DOCUMENTATION SUR UML

I. Méthode MERISE

a) Définition

MERISE est une méthode de modélisation de données et traitements orienté bases de données relationnelles.

Merise a été très utilisée dans les années 1970 et 1980 pour l'informatisation massive des organisations. Il existe plusieurs logiciels pour son implémentation ex :JMerise.

b) Démarches

La méthode Merise d'analyse et de conception propose une démarche articulée simultanément selon 3 axes pour *hiérarchiser* les préoccupations et les questions auxquelles répondre lors de la conduite d'un projet :

- **Cycle de vie** : phases de conception, de réalisation, de maintenance puis nouveau cycle de projet.
- **Cycle de décision** : des grands choix (GO-NO GO : Étude préalable), la définition du projet (étude détaillée) jusqu'aux petites décisions des détails de la réalisation et de la mise en œuvre du système d'information. Chaque étape est documentée et marquée par une prise de décision.
- **Cycle d'abstraction** : niveaux conceptuels, d'organisation, logique et physique/opérationnel (du plus abstrait au plus concret) L'objectif du *cycle d'abstraction* est de prendre d'abord les grandes décisions métier, pour les principales activités (Conceptuel) sans rentrer dans le détail de questions d'ordre de l'organisation ou technique.

L'étude *conceptuelle* Merise s'attache aux *invariants* de l'entreprise ou de l'organisme du point de vue du métier : quelles sont les activités, les métiers gérés par l'entreprise, quels sont les grands processus traités, de quoi parle-t-on en matière de données, quelles notions manipule-t-on ?... et ce indépendamment des choix techniques (comment fait-on ?) ou d'organisation (qui fait quoi ?) qui ne seront abordés que dans les niveaux suivants.

Au niveau conceptuel on veut décrire, après *abstraction*, le modèle (le *système*) de l'entreprise ou de l'organisme :

DOCUMENTATION SUR UML

- le **Modèle conceptuel des données** (ou MCD), schéma représentant la structure du système d'information, du point de vue des données, c'est-à-dire les dépendances ou relations entre les différentes données du système d'information (par exemple : le client, la commande, les produits, etc.)
- Le **Modèle conceptuel des traitements** (ou MCT), schéma représentant les traitements, en réponse aux évènements à traiter (par exemple : la prise en compte de la commande d'un client).

  Dans l'idéal, le MCD et le MCT d'une entreprise sont stables, à périmètre fonctionnel constant, et tant que le métier de l'entreprise ne varie pas. La modélisation ne dépend pas du choix d'un progiciel ou d'un autre, d'une automatisation ou non des tâches à effectuer, d'une organisation ou d'une autre, etc.
- Le **Modèle Logique de Données** (MLD) est la modélisation logique des données qui tient compte du niveau organisationnel des données. Il s'agit d'une vue logique en terme d'organisation de données nécessaire à un traitement. En symbolique, il y a des flèches entre les tables au lieu des patates entre les entités du MCD.

c) Avantages

Pour petites bases de données limitées à la troisième forme normale,MERISE est généralement l'une des meilleures solutions du point de vue de l'architecture de base de données.

d) Inconvénients

Elle est en revanche moins adaptée aux projets transverses aux organisations, qui gèrent le plus souvent des informations à caractère sociétal (environnemental et social) avec des parties prenantes.

II. METHODE UML

a) Définition

Le **Langage de Modélisation Unifié (UML)**, de l'anglais *Unified Modeling Language* (**UML**), est un langage de modélisation graphique à base de pictogrammes conçu pour fournir une méthode normalisée pour visualiser la conception d'un système. Il est couramment utilisé en développement logiciel et en conception orientée objet.

DOCUMENTATION SUR UML

b) Démarches

UML est utilisé pour spécifier, visualiser, modifier et construire les documents nécessaires au bon développement d'un logiciel orienté objet. Il offre un standard de modélisation, pour représenter l'architecture logicielle. Les différents éléments représentables sont :

- Activité d'un objet/logiciel
- Acteurs
- Processus
- Schéma de base de données
- Composants logiciels
- Réutilisation de composants

UML se décompose en plusieurs parties :

- Les *vues* : ce sont les observables du système. Elles décrivent le système d'un point de vue donné, qui peut être organisationnel, dynamique, temporel, architectural, géographique, logique, etc. En combinant toutes ces vues, il est possible de définir (ou retrouver) le système complet.

- Les *diagrammes* : ce sont des ensembles d'éléments graphiques. Ils décrivent le contenu des vues, qui sont des notions abstraites. Ils peuvent faire partie de plusieurs vues.

