
1

Concepts et Formalismes UML

www.thierrycros.net

2

Unified Modeling Language

2.1 Historique

Les concepts objet se diffusent au début des années 90, en particulier grâce au langage C++. Les méthodes s'imposent lentement à tel point que OMT devient en quelques années une référence. L'unification des formalismes qui se concrétise par la standardisation de UML en novembre 1997 provoque alors une certaine obsolescence des méthodes. Curieusement, UML est perçu comme leur suite logique alors que sa finalité est essentiellement de remplacer les formalismes natifs par une notation standard. La portabilité des notations en est l'enjeu.

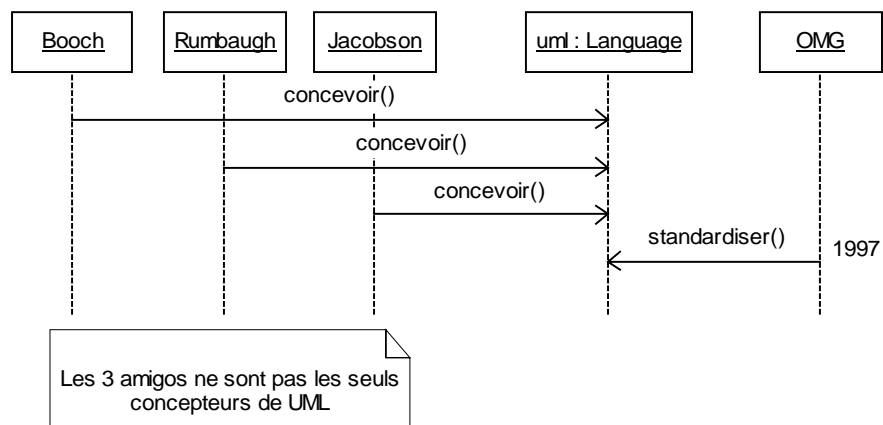


Figure 2-1 : Genèse de UML

Pourquoi percevoir UML comme une méthode ? UML offre certaines caractéristiques attribuées généralement aux méthodes¹ telles que

Abstraction (UML repose sur le concept d'objet et définit de nombreux éléments de modélisation dans son métamodèle)

Formalisme (UML propose un formalisme).

Mais UML n'est pas une méthode :

Cycle de vie (organisé en phases),

Démarches (UML propose simplement le concept de modèle)

sont absents.

Par ailleurs, UML induit quelques micro-processus ou micro-démarches. Par exemple, une démarche simplifiée d'obtention des cas d'utilisation repose essentiellement sur la nature des concepts utilisés dans les diagrammes.

1. Obtenir les acteurs
2. Lister les cas d'utilisation
3. Les décrire.

Cette procédure, bien que simpliste, peut donner l'illusion d'une méthode inhérente à UML.

Dans les faits, OMT (qui est probablement la méthode la plus utilisée en France au milieu de la décennie) est effectivement mise en œuvre sur de nombreux projets mais son avenir est plus que compromis. UML est conçu et le Unified Process prend forme. Le Unified Process est d'ailleurs nommément citée dans les spécifications UML².

2.2 Diagrammes UML

2.2.1 Modélisation des situations d'utilisation

UML propose le diagramme de cas d'utilisation. Un cas d'utilisation est une situation d'utilisation du système (ou plus généralement d'un classifieur) décrite en terme d'interactions. Les acteurs (éléments externes) interagissent avec le système. La description textuelle, le moyen d'obtention des cas, de découverte des acteurs sont autant d'aspects non décrits par le langage.

UML fournit des relations entre cas.

`<<include>>` joue exactement le rôle de la directive `#include` du C++, un cas représente ainsi un ensemble d'interactions acteur(s)/système. Pratiquement, cette

¹ Les définitions formelles des termes méthode, méthodologie, démarche n'impactent pas beaucoup le quotidien du développeur.

² OMG Unified Modeling Language Specification, Chapitre 4 : Standard Profiles

relation de dépendance stéréotypée indique l'utilisation systématique d'interactions (décrites dans le cas inclus) dans le cas origine de la dépendance.

