I - Repères du plan de cours "UML pour MOA"

Correspondances avec le contenu des supports de cours .

NB: les références de type "numéro de page" sont ici (dans cette version) par défaut exprimées vis à vis du pdf "***expr_besoins_analyse_UML_fevrier2023***" (principal support de cours).

1. Rappel des concepts Objet et d'UML

• **Le cycle de vie d'un projet en Objet.**

---> page 25 (cycle itératif et incrémental)

• **Les architectures.**

----> pages 63 (objets) , page 70 (granularité) , page 72 (points d'entrées) + CONCEPTS OBJETS

• **Le processus unifié.**

----> page 137 (UP , bonnes pratiques communes)

+ repères (liés à l'urbanisation) : page 55 .

2. Analyser le besoin

• **Analyser et formuler le besoin :**

--> a) repérer dans le cahier des charges les éléments fondamentaux (qui reviennent souvent , qui sont mis en avant, ...):

* les sujets (acteurs/utilisateurs potentiels) à par exemple souligner en vert ou en trait plein
* les noms propres et "COD" (classes potentielles) à par exemple souligner en bleu ou en double traits pleins
* les verbes (activités/tâches/traitements potentielles) à par exemple souligner en rouge ou en pointillé

b) au sens "utilisation d'UML" ,
reformuler cela sous forme de diagramme(s) de "cas d'utilisation"

où les sujets sont des acteurs , les "verbes + COD" des noms de cas utilisations .

NB1: ne pas hésiter à utiliser des synonymes (si c'est plus parlant).

NB2: ces diagrammes de "Cas d'utilisation" seront à affiner ultérieurement

(de manière progressive avec "ajustements rectificatifs" à prévoir)

• **Plan de gestion des exigences**

---> mettre en avant :

- le "MVP (Minimum Viable Product) : fonctionnalités fondamentales/minimum attendues/exigées le plus tôt possible
- tous les éléments "absolument <<indispensables>>" (sans ambiguïté) avec une idéale priorisation ("--" , "-" , "-/+", "+" , "++") si possible
- mentionner quels sont les "éléments simplement suggérés" (simples propositions , idées à débattre , plusieurs mises en ouvre possibles, ...)

  via un ou plusieurs qualificatifs (ex: <<suggestion>> ou <<proposition>> ou ...).
- + besoin d'une expression et d'une traçabilité des exigences (point qui sera détaillé un peu plus bas)

• **Comprendre les besoins des parties prenantes.**

---> au sens "système à modéliser" --> acteurs extérieurs (ou variante <<worker>>)

---> au sens "parties prenantes dans les décisions MOA" ---> réunions et gouvernance

avec matrice "**RACI**" (Responsable , Approbateur , Consulté , Informé)

et nombre impair de "votant" pour éviter réunion inutile qui ne débouche sur aucune décision.

3. Définir le système

• **La vision du projet**

---> à clairement expliquer par le(s) responsable(s) du projet

à argumenter/négocier/.... entre parties prenantes moa

à clairement comprendre par la MOE (sans ambiguïté
et aussi sans trop divulguer du "top secret")

---> peut éventuellement être un peu ré-ajustée sur un très gros projet à long terme

---> à exprimer en UML comme un paquet de digrammes fondamentaux

- surtout "diag de cas d'utilisation" pour se concentrer sur les besoins fonctionnels
- éventuel embryon de diagramme d'architecture (ex: "déploiement" ou "composant" ou "packages" ou "powerpoint libre") pour situer un peu le système à modéliser .

• **Documenter les cas d'utilisation**

---> via scénarios (pages 39,40 , 41)

---> via "diagrammes annexes" de type ("séquence simple" , "activités" proche du formalisme bpmn , "communication")

pages 40 et chapitre "diag d'activités pages 44 et suivantes).

• **Eviter les pièges pour les cas d'utilisation**

---> raisonner "session utilisateur" et "fonctionnalités/objectifs"

plutôt que "interface graphique" (voir page 36 : "à ne pas faire").

---> autres pièges : trop détailler / trop décomposer , ...

(avant tout une affaire d'expérience , d'auto-critique , de prise en compte des retours)

4. Gérer le périmètre

• **Poser des priorités sur les cas d'utilisation.**

---> idée ("--" , "-" , "-/+", "+" , "++") dans tableau récapitulatif des "cas d'utilisation"

page 41 ou bien variante de type "point_relatif" (exemple : suite de Fibonacci 1 , 2 , 3 , 5 , 8 , 13 , 21 , 34, ...)

NB: au bout des <<include>> on trouve du "toujours nécessaire" assez prioritaire

et les choses reliées par des <<extend>> sont pas systématiques
et souvent moins prioritaires

• **Déterminer le périmètre du système.**

---> diagramme de contexte (pages 30, 31)

plus une éventuelle modélisation "plus large" en amont
("business modeling" / "modelisation métier").

