COURS

ANALYSE ET CONCEPTION DES SYSTÈMES D'INFORMATION AVEC L'APPROCHE ORIENTÉE OBJET UML UNIFIED MODELING LANGAGE

Dr. Ilhem ABDELHEDI ABDELMOULA

Email : ilhemabdelmoula13@gmail.com

Université de Carthage – **Enicarthage** École Nationale des Ingénieurs à Carthage

Département: Informatique **Niveau : 1ère année / 2ème année**

INFO ING **Semestre : 2** **Année universitaire: 2019 - 2020**

OBJECTIFS DU COURS

- Importance de la modélisation
- Les méthodes d'analyse et de conception des systèmes d'information
  - Définitions
  - Types des méthodes d'ACSI
  - Evolution – unification
- Présentation générale d'UML – Unified Modeling Language
  - Modélisation objet avec UML
  - Diagrammes UML et leurs utilités
- Utiliser UML dans un contexte de projet informatique, en se penchant plus particulièrement sur les étapes d'analyse de besoins et de conception.

PLAN

1. **Introduction à l'UML**
2. **Diagrammes des cas d'utilisation**
3. **Diagrammes de classes et diagrammes d'objets**
4. **Diagrammes de séquences système et de collaboration**
5. **Diagrammes d'états-transition**
6. **Diagrammes d'activités**
7. **Diagrammes de séquences objet**
8. **Diagrammes de composants et de déploiement**

Semaines	Séance 1	Séance 2
Semaine 1	Chapitre 1 – Introduction UML	Chapitre 2 - DCU
Semaine 2	TD 1 – DCU	Chapitre 3 DCL
Semaine 3	TD 2 – DCL	TD 2 – DCL
Semaine 4	Chapitre 4 – DSEQ	TD 3 – DSEQ
Semaine 5	TD 3 – DSEQ	Chapitre 5 – DCOL
Semaine 6	TD 4 – DCOL	Révision – DS
Semaine 7	Semaine des DS	
Vacances de printemps		
Semaine 8	Ferié	Correction DS
Semaine 9	Chapitre 6 – DET	TD 5 – DET
Semaine 10	Chapitre 7 – DACT	TD 6 – DACT
Semaine 11	Chapitre 8 – DSEQ Objet	TD 7 – DSEQ Objet
Semaine 12	TD 7 – DSEQ Objet	Chapitre 9 – DCOMP/DEP
Semaine 13	TD 8 – DCOMP/DEP	Révision – Examen

OUVRAGES/DOCUMENTS UTILISÉS POUR PRÉPARER CE COURS

- Modélisation et conception orientées objet avec UML2 de Michael Blaha et James Rumbaugh, 2ème édition, Pearson Education France, 2005 - Traduction de l'ouvrage Applying Object-Oriented Modeling and Design with UML, Prentice Hall 2005
- The Unified Modeling Language Reference Manual, 2nd Edition de James Rumbaugh, Ivar Jacobson et Grady Booch, Addison Wesley Professional, 2004 - UML 2.0, Guide de Référence, CampusPress
- Le guide de l'utilisateur UML de Grady Booch, James Rumbaugh et Ivar Jacobson, Eyrolles, 2000
- The Unified Modeling Language User Guide; Addison-Wesley, 1998
- Transparents de cours de Robert Ogor : http://www-inf.int-evry.fr/COURS/IO21/COURS/CoursENSTBr.pdf ③ Transparents de cours de Marie-José Blin

RÉFÉRENCES BIBLIOGRAPHIQUES

- J. Gabay, *Merise. Vers OMT et UML*, InterÉditions, 1998.
- N. Kettani, D. Mignet, P. Paré et C. Rosenthal-Sabroux, *De MERISE à UML*, Eyrolles, 1998 · M. Lai, *Penser objet avec UML et Java*, InterÉditions, 1998.
- M. Lai, *UML : La notation unifiée de modélisation objet – De Java aux EJB*, Dunod, 2000
- N. Lopez, J. Migueis et E. Pichon, *Intégrer UML dans vos projets*, Eyrolles, 1997.
- C. Morley, B. Leblanc et J. Hugues, *UML pour l'analyse d'un système d'information – Le cahier des charges du maître d'ouvrage*, Dunod, 2000.
- P.-A. Muller, *Modélisation objet avec UML*, Eyrolles, 1998
- P. Roques et F. Vallée, *UML en action – De l'analyse des besoins à la conception en Java*, Eyrolles, 2000.
- C. Soutou, *Objet-Relationnel sous Oracle8, Modélisation avec UML*, Eyrolles, 1999
- UML 2 par la pratique : Études de cas et exercices corrigés de Pascal Roques, 4ème édition, Eyrolles, 2005
- UML 2 de Benoît Charroux, Aomar Osmani, et Yann Thierry-Mieg, coll. Synthex, Pearson Education, 2005

CHAPITRE 1 :

INTRODUCTION À UML
UNIFIED MODELING LANGUAGE

Imagination is more important than knowledge

(A. Einstein)

QUELQUES EXEMPLES DE MODÈLES

- **Modèles architecturaux** : peut par exemple permettre de voir le plan d'une maison, d'un appartement
- **Modèle de design** : peut par exemple permettre de voir le plan d'une voiture, d'une table, d'une chaise, etc..
- **Modèle économique** : peut par exemple permettre de simuler l'évolution de cours boursiers en fonction d'hypothèses macro-économiques (évolution du chômage, taux de croissance...).
- **Modèle démographique** : définit la composition d'un panel d'une population et son comportement, dans le but de fiabiliser des études statistiques, d'augmenter l'impact de démarches commerciales, etc...
- **Modèle météorologique** : à partir de données d'observation (satellite ...), permet de prévoir les conditions climatiques pour les jours à venir.

POURQUOI MODÉLISER ?

- La modélisation est une pratique **indispensable** pour le développement des systèmes d'informations
- Facilite la compréhension du fonctionnement d'un système **avant sa réalisation** et **anticiper** les résultats du codage.
- Sert à **réduire la complexité** d'un système en éliminant les détails qui n'influencent pas son comportement de manière significative
- **Sert de vecteur de communication** précis, connu par tous les membres de l'équipe de développement

QU'EST CE QU'UN MODÈLE?

- Une abstraction pour comprendre un problème de la vie réelle avant de mettre en œuvre une solution =>Représentation **abstraite et simplifiée** (excluant les détails) d'une entité (phénomène, processus, système, etc.)
- Identifie les **caractéristiques intéressantes** d'une entité, retenues par un observateur, en vue d'une utilisation précise, en vue ***de le décrire, de l'expliquer ou de le prévoir***
- Définit **une vue subjective mais pertinente :** reflète ce que l'observateur croit important pour la compréhension et la prédiction du système modélisé => Les limites du modèle dépendent de son utilité

MÉTHODES D'ANALYSE ET DE CONCEPTION DES SYSTÈMES D'INFORMATION (ACSI)

- Besoin d'une **approche méthodique** afin de respecter les délais de réalisation et les contraintes matérielles et humaines. A partir des années '70 plusieurs méthodes ont vu le jour:
  - SADT ('70),
  - Merise, puis Merise/2 ('70),
  - IEM ('80),

  fournissant des **méthodologies et des notations standards** qui aident à concevoir des logiciels de qualité.