- Les *modèles d'élément* : ce sont les éléments graphiques des diagrammes.

1. Les Vues

La vue logique

- Cette vue de haut niveau se concentre sur l'abstraction et l'encapsulation, elle modélise les éléments et mécanismes principaux du système.

DOCUMENTATION SUR UML

- Elle identifie les éléments du domaine, ainsi que les relations et interactions entre ces éléments :
  - les éléments du domaine sont liés au(x) métier(s) de l'entreprise,
  - ils sont indispensables à la mission du système,
  - ils gagnent à être réutilisés (ils représentent un savoir-faire).
- Cette vue organise aussi (selon des critères purement logiques), les éléments du domaine en ***"catégories"*** :
  - pour répartir les tâches dans les équipes,
  - regrouper ce qui peut être générique,
  - isoler ce qui est propre à une version donnée, etc...

La vue des composants

Cette vue de bas niveau (aussi appelée "vue de réalisation"), montre :

- L'allocation des éléments de modélisation dans des modules (fichiers sources, bibliothèques dynamiques, bases de données, exécutables, etc...).
- En d'autres termes, cette vue identifie les modules qui réalisent (physiquement) les classes de la vue logique.
- L'organisation des composants, c'est-à-dire la distribution du code en gestion de configuration, les dépendances entre les composants...
- Les contraintes de développement (bibliothèques externes...).
- La vue des composants montre aussi l'organisation des modules en ***"sous-systèmes"***, les interfaces des sous-systèmes et leurs dépendances (avec d'autres sous-systèmes ou modules).

La vue des processus

Cette vue est très importante dans les environnements multitâches ; elle montre :

- La décomposition du système en terme de processus (tâches).
- Les interactions entre les processus (leur communication).

- La synchronisation et la communication des activités parallèles (threads).

La vue de déploiement

Cette vue très importante dans les environnements distribués, décrit les ressources matérielles et la répartition du logiciel dans ces ressources :

- La disposition et nature physique des matériels, ainsi que leurs performances.
- L'implantation des modules principaux sur les noeuds du réseau.
- Les exigences en terme de performances (temps de réponse, tolérance aux fautes et pannes...).

La vue des besoins des utilisateurs

Cette vue (dont le nom exact est "vue des cas d'utilisation"), guide toutes les autres.

- Dessiner le plan (l'architecture) d'un système informatique n'est pas suffisant, il faut le justifier !
- Cette vue définit les besoins des clients du système et centre la définition de l'architecture du système sur la satisfaction (la réalisation) de ces besoins.
- A l'aide de scénarios et de cas d'utilisation, cette vue conduit à la définition d'un modèle d'architecture pertinent et cohérent.
- Cette vue est la "colle" qui unifie les quatre autres vues de l'architecture.
- Elle motive les choix, permet d'identifier les interfaces critiques et force à se concentrer sur les problèmes importants.

2. Les Diagrammes

Les *diagrammes* sont dépendants hiérarchiquement et se complètent, de façon à permettre la modélisation d'un projet tout au long de son cycle de vie. Il en existe quatorze depuis UML 2.3.

Diagrammes de structure ou diagrammes statiques

DOCUMENTATION SUR UML

- *Diagramme de classes* (*class diagram*) : représentation des classes intervenant dans le système.

  Le **diagramme de classes** est un schéma utilisé en génie logiciel pour présenter les classes et les interfaces des systèmes ainsi que les différentes relations entre celles-ci. Ce diagramme fait partie de la partie statique d'UML car il fait abstraction des aspects temporels et dynamiques.

