Positionnement

- UML est un langage de modélisation. Ce n'est pas une méthode il manque la démarche
- UML n'est pas propre à l'approche objet (part importante des concepts et des outils est empruntés aux approches classiques)

LA MODELISATION OBJET AVEC UML

- Les langages de modélisation orientés objets sont apparus au milieu des années 1970 et jusqu'à la fin des années 1980 comme des méthodes expérimentales. Les méthodes orientées objets sont passées de moins de 10 jusqu'à 50 pendant les années 1989-1994.
- A partir des années 1990, une nouvelle mouture de méthodes est apparue et en particulier, la méthode OOD (Object Oriented Design) et OMT (Object Modeling Technique). Puis a émergé OOSE (Object Oriented Software Engineering). Chacune de ces méthodes était complète et puissante. C'est durant l'année 1995 que les auteurs des méthodes OMT, Booch et OOSE ont terminé leurs travaux d'unification afin de créer un standard plutôt que d'évoluer chacun de son coté.
- UML (Unified Modeling Language) est donc né de la consolidation des trois méthodes objet. Cette consolidation a été marquée par trois étapes :
  - Le regroupement des trois équipes au sein d'une seule et même société Rational ;
  - Le recentrage du projet de standardisation sur le langage de modélisation, les aspects purement méthodologiques étant laissés de côté ;
  - La décision de l'Object Management Group de normaliser la version 1.1 de UML en Novembre 1997.

POSITIONNEMENT

- Comme son nom l'indique, UML (Unified Modeling Language) est un langage de modélisation. Ce n'est pas une méthode (il y manque la démarche), et UML n'est pas propre à l'approche objet (une part importante des concepts et des outils est empruntée aux méthodes classiques).
- Cette double limite explique la rapidité avec laquelle UML a été adopté par l'ensemble des acteurs concernés par le génie logiciel : UML fournit un cadre conceptuel commun mais laisse aux différents acteurs du marché toute liberté pour développer leurs méthodes et leurs outils.
- Cette unification porte aussi bien sur les méthodes objet que sur les méthodes classiques. En simplifiant, on peut y distinger trois composantes :
  - Renouvellement : c'est le cas pour les concepts utilisés pour la définition des activités.
  - Enrichissement : les concepts du modèle Entité/Relation « étendu » ont été conservés, mais la notion de type d'entité a été enrichie par les opérations pour obtenir un type d'objet.
  - Reprise : les concepts de l'analyse structurée pour représenter la dynamique des comportements ont été conservés à l'identique.

Schématiquement, UML peut être défini sur trois plans :

- Les concepts, dont la sémantique est complètement et formellement définie ;
- Les diagrammes, utilisés pour spécifier les besoins et les systèmes ;
- Un certain nombre de mécanismes d'extension, pour intégrer aux processus des stéréotypes spécifiques aux différentes méthodes et/ou contextes applicatifs.

UML

- Le langage graphique UML est le plus utilisé dans le monde, sa dénomination compléte est « The Unified Modeling Langage for Object-Oriented Developpement ». Il est directement issu de la méthode unifiée (Unified Method) qui s'est transformée en UML

UML est caractérisé par les points suivants qui permettent de mieux en comprendre la logique :

- Des concepts issus des langages de programmation
- Inspirée des modèles entitée-relation

Des concepts issus des langages de programmation

- Il s'agit de définir un langage graphique qui se veut indépendant des langages.
  Mais dans la mesure même où il tente d'en réaliser une synthèse, de n'en garder que l'essentiel on y retrouvera essentiellement des concepts langage.
  Bien que par endroit les termes utilisés s'efforcent, avec beaucoup de difficultés, d'être proche du réel.

Inspirée des modèles entitée-relation

- Ces modèles entitée-relation mis en place pour l'analyse et l'élaboration de système de gestion de bases de données ont tous une vision essentiellement statique des "choses". Ils s'intéressent à représenter des informations et leurs relations sans y associer les aspects dynamiques. Les relations envisagées sont le plus souvent symétriques et ne sont pas définies formellement.