DIVERSE MÉTHODES ACSI (70 – 80)

1. **Méthodes orientées comportement :** on s'intéresse à la dynamique du système
   Ex : réseaux de Pétri
2. **Méthodes fonctionnelles :** s'inspirent de l'architecture des ordinateurs où on s'intéresse aux fonctions du système
   Ex : SADT, IEM
3. **Méthodes orientées données :** on ne s'intéresse pas aux traitements
   Ex : MERISE

LES MÉTHODES ACSI ORIENTÉE OBJET

Une **2ème évolution fin les années 90** autour des idées:

- **Regroupement données et traitements** : Diminution de l'écart entre le monde réel et sa représentation informatique
- Le monde est avant tout **objet**
  - Localisation des responsabilités : encapsulation
  - Décomposition par identification des relations entre objets
- **Nouvelles architectures logicielles fondées sur les objets** du système où les diverses fonctionnalités du système émergent des diverses interactions entre les différents objets qui le constituent.

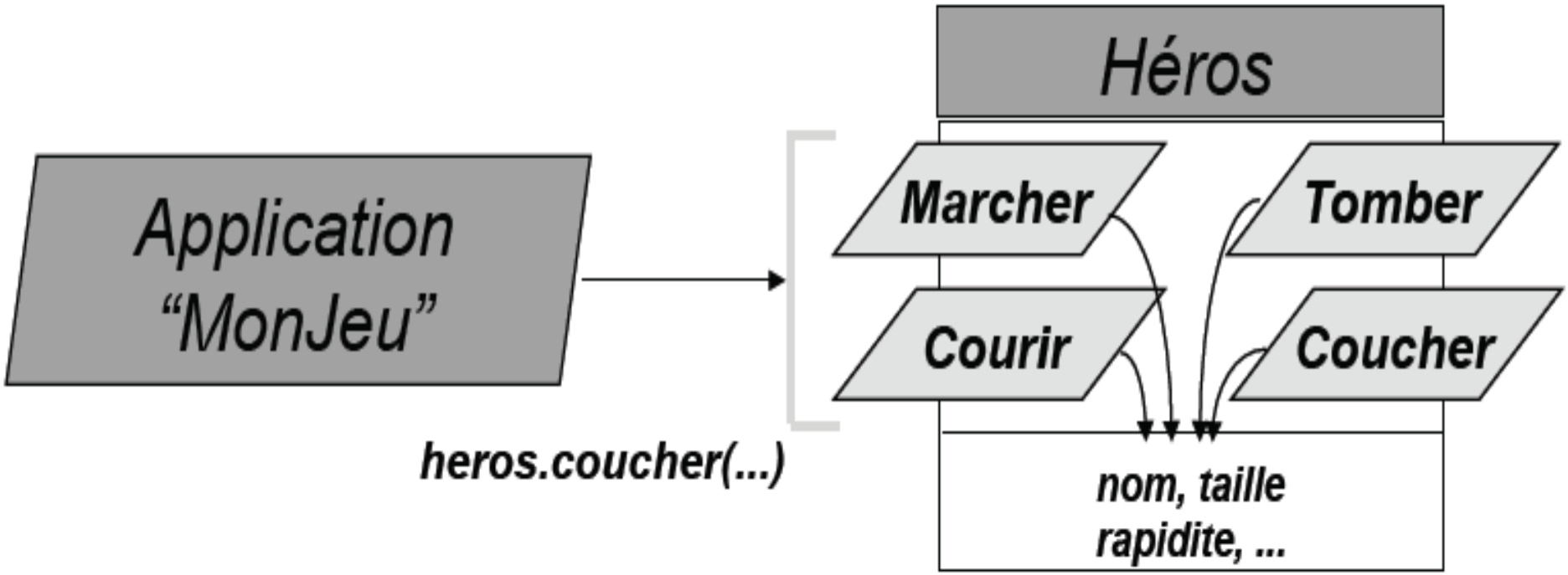
FONDEMENTS DES METHODES O.OBJET

- **Abstraction** : concentration sur ce que représente un objet et sur son comportement avant de décider de la façon de l'implémenter
- **Encapsulation :** masquage de l'information par la séparation des aspects externes d'un objet accessibles aux autres objets et des détails de l'implémentation cachés aux autres objets
- **Regroupement des données et du comportement** : polymorphisme : transfert de la décision de quelle méthode utiliser à la hiérarchie de classes
- **Partage :** héritage : possibilité de partager des portions de code commun et clareté conceptuelle (mettre en valeur le traitement commun)
- **Mise en évidence de la nature intrinsèque des objets**: sur ce qu'est un objet et non sur la façon dont il est utilisé
- **Synergie:** combinaison des concepts d'identité, classification, héritage et polymorphisme pour former un langage objet.

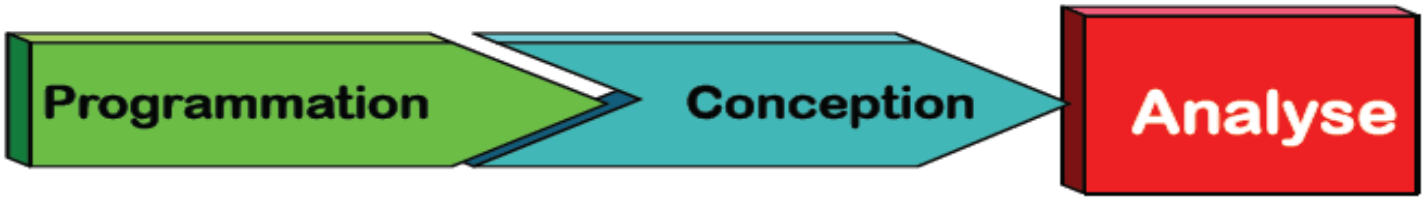
LA COMPLEXITÉ DES LOGICIELS

- Le logiciel est complexe par nature => gérer cette complexité
- Les systèmes peuvent être décomposés selon
  - ce qu'ils font (approche fonctionnelle)
  - ce qu'ils sont (approche objet)
- L'approche objet gère plus efficacement la complexité

EXEMPLE


Héros
Application
"MonJeu"
Marcher
Tomber
Courir
Coucher
heros.coucher(...)
nom, taille
rapidite, ...

