Cours UML

Autres diagrammes

Module 3

Table des matières

1 Rappel sur UML

UML est un **standard** devenu incontournable.
C'est un peu le couteau Suisse de la modélisation.

UML est un **langage** (on parle de notation) pour :

- Expression des besoins
- Spécifications
- Choix de conception
- Tests
- Implémentations
- Déploiement,…

UML n'est pas une méthode !

UML est un langage **PUBLIC** et **STANDARDISE** par l'**OMG**[1] :

- Spécifications pour les développeurs d'outils

UML est le résultat d'un consensus entre industriels et méthodologistes

UML peut être défini comme un **LANGAGE DE MODÉLISATION GRAPHIQUE** et **TEXTUEL**.

UML ne fournit aucun processus de développement. Pour cela on utilise une démarche méthodologique appelée **RUP**[2]. RUP est une base de connaissance issue de milliers de projets rassemblés durant un certain nombre d'années. Cette base est accessible via internet. Elle n'est pas traduite en Français.

[1] Object Management Group, association américaine dont le but est de standardiser et promouvoir le modèle objet sous toutes ses formes.

[2] Rational Unified Process est une implémentation de la méthode PU (Processus Unifié) = guidé par les besoins des utilisateurs, centrée sur l'architecture logicielle, itérative et incrémentale.

Nous allons voir que le langage UML permet de comprendre et décrire les besoins spécifier dans le cahier des charges, spécifier et documenter des systèmes, esquisser des architectures logicielles, concevoir et communiquer des points de vue.

Ainsi, une réalisation conforme au Processus Unifié pour transformer les besoins des utilisateurs en logiciel doit nécessairement présenter les caractéristiques suivantes :

- à base de composants
- utilise UML
- est piloté par les cas d'utilisation
- est centré sur l'architecture
- est itératif[3] et incrémental[4]

[3] Exprime l'idée d'une répétition de l'action, « le processus de développement est appliquée plusieurs fois

[4] Chaque itération augmente la quantité d'information

1.1 Aperçu de l'historique

Les grandes dates :

Les langages :

- 1962 : SIMULA[5]
- 1972 : SMALTALK[6]
- 1983 : ADA[7], C++

Les méthodes :

- 1984 à 1997 : Booch, OMT, OOSE (+ 50 autres méthodes)

Les premiers objets ont été développés pour gérer les interfaces graphiques.

Les suivants doivent être les objets métiers, c'est-à-dire ceux que l'entreprise manipule dans le cadre de son activité.

On peut considérer 1986 comme le début de l'époque des pionniers en ce qui concerne les méthodes OO (Orientée Objet).

UML naît le 11 novembre 1997 avec la version 1.1 standardisée par l'OMG (Object Management Group).

Le résultat du tri des différents concepts proposés par les méthodes existantes (environ une cinquantaine) a permis **l'Unification des méthodes de modélisation objet** considérées comme les plus efficaces.

[5] Simple Universal Language : premier langage à base de classes.

[6] Premier langage avec IDE intégré

[7] Ada Lovelace, première femme informaticienne de l'histoire, langage POO de Bull

1.2 Histoire récente

Les trois locomotives en matière de méthodes Orientées Objets sont :

- Booch,
- Rumbaugh
- Jacobson

Ils ont travaillé à une convergence de leurs méthodes au sein de la société américaine *Rational Software*.

Leurs travaux ont aboutis à la naissance d'**UML** (**Unified Modeling Language**) :

- 1993 : Booch 93 et OMT-2
- 1995 : OOPSLA 95 : Méthode unifiée 0.8 (méthodes de 1993 + Autres)
- Juin 1996 : UML 0.9 : reprise version 1995 + méthode OOSE
- 1996 : version bêta avec OOPSLA 96
- 1997 : Soumission à l'OMG en janvier : UML 1.0 (+ partenaires)
- 1997 : révision en septembre : UML 1.1
- 2000 : UML 1.3 (9 diagrammes)
- 2001 : UML 1.4
- 2003 : UML 1.5
- 2006 : UML 2.00
- 2008 : UML 2.2
- 2010 : UML 2.3 (13 diagrammes)
- 2011 : UML 2.4.1
- 2013 : UML. 2.5 (14 types de diagrammes)

Voici un lien pour avoir un aperçu de l'évolution d'UML sur Wikipédia

Les créateurs d'UML sont arrivés à un **consensus en terme de modélisation** mais pas encore en temps que démarche.

Raisons conduisant à préconiser l'utilisation d'UML :

- ***normalisation*** par l'OMG[8]
- disposer de modèles communs ***facilitant le dialogue*** entre maître d'ouvrage et maître d'œuvre[9]
- possibilité d'utiliser ***un seul AGL***[10] depuis l'expression des besoins jusqu'à la génération de tout ou partie du code
- s'appuyer sur des principes et ***concepts objets***
- intention de nombreuses entreprises de soutenir UML: Microsoft, IBM, ORACLE, Unisys, softeam, Hewlett-Packard, et bien d'autres...

[8] Object Management Group qui est un organisme à but non lucratif crée en 1989 à l'initiative de grandes sociétés. Plus de 700 entreprises adhérentes (DEC, HP, IBM, Microsoft, Oracle …)

[9] Le maître d'ouvrage est le donneur d'ordre au profit de qui l'ouvrage est réalisé. Le Maître d'œuvre est chargé de concevoir, de construire.