Situer UML

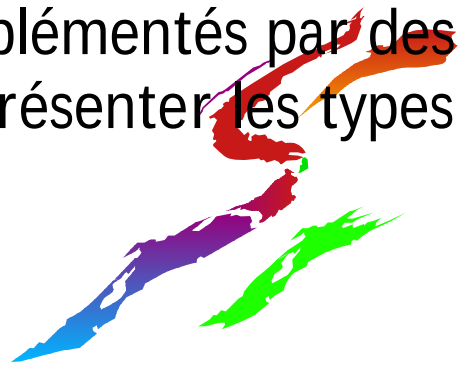
- Dans la mesure même où UML tente une synthèse de ce que l'on trouve dans différents langages Objets parmi les plus utilisés, il en est l'émanation, il est lui même un langage de type informatique.
- En réalité UML est fait pour permettre aux informaticiens de communiquer entre eux mais il n'est pas réellement accessibles aux autres faisant trop appel aux concepts des langages contrairement aux objets abstraits. UML passe ainsi à côté de l'opportunité qu'offre l'approche objet de communiquer en s'appuyant sur le réel. Pour les mêmes raisons le passage directe de l'objet réel à l'objet UML est problématique.

Apport d'UML

- UML préconise de séparer les ascpects
  - fonctionnels,
  - technologiques
  - architecturaux
- UML propose des diagrammes pour éclairer les spécifications.

Objets, types, classes et interfaces

- La modélisation des objets avec UML supporte la distinction entre les objets du domaine (entités) et les objets du système, ainsi que la distinction entre les types (description des comportements) et les classes (implémentation des types).
- UML peut ainsi supporter une approche classique (modèle Entité/Relation avec ou sans sous-types) ou une méthode objet (types caractérisés par des opérations). Le choix méthodologique se fera avec l'analyse des interactions, puisque alors les entités se verront attribuer des responsabilités, ce qui en fera des objets dont le comportement sera décrit par des types d'objets, eux-mêmes implémentés par des classes. La notion d'interface est utilisée pour représenter les types non implémentés.

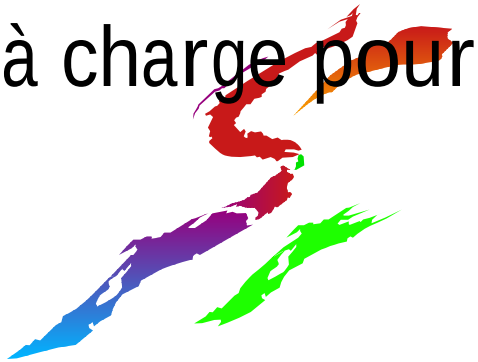


Relations, associations et compositions

- De même que les objets sont représentés par des types (ou classes), les liens entre les objets sont représentés par des relations. Lorsque la relation est caractérisée par des informations, on parle d'association. lorsque les relations sont structurelles, on parle de composition. Comme dans les méthodes classiques, les relations sont décrites par des rôles et par des cardinalités minimales et maximales.

Messages et opérations

- L'activité du système se réduit à l'activité des objets, qui se coordonnent par échange de messages. Pour qu'un objet traite un message il faut que celui-ci corresponde à une opération connue de l'objet, c'est-à-dire qu'elle soit définie par l'interface de la classe. A noter cependant que les messages et les opérations sont définis indépendamment les uns des autres, à charge pour l'analyste d'assurer la consolidation.



Etats, événements et transitions

- Pour modéliser la dynamique des états, UML reprend les concepts utilisés par les méthodes classiques : lorsque le comportement d'un objet est déterminé par un ensemble fini de situations, celles-ci sont représentées par des états. Ces états peuvent être définis de deux manières :
  - En termes d'activités : les états sont associés aux opérations exécutées par les objets, et les transitions associées aux événements,
  - En termes d'objets : les états sont définis en termes de conditions sur les attributs, et les transitions associées aux opérations.

Actions, activités et opérations

- La notion de temps (durée) est définie par domaine : c'est l'intervalle entre deux événements. Une action peut être instantanée pour un domaine (un seul événement) et avoir une durée pour un autre (deux événements distincts pour marquer le début et la fin). C'est généralement le cas lorsque l'on passe d'un domaine applicatif au domaine physique. Pour un domaine donné, on parlera d'opération pour une action instantanée, d'activité dans le cas général. La décomposition des états et des transitions joue un rôle essentiel pour modéliser les différents niveaux d'abstraction : une action sera représentée par une transition dans un domaine et par un état dans le contexte de l'objet qui l'implémente.