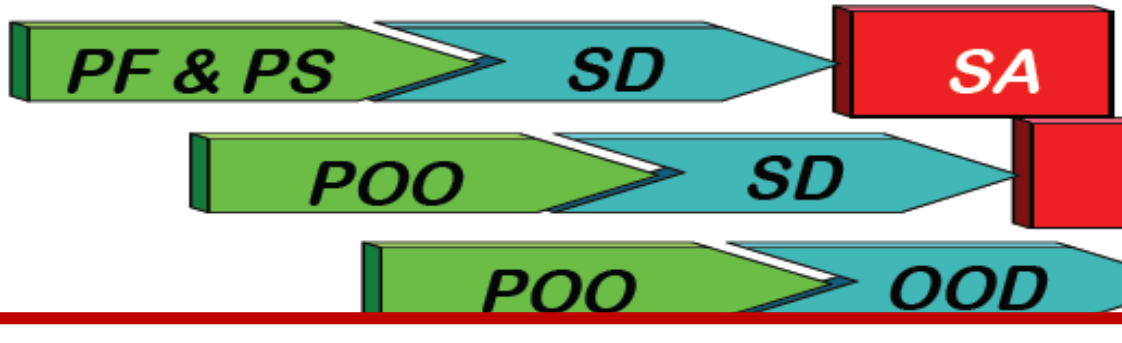
2ÈME ÉVOLUTION DES MÉTHODES



L'évolution des méthodes s'est faite de la programmation vers l'analyse

PF & PS : Prog. fonctionnelle et structurée
SD : Conception structurée
SA : Analyse structurée

POO : Programmation orientée objet
OOD : conception objet
OOA : Analyse objet



Les méthodes objet couvrent l'ensemble du cycle logiciel

AVANTAGES DES MÉTHODES ACSI O.O

- Réutilisation (code et conception)
- Langage haut niveau facilite l'analyse (description du problème), la conception (description de la solution), et l'implémentation
- Représentation de concepts abstraits (vues graphiques) couvrant tous les aspects d'un système
- Offre une assise théorique et des façons de modéliser des traitements qui seront ensuite codés en un langage orienté objet.
- Gère plus efficacement la complexité des SI
- Description de l'architecture complexe à base de composants distribués et hétérogènes.

EXEMPLES DE MÉTHODES C.O.O

1. **OMT (Object Modeling Technique)** *1991 de James Rumbaugh (General Electric).*

   La méthode fournit une représentation graphique des aspects statique, dynamique et fonctionnel d'un système. Elle vise à représenter un système comme un assemblage d'éléments auxquels on attache des comportements. Un élément (un composant) contient les informations et encapsule les actions qui peuvent être exécutées à la réception d'un message.

2. **OOD (Object Oriented Design)** *1991 de G. Booch*

   définie pour le *Department of Defense*, introduit le concept de paquetage (*package*). Elle est conçue pour préparer de façon rigoureuse la structuration de programmes en ADA et C++ ; proche de la programmation. Son auteur recommande de s'appuyer, pour la phase d'analyse, sur des méthodes générales type SADT.

3. **OOSE (Object Oriented Software Engineering) 1991** d'Ivar Jacobson (Ericsson)

   fonde l'analyse sur la description des besoins des utilisateurs (cas d'utilisation ou *use case*). L'analyse repose sur l'expression dont l'utilisateur

La guerre des méthodes ne fait plus avancer la technologie des objets

SOLUTION : UNIFICATION DES MÉTHODES

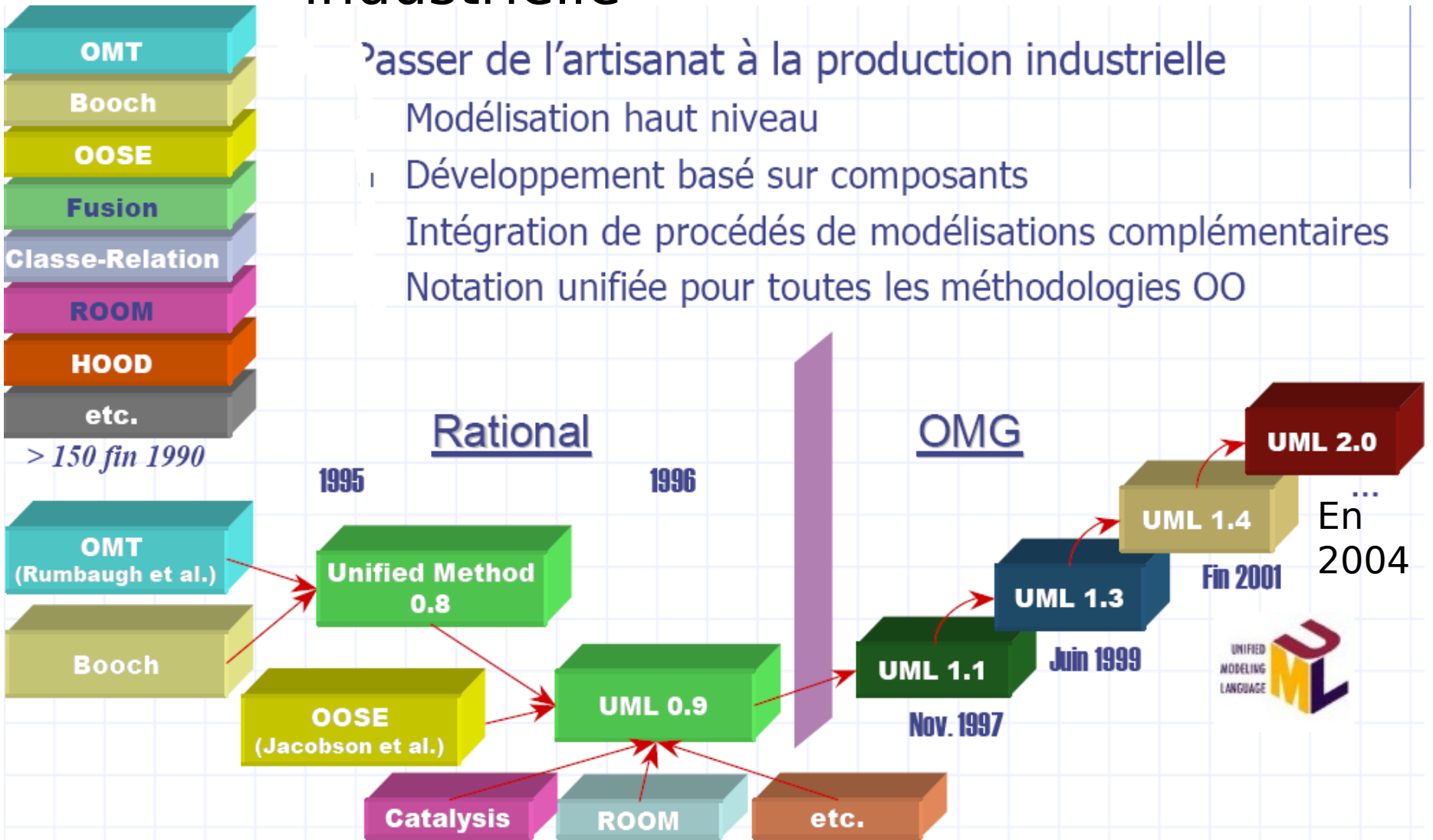
- Recherche d'un **langage commun unique**
  - Utilisable par toutes les méthodes
  - Adapté à toutes les phases du développement
  - Compatible avec toutes les techniques de réalisation
- sur **plusieurs domaines d'applications**
  - Logiciels => Ingénierie des logiciels
  - Logiciels et matériels => Ingénierie des systèmes
  - Personnes => Ingénierie des affaires