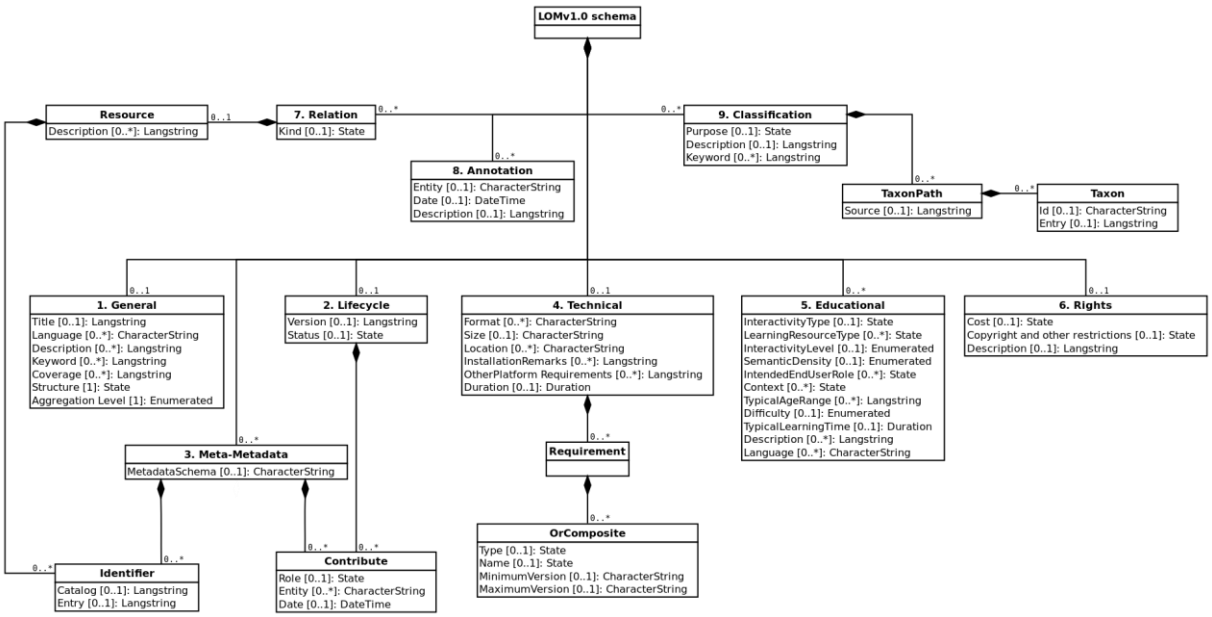
  Une classe décrit les responsabilités, le comportement et le type d'un ensemble d'objets. Les éléments de cet ensemble sont les instances de la classe.

  Une classe est un ensemble de fonctions et de données (attributs) qui sont liées ensemble par un champ sémantique. Les classes sont utilisées dans la programmation orientée objet. Elles permettent de modéliser un programme et ainsi de découper une tâche complexe en plusieurs petits travaux simples.

  Les classes peuvent être liées entre elles grâce au mécanisme d'héritage qui permet de mettre en évidence des relations de parenté. D'autres relations sont possibles entre des classes, chacune de ces relations est représentée par un arc spécifique dans le diagramme de classes.

  Elles sont finalement instanciées pour créer des objets (une classe est un *moule à objet* : elle décrit les caractéristiques des objets, les objets contiennent leurs valeurs propres pour chacune de ces caractéristiques lorsqu'ils sont instanciés).

  Ex :