Paquetages et composants

- Un paquetage regroupe un ensemble d'objets logiques solidaires du point de vue du développement. Un composant est un élément physique du système qui implémente une ensemble d'interfaces. Un composant est l'équivalent physique d'une classe (ou de plusieurs).

Le contenu et la forme

- Un système peut être décrit sous différents aspects : ses composants, ses fonctionnalités, son comportement dynamique. Il faut donc construire différents modèles, dont il faudra vérifier la cohérence. Très souvent une même spécification pourra être également renseignée de plusieurs manières : graphique, textuelle, ou à l'aide d'un langage formalisé.

Cycle de vie

- Traçabilité
  - les concepts objets n'engendre plus de rupture entre les différentes étapes d'abstractions
- Itératif : évolutions successives
    - réduit les risques suivants :
      - méconnaissance des besoins
      - incompréhension
      - instabilité
- Incrémental : prototypes successifs

Technique de modélisation

- Cas d'utilisation (use cases)
- Cartes CRC (CRC Cards)
- Diagrammes de Classe
- Diagrammes d'interaction
- Diagrammes de transition d'états
- Diagrammes de paquetages
- Modèle By Contract

Les modèles

- La notion de modèle est souvent confondue avec la notion de vue, alors que ces deux notions renvoient à des logiques différentes
- Le modèle renvoie à un contenu : il constitue une spécification plus ou moins abstraite d'un domaine ou d'un système, réalisée avec un langage,
- La vue renvoie à une démarche : elle constitue une manière d'aborder les choses dans un contexte méthodologique donné.

Modèles traditionnels

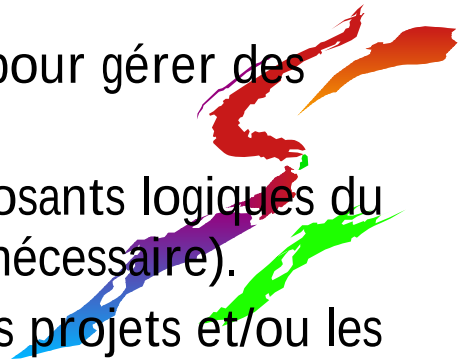
- Les méthodes classiques utilisent trois types de modèles : les modèles statique, dynamique et fonctionnel.
  - **Modèle statique.** Il représente la structure des informations gérées par le systume ainsi que les invariants. Il correspond au modèle Entité/Relation étendu. Il faut y spécifier les règles et les contraintes applicables en permanence.
  - **Modèle dynamique.** Il décrit le comportement des objets. Il s'appuie sur la vue statique pour décrire les effets des popérations sur les objets. Il faut y spécifier les règles de pilotage et les contraintes d'exécution.
  - **Modèle fonctionnel.** Il décrit les interactions du système. Il s'appuie sur la vue statique pour décrire les scénarios en termes de responsabilités et de collaborations. Il faut y spécifier les règles et les contraintes applicables dans le contexte des cas d'usage.

Modèles du processus « unifié » de Rational

- Bien qu'UML ne définisse pas formellement de modèles, un certain consensus existe, dont l'approche unifiée de Rational es assez représentative. Ce découpage sert essentiellement de référence et doit être adapté à chaque situation : nature du domaine, taille du projet, maturité des besoins et évolution.

Modèles du processus « unifié » de Rational

- **Cas d'usage.** La définition d'un modèle à partir des seuls cas d'usage se justifie dès lors que l'on souhaite gérer l'évolution des spécifications.
- **Modèle d'analyse.** lorsque les projets sont importants, l'organisation doit gérer séparément et parallèlement deux niveaux de modélisation : le premier, tourné vers le domaine, privilégie la lisibilité, l'exhaustivité et la cohérence ; le second, tourné vers le système, porte sur l'implémentation. Ainsi, le diagramme de classes sera utilisé pour construire les modèles d'analyse (types d'entités, d'interfaces, de contrôle) et de conception (les classes). De même, le diagramme de collaborations sera utilisé pour l'analyse et le diagramme de séquences pour la conception.
- **Modèle de conception.** Le modèle de conception est souvent construit avec les mêmes outils que le modèle d'analyse. Mais contrairement au précédent, les choix de modélisation sont liés à une architecture particulière.
- **Modèle de déploiement.** Ce type de modèles est utilisé pour gérer des environnements d'exploitation multiples et complexes.
- **Modèle d'implémentation.** Ce modèle spécifie les composants logiques du système et la manière dont ils sont obtenus (il est toujours nécessaire).
- **Modèle de tests.** Ce modèle est nécessaire lors des grands projets et/ou les projets à livraisons multiples.



