu

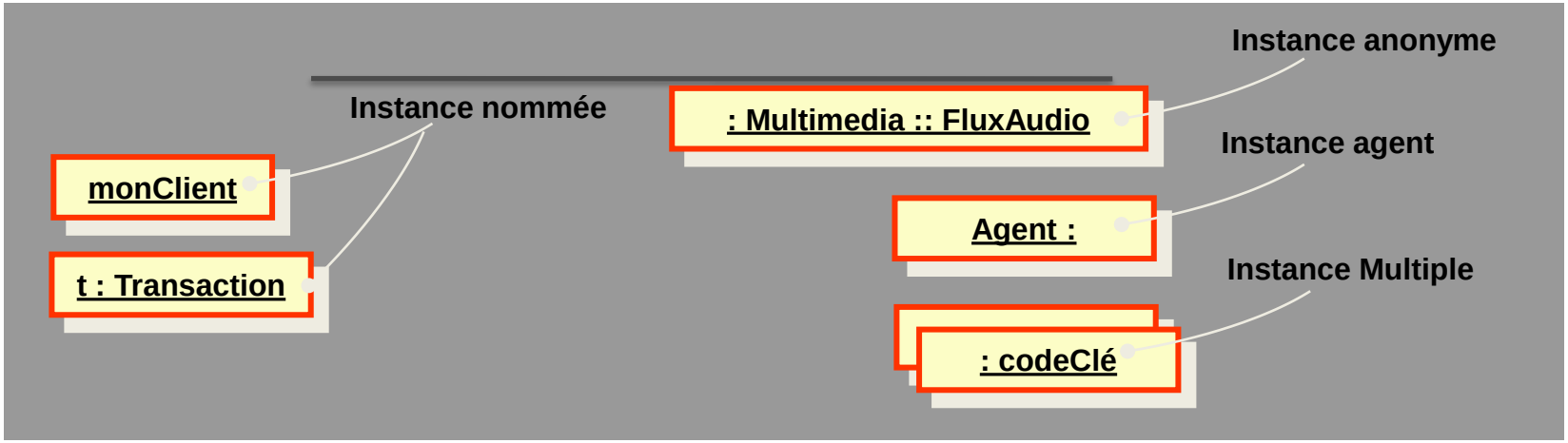


Diginamic Formation

« Dynamisez votre carrière digitale ! »

Instances

- Les termes " instance " et " objet " sont largement synonymes et peuvent se substituer l'un à l'autre dans la plupart des cas.
- Une instance est une manifestation concrète d'une abstraction à laquelle un ensemble d'opérations peut être appliqué et qui peut avoir un état qui en stocke les effets.
- On utilise les instances pour modéliser des éléments concrets ou prototypes qui existent dans le monde réel.



Diginamic
Formation
« Dynamisez votre carrière digitale ! »

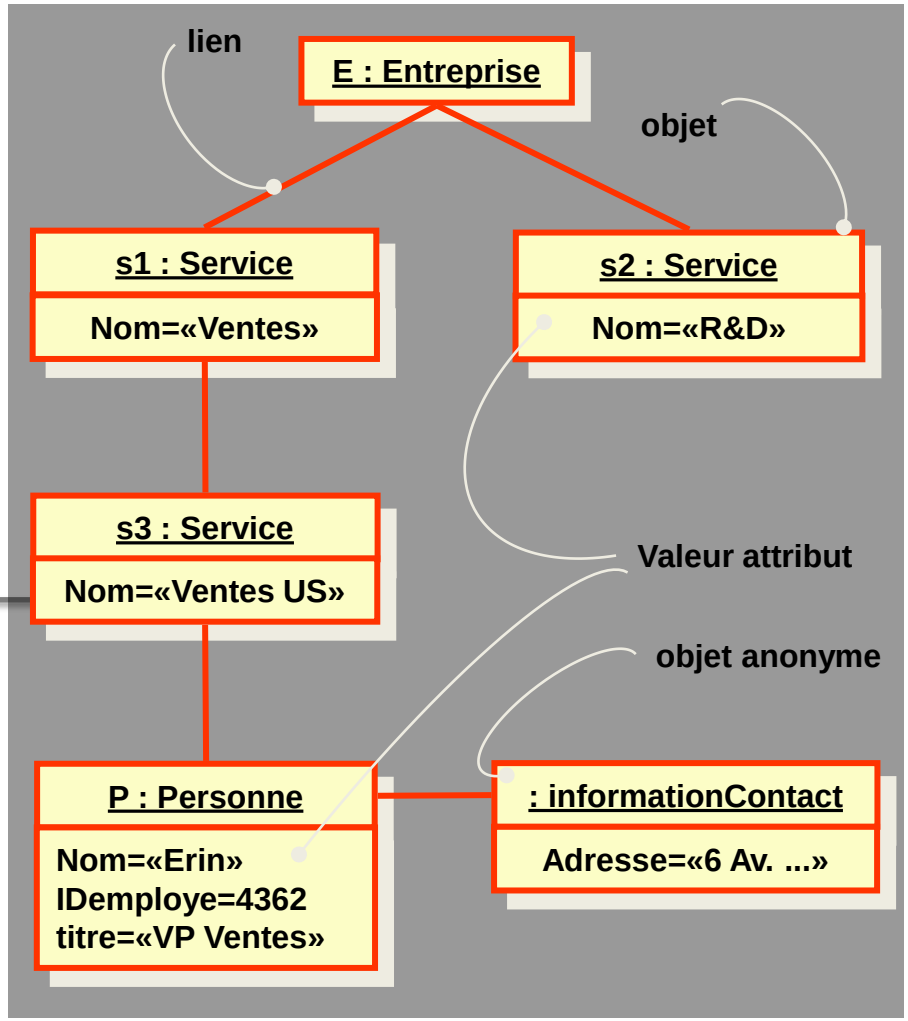
Diagramme d’objet

Diginamic Formation

« Dynamisez votre carrière digitale ! »

Diagramme d'objet

- Les diagrammes d'objets modélisent les instances d'éléments qui apparaissent sur des diagrammes de classes.
- Ils montrent un ensemble d'objets et leurs relations à un moment donné.
- On utilise les diagrammes d'objets pour modéliser les vues de conception statiques ou de processus statiques.
- En UML, on utilise les diagrammes de classes pour visualiser les aspects statiques des briques de base d'un système.
- Les diagrammes d'interaction servent à visualiser les aspects dynamiques du système, qui comprennent les instances de ces briques de base et les messages qui peuvent être échangés entre elles.



Diginamic
Formation
« Dynamisez votre carrière digitale ! »

Diagramme de classe

Diginamic Formation

« Dynamisez votre carrière digitale ! »