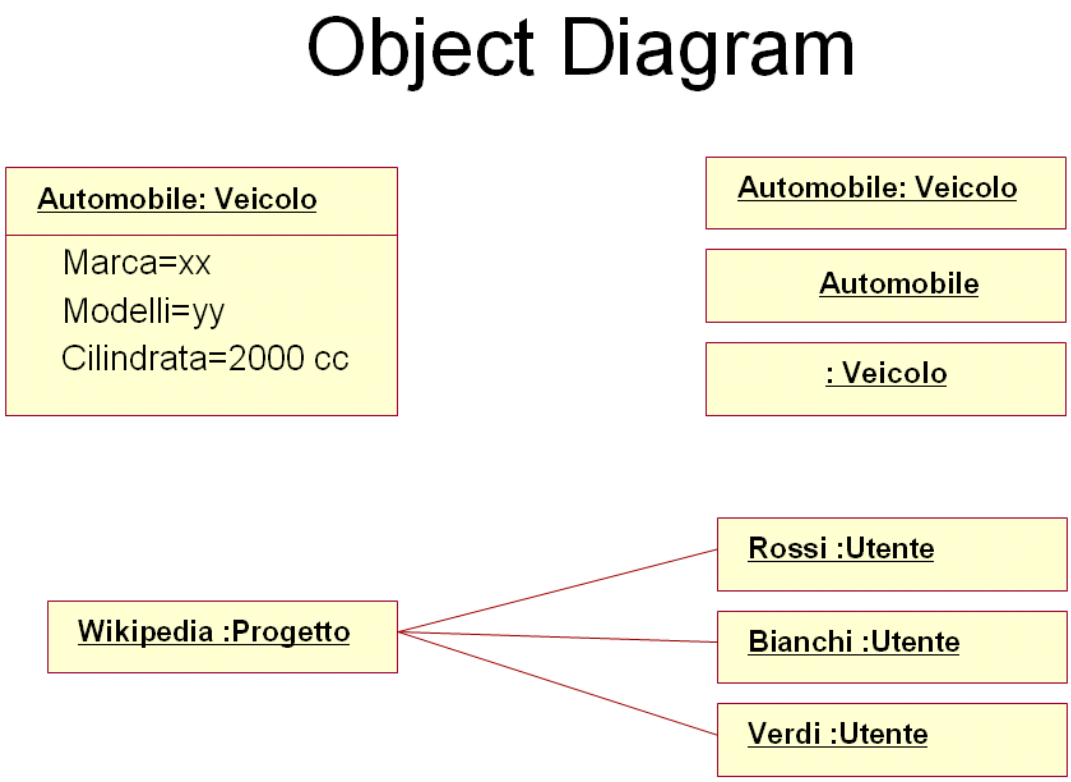


- Diagramme d'Objets

  Le **diagramme d'objets**, dans le langage de modélisation de donnée UML, permet de représenter les instances des classes, c'est-à-dire des objets. Comme le diagramme de classes, il exprime les relations qui existent entre les objets, mais aussi l'état des objets, ce qui permet d'exprimer des contextes d'exécution. En ce sens, ce diagramme est moins général que le diagramme de classes. Les diagrammes d'objets s'utilisent pour montrer l'état des instances d'objet avant et après une interaction, autrement dit c'est

DOCUMENTATION SUR UML

une photographie à un instant précis des attributs et objet existant. Il est utilisé en phase exploratoire.

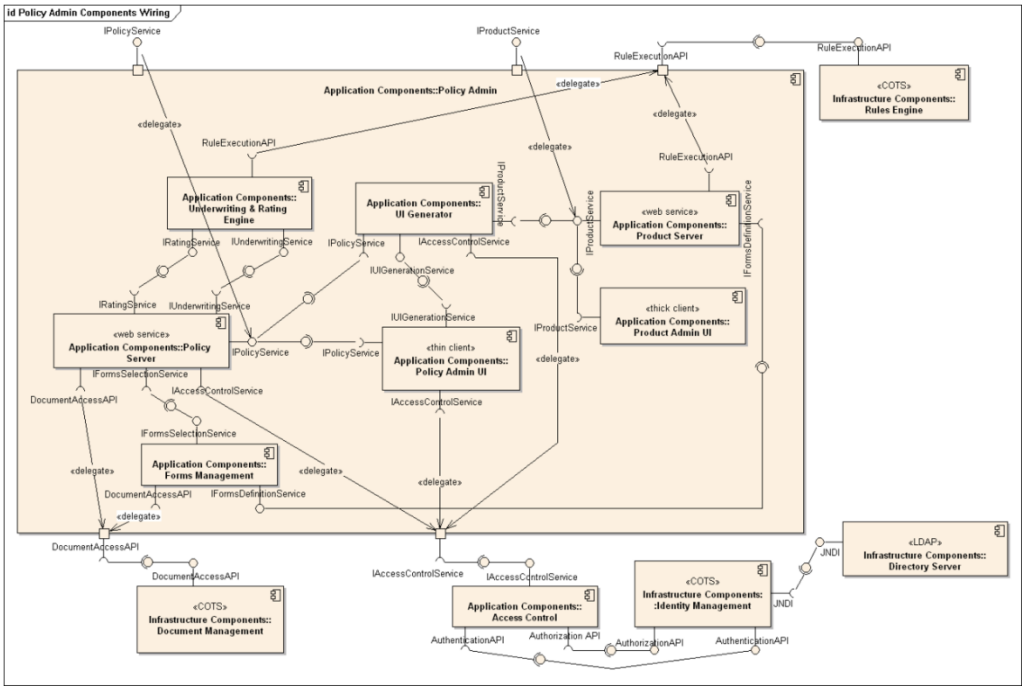


- Diagramme de Composants

  Le **diagramme de composants** décrit l'organisation du système du point de vue des éléments logiciels comme les modules (paquetages, fichiers sources, bibliothèques, exécutables), des données (fichiers, bases de données) ou encore d'éléments de configuration (paramètres, scripts, fichiers de commandes). Ce diagramme permet de mettre en évidence les dépendances entre les composants (*qui utilise quoi*).