[10] Atelier de Génie Logiciel

1.3 Modéliser avec UML

Les 8 diagrammes que nous allons décrire dans la suite du cours vont nous permettre de :

REPRESENTER GRAPHIQUEMENT UN SYSTEME

Comme les systèmes sont des choses complexes, on utilise une simplification de la réalité :

LE MODELE (réalité perçue)

Qui sera représenté sous différents angles (à l'aide de nos 8 diagrammes). Cela nous permettra de :

DÉFINIR ET VISUALISER NOTRE SYSTÈME

Le recours à la modélisation est une pratique indispensable servant à :

REPRESENTER DE MANIERE ABSTRAITE UN SYSTÈME LOGICIEL

Chaque diagramme va s'intéresser à un aspect précis du modèle. Un diagramme possède :

UNE STRUCTURE ET UNE SÉMANTIQUE

La combinaison des différents types de diagrammes va nous offrir une vue complète des aspects :

STATIQUES ET DYNAMIQUES d'UN SYSTÈME

1.4 Apperçu de 8 diagrammes UML

1.4.1 Diagramme de classes et diagramme d'objets

Il décrivent la STRUCTURE STATIQUE d'un système en terme de CLASSES et de RELATIONS entre les CLASSES. On utilise des concepts d'agrégation, de composition, d'héritage, d'association, de multiplicité, de rôles, de classes abstraites et de visibilité, ainsi que les attributs et les opérations qui caractérisent chaque classe d'objets. Le diagramme d'objets permet de montrer un contexte pour faciliter la compréhension des structures de données complexes.

(Proposés par *OMT* et *Booch)*.

Mots clés

CLASSES - STRUCTURE – STATIQUE – RELATIONS – OBJETS

1.4.2 Diagramme de cas d'utilisation (ou Use Cases)

Ils permettent de CAPTURER LES BESOINS FONCTIONNELS du client sous la forme d'une interaction entre l'UTILISATEUR et le SYSTEME. Les cas d'utilisation décrivent le système en privilégiant le point de vue de l'utilisateur. Les cas d'utilisation mettent en évidence les ACTEURS (rôles) qui participent et les RELATIONS entre ces derniers. Ils donnent lieu à l'élaboration de scénario.

(Ils constituent l'approche principale de la méthode *OOSE* de *Jacobson)*.

Mots clés

BESOINS - SYSTEME – ACTEURS – INTERACTIONS – SCENARIO

1.4.3 Diagramme de séquence (Event trace dans *OMT*)

Ce diagramme permet de représenter un SCENARIO montrant les INTERACTIONS entre les OBJETS de manière SÉQUENTIELLE par l'intermédiaire de MESSAGES. Ce diagramme nous permet de voir comment les objets interagissent entre eux du point de vue séquentiel.

Mots clés

DYNAMIQUE - TEMPOREL – MESSAGES – SCENARIO

1.4.4 Diagramme de collaborations (Event flow dans *Booch)*

Il montre les INTERACTIONS entre OBJETS avec les MESSAGES qu'ils s'échangent. Il met en évidence les LIENS entre objets et la NATURE de ces liens, l'aspect STRUCTUREL.

Mots clés

DYNAMIQUE - INTERACTIONS – MESSAGES – STRUCTUREL

1.4.5 Diagramme d'états (STD : State Transition Diagram de *Harel*)

Il permet de mettre en évidence les ÉTATS et de montrer leur ENCHAÎNEMENT au cours de la vie d'un OBJET. Généralement associé à une classe, il décrit le COMPORTEMENT COMPLET d'un objet de la classe en représentant toutes les TRANSITIONS dans lesquelles il peut se trouver ainsi que les événements provoquant un changement d'état.

Mots clés

ETAT - COMPORTEMENT – OBJETS – TRANSITIONS

1.4.6 Diagramme d'activité (proche d'un MCT/MOT Merise)

Il permet de décrire les TRAITEMENTS (PROCESSUS) et le fonctionnement d'une classe en schématisant leur déroulement. Les COULOIRS d'ACTIVITÉ répartissent les responsabilités entre ACTEURS opérationnels.

Il dérive à la fois des statecharts de *Harel*, mais aussi des Work flow diagrammes existants bien avant l'orienté objet.

Mots clés

TRAITEMENT - ACTEURS – ACTIVITÉ - PROCESSUS

1.4.7 Diagramme de composants

Il montre les dépendances physiques entre les composants logiciels (*Booch*).

1.4.8 Diagramme de déploiement

Il décrit la disposition physique des matériels qui composent le système et la répartition des composants sur ces matériels (*Booch*).

UML prend le meilleur de chacune des 3 méthodes !

2 Modèles dynamiques : Séquence et Collaboration

- Ils permettent de comprendre et de décrire le comportement des objets et leurs interactions.

- Servent à définir ou à préciser les opérations.

