CONCEPTION des SYSTÈMES d'INFORMATION

UML

1 : INTRODUCTION

Epitech 3 – Automne 2007

Bertrand LIAUDET

SOMMAIRE

INTRODUCTION

Il est facile de décrire la méthode encore que son application exige à coup sûr savoir et pratique.

Conception des Systèmes d'Information

Langage UML

Trois notions dans le titre du cours :

Conception : c'est une partie du développement du logiciel.

Système d'information : c'est un ou plusieurs logiciels manipulant un ensemble d'informations structurées et cohérentes. Par exemple : l'intra de l'EPITECH : des cours, des élèves, des profs, des horaires, des projets, des groupes, des notes, etc. On peut les consulter, les créer, les modifier, les détruire. Plus largement, aujourd'hui, tout logiciel peut être considéré comme un système d'information.

UML : c'est un langage qui permet de représenter graphiquement les éléments de la conception. UML est connoté programmation objet. Attention, ce n'est pas une méthode.

Dans ce cours, on va dérouler une méthode de conception pour le développement d'applications type système d'information avec un langage orienté objet en utilisant le langage UML.

1. Développement d'un logiciel : les quatre distinctions capitales

Il y a quatre distinctions capitales dans le développement d'un logiciel.

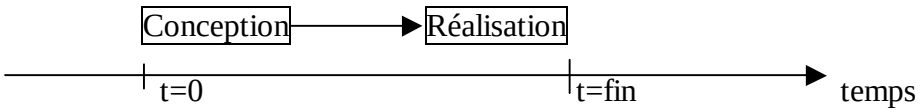
Première distinction : Développement = Conception + Réalisation

Le développement se compose de deux activités qu'on peut distinguer : la conception et la réalisation.

- **La conception** consiste à comprendre et prévoir ce qu'il a à faire.
- **La réalisation** consiste à faire concrètement ce qu'il y a à faire.

La distinction entre la conception et la réalisation est une façon d'organiser la division du travail.

Le premier principe de la méthode consiste à considérer ces deux activités comme deux étapes successives :



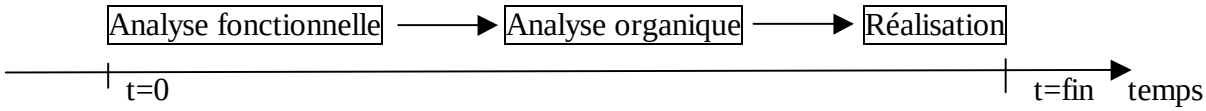
Le projet se déroule dans le temps : il commence avec la conception, il se termine avec la réalisation.

La division du travail consiste à mettre en évidence les étapes de la réalisation d'un logiciel.

Deuxième distinction : Conception = Analyse fonctionnelle + Analyse organique

La conception se divise en deux parties :

- L'analyse fonctionnelle d'abord. L'analyse fonctionnelle s'occupe des fonctions (ou des services) que le système offre à ses utilisateurs. Ce sont les « cas d'utilisation » du logiciel.
- L'analyse organique (ou architectonique[1]) ensuite. L'analyse organique s'occupe de la façon dont sera construit le système pour répondre aux attentes de l'analyse fonctionnelle.



Analyse fonctionnelle	**Analyse organique**
Point de vue de l'utilisateur	Point de vue de l'informaticien
Point de vue du maître d'ouvrage	Point de vue du maître d'œuvre
Point de vue de celui qui commande le logiciel	Point de vue de celui qui réalise le logiciel
Le QUOI	Le COMMENT
Externe	Interne
Build the right system	*Build the system right*
Construire le bon système	Construire bien le système

Avec cette distinction, on fait apparaître :

- le point de vue de l'utilisateur : le maître d'ouvrage (l'utilisateur, le client)
- le point de vue de l'informaticien : le maître d'œuvre.

Pour l'utilisateur, ce qui compte, c'est l'usage du système : les cas d'utilisation (vocabulaire UML). L'analyse fonctionnelle permettra de modéliser l'ensemble des cas d'utilisation.