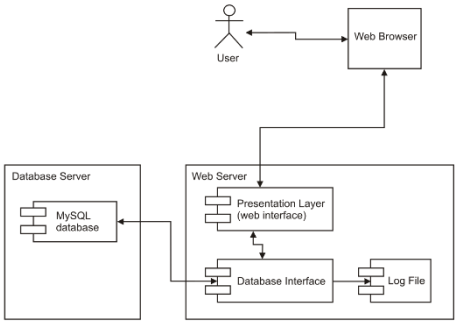
  Ex :

DOCUMENTATION SUR UML



- Diagramme de Déploiement

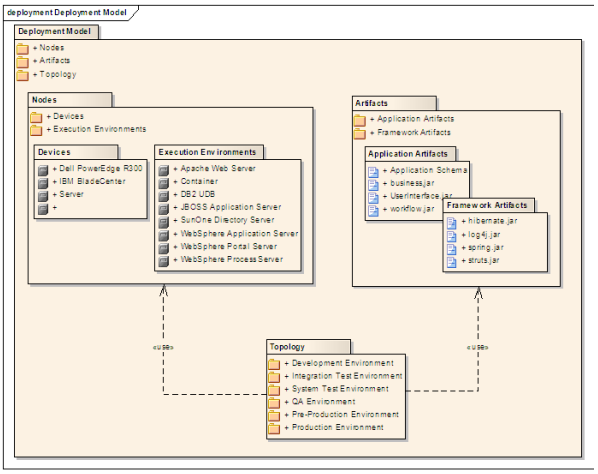
  En UML, un **diagramme de déploiement** est une vue statique qui sert à représenter l'utilisation de l'infrastructure physique par le système et la manière dont les **composants** du système sont répartis ainsi que leurs relations entre eux. Les éléments utilisés par un **diagramme de déploiement** sont principalement les **nœuds**, les **composants**, les **associations** et les **artefacts**. Les caractéristiques des ressources matérielles physiques et des supports de communication peuvent être précisées par stéréotype.



- Diagramme de Paquetages

DOCUMENTATION SUR UML

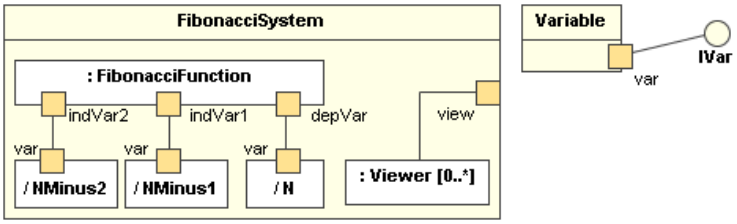
Les **diagrammes de paquetages** sont la représentation graphique des relations existant entre les paquetages (ou espaces de noms) composant un système, dans le langage Unified Modeling Language (UML).



- Diagramme de structure composite

Dans le langage UML, le **diagramme de structure composite** expose la structure interne d'une classe ainsi que les *collaborations* que cette dernière rend possible. Les éléments de ce diagramme sont les *parties* (en anglais *parts*), les *ports* par le biais desquels les parties interagissent entre elles, avec différentes instances de la classe ou encore avec le monde extérieur, et enfin les *connecteurs* reliant les parties et les ports.

Une *structure composite* est un ensemble d'éléments interconnectés collaborant dans un but commun lors de l'exécution d'une tâche. Chaque élément se voit attribuer un *rôle* dans la collaboration.

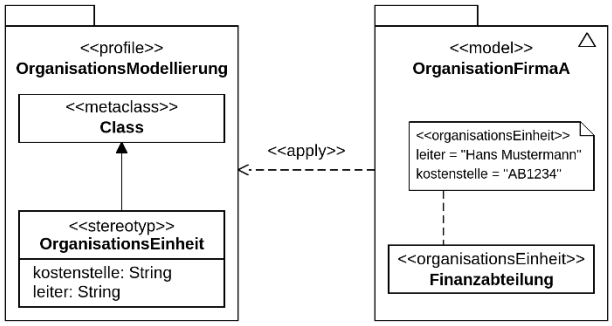


- Diagramme de Profils

En UML, un **diagramme de profils** est un diagramme de structure permettant l'utilisation de profils pour un métamodèle donné. Apparu avec UML 2.2, ce diagramme fournit une représentation des concepts

DOCUMENTATION SUR UML

utilisés dans la définition des profils (packages, stéréotypes, application de profils, etc.)