Intérêt d'un standard de modélisation universel

UML 2.0 UNIFIED MODELING LANGUAGE

OBJECT MANAGEMENT GROUP

- Passer de l'artisanat à la production industrielle



UML – UNIFIED MODELING LANGU

- Langage unifié pour la modélisation objet
- Langage de modélisation des applications construites à l'aide d'objets, indépendant de la méthode utilisée

- Différence Langage – Méthode
  - Langage de modélisation = notations, grammaire, sémantique
  - Méthode : comment utiliser le langage de modélisation (recueil des besoins, analyse, conception, mise en œuvre, validation…)
  - Objet = représentation du problème basée sur des entités

Langage standard de modélisation des systèmes d'information permettant de représenter des modèles, mais il ne définit pas le processus d'élaboration des modèles

UML : LANGAGE DE MODÉLISATION OBJET

- Un langage pas une méthode : définit des modes de représentation (diagrammes et notations) mais n'impose pas de démarche standardisée

  Langage = syntaxe + sémantique

  - Syntaxe : Règles selon lesquelles les éléments du langage (ex. les mots) sont assemblés en des expressions (ex. phrases, clauses).
  - Sémantique : Règles permettant d'attribuer une signification aux expressions syntactiques

- Documente toutes les phases du développement (analyse, conception et implantation)

PORTÉE

- universel : indépendant du langage d'implémentation/ domaine d'application/ processus…

⇒Convient pour toutes les démarches et langages de programmation objet

- Formalisme unique pour tout type d'application
  - gestion, scientifique, temps réel, industrielle, multimédia…
- Reste au niveau d'un langage
  - ne propose pas un processus de développement
  - ni ordonnancement des tâches,
  - ni répartition des responsabilités,
  - ni règles de mise en œuvre

LES BASES DE LA MODÉLISATION O.O

Les modèles sont les parties distinctes de la description du système mais interdépendantes

Deux dimensions sont associées à un modèle :

1. Une vue d'un système (modèle de classes, d'états ou d'interactions)
2. Un stade de développement(analyse, conception ou implémentation)

Trois modèles les plus importants en UML :

3. **Modèle de classes** : le plus fondamental décrit les aspects orientés données du système.

« Il est nécessaire de décrire ce qui change ou se transforme avant de décrire quand et comment les changements ont lieu »

2. Modèle d'états de transitions : aspects temporels, comportementaux et de contrôle du système

3. Modèle d'interactions : collaboration entre objets.

Un seul aspect du système traité par chaque modèle, mais relations entre les trois modèles.

Variation du poids des modèles en fonction du problème posé

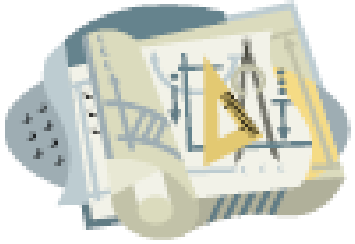
TROIS MODÈLES PRINCIPAUX

- **Modèle de classes**
  - Description de la structure statique des objets du système et de leurs relations
  - **Diagramme de classes** : graphe avec pour sommets les classes et pour arcs les relations entre les classes
- **Modèle d'états**
  - Description des états successifs d'un objet au cours du temps
  - **Diagramme d'états** : graphe avec pour sommets les états et pour arcs les transitions entre états déclenchées par des événements
- **Modèles d'interactions**
  - Description de la manière de coopérer des objets pour obtenir un résultat
  - **Cas d'utilisation** axé sur une fonctionnalité
  - **Diagramme de séquence** : représentation des interactions entre objets et ordonnancement des interactions
  - **Diagramme d'activités** : détails des étapes importantes du traitement