Pour l'informaticien, ce qui compte c'est l'architecture interne du système.

[1] L'architectonique c'est la technique de la construction, mais aussi la structure ou l'organisation de la construction. Est architectonique ce qui est conforme à la technique de l'architecture.

L'analyse fonctionnelle garantit qu'on va bien faire ce qui est demandé : répondre aux exigences du client.

L'analyse organique garantit que ce qu'on va faire, on va bien le faire.

Troisième distinction : Analyse organique = Architecture système + Analyse détaillée

L'analyse organique se divise en deux parties :

- L'architecture système (ou analyse organique générale): elle s'occupe de l'organisation des sous-systèmes logiciels et matériels du système complet. C'est aussi à ce niveau qu'on situera l'architecture des données.
- L'analyse détaillée (ou analyse organique détaillée) : elle s'occupe du découpage en procédure et en fonctions informatiques de chacun des sous-systèmes logiciels. A ce niveau vont apparaître les en-têtes des fonctions, voir leurs pseudo-codes.

Quatrième distinction : les données et les traitements

Les trois distinctions précédentes sont centrées sur la questions des traitements.

La dernière distinction est celle qui est faites entre les données et les traitements.

Les données seront analysées pour elle-même, indépendamment des traitements qu'on leur appliquera.

Dans une approche « base de données », la distinction entre données et traitement sera radicale.

Dans une approche « orientée objet », aux données, analysées pour elles-même, seront associés les traitements de l'application (les méthodes des objets).

Avec la méthode MERISE et la programmation objet, les données sont revenues au cœur de la conception des logiciels. C'est l'apport de la méthode systémique (système d'information) par rapport à la méthode analytique (descendante) plus classique.

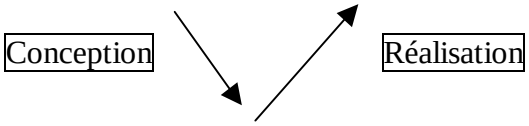
Le MCD de MERISE et le diagramme des classes d'UML sont les deux outils de conception qui permettent de faire une analyse systématique des données.

2. Le cycle en V

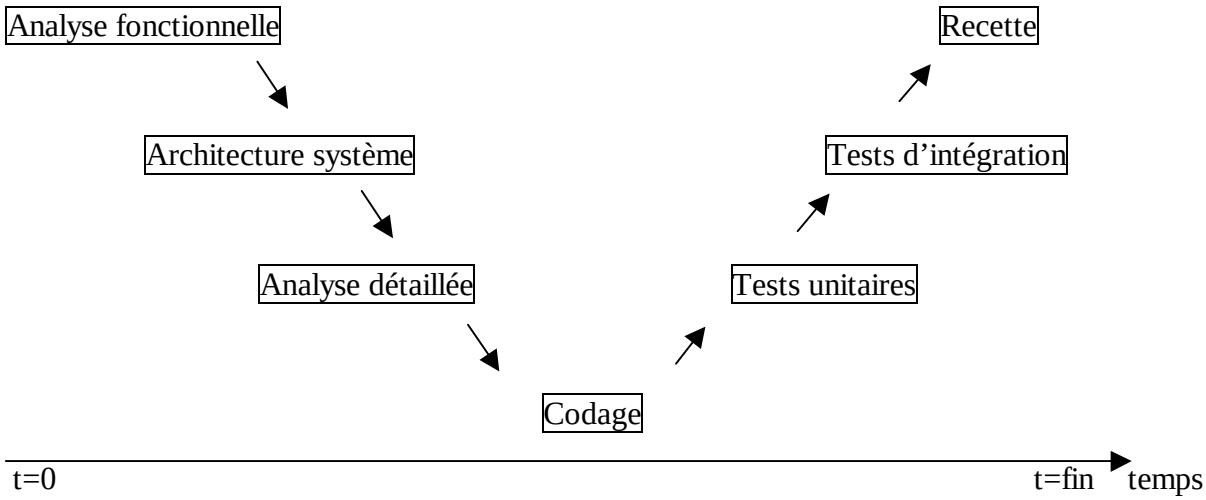
Présentation

Le cycle en V c'est une méthode de développement du logiciel.