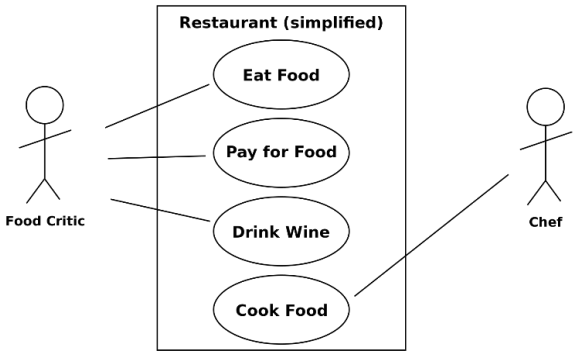


Diagrammes de comportement

Les **diagrammes de cas d'utilisation** (**DCU**) sont des diagrammes UML utilisés pour donner une vision globale du comportement fonctionnel d'un système logiciel. Ils sont utiles pour des présentations auprès de la direction ou des acteurs d'un projet, mais pour le développement, les cas d'utilisation sont plus appropriés. Un cas d'utilisation représente une unité discrète d'interaction entre un utilisateur (humain ou machine) et un système. Il est une unité significative de travail. Dans un diagramme de cas d'utilisation, les utilisateurs sont appelés acteurs (actors), ils interagissent avec les cas d'utilisation (use cases).

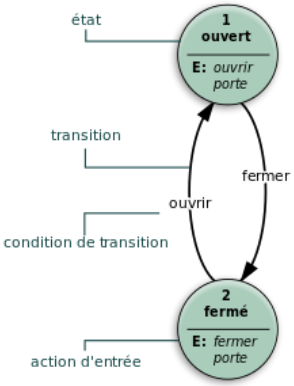
UML définit une notation graphique pour représenter les cas d'utilisation, cette notation est appelée diagramme de cas d'utilisation. UML ne définit pas de standard pour la forme écrite de ces cas d'utilisation, et en conséquence il est aisé de croire que cette notation graphique suffit à elle seule pour décrire la nature d'un cas d'utilisation. Dans les faits, une notation graphique peut seulement donner une vue générale simplifiée d'un cas ou d'un ensemble de cas d'utilisation. Les **diagrammes de cas d'utilisation** sont souvent confondus avec les cas d'utilisation. Bien que ces deux concepts soient reliés, les cas d'utilisation sont bien plus détaillés que les diagrammes de cas d'utilisation. Cela permet donc de comprendre qui est l'acteur et ce que le système doit réaliser.

DOCUMENTATION SUR UML



- Diagramme états-transitions

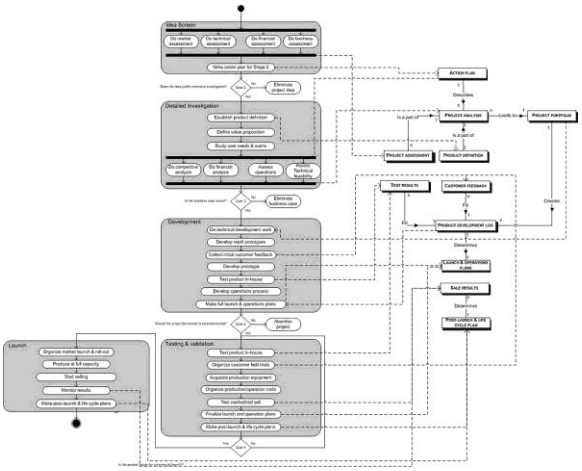
Un **diagramme états-transitions** est un schéma utilisé en génie logiciel pour représenter des automates déterministes. Il fait partie du modèle UML et s'inspire principalement du formalisme des statecharts et rappelle les grafcets des automates. S'ils ne permettent pas de comprendre globalement le fonctionnement du système, ils sont directement transposables en algorithme. En effet, contrairement au diagramme d'activité qui aborde le système d'un point de vue global, le diagramme états-transitions cible un objet unique du système. Tous les automates d'un système s'exécutent parallèlement et peuvent donc changer d'état de façon indépendante.



- **Diagramme d'activité**

Le **diagramme d'activité** est un diagramme comportemental d'UML, permettant de représenter le déclenchement d'événements en fonction des états du système et de modéliser des comportements parallélisables (multi-threads ou multi-processus). Le diagramme d'activité est également utilisé pour décrire un flux de travail (workflow).

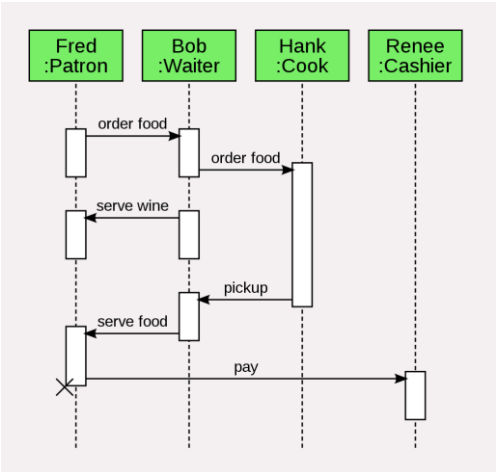
DOCUMENTATION SUR UML



Diagrammes d'interaction ou diagrammes dynamiques

- **Diagramme de Séquences**

Les **diagrammes de séquences** sont la représentation graphique des interactions entre les acteurs et le système selon un ordre chronologique dans la formulation Unified Modeling Language.



