Introduction à UML

Branches SRT et ISI – Unité de valeur LO02

Guillaume Doyen

Contact : guillaume.doyen@utt.fr

Pourquoi modéliser ?

- **Faire face à la complexité des systèmes**
  - Le raisonnement humain seul ne peut abstraire que des systèmes simples
    - Les systèmes qui nous entourent sont de complexité plus importante que l'autorise notre raisonnement
    - Ex : une application n-tiers (serveur web, base de données)
  - Comment garantir que chaque aspect du système est correctement appréhendé par l'ensemble des parties prenantes
    - Client, maîtrise d'œuvre, maitrise d'ouvrage, …
    - Quid de la communication ?
  - La modélisation abstrait, simplifie la réalité et permet de ne se concentrer que sur les aspects pertinents du système
    - Ex : On peut modéliser une voiture du point de vue mécanique, du point de vue environnemental, du point de vue financier, …
    - La modélisation permet aussi de ne pas travailler en conditions réelles car elle est par essence plus complexe

Notion de langage de modélisation

- **Une notation**
  - Un langage de modélisation de système contient tout type de représentation qui décrit le système (image, diagramme, texte, …)
  - L'ensemble des éléments constituant le langage définissent sa notation
- **Une sémantique**
  - La notation doit posséder un sens connu qui est défini par sa sémantique
    - Sens d'un symbole, d'un diagramme, …
  - La sémantique est définie dans le méta-modèle
  - Le méta-modèle est une description du modèle

Quel type de langage pour la modélisation ?

- **Le code**
  - Spécification outrancière de détails
  - Offre une vue localisée du système
- **Les langages informels**
  - Le langage naturel, les langages propriétaires
  - Sujets à ambiguïtés
- **Les langages formels**
  - Le juste équilibre entre verbosité, détails et absence d'ambiguïtés

Pourquoi utiliser UML ?

- **Langage formel**
  - Sémantique précise qui évite toute confusion dans la modélisation
- **Concis**
  - La notation est simple et directe
- **Complet**
  - Décrit les aspects importants du système
- **Supporte le facteur d'échelle**
  - Tout aussi puissant pour la modélisation de petits que de très gros systèmes
- **Expérience acquise**
  - Le résultat de la communauté orientée-objet depuis 20 ans
- **Une norme ouverte**
  - Défini par un consortium composé d'industriels, académiques donc indépendant de tout fournisseur.
- **Standard pour la modélisation**
  - Utilisé… massivement pour la modélisation des systèmes

Que modélise-t-on avec UML ?

Diagramme	Modélisations possibles	Depuis
Cas d'utilisation	Interactions entre le système et l'extérieur (utilisateur, autre système, …)	UML 1.X
Activité	Activités parallèles et séquentielles du système	UML 1.X
Classe	Classes, types, interfaces et relations	UML 1.X
Objet	Instances des classes dans des configurations importantes	UML 1.X
Séquence	Interactions entre objets lorsque l'ordre est important	UML 1.X
Communication	Interactions entre les objets et connexions pour cette interaction	UML 1.X
Chronométrage	Interactions entre objets lorsque le minutage est important	UML 2.0
Global d'interaction	Réunion des diagrammes de séquence, communication et chronométrage pour représenter des interactions importantes dans le système	UML 2.0
Structure composite	Les relations de composition entre les éléments du système	UML 2.0
Composant	Les composants importants du système et leurs interfaces d'interaction	UML 1.X
Paquetage	Regroupement de classes et de composants	UML 2.0
Machine à états	L'état d'un objet tout au long de sa vie	UML 1.X
Déploiement	Manière dont le système est déployé dans des conditions réelles	UML 1.X