Dans cette méthode, la conception et la réalisation forment les deux branches du cycle en V :



Ces deux étapes sont détaillées en reprenant les 3 distinctions abordées précédemment et en ajoutant des distinctions dans la réalisation :



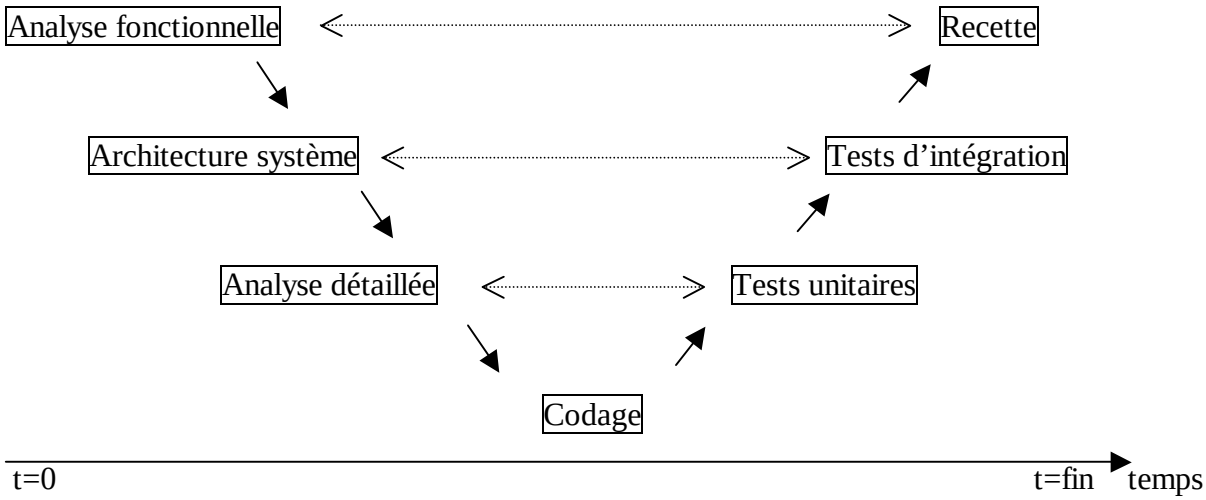
Conception = Analyse fonctionnelle + Architecture système + Analyse détaillée.

Réalisation = Codage + Tests unitaires + Tests d'intégration + Recette.

C'est le lien entre les étapes de chaque branche qui justifie le cycle en V :

- Quand on fait l'analyse fonctionnelle, on peut préparer la procédure de recette.
- Quand on fait l'architecture système, on peut préparer les tests d'intégration des sous-systèmes.
- Quand on fait l'analyse détaillée, on peut préparer les tests unitaires

Ainsi, cela permettra, en cas de problème de test (unitaire, d'intégration ou de recette), de revenir facilement à la partie de la conception à laquelle le problème correspond.



La production des documents

Le cycle en V correspond aussi à un cycle de consommation et de production des documents.

Les activités du cycle en V utilisent les documents des activités précédentes et produisent des documents.

L'analyse fonctionnelle se base sur le cahier des charges. Elle aboutit à un document d'analyse fonctionnelle. Ce document pourra être validé par le client de façon à vérifier la bonne compréhension du cahier des charges par l'informaticien. Ce document servira d'entrée pour l'architecture et l'analyse détaillée.

L'architecture se base sur le document d'analyse fonctionnelle et sur le cahier des charges. Elle aboutit à une document d'architecture.