<<extend>> correspond à une *option* en terme d'interactions. Le cas étendu existe en tant que tel sans l'apport du cas source de l'extension. Un ou plusieurs points d'extension précisent l'endroit de l'insertion. Le label de la relation définit la condition d'extension.

l'héritage entre cas : le principe de substitution lié à cette relation entre classificateurs est ici l'essence même de la spécialisation.[TC1]

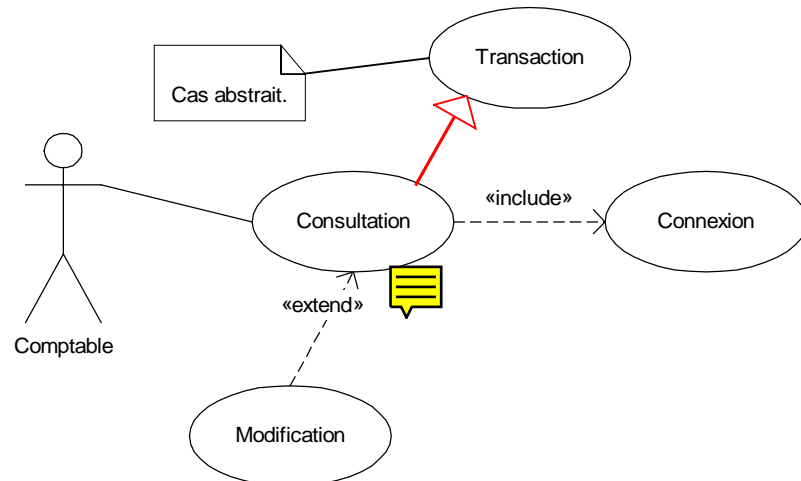


Figure 2-2 : Relations entre cas d'utilisation

Les diagrammes de cas d'utilisation offrent aussi la modélisation de procédures "métier". Ce sont alors des "business use case" ou "cas d'utilisation métier". La distinction provient essentiellement du contexte : le système modélisé est l'organisation elle-même.

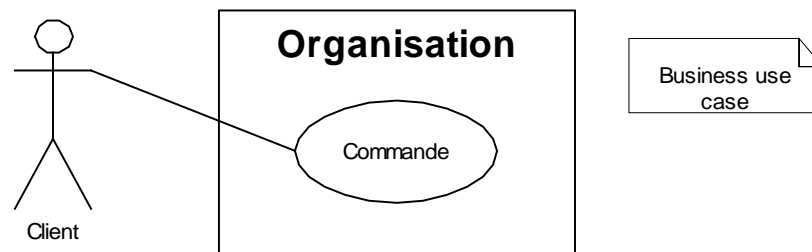


Figure 2-3 : Exemple de cas d'utilisation métier

2.2.2 Modélisation de la dynamique d'un système

Les diagrammes d'interaction

Collaboration

Séquence

modélisent des scénarios, instances de cas d'utilisation et plus généralement toute suite de messages entre objets. Ces deux diagrammes sont dans un premier temps des diagrammes d'objet : ils représentent les objets en présence dans une interaction. Les deux diagrammes visualisent aussi les messages échangés. Le diagramme de séquence met l'accent (visuellement) sur le temps, autrement dit le séquençement des messages. Au contraire, un diagramme de collaboration met en exergue les objets : c'est plus *spatial*, par opposition à l'aspect temporel du diagramme précédent. Contrairement à une idée reçue, les diagrammes de collaboration sont plus adaptés à la modélisation d'interactions, de séquençements, complexes. La numérotation lexicographique des messages offre un moyen sophistiqué de représentation. Par ailleurs, seuls les diagrammes de collaboration indiquent la nature des liens entre objets (connaissance d'un objet destinataire d'un message parce qu'il est local, global...). Les méthodes insistent plus ou moins sur l'utilisation de ces diagrammes, en fonction des activités de développement (analyse, conception en particulier).

Les diagrammes d'état transition correspondent à une synthèse de la dynamique d'un classificateur.

Les diagrammes d'activité sont une variation des diagrammes d'état, focalisée sur les activités dans les états.

2.2.3 Modélisation de la structure d'un système

UML offre plusieurs diagrammes de modélisation de l'aspect statique du système.

Diagramme d'objet

Diagramme de classes

Diagramme de déploiement

Diagramme de composants.

Les diagrammes d'objet visualisent les objets qui collaborent à un instant donné. Ils correspondent souvent à la première version d'un diagramme de collaboration ou de séquence.