Deux types de représentations (vues précédemment) :

- **dynamique entre objets avec diagrammes d'interaction** :

    - diagramme de séquence

    - diagramme de collaboration

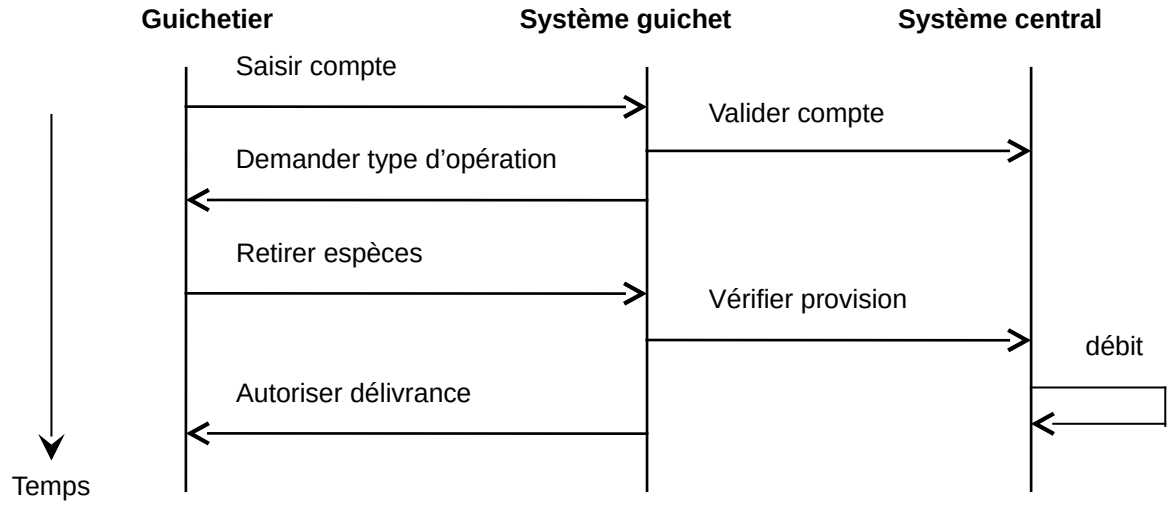
- **dynamique interne à un objet** :

    - diagramme d'états-transitions

2.1 Diagramme de séquence

Met en avant l'aspect ***temporel*** des interactions. Le temps s'incrémente du haut vers le bas de la figure, mais les espaces ne sont pas significatifs. Seules les séquences d'événement sont représentées, non le temps qui les sépare. Chaque objet concerné est représenté par une ligne verticale. Les ***messages*** sont représentés par des lignes horizontales sur lesquelles on précise le type de message par un verbe.

Pour décrire textuellement un diagramme de séquence ou bien un diagramme de collaboration, il suffit d'écrire **émetteur (acteur) / message / récepteur (acteur)**. Le diagramme de séquence prend en compte le **TEMPS**. Il nous suffit d'ordonner les opérations, voire de les numéroter. Dans ce diagramme, les acteurs sont symbolisés par des lignes verticales d'où partent et arrivent les messages les concernant. Les opérations se déroulent du haut vers le bas.



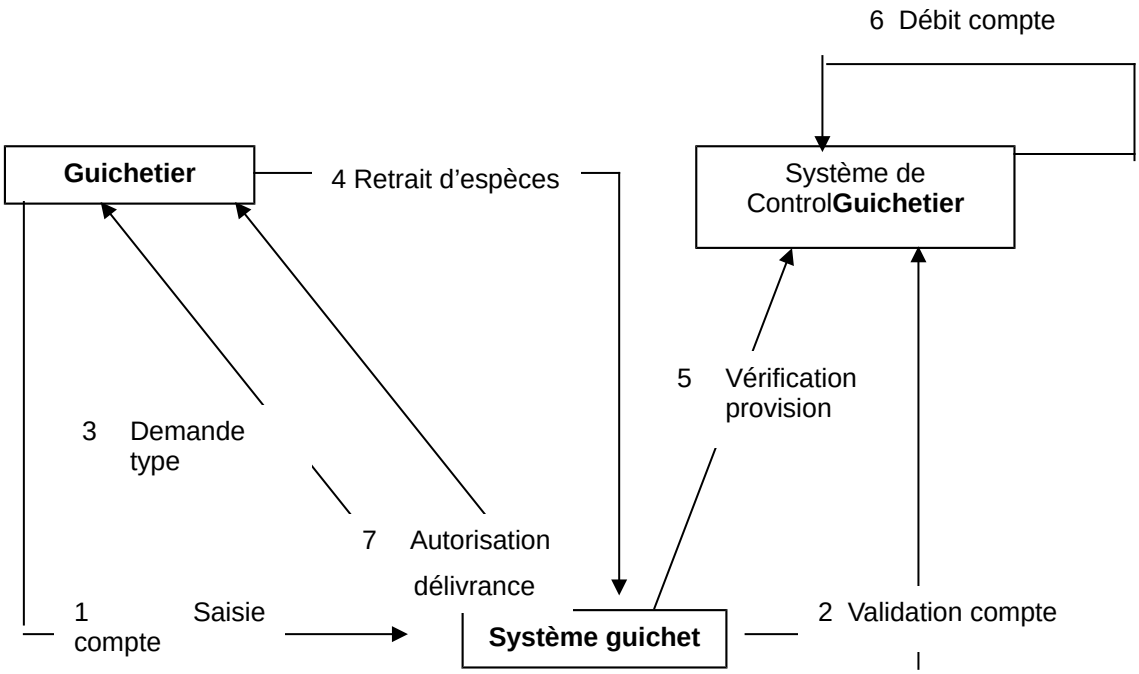
Notations complémentaires :

Les rectangles sur les barres verticales sont des **blocs d'opérations** qui montrent les périodes d'activité des objets.

Les **retours de messages** ne sont indiqués que s'ils augmentent la compréhension du modèle.

Les stimuli inter-processus, ceux qui franchissent la frontière du système, sont plutôt appelés des **signaux**. Les autres stimuli intra-processus sont appelés les **messages**.