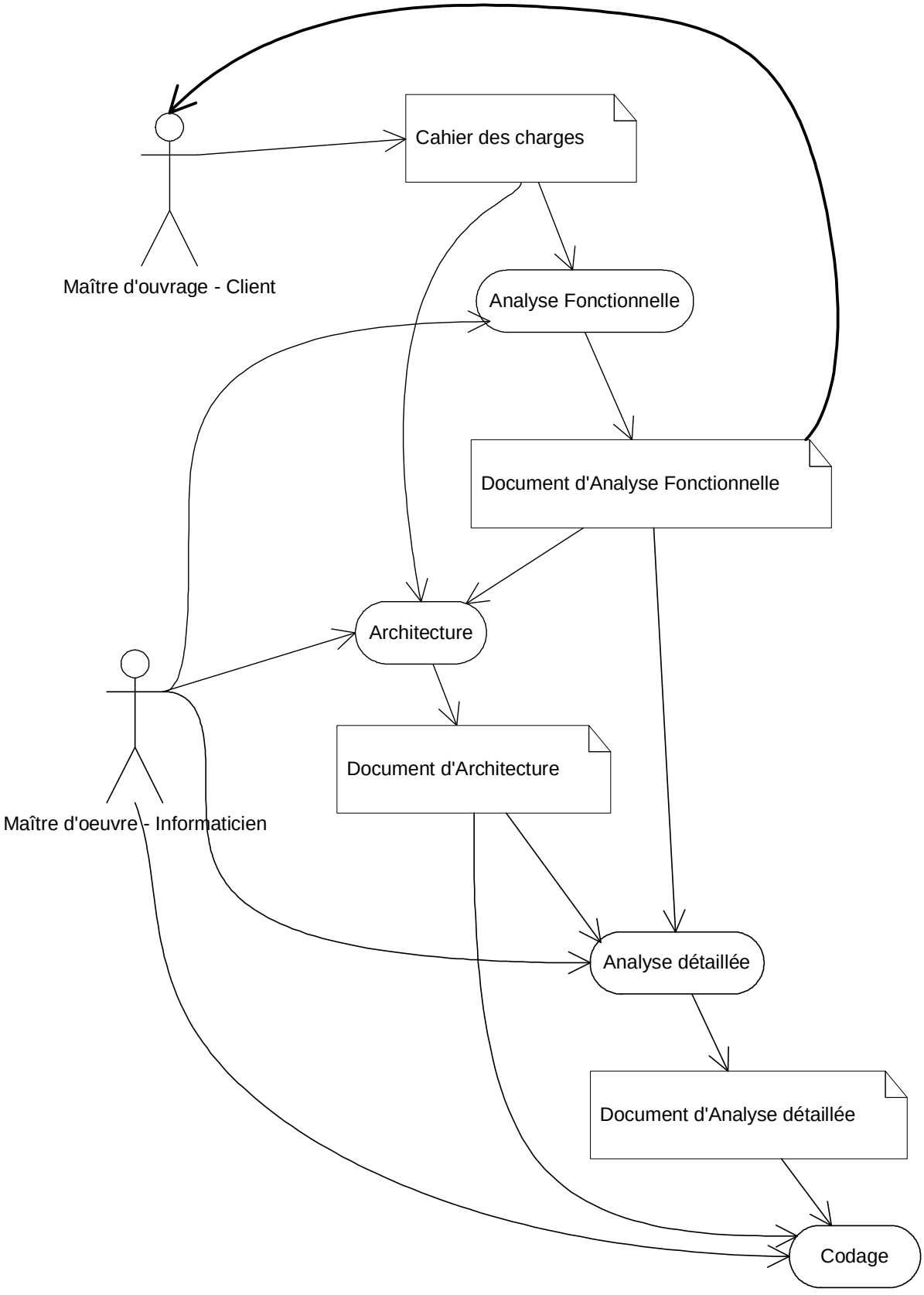
L'analyse détaillée se base sur le document d'analyse fonctionnelle et sur le document d'architecture. A ce niveau, on n'utilise plus le cahier des charges. Elle aboutit à un document d'analyse détaillée.

Remarques :

L'analyse fonctionnelle produit aussi un document de recettes qui sera utilisé à la fin par l'activité de recette.