Les Diagrammes

- Comme son nom l'indique, UML est fait pour construire des modèles. Pour ce faire, il suffit d'utiliser un certain nombre d'outils : les diagrammes. Quatre groupes de diagrammes existent regroupés selon la nature des éléments modélisés :
  - Cas d'usage pour modéliser les fonctionnalités des applications.
  - Classes pour modéliser les objets gérés par le système.
  - Interactions pour modéliser la manière dont les objets collaborent à la réalisation des cas d'usage.
  - Composants et déploiement pour modéliser les éléments logiques et physiques du système.

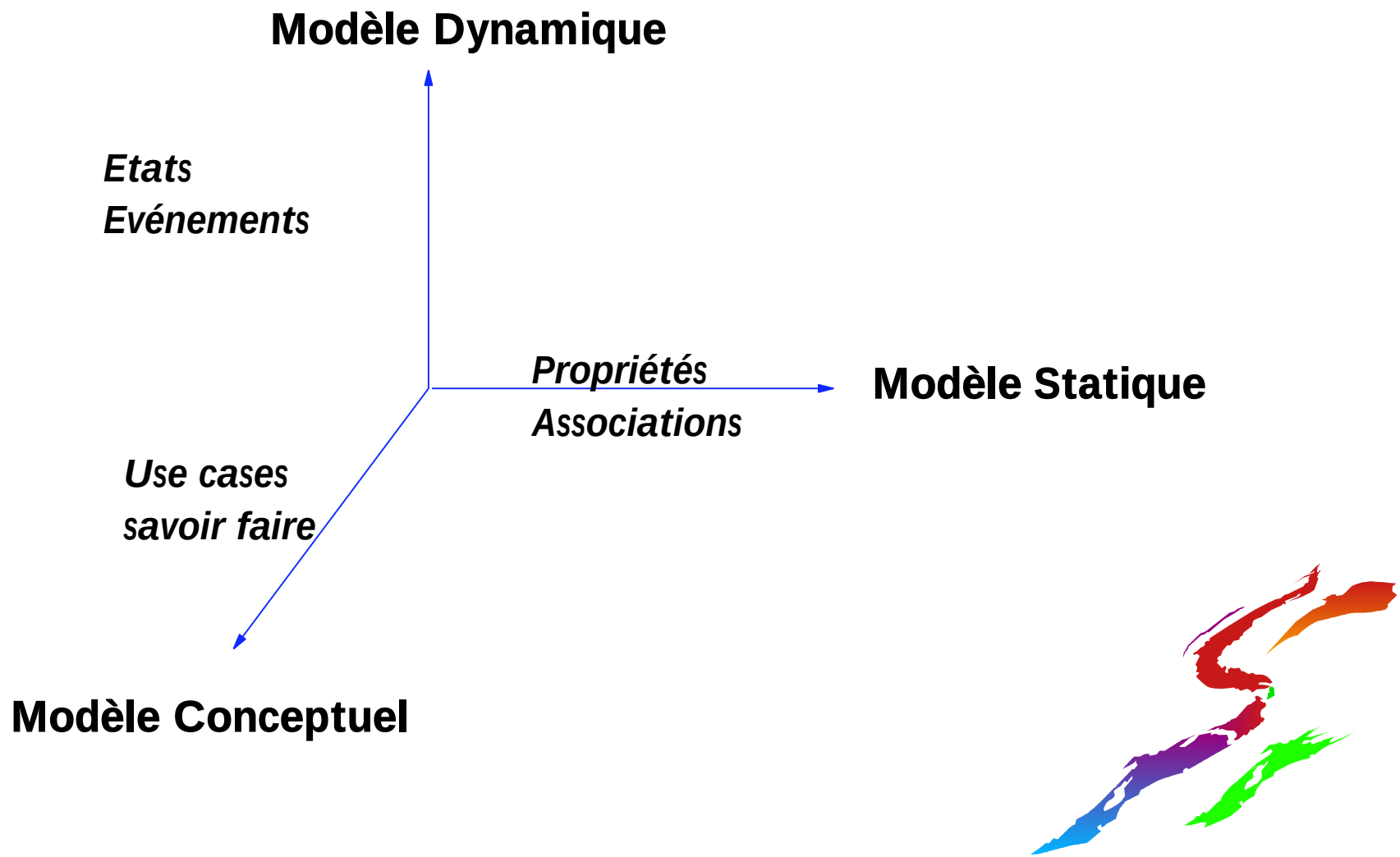
Diagrammes des cas d'utilisation (ou d'usage)