• **Contrôler les changements.**

---> Très souvent besoin d'une traçabilité entre "exigences principales" et "sous exigences induites"

(alias "objectifs" et "sous objectifs" nécessaires)

à définir/modéliser (par exemple par diagramme de "Ishikawa" page 59 du pdf complémentaire "Modélisation_SOA_avec_UML_et_BPMN")

à exprimer d'une certaine façon (exemple : référence vers "exigence parente" si nécessaire)

à versionner (via "tag GIT" ou tout autre système de gestion de version/révision)

en fonction des changements (la suppression ou modification d'une exigence principale peut entraîner une cascade de suppressions ou modifications).

5. Affiner la modélisation des exigences

• **Définir le contenu de la spécification**

--> pages 53,54

Début de "Spécifications fonctionnelles générales" (et éventuellement détaillées)

+

Début/ébauche du "Dossier d'architecture technique" (avec quelques "exigences techniques" (temps de réponse , sécurité, ...)

plus "exigences d'intégration" au sein d'un système hôte existant ou partiellement défini).

• **Spécifier les exigences fonctionnelles et non fonctionnelles.**

--> "idée méthodologique" , "fil conducteur" "2TUP" page 118

• **Décrire l'interface utilisateur, les protocoles, les contraintes.**

--> chapitre "expr besoins IHM/GUI" ---> pages 59 et suivantes

--> contraintes : pages 52 et suivantes (règles de gestion)

et aussi "conséquences indirectes" d'un "diag d'états" : pages 90, 95 .

--> éventuels protocoles (ex: HTTP , OAuth2/OIDC, ...)

sur diag de "déploiement" UML page 110

• **Qualités requises d'une spécification**

--> pages 53,54

6. Analyse, conception et architecture

• **Les rôles.**

---> rôles et "job" des personnes qui interviennent sur les spécifications et modélisation d'un système/projet

---> pages 24 + "discussion avec formateur"

• **Les concepts clés.**

--> pages 23 et 24 + "concepts objets" (chapitre "fondamentaux orientés objet" pages 63-72)

+ "glossaire" alias "dictionnaire des données" page 73

+ packages (pages 86 et 87)

+ principales syntaxes du diagramme de classes (pages 74-82)

+ BONNE COMPREHENSION DE TOUT CELA via "petite étude de cas (ex: bibliothèque)" .

• **L'architecture logicielle dans le contexte de l'analyse et de la conception.**

--> c'est généralement pas le boulot de la MOA mais le boulot de la MOE.

La MOA peut néanmoins quelquefois imposer certaines technologies considérées "matures" ou "classiques"

pour simplifier/maîtriser un peu la future "maintenance".

Certaines "exigences techniques et d'intégration" formulées par la MOA vont fortement influencer l'architecture logicielle

qui sera proposée par la MOE

Une ébauche de découpage en "secteurs/catégories 'métier' " c'est déjà un très bon début de la part de la MOA.

La MOA peut solliciter "une assistance MOA/MOE" pour bien qualifier certains points un peu techniques et importants de manière à ce que le futur système développé s'interface/s'intègre bien dans son contexte

(en communiquant avec d'autres logiciels autours) .

7. Analyse des cas d'utilisation

• **Identification des cas d'utilisation.**

---> ceux présents sur diagramme(s) de "cas d'utilisation" rédigés en début d'étude de cas.

• **Allocation des responsabilités.**

---> pages 97 , 98 ...

• **Modélisation des relations entre classes : associations et agrégations.**

---> concepts/syntaxes UML pages 77 à 83 (pour les aspects "statiques")

---> "diag de séquence" et "réalisation des cas d'utilisation" pages 100-104

+ démo de la part du formateur sur étude de cas

Attention: c'est en général plutôt le boulot de la MOE

mais c'est intéressant de comprendre l'idée du coté MOA pour :

- bien comprendre les liens de cohérences entre les différents diagrammes UML
- identifier un peu les "points d'entrées nécessaires" de l'application (ex: WS/API REST).

8. Créer des modèles de qualité

• **Les « patterns » à disposition.**

--> quelques "principes génériques" de "bonne structuration" ---> page 71

--> quelques idées "un peu compréhensibles" pour "moa" dans certains design-patterns "GRASP"
---> pages 128 et suivantes

Attention: la plupart des autres "design-patterns" (design-principles , Gang-of-4 (singleton, factory , ...) , MVC, ...) sont très techniques et pour la MOE (pas pour la MOA).

--> critères/attributs de qualité dans le pdf
"pres_architecture_logicielle_et_ATTRIBUTS_DE_QUALITE" (premières pages)

+ idée "scénario de qualité" du même pdf en page .

C'est à peu près équivalent aux "critères d'acceptation" d'une "user story"
de la méthodologie agile "SCRUM"

en page 32 du pdf "agile_scrum" (GIVEN .... WHEN .... THEN ...) : de "Gherkin" .