- Diagramme de Communication

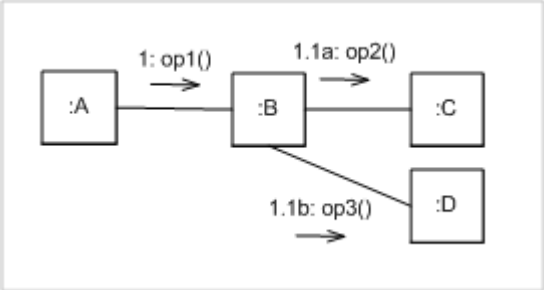
Un **diagramme de communication** est un diagramme d'interactions UML 2.0 (appelé **diagramme de collaboration** en UML 1), représentation simplifiée d'un diagramme de séquence se concentrant sur les échanges de messages entre les

DOCUMENTATION SUR UML

objets. En fait, le diagramme de séquence et le diagramme de communication sont deux vues différentes mais logiquement équivalentes (on peut construire l'une à partir de l'autre) d'une même chronologie, ils sont dits isomorphes.

C'est une combinaison entre le diagramme de classes, celui de séquence et celui des cas d'utilisation. Il rend compte à la fois de l'organisation des acteurs aux interactions et de la dynamique du système.

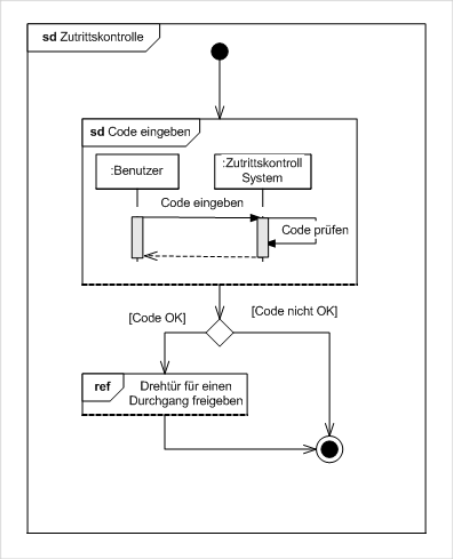
C'est un graphe dont les nœuds sont des objets et les arcs (numérotés selon la chronologie) les échanges entre objets.



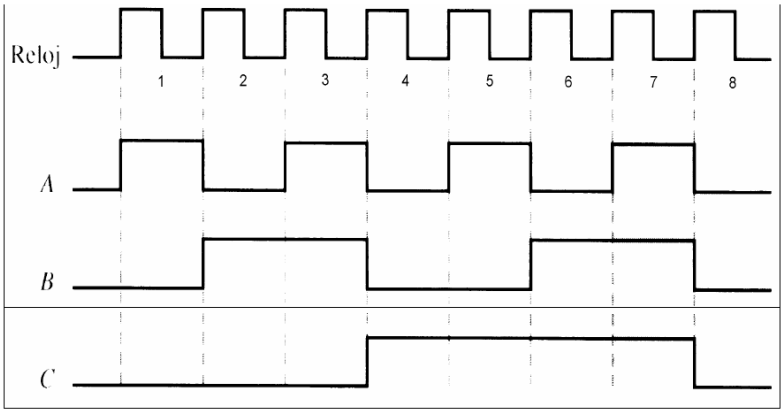
- Diagramme globale d'intéraction

Le **Diagramme global d'interaction** ou **diagramme d'interactivité** est un diagramme UML version 2.0 utilisé pour rendre compte de l'organisation spatiale des participants à l'interaction.
Les diagrammes globaux d'interaction définissent des interactions par une variante des diagrammes d'activité, d'une manière qui permet une vue d'ensemble de flux de contrôle. Ils se concentrent sur la vue d'ensemble de flux. de contrôle où les nœuds sont des interactions ou InteractionUses. Les lignes de vie et les messages n'apparaissent pas à ce niveau de vue d'ensemble.