- Les diagrammes de cas d'utilisation se composent d'acteurs (représentés par des silhouettes) et des cas d'utilisation (représentés par des ellipses). Les traits entre les cas d'utilisation et les acteurs représentent les interactions.
- Ces diagrammes montrent les relations qui existent entre des acteurs et des fonctionnalités du système. Ils sont utilisés tout au long du processus de développement pour formaliser les comportements et la synchronisation des actions (souvent utilisés pour modéliser le processus de développement lui-même). Ils définissent de manière informelle les exigences auxquelles devra répondre le système.

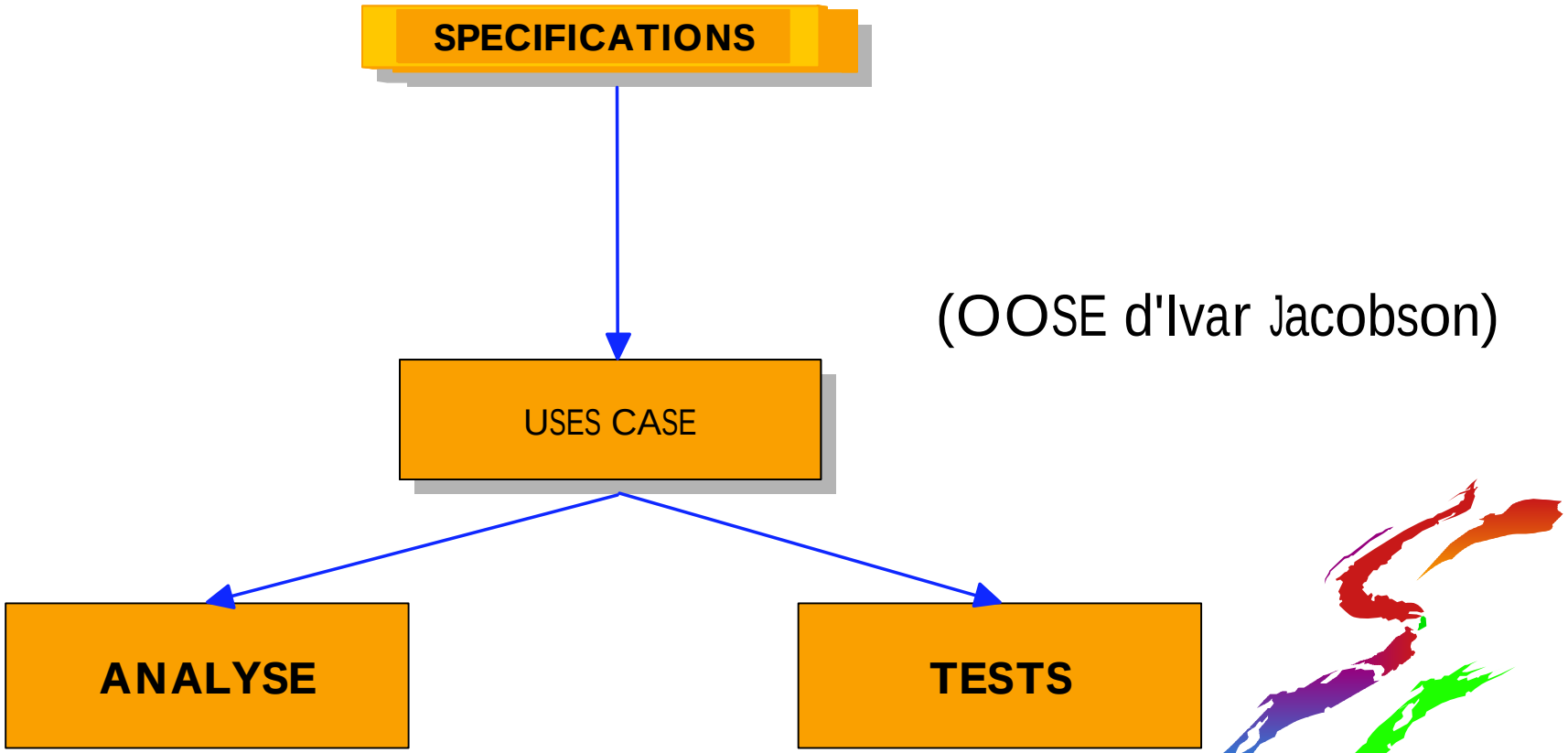
Les diagrammes

- Diagramme des activités
- Diagramme des cas d'utilisation
- Diagramme de classe
- Diagramme de collaboration
- Diagramme de composants
- Diagramme de déploiement
- Diagramme d'états de transitions
- Diagramme d'objets
- Diagramme de séquence (interaction)

Etapes pour élaborer les diagrammes

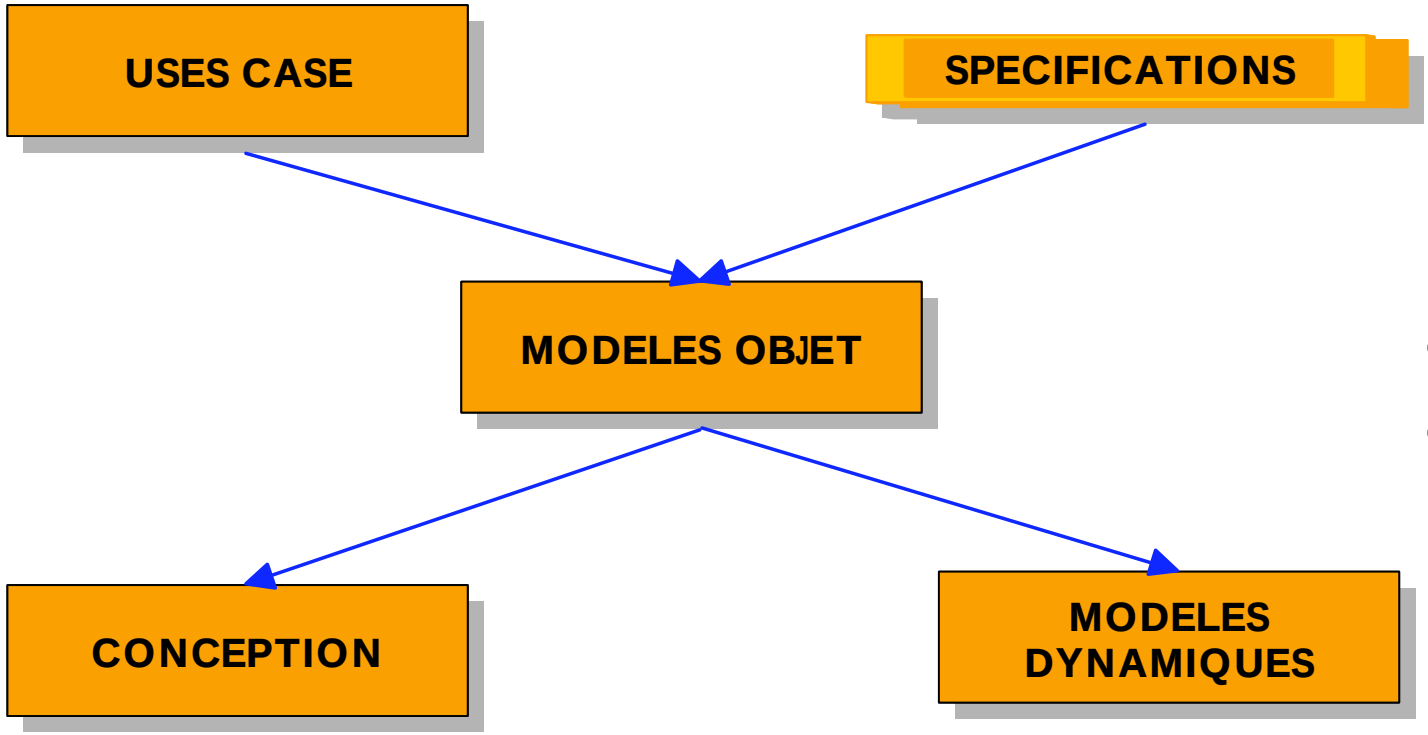


Les cas d'utilisation (vue fonctionnelle)



(OOSE d'Ivar Jacobson)

Le modèle objet (analyse statique)