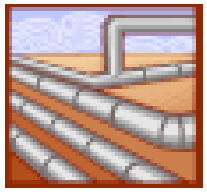
UML : un langage de modélisation objet

- modèle : *simplification* de la réalité dont les buts sont

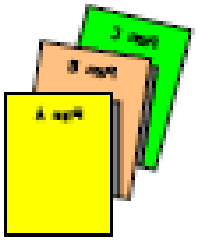
Visualiser le système



Spécifier la **structure et le comportement** du système



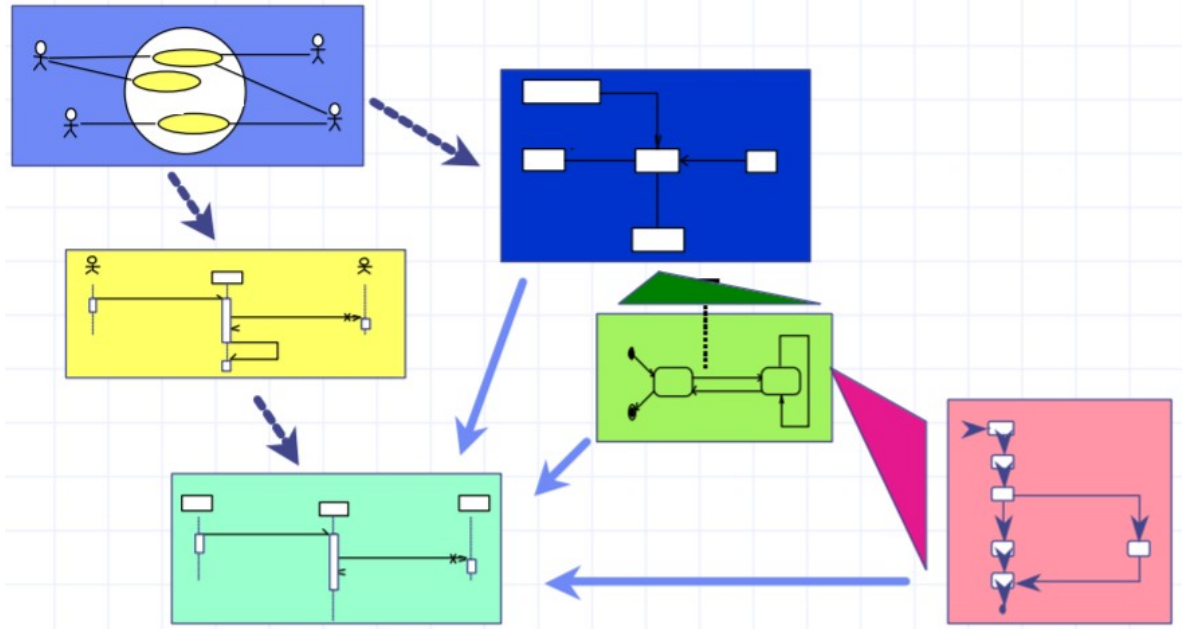
Aider à la construction du système



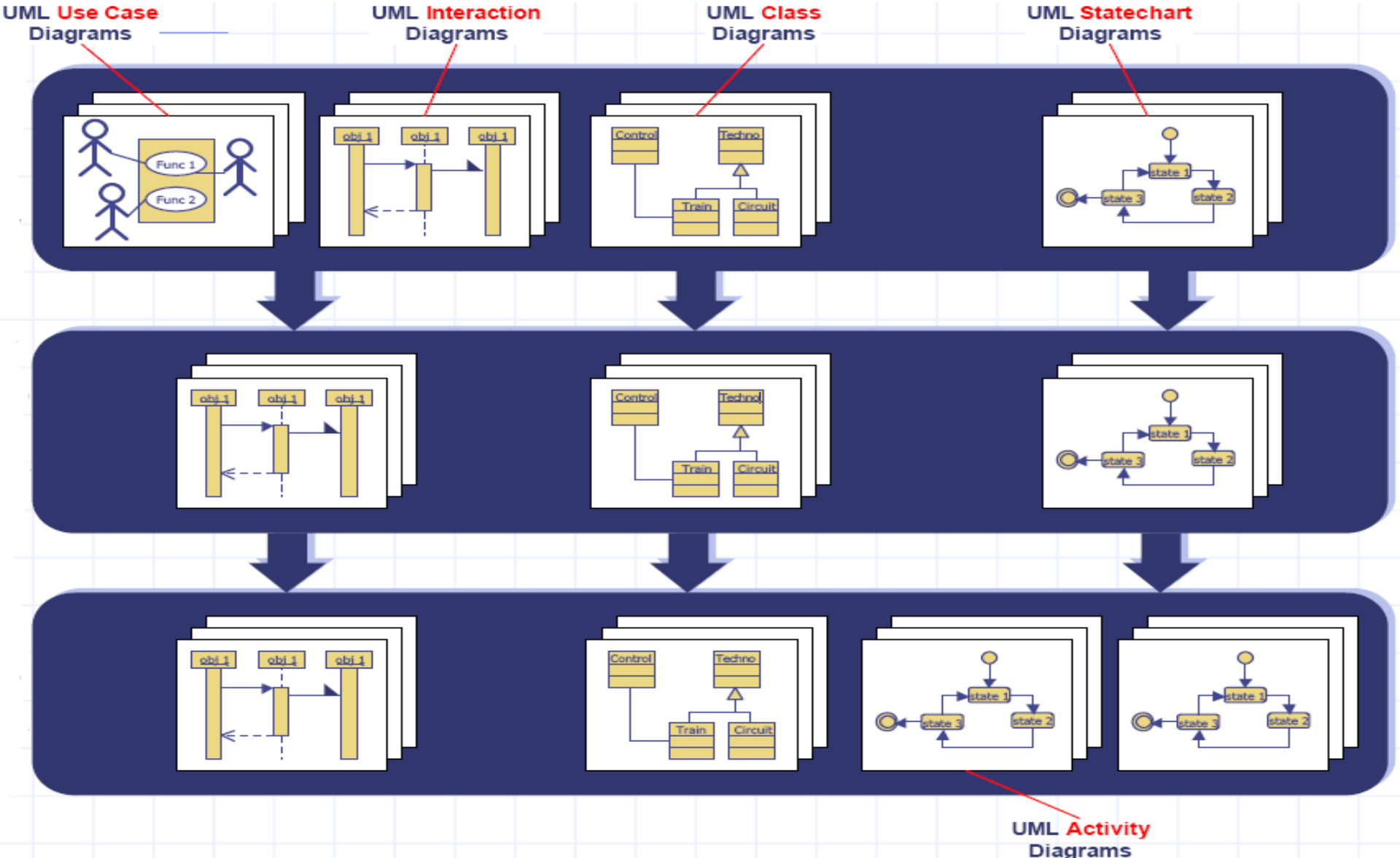
Documenter les décisions

DES MODÈLES COMPLÉMENTAIRES

- Démarches :
  - Ensemble de point de vues complémentaires pour un modèle complet
  - Développement par raffinages successifs
  - Des règles de cohérence pour une modélisation non ambiguë
  - L’analyse formelle du modèle devient possible



MODÈLES DE PLUS EN PLUS DÉTAILLÉS



EN RÉSUMÉ

- UML est un langage de modélisation objet
- UML est une notation, pas une méthode
- UML convient pour toutes les méthodes objet
- UML est dans le domaine public

UML est la notation standard pour documenter les modèles objets

QUAND UTILISER UML ?

Pas limité à un domaine précis

- Systèmes à forte composante logicielle
- Systèmes d'information pour les entreprises
- les services bancaires et financiers
- les télécommunications
- les transports
- la défense / l'aérospatiale
- le commerce de détail
- l'électronique médicale
- les services distribués et les applications WEB

UML – UNIFIED MODELING LANGUAGE

- Support de communication offrant
  - différentes vues (perspectives) complémentaires d'un système, qui guident l'utilisation des concept objets,
  - et plusieurs niveaux d'abstraction, qui permettent de mieux contrôler la complexité dans l'expression des solutions objets.
- Concepts de base : les objets, les classes, les entités, les acteurs et les processus
  - Son aspect formel de sa notation limite les ambiguïtés et les incompréhensions.
  - Son aspect visuel facilite la comparaison et l'évaluation de solutions.

DÉFINITION DE DIAGRAMME UML

- Représentation graphique, qui s'intéresse à un aspect précis du modèle = une perspective du modèle
  - Chaque type de diagramme possède une structure
  - les types des éléments de modélisation qui le composent sont prédéfinis.
- Un **type de diagramme** véhicule une sémantique précise (offre toujours la même vue d'un système)
  - Combinés, les différents types de diagrammes UML offrent une vue complète des aspects statiques et dynamiques d'un système.