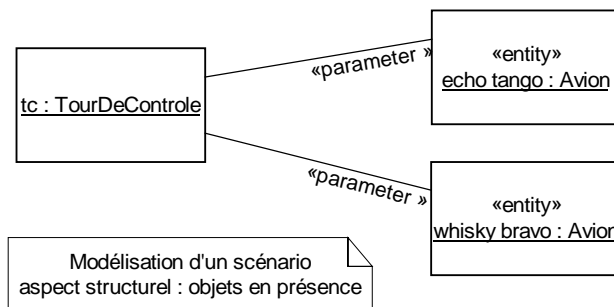


Figure 2-4 : Diagramme d'objets

Les diagrammes de classes sont au coeur de nombreuses modélisations. Ce sont très certainement les plus utilisés dans la panoplie UML. Pourtant, ils ne représentent jamais que l'aspect structurel du modèle, alors que *l'essence même du système est dynamique*.

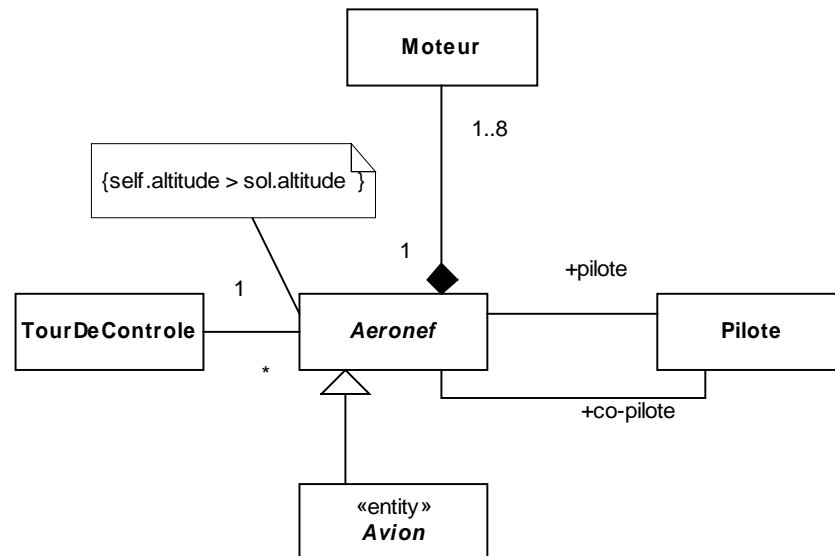


Figure 2-5 : Eléments de base d'un diagramme de classes

Les nœuds et composants sont des classificateurs particuliers qui “résident dans le monde des bits”. Les formalismes graphiques sont apparentés : ils sont basés sur un rectangle muni de décorations spécifiques. Un nœud est représenté par un cube, ce qui suggère le volume d'une unité centrale, d'un routeur, etc.

A noter : les diagrammes de déploiement et composants, dits diagrammes d'implémentation, jouent aussi le rôle de diagrammes d'objets (cas où les labels sont soulignés) ou de classes.