2.2 Diagramme de collaboration



Dans ce diagramme, les **interactions** sont représentées par des flèches et la **chronologie par des numéros**. Cette notation devient vite difficile à lire pour les cas d'utilisation qui nécessitent beaucoup d'interactions. Par contre, **elle met en évidence les acteurs dont le rôle est important.** Les acteurs sont représentés par des rectangles placés n'importe où dans le diagramme. **Chaque acteur émet et reçoit des messages comme dans le diagramme de Séquence vu précédemment.**

3 Exemple d'utilisation des diagrammes de séquence et collaboration

Dans UML, l'élaboration de la liste des scénarios est une activité importante de l'analyse.

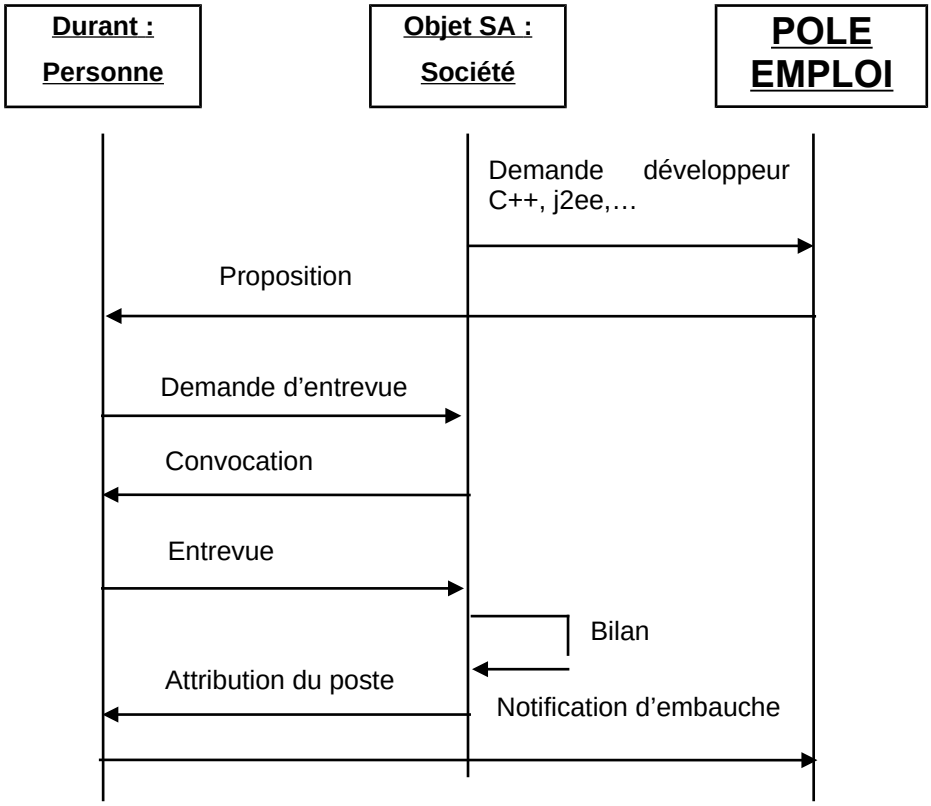
On peut proposer un début de démarche :

1. définir et décrire les cas d'utilisation : forme textuelle
2. pour chaque cas d'utilisation, choisir les scénari.
3. construire les diagrammes d'interaction pour chaque scénario.

Un *scénario* est une série d'événements ordonnés dans le temps, simulant une exécution particulière du système.

Exemples de scénari :

Ce premier scénario décrit le recrutement d'un développeur en JavaEE par la société Objet SA qui s'adresse à l'ANPE.



Remarque : A priori on construit le *scénario de base* sans tenir compte des ***exceptions*** (erreurs, absence de réponse quand le temps prévu est dépassé).