LES DIAGRAMMES UML

- une représentation graphique, qui s'intéresse à un aspect précis du modèle ; c'est une perspective du modèle
- Chaque type de diagramme possède une structure
- les types des éléments de modélisation qui le composent sont prédéfinis.
- Un type de diagramme véhicule une sémantique précise (offre toujours la même vue d'un système)
  - Combinés, les différents types de diagrammes UML offrent une vue complète des aspects statiques et dynamiques d'un

MODÈLES ET VUES

- Les utilisateurs regardent et manipulent les modèles au moyen de vues graphiques, véritables projections au travers des éléments de modélisation contenus dans un ou plusieurs modèles.
- De nombreuses vues peuvent être construites à partir des modèles de base (montrant tout ou une partie des modèles).
- À chaque vue correspondent un ou· plusieurs diagrammes.
- UML définit 9 types de diagrammes.

MODÈLES ET VUES D'UML

Vues statiques du système	Vues dynamiques du système
diagrammes de classes;	diagrammes de séquence ;
diagrammes d'objets;	diagrammes de collaboration;
diagrammes de cas d'utilisation;	diagrammes états-transitions
diagrammes de composants;	diagrammes d'activités.
diagrammes de déploiement ;	

9 DIAGRAMMES D'UML

Aspects / Vue	Diagrammes d'UML
Besoins des utilisateurs	Diagramme des cas d'utilisation Diagramme de séquences système
Structure statique	Diagramme de classes Diagramme objet
Dynamique des objets	Diagramme états-transition Diagramme d'activités
Interactions entre objets	Diagramme de séquence objets Diagramme de collaboration
Mise en œuvre et déploiement	Diagramme de composants Diagramme de déploiement

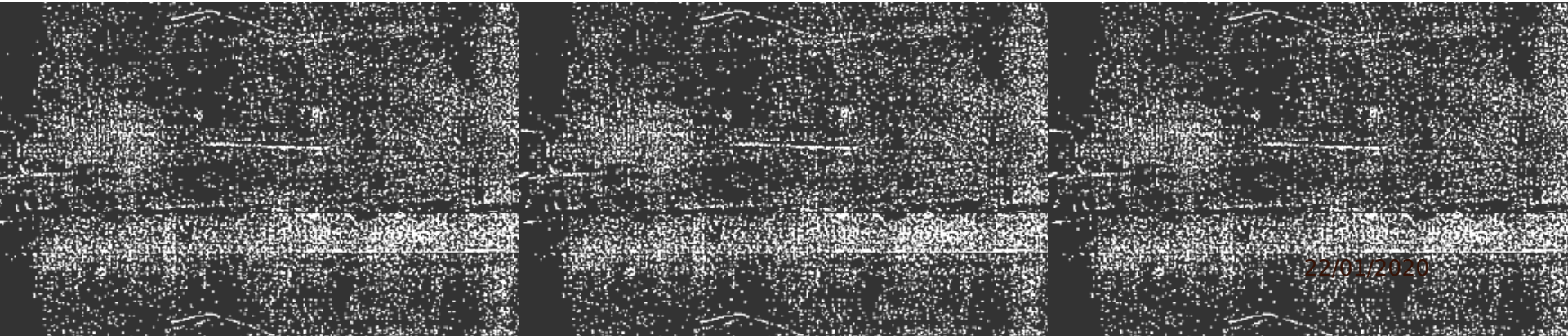
OMG UML 1.4 SPECIFICATION: HTTP://WWW.OMG.ORG

- UML Summary
- UML Semantics
- UML Notation Guide
- UML Standard Profiles
- UML CORBAFacility Interface Definition
- UML XML Metadata Interchange DTD
- Object Constraint Language

AUTRES RÉFÉRENCES

- CORBA 2.4.2 (inclut Real-Time CORBA specifications)
- Meta-Object Facility (MOF)
- Base commune CORBA, UML, etc.
- UML v. 1.4
  - OMG UML Tutorials: http://www.celigent.com/omg/umlrtf/tutorials.htm
  - OMG UML Resources: http://www.omg.org/uml/
  - Pierre-Alain Muller, Essaim Mulhouse : www.uml.crespim.uha.fr
- Profile for Action semantics
  http://www.kc.com/as_site/home.html
- Profile for Scheduling, Performance and Time
- UML Profile for CORBA

DÉMARCHE DE MODÉLISATION UML

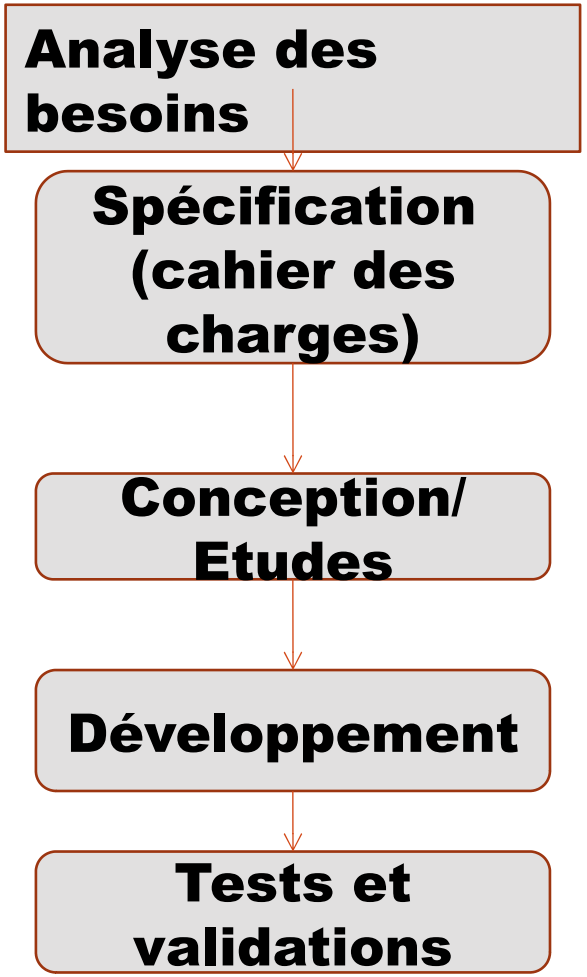


DÉMARCHE DE MODÉLISATION ?