Diagramme de classe

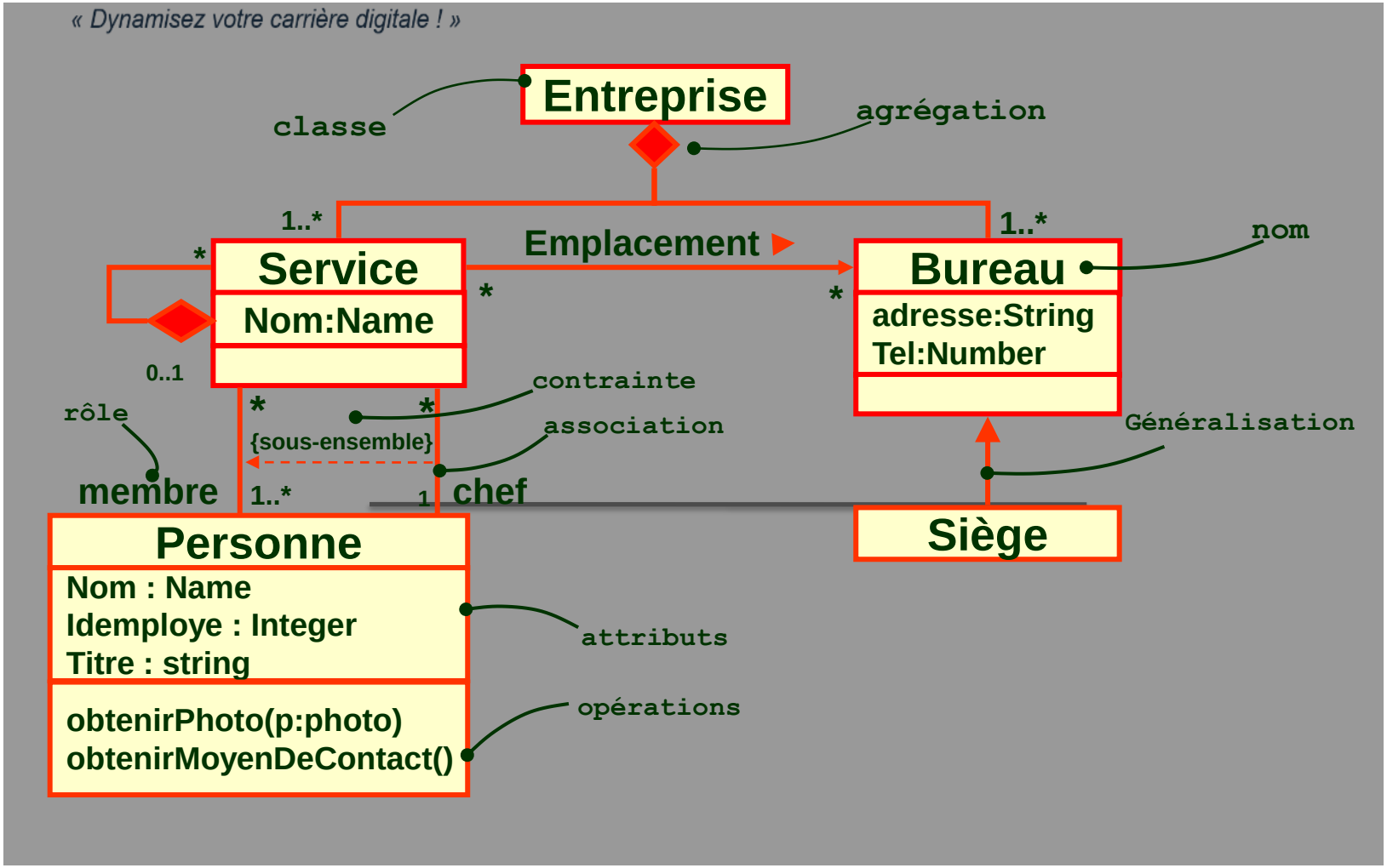
- Les diagrammes de classes sont les diagrammes les plus courants dans la modélisation des systèmes orientés objet. Ils représentent un ensemble de
  - classes,
  - d'interfaces
  - de collaborations
  - ainsi que leurs relations.
- On utilise des diagrammes de classes pour modéliser la vue de conception statique d'un système.
- Pour l'essentiel, cela implique
  - la modélisation du vocabulaire du système,
  - la modélisation de collaborations
  - ou la modélisation de schémas.

- Les diagrammes de classes sont aussi le fondement de diagrammes apparentés: diagrammes de composants et diagrammes de déploiement.
- Les diagrammes de classes sont importants non seulement pour
  - visualiser,
  - construire,
  - spécifier
  - et documenter des modèles structurels.

Diginamic Formation

« Dynamisez votre carrière digitale ! »

Diagramme de classe



Diginamic Formation

« Dynamisez votre carrière digitale ! »

Diagramme de composant

Diginamic Formation

« Dynamisez votre carrière digitale ! »

Diagramme de composant

- Les composants vivent dans le monde matériel des bits et constituent des briques de base importantes dans la modélisation des aspects physiques d'un système.
- Un composant est une partie physique remplaçable d'un système qui fournit la réalisation d'un ensemble d'interfaces et s'y conforme.
- On utilise les composants pour modéliser les éléments physiques qui peuvent se trouver sur un nœud, comme
  - les exécutables,
  - les bibliothèques,
  - les tables,
  - les fichiers
  - et les documents.

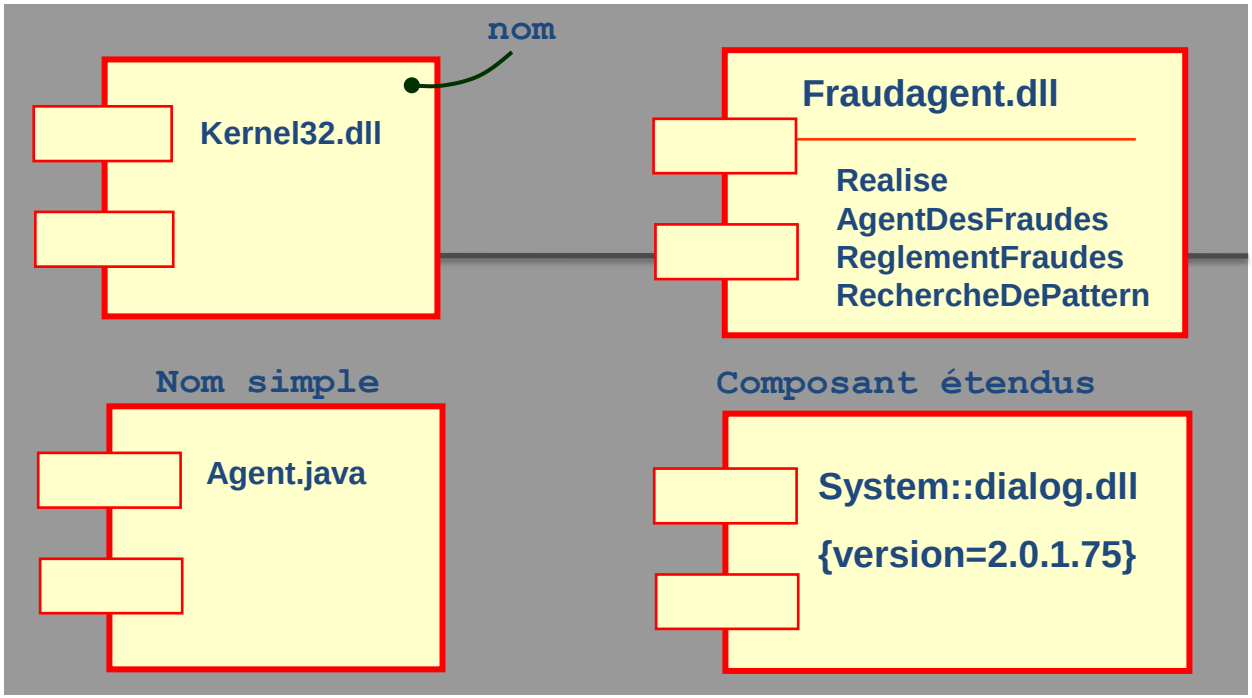
- Un composant représente en général le regroupement physique d'éléments habituellement logiques, tels que
  - les classes,
  - les interfaces
  - et les collaborations.
- Les bons composants déterminent des abstractions bien délimitées à l'aide d'interfaces bien définies, ce qui permet de remplacer facilement les composants plus anciens par des composants plus récents et compatibles.

Diginamic Formation

« Dynamisez votre carrière digitale ! »

Représentation des composants

- UML fournit une représentation graphique pour les composants.
- Cette notation canonique permet de visualiser un composant en dehors de tout système d'exploitation ou langage de programmation.
- En utilisant les stéréotypes, qui font partie des mécanismes d'extensibilité d'UML, on peut adapter cette notation pour représenter des types de composants spécifiques.