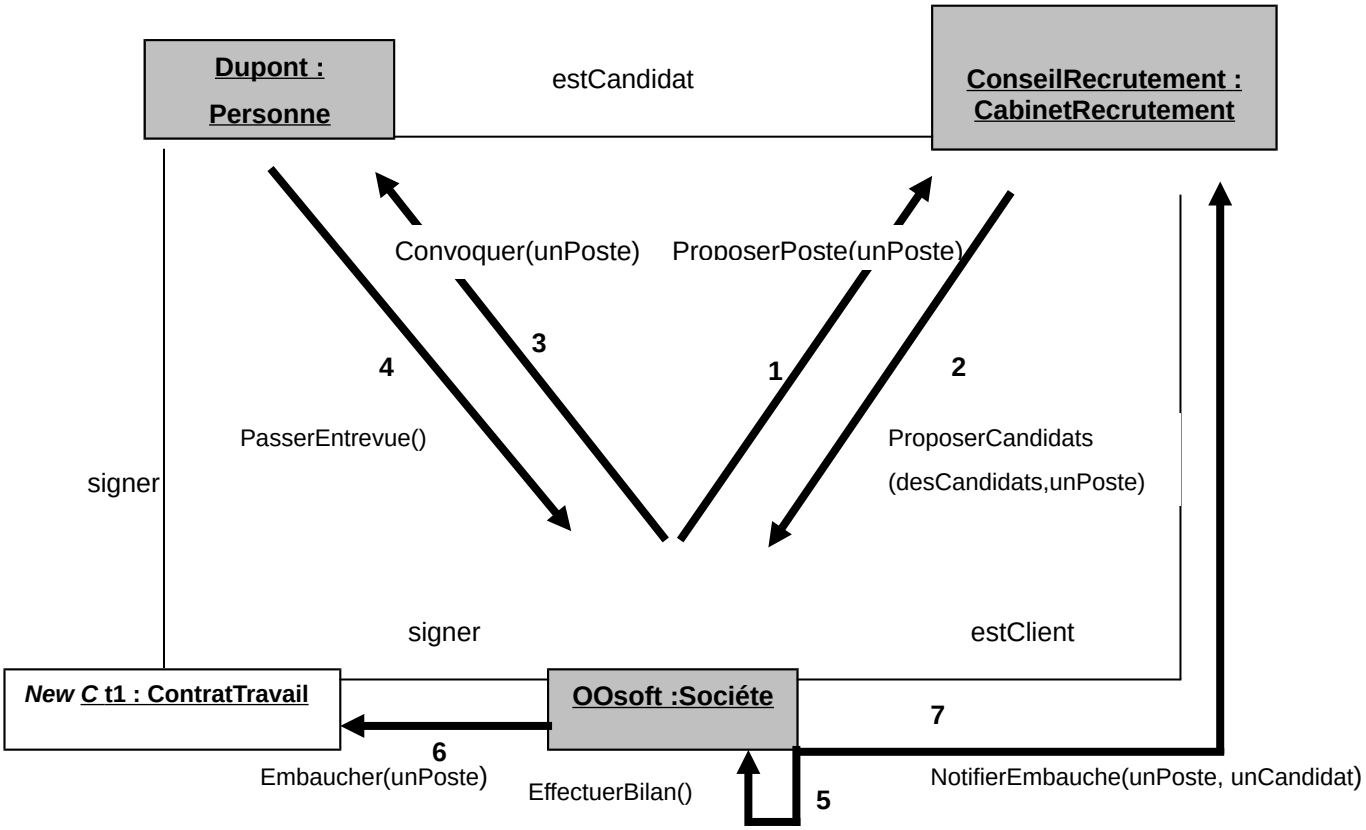
Diagramme de collaboration (point de vue structurel) :

Il représente du point de vue statique et dynamique les objets impliqués dans la mise en place d'une fonction.

Sa ***granularité*** est plus ou moins fine, selon qu'il décrit un ensemble d'opérations utilisées dans l'exécution d'une fonction ou une opération plus simple.

Deux notions fondamentales sont utilisées dans un diagramme de collaboration :

- **Le contexte** : c'est une vue statique partielle des objets qui collaborent pour réaliser une fonction.. Il correspond donc à une partie du modèle objet.
- **Les interactions** : ce sont des séquences de ***messages*** échangés par les ***objets*** dans le cadre de la réalisation d'une opération.



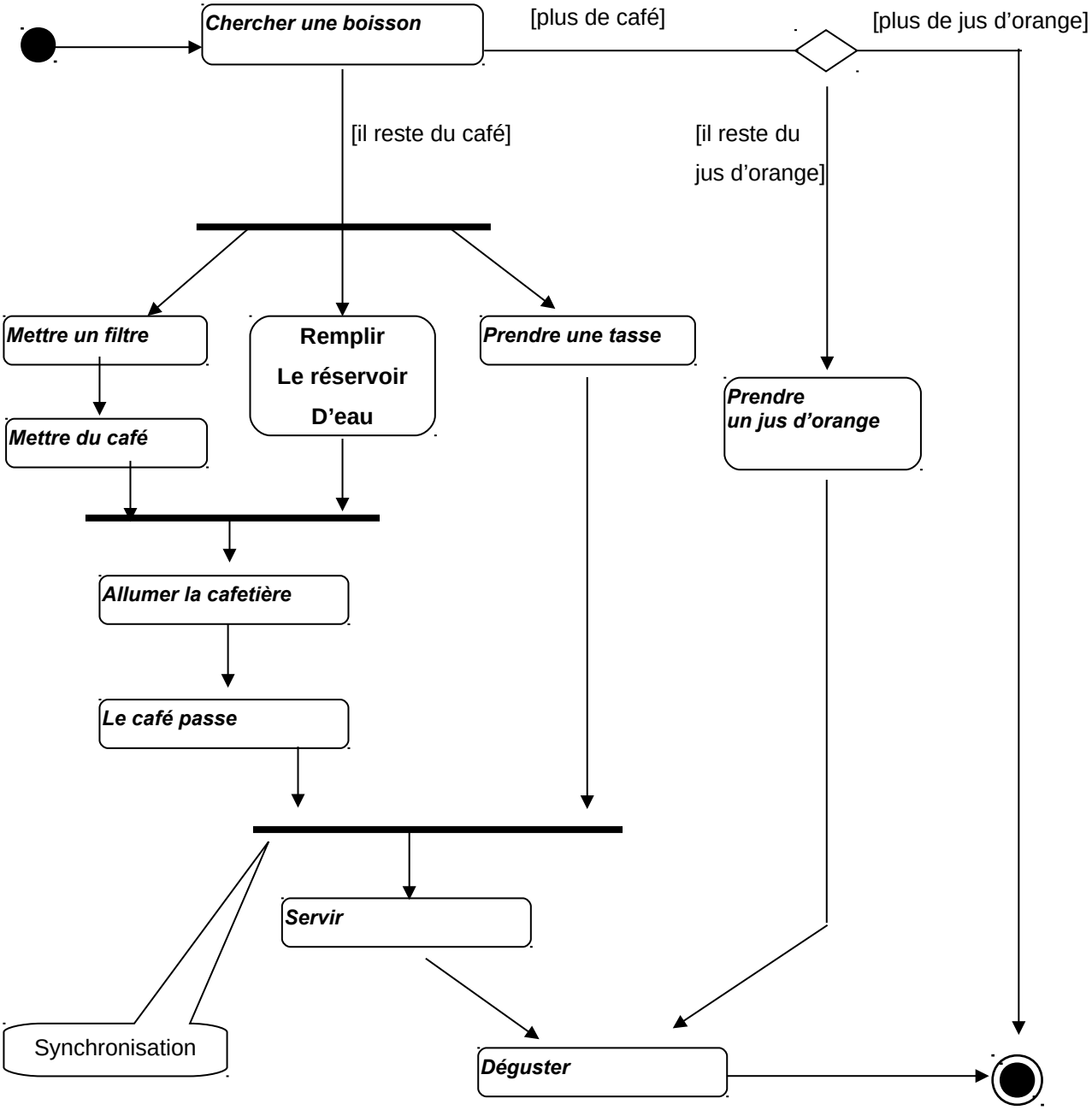
Le message 6 entre Oosoft et Dupont va générer un Nouveau contrat de travail (New ContratTravail).