- Langage standard de modélisation des systèmes d'information permettant de représenter des modèles,
- mais **il ne définit pas le processus d'élaboration des modèles**.
- Toutefois il définit une démarche de modélisation

LA DÉMARCHE DE MODÉLISATION D'UML

1. **Itérative et incrémentale,**
2. **Guidée par les besoins des utilisateurs du système,**
3. **Centrée sur l'architecture logicielle**
4. **basée sur les niveaux d'abstraction.**



1. Démarche Itérative et incrémentale

- Modéliser un système complexe = comprendre et représenter
- construire les modèles de spécification et de conception en plusieurs étapes

  ⇒Affiner l'analyse par étapes et en plusieurs fois
- S'applique au cycle de développement dans son ensemble, en favorisant le prototypage

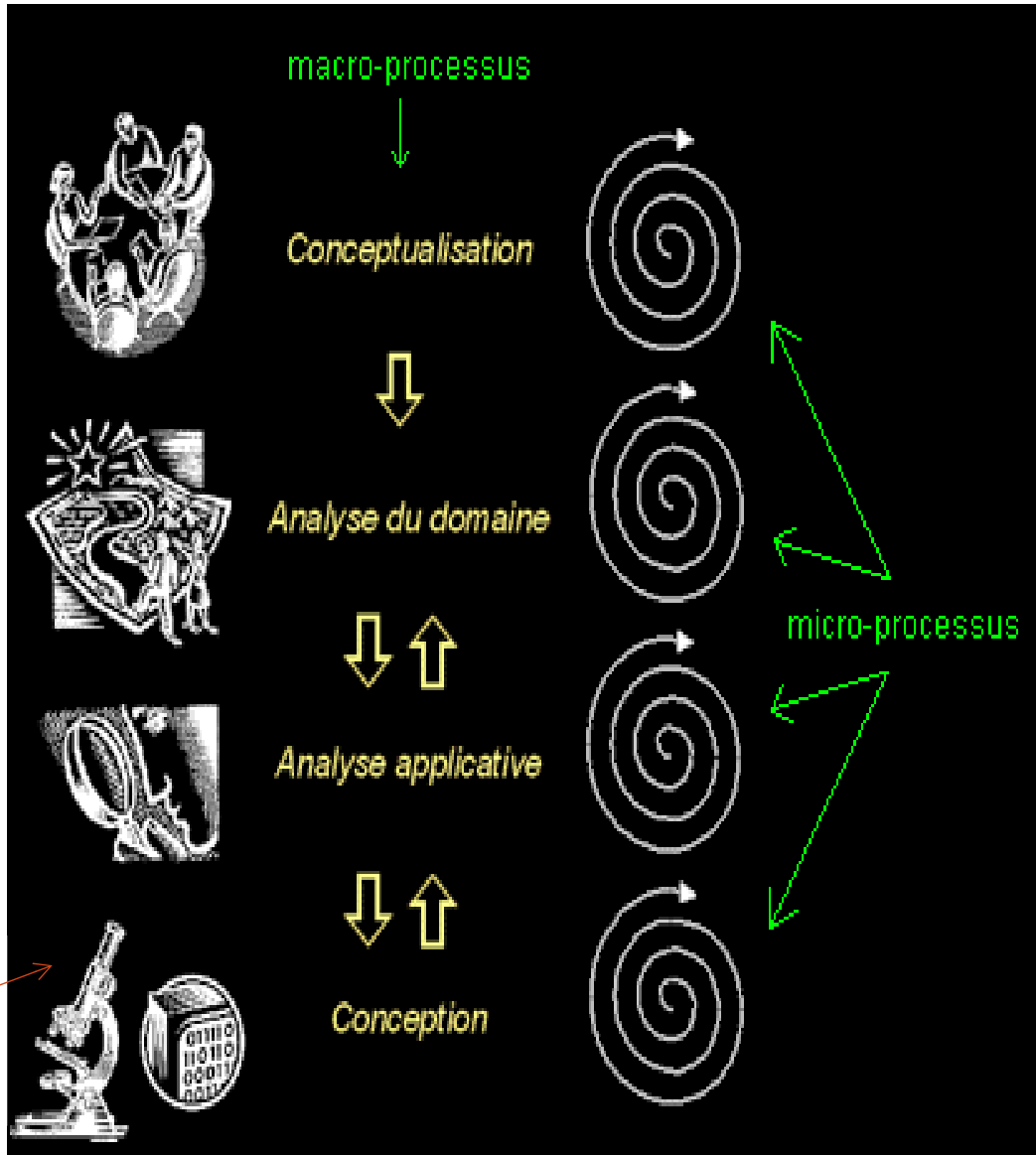
⇒**Mieux maîtriser la part d'inconnu et d'incertitudes qui caractérisent les systèmes complexes.**

2. Guidée par les besoins des utilisateurs

- Le périmètre du système à modéliser est défini par les besoins des utilisateurs
- Le but du système à modéliser est de répondre aux besoins de ses utilisateurs
- Les besoins servent de fil rouge, tout au long du cycle de développement :
  - A chaque itération de la phase d'analyse, on clarifie, affine et valide les besoins des utilisateurs.
  - A chaque itération de la phase de conception et de réalisation, on veille à la prise en compte des besoins des utilisateurs.
  - A chaque itération de la phase de test, on vérifie que les besoins des utilisateurs sont satisfaits.

3. BASÉE SUR LES NIVEAUX D'ABSTRACTION

- Les niveaux d'abstraction permettent de structurer les modèles.
- Un micro-processus régit les itérations à niveau d'abstraction constant.
- Un macro-processus régit le passage de niveau à niveau



les niveaux d'abstraction principaux, dans un processus de développement logiciel

3. BASÉE SUR LES NIVEAUX D'ABSTRACTION

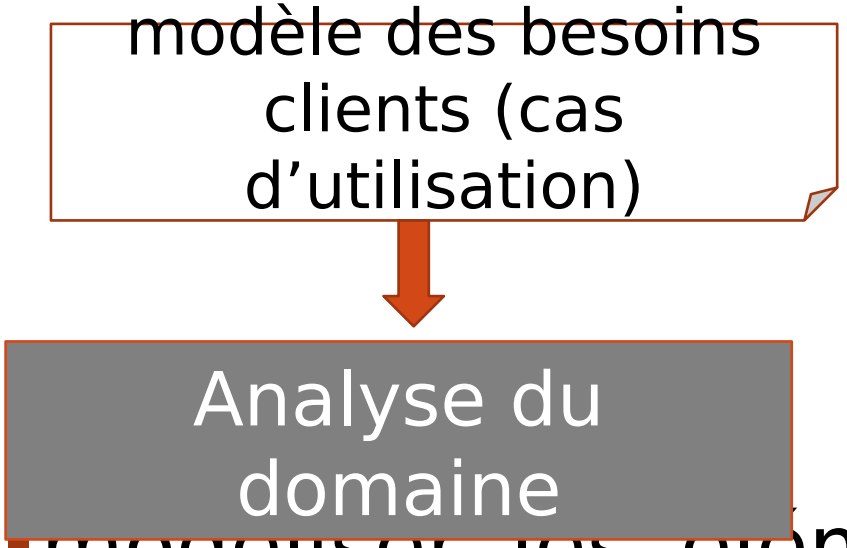
Dossier d'expression des besoins client

Conceptualisation

clarifier, filtrer et organiser les besoins

- capturer les besoins principaux des utilisateurs.
- définir le contour du système à modéliser (quoi),
- capturer les fonctionnalités principales du système, afin d'en fournir une meilleure compréhension (le modèle produit sert d'interface entre les acteurs du projet),
- fournir une base à la planification du projet

3. BASÉE SUR LES NIVEAUX D'ABSTRACTION



- modéliser les éléments et les relations et interactions entre ces éléments

⇒les éléments du domaine sont organisés en *"catégories"* :

- pour répartir les tâches dans les équipes,
- regrouper ce qui peut être générique, etc...