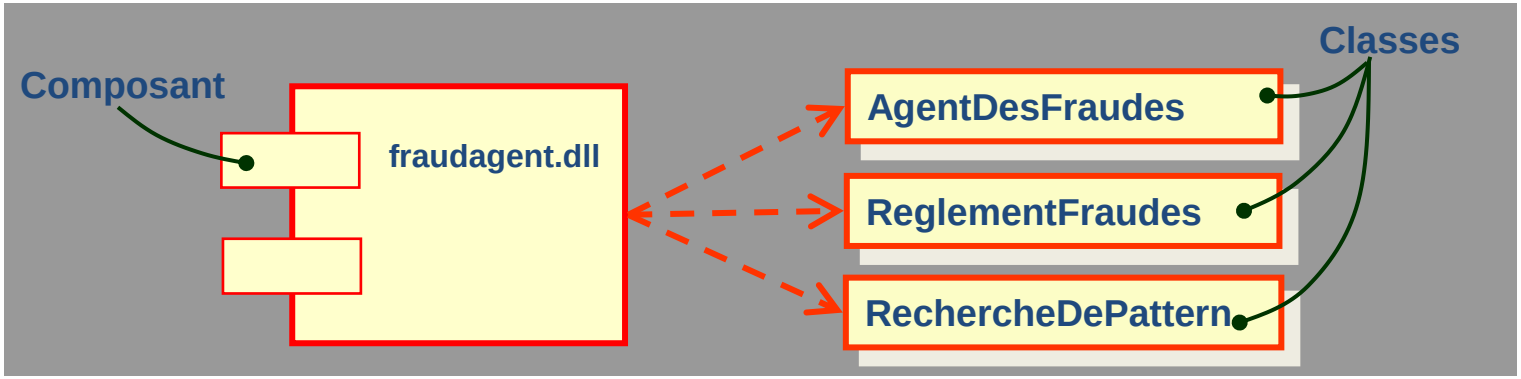


Diginamic Formation

« Dynamisez votre carrière digitale ! »

Composants et classes

- Par de nombreux aspects, les composants sont similaires aux classes:
  - ils ont tous deux un nom,
  - peuvent réaliser un ensemble d'interfaces,
  - participer à des relations de dépendance, de généralisation et d'association, et à des interactions,
  - être emboîtés et avoir des instances.
- Il existe cependant des différences importantes entre les composants et les classes:
  - les classes représentent des abstractions logiques alors que les composants représentent des éléments physiques qui existent dans le monde des bits. Pour résumer, cela signifie que les composants peuvent résider sur des nœuds, mais pas les classes.
  - les composants représentent le regroupement physique de ce qu'on pourrait appeler des composants logiques et se situent à un niveau d'abstraction différent.
  - les classes peuvent avoir directement des attributs et des opérations. En général, les composants comportent seulement des opérations que l'on peut atteindre uniquement par leur interface.



Diginamic Formation
« Dynamisez votre carrière digitale ! »

Remplaçabilité binaire

- Un composant est un élément physique remplaçable d'un système qui fournit la réalisation d'un ensemble d'interfaces et s'y conforme.
  - Premièrement, un composant est *physique.* Il vit dans le monde des bits et non dans celui des concepts.
  - Deuxièmement, un composant est *remplaçable.* Il peut être remplacé par un autre composant qui se conforme aux mêmes interfaces.
  - Troisièmement, un composant est un *élément d'un système.*
    - Un composant existe rarement indépendamment des autres.
    - Un composant donné collabore avec d'autres composants et,
    - ce faisant, existe dans le contexte d'architecture ou de technologie dans lequel il est supposé être utilisé.

    - Un composant est logiquement et physiquement cohérent et dénote donc une partie structurelle et/ou comportementale importante d'un plus grand système.
    - Il peut être réutilisé à travers de nombreux systèmes.
    - simple composant à un plus haut niveau d'abstraction.
  - Quatrièmement, comme cela est expliqué dans la partie précédente, un composant *fournit la réalisation d'un ensemble d'interfaces et s'y conforme.*

Diginamic Formation

« Dynamisez votre carrière digitale ! »

Types de composants

- Types de composants
- On distingue trois types de composants.
  - Premièrement, les *composants de déploiement* qui sont nécessaires et suffisants pour former un système exécutable,
    - comme les bibliothèques dynamiques (DLL)
    - les exécutables (EXE). .
- Deuxièmement, les *composants produits par le développement.* Ces composants sont essentiellement le résultat du processus de développement;

  - ils se composent d'éléments tels que
    - des fichiers code source
    - des fichiers de données à partir desquels les composants de déploiement sont créés.
  - Ils ne participent pas directement à un système exécutable mais sont les produits du travail de développement qui servent à créer le système exécutable.
- Troisièmement, les *composants d'exécution.* Ces composants sont créés en tant que produits d'une exécution, comme par exemple un objet COM+ qui est instancié à partir d'une DLL.

Diginamic Formation

« Dynamisez votre carrière digitale ! »

Organisation des composants

- On organise les composants en les groupant dans des paquetages de la même façon que l'on organise les classes.
- On peut également organiser des composants en spécifiant leurs relations de dépendance, de généralisation, d'association (y compris d'agrégation) et de réalisation.

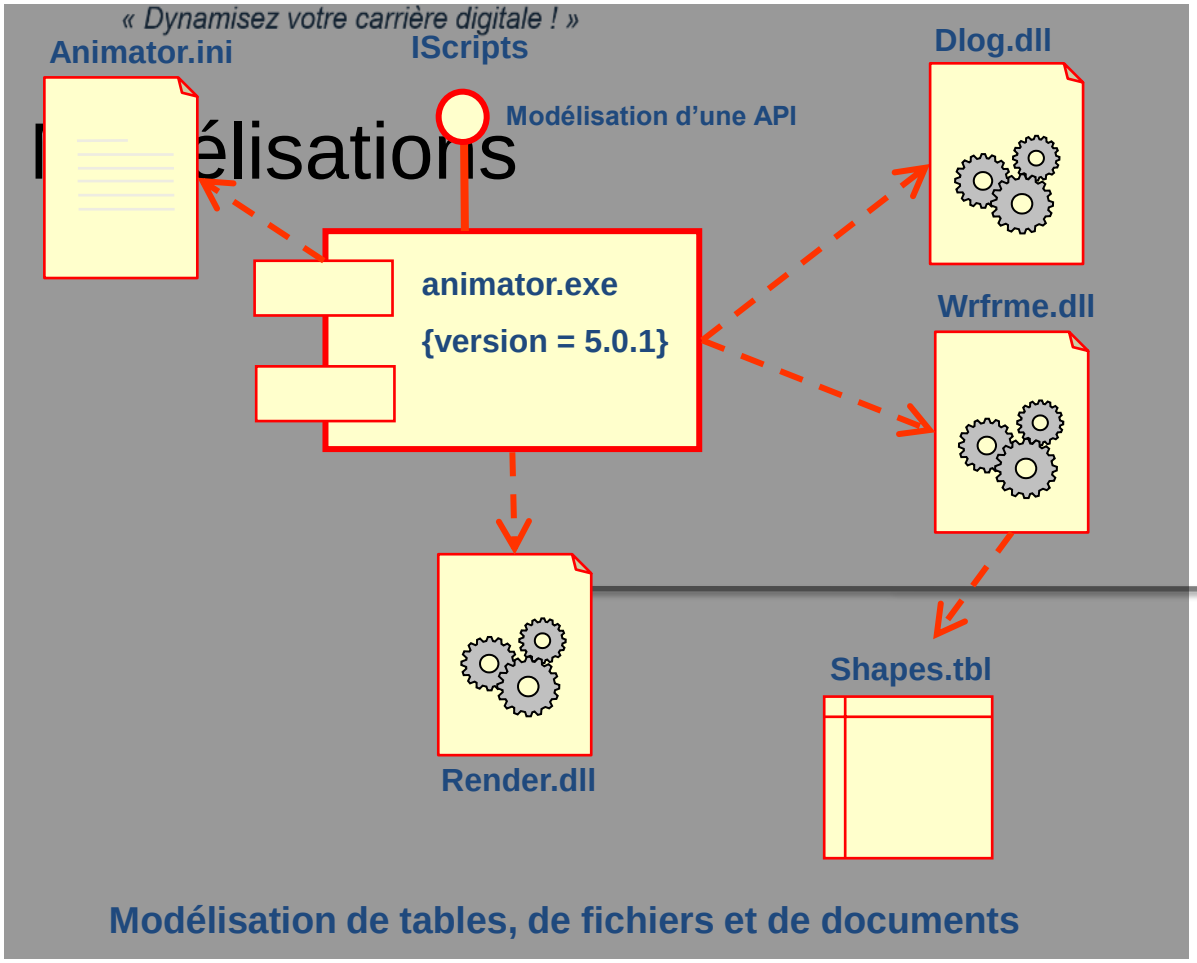
Diginamic Formation
« Dynamisez votre carrière digitale ! »