DOCUMENTATION SUR UML



- Diagramme de Temps



3) Modèles d'éléments

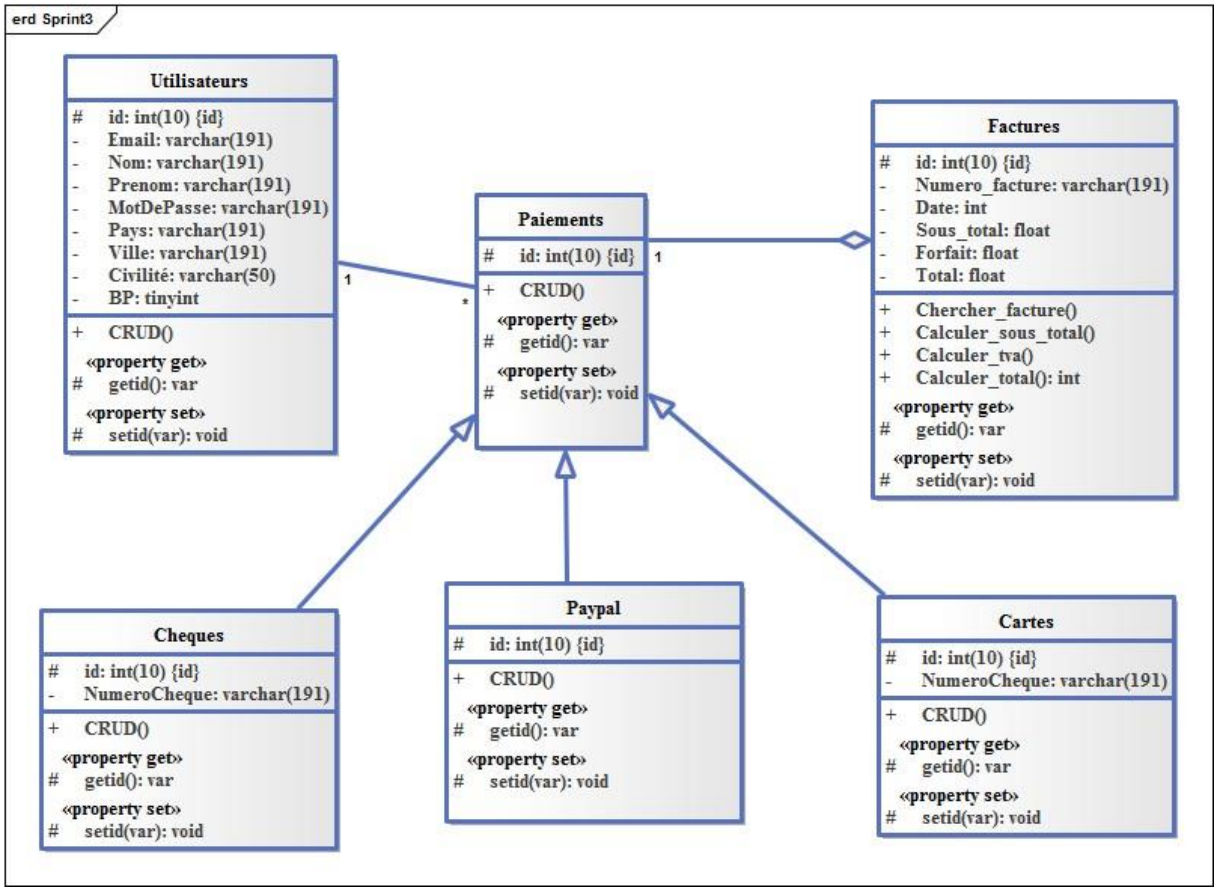
- Un *stéréotype* est une marque de généralisation notée par des guillemets, cela montre que l'objet est une variété d'un modèle.
- Un *classeur* est une annotation qui permet de regrouper des unités ayant le même comportement ou structure. Un classeur se représente par un rectangle conteneur, en traits pleins.
- Un *paquet* regroupe des diagrammes ou des unités.

DOCUMENTATION SUR UML

- Chaque classe ou objet se définit précisément avec le signe « :: ». Ainsi l'identification d'une classe X en dehors de son paquet ou de son classeur sera définie par « Paquet A::Classeur B::Classe X ».

Grâce aux outils de modélisation UML, il est également possible de générer automatiquement tout ou partie du code d'une application logicielle, par exemple en langage Java, à partir des divers documents réalisés.

Exemple de Modélisation avec UML :



Donc on peut en déduire que MERISE est une méthode de modélisation des données alors qu'UML est un langage.

Cette méthode reste adaptée pour la gestion des projets internes aux organisations, se limitant à un domaine précis.

DOCUMENTATION SUR UML

Elle est en revanche moins adaptée aux projets transverses aux organisations, qui gèrent le plus souvent des informations à caractère sociétal (environnemental et social) avec des parties prenantes.

III. Différences entre MERISE et UML

MERISE est une méthode de conception de Base de Données alors qu'UML est un langage de modélisation de Base de Données orientés objet.