L'architecture produit aussi un document d'intégration qui sera utilisé par l'activité d'intégration.

L'analyse détaillée produit aussi un document de tests unitaires qui sera utilisé par l'activité de tests unitaires.



Cycle de la documentation

Réalisation et langage de programmation

Une fois la conception terminée, on passe à la réalisation.

La réalisation peut se faire avec n'importe quel langage.

Toutefois, dans le cas d'un SI centré sur une base de données, on utilisera probablement le SQL pour la partie directement liée à la base de données.

Pour l'interface utilisateur, on utilisera indifféremment : C,C++, Java, php, mais aussi des environnements de développement rapide du type de 4D (quatrième dimension) ou de Oracle Database XE (freeware depuis mars 2006).

Cycle en V et analyse des données

Le défaut du cycle en V est qu'il ne prend pas en compte explicitement la question des données.

L'analyse des données peut être considérée comme une partie de chaque étape de la conception. On commence au niveau de l'analyse fonctionnelle (c'est le MCD MERISE et la première ébauche du diagramme des classes), on continue au niveau de l'architecture (c'est le MOD MERISE) et encore au niveau de l'analyse détaillée (avec les méthodes ajoutées aux classes et le MPD et l'ajout d'index pour l'optimisation).

3. UML

Présentation

UML est un langage graphique qui permet de réaliser les étapes de la conception.

UML propose 9 diagrammes différents. Chaque diagramme est adapté à une étape ou à un point de vue de la conception.

Ce n'est qu'un langage, ce n'est pas une méthode.

C'est un langage formalisé : les symboles graphique ont une signification précise.

Les 9 diagrammes d'UML

	Diagramme	**Cf.**	**Fonctionnel**	**Données**	**Organique détaillée**	**Architecture**	**Objet**
1	Cas d'utilisation		**Oui**				
2	Séquence	Collab.	**Oui (scénarios)**		**Oui (séquence objet)**		Oui
3	Activités	Etats-trans.	**Oui (flots)**		(peu)		
4	Classes	Objets		**Oui**	**Oui**		Oui
5	Objets	Classes			**Oui**		Oui
6	Collaboration	Séquence			**Oui**		Oui
7	Etats-transitions	Activités	(peu)		**Oui**		
8	Composants					**Oui**	
9	Déploiement					**Oui**	

1 : Diagramme de cas d'utilisation

Il représente les fonctions du système du point de vue de l'utilisateur.

2 : Diagramme de séquence (cf. collaboration)

Tout comme les diagrammes de collaboration, il montre les interaction entre les objets.

Il se concentre sur la séquence temporelle de ces interactions.

Avec les objets « acteur » et « système », il permet de représenter le déroulement des scénarios (instances des cas d'utilisation).

3 : Diagramme d'activité (cf. états-transition)

Le diagramme d'activités visualise un graphe d'activité (équivalent au MOT MERISE).

Le diagramme d'activités est une variante du diagramme d'états-transitions qui met en avant les activités (fonctionnel) et les transitions plutôt que les états (architectonique) et les transitions.

4 : Diagramme de classes

Il représente la structure statique du système en terme de classes et de relations.

Outre les classes, on y trouve les interfaces et les paquetages.

5 :Diagramme d'objets (cf. classes)

Il représente les objets et leurs liens. Il représente la structure statique du système. C'est un diagramme d'instances du diagramme de classes.

6 : Diagramme de collaboration (cf. séquence)

Tout comme les diagrammes de séquence, il montre les interaction entre les objets.

Il montre les interactions entre objets à travers la représentation d'envois de messages.

7 : Diagramme d'états-transition (cf. activité)

Il représente des automates d'états finis du point de vue des états et des transitions.

8 : Diagramme de composants

Il représente les composants physiques d'une application (fichiers, bibliothèques, etc.).

9 : Diagramme de déploiement

Il représente le déploiement des composants sur les dispositifs matériels.