Stéréotypes standard

- UML définit cinq stéréotypes standard qui s'appliquent aux composants:
  1. executable précise un composant qui peut être exécuté sur un nœud.
  2. library précise une bibliothèque objet statique ou dynamique.
  3. table précise un composant qui représente une table de base de données.
  4. file précise un composant qui représente un document contenant un code source ou des données.
  5. document précise un composant qui représente un document.
- Remarque: UML ne fournit pas d'icônes définies pour ces stéréotypes.

Diginamic
Formation

« Dynamisez votre carrière digitale ! »



Diginamic
Formation
« Dynamisez votre carrière digitale ! »

Diagramme de déploiement

Diginamic Formation
« Dynamisez votre carrière digitale ! »

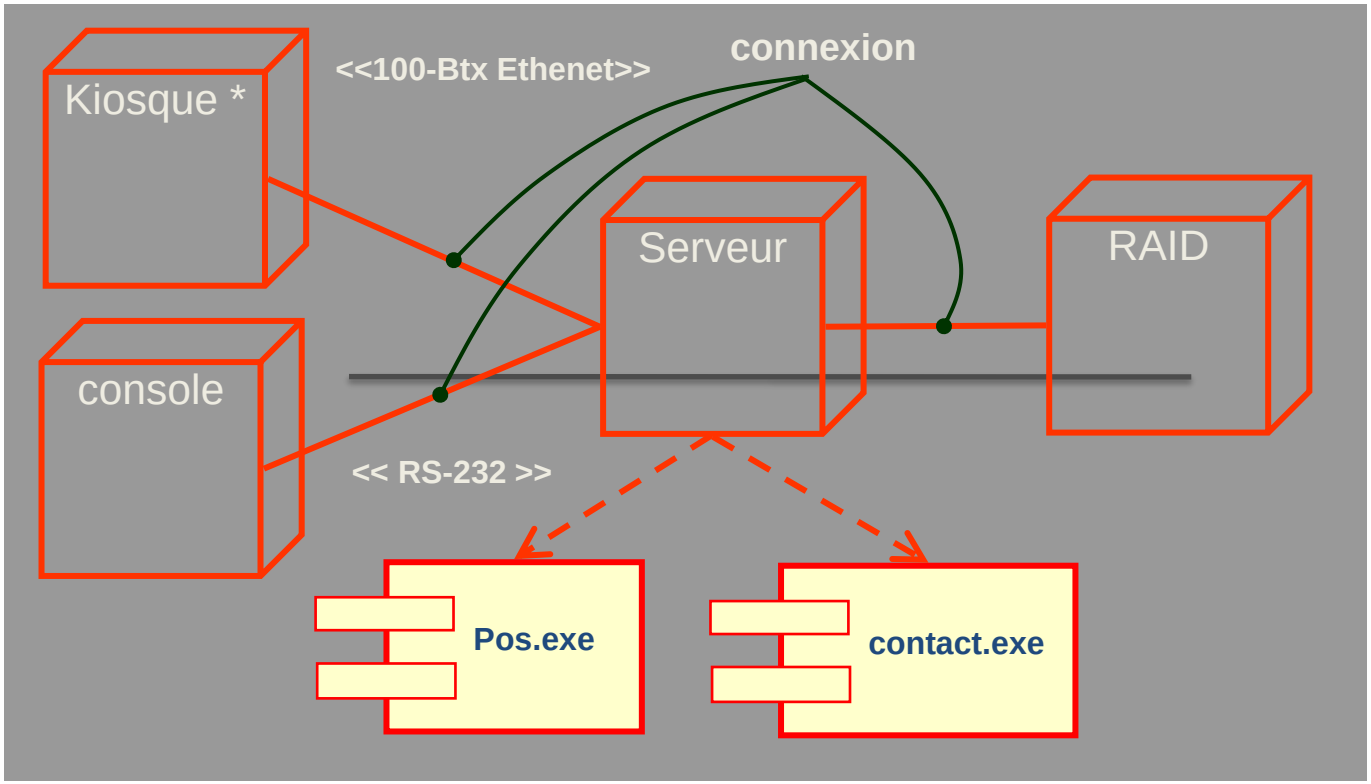
Diagramme de déploiement

- Comme les composants, les nœuds vivent dans le monde du matériel et constituent des briques de base importantes pour la modélisation des aspects physiques d'un système.
- Un nœud est un élément physique qui existe au moment de l'exécution et représente une ressource de calcul. En général, il a au moins de la mémoire et souvent, en plus, des capacités de traitement.
- On utilise les nœuds pour modéliser la topologie du matériel sur lequel le système s'exécute.

- La plupart du temps, un nœud représente un processeur ou un périphérique sur lequel les composants peuvent être déployés.
- Les bons nœuds représentent clairement le vocabulaire du matériel dans le domaine de solution concerné.

Représentation

- UML propose une représentation graphique des nœuds. Cette notation canonique permet de visualiser un nœud en dehors de tout matériel spécifique.