Identifier les classes
associations
attributs
opérations

Diagrammes dynamiques

USES CASE

SPECIFICATIONS

MODELES OBJET

MODELES DYNAMIQUES

Diagramme états transition
Diagramme des scénarios
Diagramme de collaboration

CONCEPTION

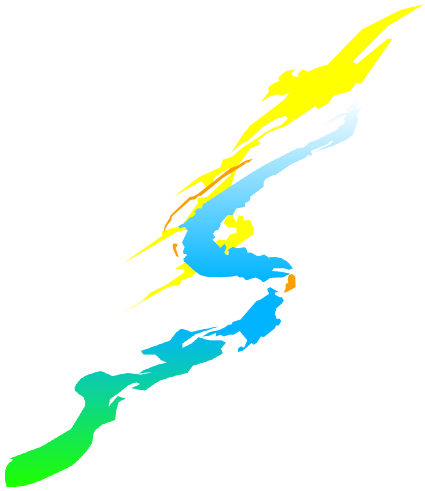
TESTS

Maintenance

Diagrammes

- Fonctionnalités du système : cas d'usage
- Objets du système : diagramme de classes
- Comportements et interactions
  - diagramme de collaboration
  - diagramme de séquences
  - diagramme d'états
  - diagramme d'activités
  - diagramme de flux
- Architecture du système composants et déploiement

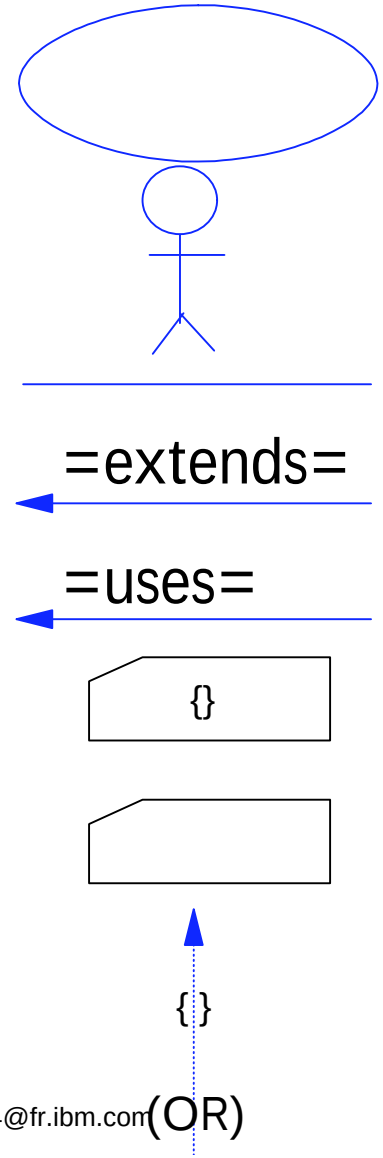
Cas d'utilisation



Diagrammes des cas d'utilisation (ou d'usage)