Ce diagramme est souvent « chargé » en notation et parfois moins lisible que les scénari. Pourtant, s'il est peu **utile** durant la phase d'analyse, il le devient particulièrement **lors de la phase de conception**. car il permet de faire le lien entre le modèle objet et le modèle dynamique. Les méthodes (messages) y apparaissent entre les différents objets.

4 Diagrammes d'activité : états de la machine à café

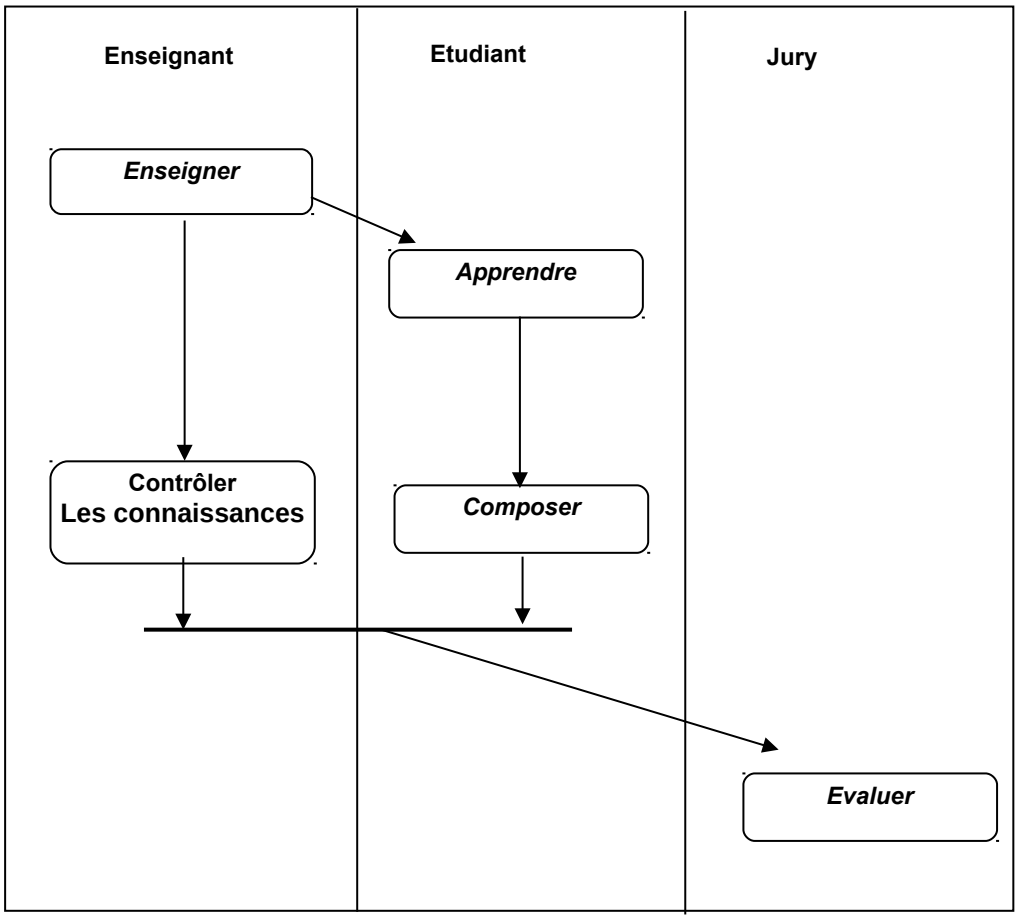
Ce sont des cas particuliers de **diagramme d'état** dans lesquels les états représentent des **activités** et les transitions des **transitions automatiques**.

Il sont plutôt rattachés à une opération ou un cas d'utilisation qu'à une classe. Utilisés pour décrire l'algorithmique d'une opération : séquences d'étapes avec représentation de décisions et de la synchronisation des flots de contrôles. Focalisés sur les traitements internes et non pas sur la réception d'événements externes.



4.1.1 Faire apparaître les Responsabilités dans le diagramme

Un diagramme d'activité peut être divisé en couloirs verticaux assignant la responsabilité de certaines activités à des objets particuliers.