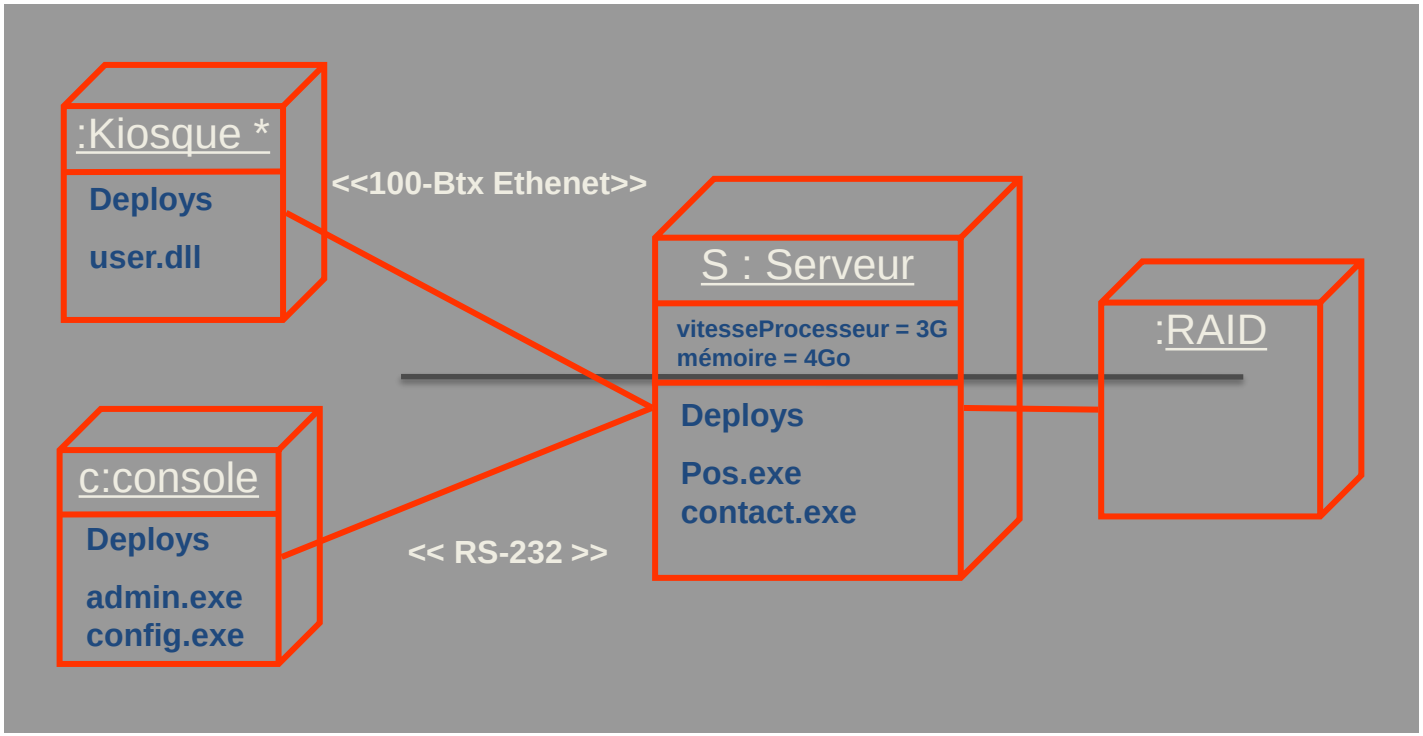


Diginamic Formation

« Dynamisez votre carrière digitale ! »

Modélisation de la répartition de composant

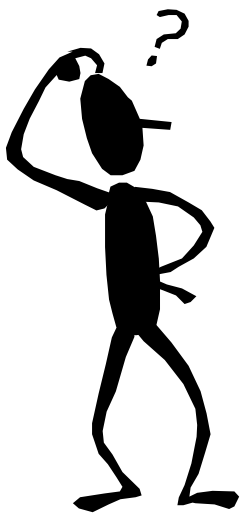
- Visualise l'instance spécifique de chaque nœud



Diginamic Formation

« Dynamisez votre carrière digitale ! »

A vous de jouer (échauffement)



Exercice 1 :

Réaliser les diagrammes de classe des expressions suivantes:

1. Un pays possède une capitale.
2. Une personne dîne avec une fourchette.
3. Un chemin peut représenter un fichier ou un répertoire.
4. Un chemin est un répertoire avec éventuellement un nom de fichier.
5. Un fichier contient des enregistrements.

---

6. Un fichier est accessible par un utilisateur selon des droits d'accès.
7. Un dessin est soit du texte, soit une forme géométrique, soit un groupe de dessins.
8. Des personnes utilisent un langage pour un projet.
9. Une personne joue dans une équipe pour une certaine durée.
10. Une équipe est composée de plusieurs personnes.
11. Une route connecte deux villes.

Diginamic
Formation

« Dynamisez votre carrière digitale ! »

Corrections exercices 1

Diginamic Formation

« Dynamisez votre carrière digitale ! »

Architecture

- Le système doit être envisagé sous plusieurs perspectives.
- Les différents intervenants (utilisateurs finals, analystes, développeurs, intégrateurs de système, testeurs, rédacteurs techniques et chefs de projet) ont chacun des besoins distincts pour un même projet
- Tous, envisagent ce système sous des approches et à des moments différents.

  L'architecture d'un système est peut-être l'artefact le plus important qui puisse être utilisé pour gérer ces différents points de vue et, ainsi, diriger le développement itératif et incrémental.

- L'architecture est l'ensemble des décisions fondamentales en matière:
  - d'organisation d'un système logiciel,
  - de choix des éléments structurels qui composent le système et de leurs interfaces,
  - de comportement de ces éléments structurels, tel qu'il est spécifié dans leurs collaborations,
  - de composition de ces éléments structurels et comportementaux en des sous-ensembles progressivement plus vaste.
  - de style architectural qui guide cette organisation.

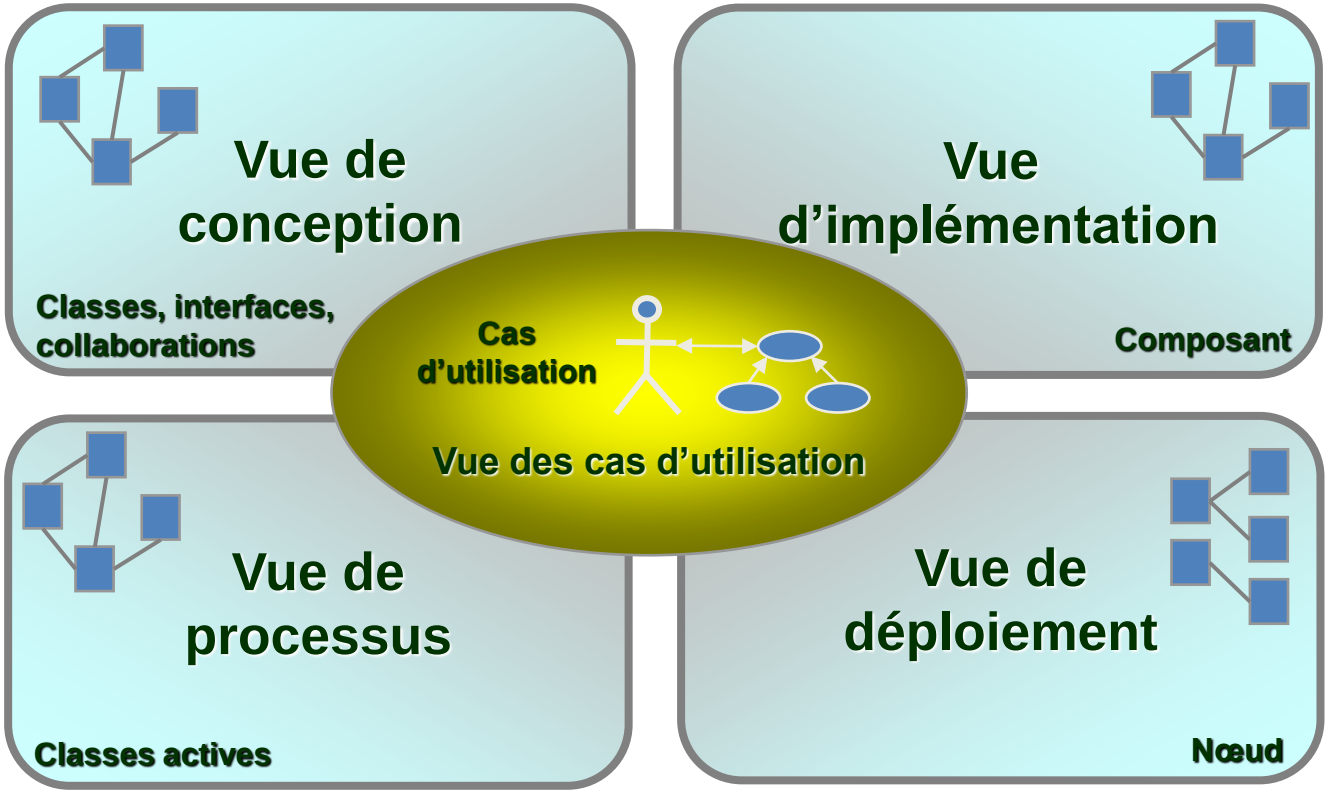
Diginamic Formation

« Dynamisez votre carrière digitale ! »