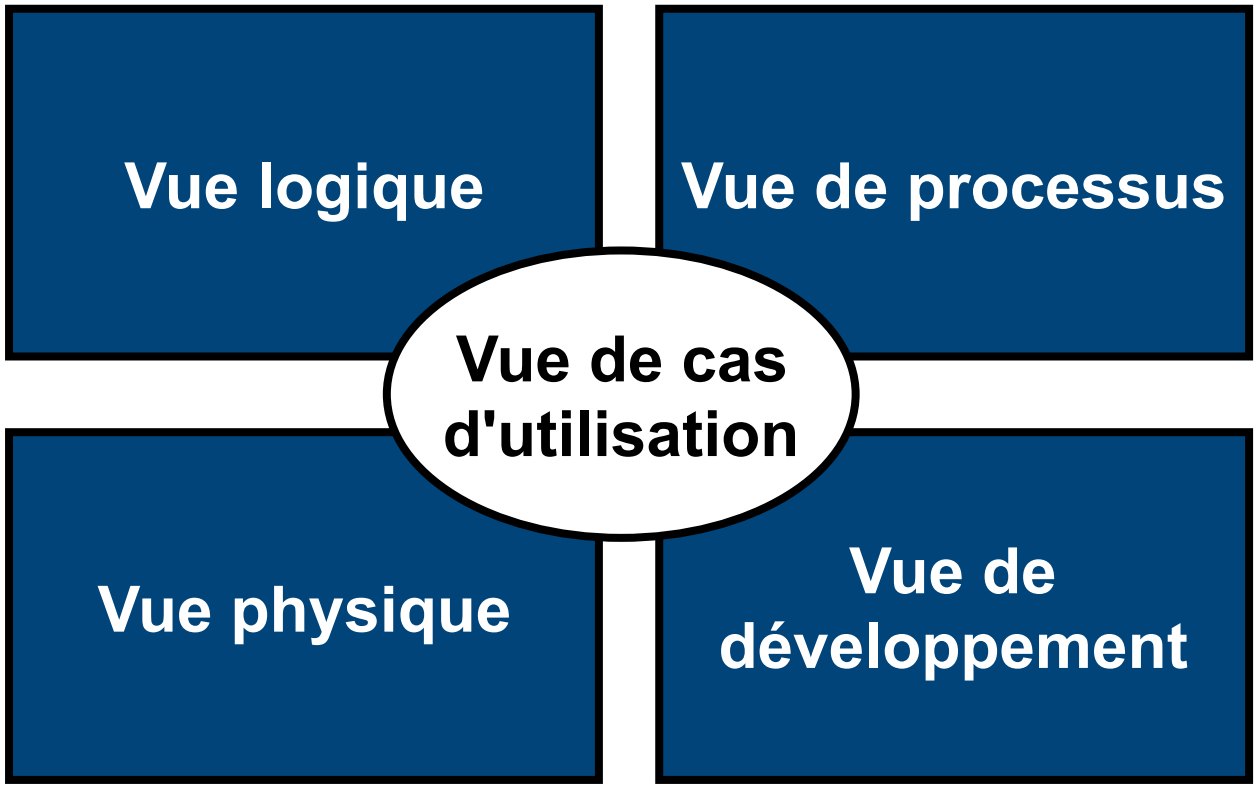
A quel niveau utiliser UML ?

- **Esquisse**
  - Utilisé comme formalisme d'expression pour des petits morceaux du système lors d'échanges informels
    - Dessins faits sur « un coin de table »
- **Plan**
  - Spécification détaillée du système
  - Utilisée avec des outils de retro-ingéniérie qui synchronise le modèle et le code produit
- **Langage de programmation**
  - Pas de recours au code
  - Le modèle est détaillé au point de pouvoir générer du code exécutable directement

UML et le processus de développement logiciel

- **Le développement en cascade**
  - Chaque phase du développement est effectuée dans son intégralité avant de passer à la suivante
    - Ex : recueil des besoins -> Analyse -> Conception -> Implantation -> Test
  - Inconvénient : méthode rigide qui rend couteux les changements qui peuvent se produire au cours du processus de développement
- **Le développement itératif**
  - Permet d'assouplir le développement en cascade
  - Différentes méthodes existent (ex. le Processus Unifié)
- **Les méthodes agiles**
  - Méthodes de développement très souples qui réduisent l'impact du changement sur le processus de développement
  - Ex : développement par les tests, développement par paires

Vues du modèle



Vues du modèle