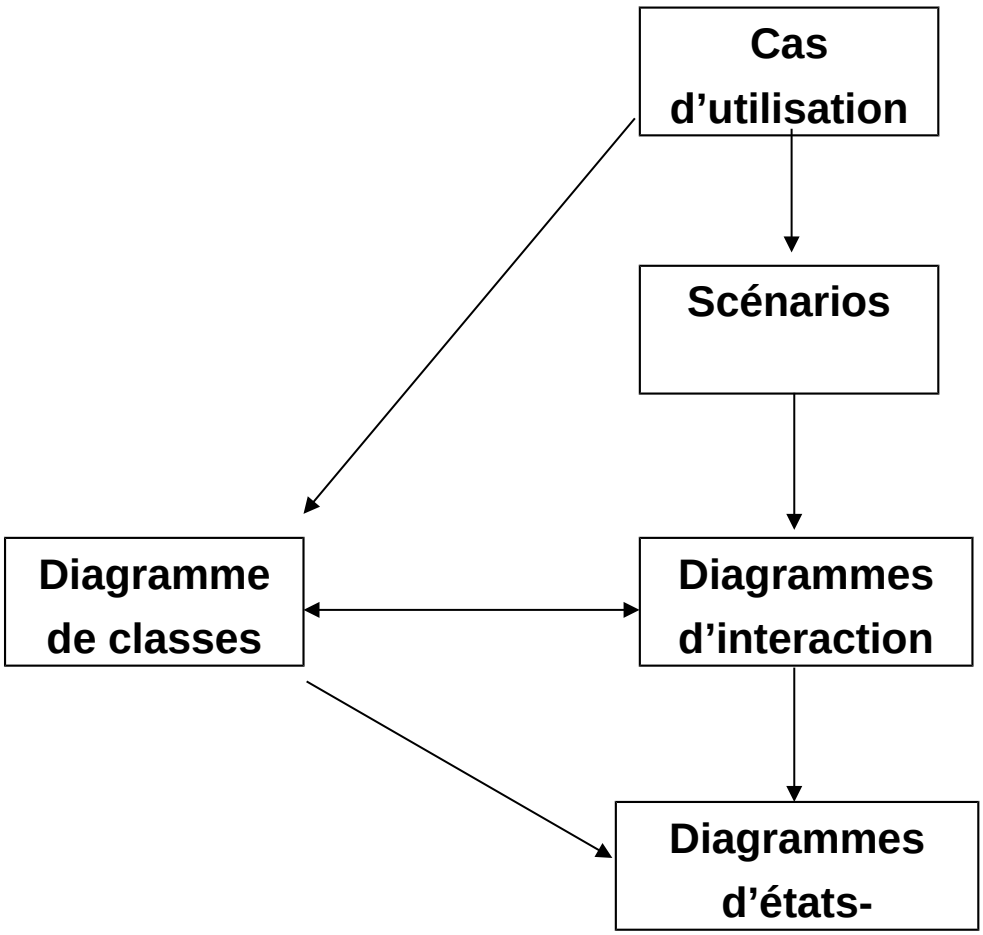


5 Synthèse de démarche d'analyse avec UML

UML propose un standard de notation (8 diagrammes) pour représenter l'analyse et la conception orientée objet. Ce n'est pas une notation fermée : elle est générique, extensible et configurable par l'utilisateur. Une grande liberté est donnée aux outils pour le filtrage et la visualisation d'information. Pour UML, on utilise souvent la méthode RUP malheureusement compliquée à mettre en oeuvre.

Il n'y a pas de démarche standard dans UML. On peut cependant en proposer une :

- définir et décrire les cas d'utilisations
- pour chaque cas d'utilisation, choisir les scénarios
- construire les diagrammes d'interaction pour chaque scénario
- élaborer en parallèle le diagrammes de classes.



Diagrammes	Etape du cycle en V
Cas d'utilisation	Spécifications et cahier des charges
Séquence	
Activité (processus métier)	
Activité (cinématique)	
Classe	Conception Architecturale
Objets	
Communication	
Déploiement	
Composants	

5.1 Comment aborder un projet Web avec UML ?

- Comprendre les besoins
- Analyser le problème
- Concevoir une solution
- Développer la solution
- Déployer la solution

5.1.1 Comprendre les besoins

Pour commencer, on **identifie les acteurs** (utilisateurs) et les **besoins fonctionnels** avec un ou plusieurs diagrammes de cas d'utilisation (les Uses Cases).

L'étape suivante consiste à détailler les différents Cas d'utilisation sous forme textuelle qui comprend un **scénario** qui décompose, étape par étape, les interactions entre le ou les acteurs et le Cas d'utilisation en précisant qui réalise chaque étape. On utilise fréquemment les maquettes des écrans destinés à l'utilisateur pour décrire les scénarios.

Il nous faut aussi rédiger les scénarios lorsque tout ne se passe pas comme prévu. On appelle cela les **scénarios alternatifs**.

On peut aussi utiliser le **diagramme UML d'Activité** pour représenter graphiquement la version textuelle du scénario.

5.1.2 Analyser le problème

Comme dans la méthode Merise lors de la conception d'un MCD, nous devons **lister chaque concept** et en donner la définition pour éviter toute confusion lors de la modélisation. Nous allons constituer un glossaire.

On peut alors commencer le **diagramme de Classes** pour représenter les concepts et les relations.

Puisque l'on sait que les classes deviendront des objets, nous pouvons réaliser le ou les **diagrammes de Séquences** : l'objectif étant de **montrer les échanges de message** (méthodes) entre les différents objets **de manière séquentielle**.

Dans ce type de diagramme, on fait apparaître les acteurs, les interfaces et les classes métiers. On peut faire apparaître les classes **Entity**, **Boundary** et **Control**.