Modélisation de l'architecture d'un système

Vocabulaire
fonctionnalité

Assemblage du système
gestion de configuration



Comportement

Performance
monter en charge
débit

Topologie du système
distribution
livraison
installation

Diginamic Formation
« Dynamisez votre carrière digitale ! »

Processus de développement (de) logiciel

Un processus correctement défini aide à choisir:

- *les artefacts à produire*
- *les procédures à développer*
- *les intervenants chargés de leur création et de leur gestion*
- *la manière d'employer ces artefacts pour évaluer et diriger le projet dans son ensemble*

Diginamic Formation

« Dynamisez votre carrière digitale ! »

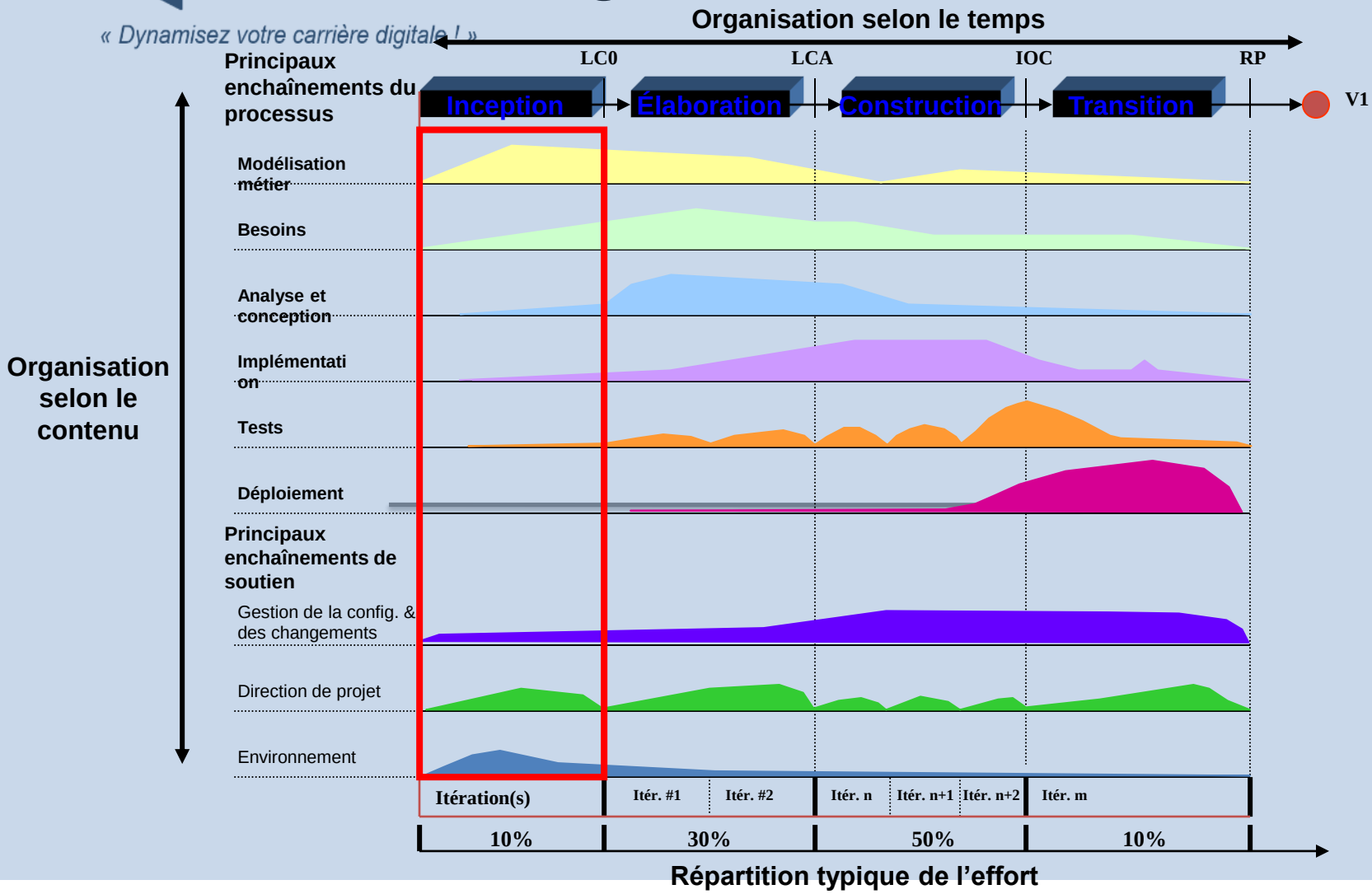
Cycle de développement logiciel

- UML est indépendant des processus utilisés,
- Pour exploiter UML au maximum, il faut envisager un processus qui soit:
  - Piloté sur les cas d'utilisation,
    - les cas d'utilisation sont utilisés comme artefacts de base pour définir le comportement attendu du système, pour vérifier et valider l'architecture du système, pour faire des tests et pour permettre à tous les intervenants du projet de communiquer.
  - Centré sur l'architecture,
    - Signifie que l'architecture d'un système est utilisée comme artefact de base dans la:
      - conceptualisation,
      - la construction,
      - la gestion et l'évolution du système en cours de développement.
  - Itératif
    - implique la gestion d'une succession de versions exécutables
  - Incrémental
    - signifie que l'architecture du système est constamment améliorée, pour produire ces nouvelles versions qui apportent toutes des améliorations incrémentales par rapport à la précédente.
- Un processus à la fois itératif et incrémental est *centré sur les risques:* chaque nouvelle version se concentre sur la prise en charge et la réduction des risques les plus importants qui pourraient menacer la réussite du projet.
- Il est divisé en plusieurs 4 phases. Une *phase* est le laps de temps qui s'écoule entre deux étapes déterminantes du processus, quand un ensemble bien défini d'objectifs est atteint, quand les artefacts sont terminés et quand la décision de passer ou non à la phase suivante est prise.

Diginamic Formation

« Dynamisez votre carrière digitale ! »