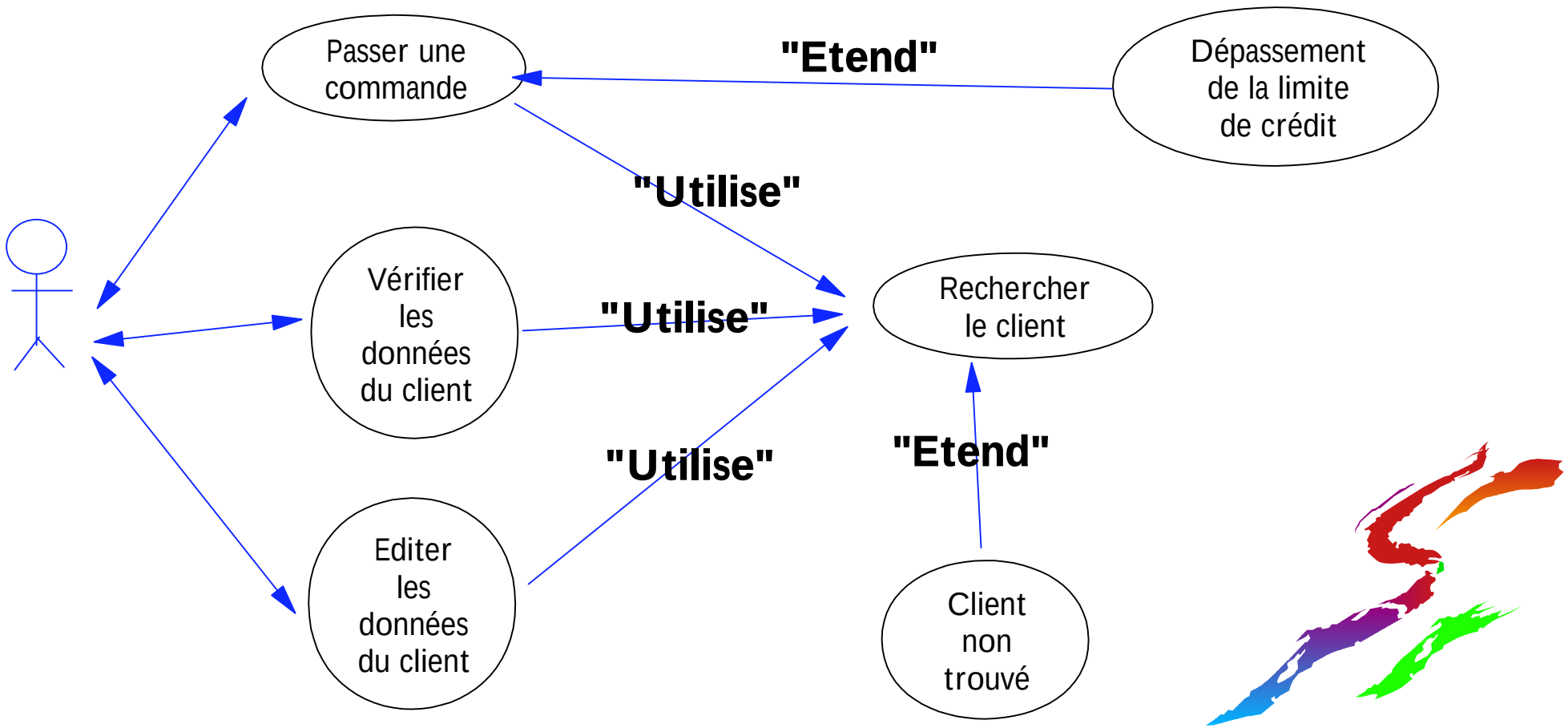
- Les diagrammes de cas d'utilisation se composent d'acteurs (représentés par des silhouettes) et des cas d'utilisation (représentés par des ellipses). Les traits entre les cas d'utilisation et les acteurs représentent les interactions.
- Ces diagrammes montrent les relations qui existent entre des acteurs et des fonctionnalités du système. Ils sont utilisés tout au long du processus de développement pour formaliser les comportements et la synchronisation des actions (souvent utilisés pour modéliser le processus de développement lui-même). Ils définissent de manière informelle les exigences auxquelles devra répondre le système.

Représentation des cas d'usage



- **Uses case** : unité cohérente de fonctionnalité par un système ou une classe
- **Actor** Acteur, personne ou système qui joue un rôle dans le système
- indique la participation d'un acteur à un cas d'utilisation
- Indique qu'une instance d'un cas d'utilisation peut inclure le comportement décrit par un autre
- Indique qu'une instance d'un cas d'utilisation contient le comportement décrit par un autre
- Représente une relation sémantique des éléments d'un modèle spécifiant des propositions et conditions qui doivent vérifiées
- Commentaires additionnels placés sur un diagramme
- Représente une relation sémantique entre les relations d'un modèle spécifiant des propositions et conditions qui doivent vérifiées

Exemple (Programmez no 15 11/1999)



Ce que font les cas d'utilisation

- Ils maintiennent les exigences fonctionnelles sous une forme facile à lire facile à maintenir
- Ils représentent le but d'une interaction entre acteur et système. Ce but est l'image d'un objectif compréhensible et mesurable pour l'acteur
- Ils enregistrent le scénario que joue un acteur depuis un point de départ (événement déclencheur) j'usqu'au but à atteindre
- Ils mémorisent le scénario que joue un acteur depuis un point de départ (événement déclencheur) j'usqu'à un point d'échec

Utilisation des cas d'utilisation

- Capturer les exigences d'un système
- agir comme support de conception logicielle
- utilisés lors des étapes de validation
- utilisés lors des étapes de tests et d'assurance Qualité
- synopsis initiaux de l'aide en ligne et du manuel utilisateur