Les différents diagrammes selon l'étape de la conception

	Point de vue	Diagramme UML
ANALYSE FONCTIONNELLE	Statique – non objet	Cas d'utilisation
	Dynamique – non objet	Séquence
		Activités
ANALYSE ORGANIQUE	Statique - objet	Classes
		Objets
	Dynamique - objet	Séquence
		Collaboration
	Dynamique Plutôt non objet	Etats-transitions
		Activités

Précisions

Etapes	**Point de vue**		**Diagramme**	**Paternité**	**Nb de diag.**
Analyse fonctionnelle	Tout	Dynamique	Flux (contexte)	MERISE	1
	Tout	Statique	Cas d'utilisation	UML	1
	Parties	Dynamique	Séquence système	UML	* / UC
	Parties	Dynamique	Activité	UML	1 / UC
Analyse des données	Tout	Statique	Classes	UML - MERISE	1

Qui fait quoi

QUI	QUOI
Le maître d'ouvrage (utilisateur)	Exprime les besoins dans un cahier des charges
L'analyste	Comprend les besoins exprimés et produit un diagramme des cas d'utilisation et de séquence.
L'architecte	Conçoit les besoins compris et produit diagrammes de classes et autres diagrammes.
Le programmeur	Réalise les besoins conçus.
Le testeur	Teste les besoins réalisés

MO.UML p. 156

Pour établir le diagramme des cas d'utilisation, on part du cahier des charges. C'est-à-dire d'un document qui précise ce que le maître d'ouvrage attend du système qu'il souhaite faire réaliser par le maître d'œuvre.

Il faut établir un dialogue constant avec le client et les utilisateurs.

4. Conclusion : plan du cours

Le plan de ce cours correspondra donc aux étapes analysés dans la méthode :

1. **Analyse fonctionnelle :** avec des diagramme des cas d'utilisation (1), de séquence (2) et d'activités (3).
2. **Analyse des données :** avec un diagramme des classes (4).
3. **Analyse organique détaillée :** avec des diagramme de séquence (3) et éventuellement des diagrammes d'objets (5), de collaboration (6), d'états (7).
4. **Architecture :** avec des diagramme de composants (8) et de déploiement (9).

5. Bibliographie

Théorie UML

- **Modélisation objet avec UML**, Pierre-Alain Muller et Nathalie Gaertner, Eyrolles, 2000, 2ème édition (MO.UML)

  C'est la référence française sur UML. A noter particulièrement l'excellent chapitre 2 sur l'approche objet. Existe en format poche : collection « Best of » chez Eyrolles.

- **Introduction à UML**, Tom Penders, OEM 2002.

- **Mémento UML**, Pascal Roques, Eyrolles, 2005 (UML 2)

  La synthèse la plus concise !

Théorie RUP

- **Le processus unifié de développement logiciel**, Grady Booch, Ivan Jacobson et James Rumbaugh, Eyrolles, 1999 (RUP)

  La bible du RUP

- **Introduction au RUP**, Philippe Kruchen, Eyrolles 2000.

  Le RUP vu par Rational Rose

- **Ingéniérie des systèmes d'information : Merise - Deuxième génération**, Dominique Nanci et Bernard Espinasse, Vuibert, 2001, 4ème édition (ISIM)

  L'état de l'art sur MERISE

Pratique

- **UML 2 par la pratique**, Pascal Roques, Eyrolles 2005.

  Apprentissage d'UML par la pratique en suivant la méthode RUP. Une formation à 1500 euros pour 29,90 euros : très rentable !

- **UML2 en action**, Pascal Roques et Franck Vallée, Eyrolles 2004.

  Le même que le précédent en plus développé mais aussi plus abstrait (et plus cher !).

- **UML2 et les design pattern**, Craig Larman, Pearson Education 2005.

- **Modélisation UML avec Rational Rose 2000**, Terry Quatrani, Eyrolles 2000.

  UML et Rose.

- **UML 2**, Benoît Charroux, Aomar Osmani, Yann Thierry-Mieg, Pearson Education, 2005 (UML 2)

  Quelques exercices intéressants.