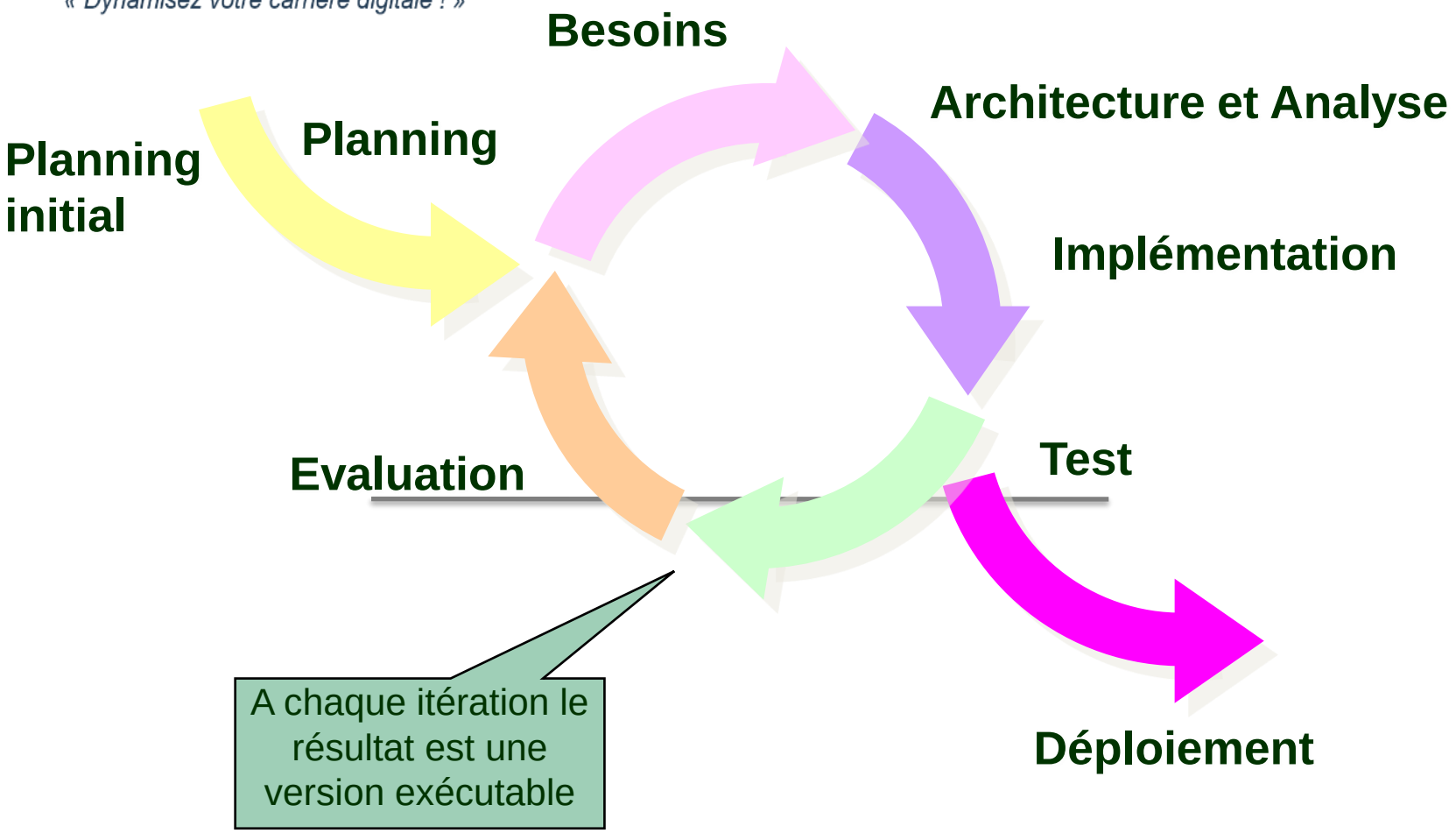
Cycle de développement logiciel UP



Diginamic Formation

« Dynamisez votre carrière digitale ! »

Le processus unifié



Diginamic Formation
« Dynamisez votre carrière digitale ! »

UP est agile

- Les spécialistes en méthodologie distinguent:
  - les processus "lourds" des processus "légers",
    - est un terme péjoratif qui sert à suggérer que le processus présente l'une des caractéristiques suivantes.
      - nombreux artefacts créés dans un environnement bureaucratique;
      - rigidité et contrôle;
      - planification à long terme;
      - élaborée et détaillée;
    - processus prédictif plutôt qu'adaptatif.

  - les processus" prédictifs
    - tente de prévoir et de planifier en détail toutes les activités et les allocations de ressources (les individus) sur une période relativement longue.
    - Les processus prédictifs ont généralement un cycle de vie en "cascade" ou séquentiel, qui consiste à :
      1. Définir tous les besoins.
      2. Mettre au point une conception détaillée.
      3. Implémenter.

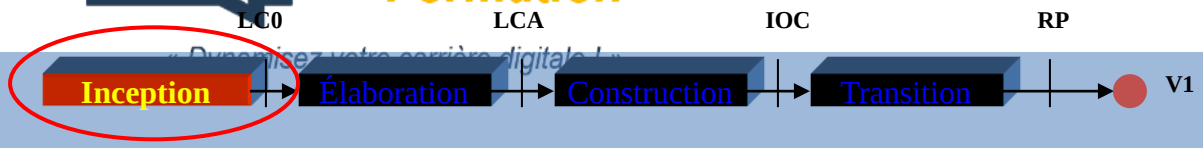
Diginamic Formation
« Dynamisez votre carrière digitale ! »

UP est super agile

- des processus" adaptatifs".
  - accepte que le changement soit un moteur inévitable, et il favorise souplesse et adaptation. Son cycle de vie est habituellement itératif. Un processus "agile" est léger et adaptatif.
    - De préférence, l'ensemble des activités et des artefacts doit être restreint. Certains projets en nécessiteront plus que d'autres, mais il faut généralement privilégier la simplicité.
    - Comme le Processus Unifié est itératif, les spécifications et la conception ne sont pas terminées avant le début de l'implémentation. Elles sont développées sur un mode adaptatif lors de la suite des itérations, grâce au feed-back.

    - Le projet global n'a pas de plan détaillé.
      - Il existe un plan de haut niveau, nommé plan de phases, qui estime une date d'achèvement et d'autres échéances majeures, mais ne détaille pas précisément les étapes, nécessaires pour les atteindre.
      - Un plan détaillé, nommé plan d'itération, ne planifie de façon détaillée qu'une itération à la fois.
      - La planification détaillée s'effectue de manière adaptative d’itération en itération.

Phase : inception



La charge

- Établir le périmètre du projet
- Déterminer les cas d'utilisation critiques du système
- Présenter au moins une architecture compatible avec les scénarios principaux
- Estimer les risques potentiels (les sources d'incertitudes)
- Estimer une enveloppe globale de coûts et délais du projet

Cas d'utilisation* (besoins d'écrit graphiquement)

les cas d'utilisation comportent généralement les informations suivantes:	les principaux artefacts
- Un identificateur unique - L'énoncé de l'objectif - Les auteurs - La priorité - Le risque - Les hypothèses - Les préconditions - Les postconditions - Les problèmes non résolus - Les besoins satisfaits	- Un modèle initial des cas d'utilisation (10% et 20% du nombre total de cas) - Diagramme de cas d'utilisation - Diagramme de séquence - Diagramme de robustesse - Un modèle de domaine, plus sophistiqué qu'un glossaire - Un modèle du métier

Passer le jalon LC0 (Life-Cycle-Objective)

- Les parties s'accordent sur la définitions des ambitions du projet et l'estimation des coûts et délais	- Les estimations des coûts, des délais, des priorités, des risques et du plan de développement sont crédibles
- Les exigences décrites dans les cas d'utilisations primaires sont compris	- Les dépenses réelles sont conformes aux prévisions.

La gestion des exigences

Besoins fonctionnels

- Le système doit …
- Le système doit produire …

Besoins non-fonctionnels

- Les besoins d'utilisabilité
- Les besoins de performance
- Les besoins de disponibilité/fiabilité
- Les besoins de sécurité
- Les besoins matériels
- Les besoins de déploiement

Collecte des besoins

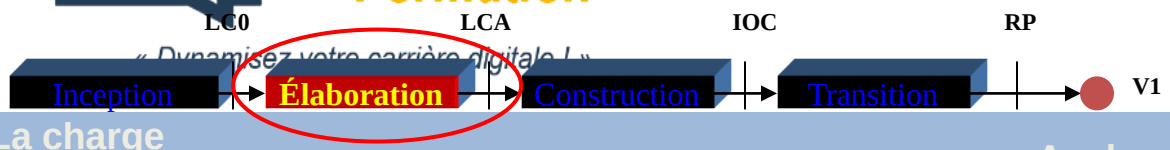
- Formulation de chaque besoin claire et concis en évitant les descriptions verbeuses prêtant à des interprétations multiples.
- L'énoncé d'un besoins doit être concentré sur une seule idée.
- Chaque besoins d'un être vérifiable (« l'interface doit être intuitive » ne pas être testé objectivement)

Les acteurs principaux de l'inception

- le client
- Le consultants
- Le chef de projet
- Autres …

Diginamic Formation

Phase : élaboration



La charge

Traduire l'ensemble des exigences en une forme plus adaptée au monde des concepteur de logiciel

Choisir la meilleur **stratégie d'implémentation**

Tôt dans le projet, il faut mettre en place une architecture **robuste** qui permet de concevoir un système **facile** à comprendre, à **construire** et à faire **évoluer.**

Dresse une image plutôt idéale du système

Analyse (des besoins fonctionnels)

Itérations

- Découpage du projet « diviser pour mieux régner »
- Détection du risque « au plus tôt »

Définition du modèle de haut niveau : paquetages

- Le modèle doit être:

Intelligibles : un individu doit pouvoir saisir la sémantique, la justification, les éléments majeurs, et la responsabilité du paquetage.