ANALYSE APPLICATIVE

- Modéliser les aspects informatiques du système, sans rentrer dans les détails d'implémentation.
- Définir les interfaces des éléments de modélisation
- Définir les relations entre les éléments des modèles

Conception

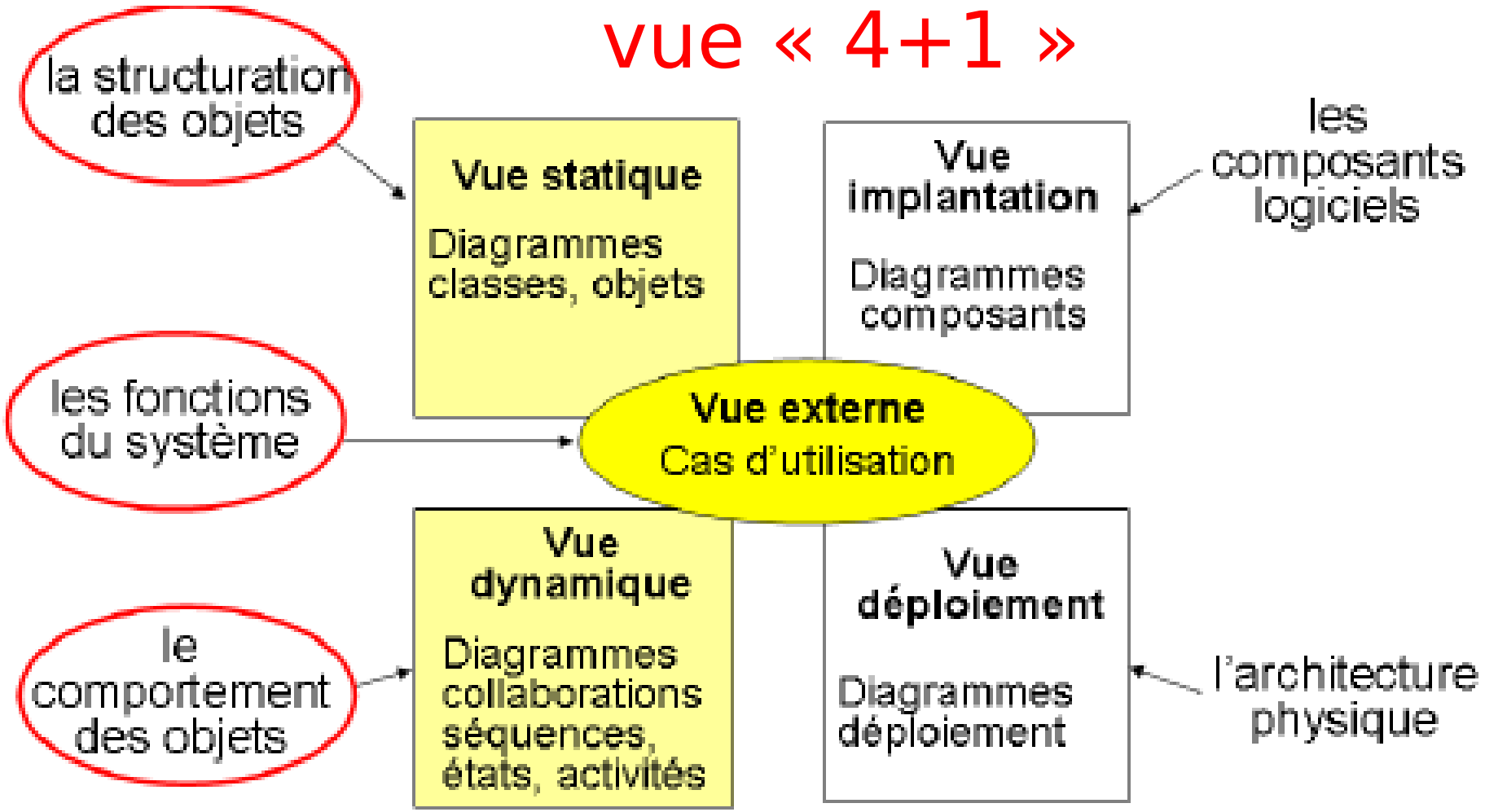
• Modéliser tous les rouages d'implémentation
•Détailler tous les éléments de modélisation issus des niveaux supérieurs.
•Les modèles sont optimisés ⇒destinés à être implémentés.

4. CENTRÉE SUR L'ARCHITECTURE

- **Ph. Kruchten propose différentes perspectives, indépendantes et complémentaires, qui permettent de définir un modèle d'architecture (IEEE, 1995).**

décrit des choix stratégiques qui déterminent en grande partie les qualités du logiciel (adaptabilité, performances, fiabilité...).

4. CENTRÉE SUR L'ARCHITECTURE



1. LA VUE LOGIQUE

- vue de haut niveau - abstraction et l'encapsulation, elle modélise les éléments et mécanismes principaux du système.
  - éléments du domaine, les relations et les 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 (selon des critères purement logiques), les éléments du domaine en ***"catégories"*** :
  - pour répartir les tâches dans les équipes,
  - regrouper ce qui peut être générique,

2. LA VUE DES COMPOSANTS:

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

3. LA VUE DES PROCESSUS :

- très importante vue 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).

4.LA VUE DE DÉPLOIEMENT:

- très importante vue dans les environnements distribués.
- Elle décrit les ressources matérielles et la répartition du logiciel dans ses ressource. Elle montre :
  - 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...).

VUE DES BESOINS DES UTILISATEURS (VUE DES CAS D'UTILISATION)

- Guide toutes les autres. La "colle" qui unifie les quatre autres vues de l'architecture.
- Elle 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.
- Elle motive les choix, permet d'identifier les interfaces critiques et force à se concentrer sur les problèmes importants.

EN RÉSUMÉ

- Modéliser une application n'est pas une activité linéaire.
- Une tâche très complexe, nécessitant une approche : **itérative et incrémentale** (grâce aux niveaux d'abstraction), car il est plus efficace de construire et valider par étapes, ce qui est difficile à cerner et maîtriser,
- **centrée sur l'architecture** (définie par la vue "4+1"), car il s'agit de la clé de voûte qui conditionne la plupart des qualités d'une application informatique,
- **guidée par la prise en compte des besoins des utilisateurs** (à l'aide des cas d'utilisation), car ce qui motive l'existence même du système à concevoir, c'est la satisfaction des besoins de ses utilisateurs.