Si l'on souhaite davantage faire ressortir les liens entre les objets sans se soucier de l'aspect séquentiel, on peut utiliser **le diagramme de Collaboration** (en numérotant les différents liens).

A cette étape, on peut concevoir une solution.

5.1.3 Développer la solution

Nous devons passer à la phase de développement à partir des informations que nous donnes les différents schémas réalisés lors des phases précédentes.

Les classes, les interfaces et les scripts de génération de tables de la bases de données peuvent être générées à partir des diagrammes de classes UML. Ainsi le squelette de l'architecture sera généré automatiquement.

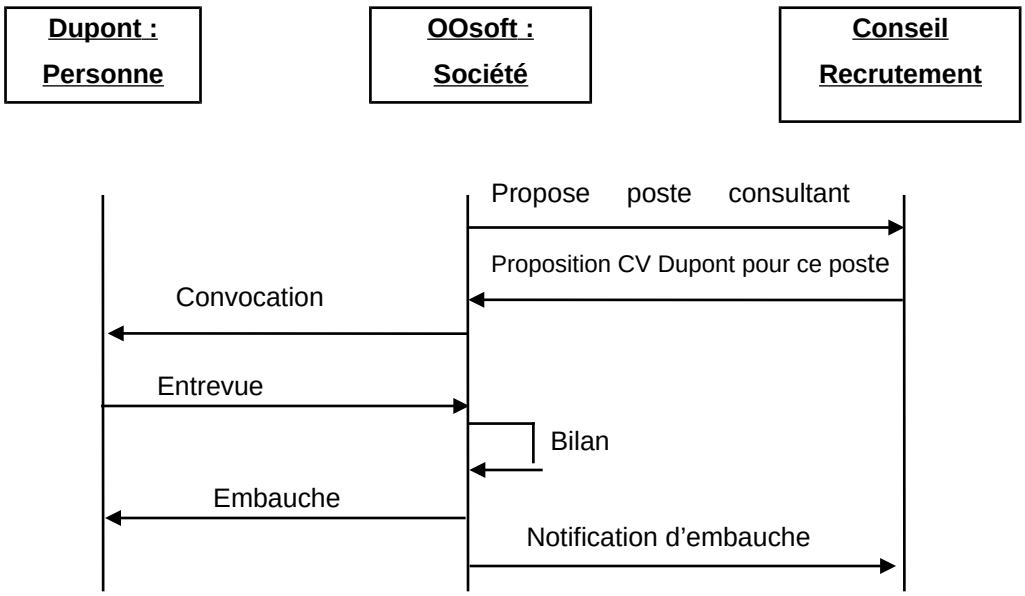
Les traitements seront écrits par les développeurs.

5.1.4 Déployer la solution

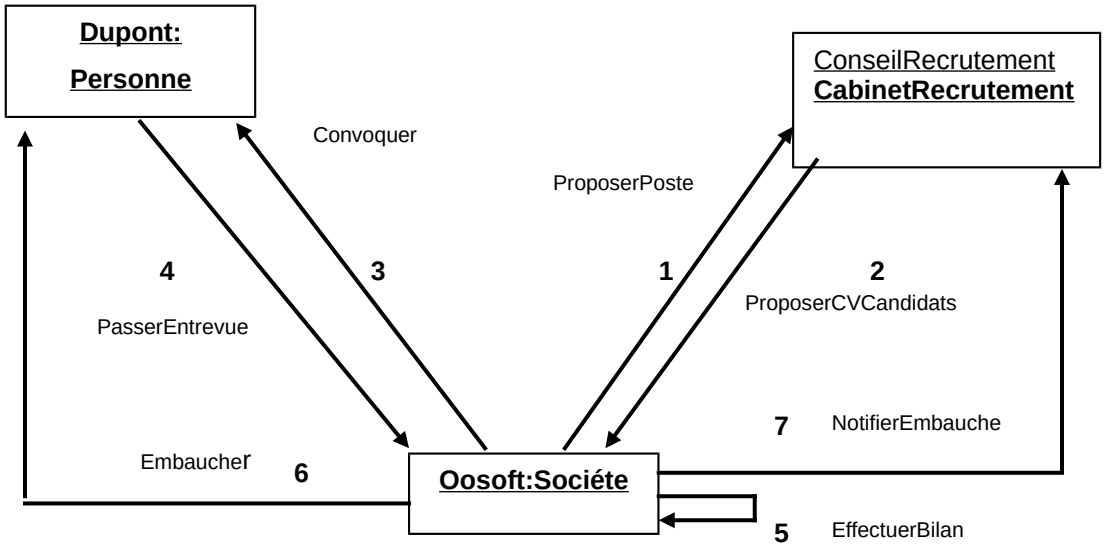
Si besoin, on peut réaliser un diagramme de Déploiement pour préciser les composants et les nœuds correspondants.

6 Annexe

Ce deuxième scénario décrit le recrutement d'un consultant objet par la société OOsoft qui s'adresse au cabinet « Conseil Recrutement ».



Ce scénario peut aussi se représenter sous la forme d'un diagramme de collaboration :



Les ***messages*** qui se déroulent de façon séquentielle sont numérotés.

7 Quelques outils gratuits pour la modélisation avec UML

- BOUML
- ArgoUML
- Papyrus UML2
- Visual Paradigm for UML 8.2
- StarUML
- Win'Design (payant)
- Sybase PowerAMC version 16 (payant)
- Modélio
- Objecteering/UML (version gratuite et payante)
- Voir les plugins pour Eclipse.
- Umbrello (KDE)