Cohérents : d'un point vue logique les classes sont liées. A un certains niveau d'arborescence les classes du paquetage ne forment plus q'un groupe.

Faiblement couplés : généralement, chaque classe a plus de relations avec des classes du même paquetage qu'avec des classes qui se trouvent en dehors de celui-ci.

Hiérarchiquement peu profonds : les hiérarchies trop profondes sont difficiles à comprendre, d'autant plus que chaque niveau draine avec lui ses propres signification. Il est conseillé de ce contenter de deux ou trois niveaux par hiérarchie.

Méthodologies de modélisation orientée objet

Plusieurs méthodes de modélisation	les principaux artefacts
-UML -OMT -SADT -ER -Gantt -…	-Paquetage -Diagramme de classe -Diagrammes d'état ou d'activités -Diagramme de robustesse -Diagrammes de séquence -Diagrammes de collaboration -Schéma de base de données - Intégrité de domaine - Intégrité de référence -Plan d'abstraction du système de donnée

Les acteurs principaux de l'élaboration

- Le chef de projet
- Concepteur méthode
- Qualiticien
- autres …

Passer le jalon LCA (Life-Cycle-Architecture)

La vision du produit est-elle stable ? L'architecture est-elle stable ? Peut-on démontrer, que les risques majeurs ont été effectivement traités ? Le plan de la phase de construction est-il suffisamment détaillé et précis ?	-Toutes les parties concernés reconnaissent-elles, sur la base de l'architecture définie, qu'il est effectivement possible d'atteindre la vision du produit en exécutant le plan de développement proposé ? Les dépenses réelles sont-elles acceptables par rapport aux dépenses prévues ?

Diginamic Formation

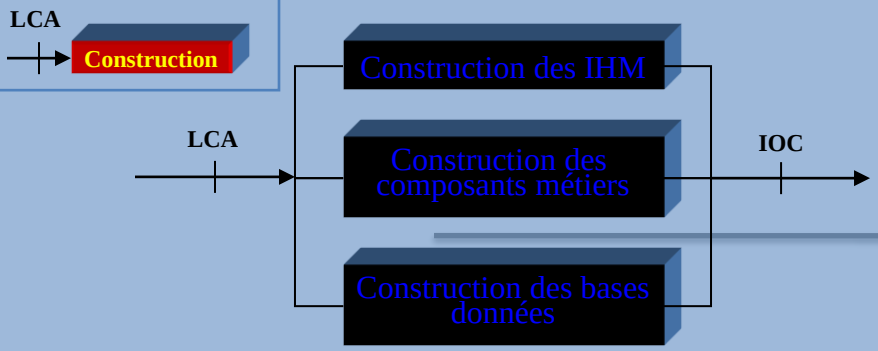
« Dynamisez votre carrière digitale ! »

Phase : construction

La charge

Tous les éléments manquants sont développés et intégrés dans le produits et toutes les fonctionnalités testées unitairement et globalement.

Organiser la production pour supporter l'ingénierie concurrente



- **Pour accélérer de façons significative la mise au point de versions livrables. Il faut effectuer des itérations parallèles**
- **On rend plus complexe la gestion des ressources et la synchronisation des activités entres les équipes.**

Objectifs de la phase de construction

- **Minimiser les coûts de développement en optimisant les ressources et en évitant d'exécuter des travaux inutiles (que l'on doit abandonner ou recommencer)**
- **Obtenir un produit de qualité satisfaisante aussi rapidement que possible**
- **Mettre au point au plus vite les versions utiles (alpha, bêta, RC1, RC2, autres versions de test).**

Les acteurs principaux de l'élaboration

- **Concepteur méthode**
- **Développeur scripting ou intégrateur IHM**
- **Développeur Objet**
- **Développeur base de données**
- **Qualiticiens**
- **Autres …**

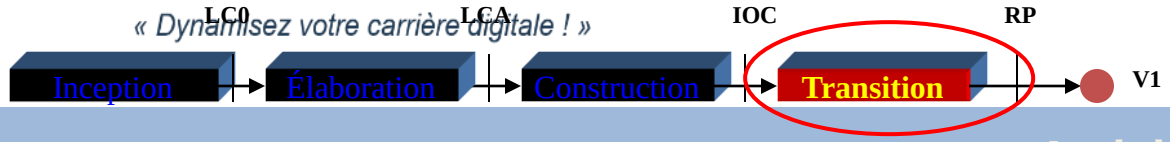
Passer le jalon IOC (Initial Operational Capability)

- **La version du produit est-elle suffisamment stable et au point pour être déployée chez les utilisateurs ?**
- **Toutes les parties concerné sont-elles prêtes pour la mise en place du produit chez les utilisateur ?**
- **Les dépenses réelles sont-elles acceptables par rapport aux dépenses prévues**

Diginamic Formation

« Dynamisez votre carrière digitale ! »

Phase : transition



La charge

- Rendre l'utilisateur opérationnel et autonome.
- S'assurer auprès de toutes les parties concernées que les versions livrées sont complètes et répondent aux critères d'évaluation de la vision.
- Livrer une version finale aussi rapidement que possible.

Activités

- Mettre le produit en bêta test pour le valider
- Mettre en service le nouveau système, tout en maintenant le système existant en parallèle.
- Convertir la base de données opérationnelles
- Former les utilisateurs et les responsables de la maintenance
- Transmettre le produit aux équipes marketing, de distribution et de vente
- Mettre au point le logiciel, en particulier corriger les erreurs et améliorer les performances et la facilité d'utilisation.
- Évaluer la version finale par rapport à la vision et aux critères d'acceptation du produit.

Les acteurs principaux de l'élaboration

- Le chef de projet
- Concepteur méthode
- Qualiticien
- Développeurs
- Clients
- Autres …

Passer le jalon RP (Release Product)

- Les utilisateurs sont-ils satisfaits
- Les dépenses réelle sont-elles encore acceptables par rapport aux dépenses prévues.

Diginamic Formation

« Dynamisez votre carrière digitale ! »

Vous savez que vous n'avez pas compris le Processus Unifié quand...

- Vous pensez que initialisation = spécifications, élaboration = conception et construction = implémentation (autrement dit vous superposez un cycle de vie en cascade au Processus Unifié).
- Vous pensez que l'élaboration a pour but de définir entièrement et rigoureusement des modèles qui sont traduits en code durant la construction.
- Vous essayez de définir la plupart des besoins avant d'entamer la conception ou l'implémentation.
- Vous essayez de terminer la plus grande partie de la conception avant de commencer l'implémentation;
- vous essayez de définir entièrement une architecture et de vous y engager avant de commencer la programmation itérative et les tests.
- Vous consacrez "beaucoup de temps" à la définition des spécifications ou au travail de conception avant de programmer.
- Vous croyez que la durée convenable d'une itération est de quatre mois et non de quatre semaines (non compris les projets impliquant des centaines de développeurs).

- Vous pensez que les activités de conception et le temps consacré à tracer des diagrammes UML doivent servir à définir exactement des modèles très détaillés, et que la programmation n'est qu'une simple transcription mécanique de ceux-ci en code.
- Vous pensez que l'adoption d'UML implique beaucoup d'activités et beaucoup de documents, et vous considérez ou vous ressentez le Processus Unifié comme un processus
- formel et tatillon qui comprend de nombreuses étapes à suivre.
- Vous essayez de planifier un projet en détail du début à la fin; vous essayez de prédire spéculativement toutes les itérations et ce qui doit se passer dans chacune d'elles.
- Vous voulez disposer de plans et d'estimations crédibles avant la fin de la phase d'élaboration.