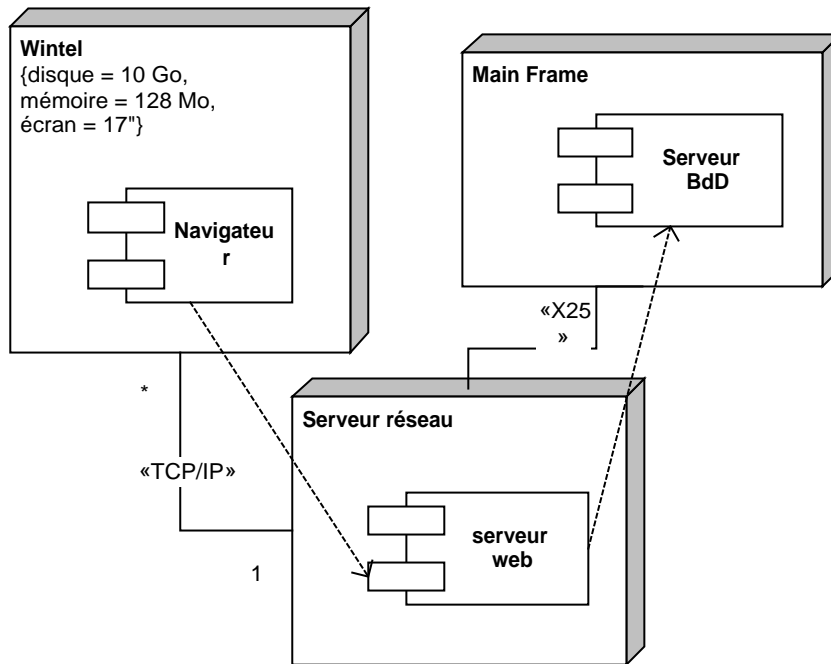


Figure 2-6 : Diagramme de déploiement avec composants

Ce type de diagramme visualise des classificateurs de type nœud et composant. En fonction du degré d'avancement de l'architecture, les diagrammes d'implémentation deviennent de plus en plus précis. Les relations de communication entre nœuds sont stéréotypées, ce qui indique le type de la liaison.

Le déploiement ou une architecture de développement, de test, peut être représentée par des diagrammes de nœuds et de composants au sens objets et non plus classificateurs.

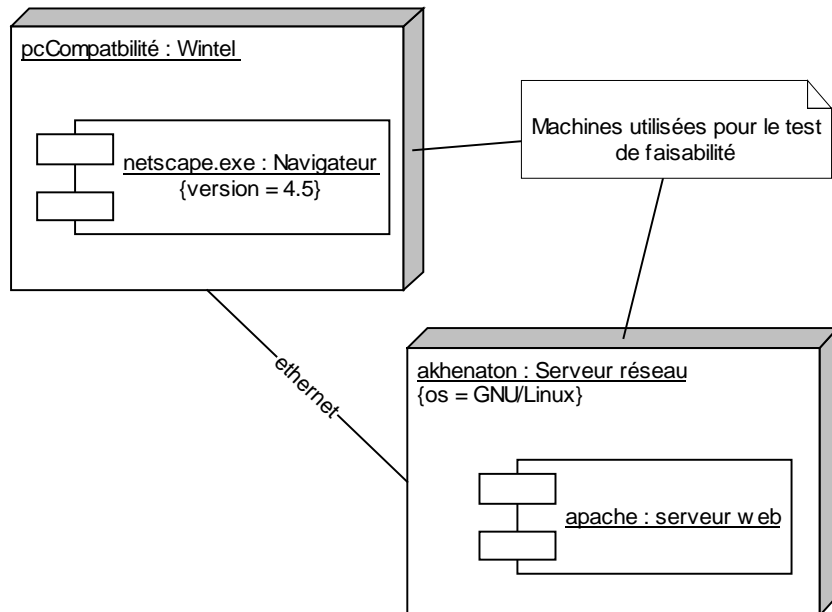


Figure 2-7 : Diagrammes de déploiement : instances de nœuds et composants

2.2.4 Autres diagrammes

Les neuf diagrammes proposés par UML couvrent l'essentiel des besoins en modélisation. Toutefois, certains aspects ne sont pas traités dans ces diagrammes de base : évolution d'un élément dans les modèles par exemple, ce qui correspond à la relation de dépendance stéréotypée <<refine>> ou <<trace>>. Dans ce cas, rien n'empêche la définition de diagrammes supplémentaires³, la contrainte étant le respect des concepts manipulés dans les formalismes.

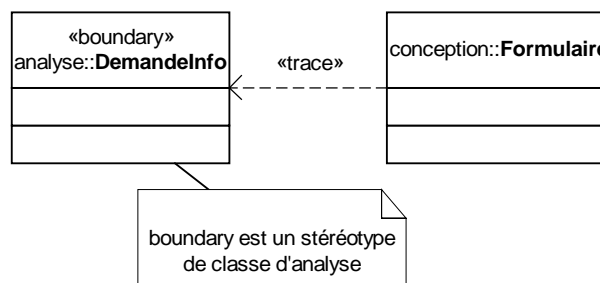


Figure 2-8 : Exemple de diagramme supplémentaire

³ Voir par exemple « Modélisation objet avec UML » de Pierre-Alain Muller 2^{ème} édition page 149

2.2.5 Relations entre éléments

UML propose quatre relations.

- Association
- Héritage
- Dépendance
- Réalisation.

L'association se décline en communication (par exemple entre acteurs et cas d'utilisation), agrégation, composition. D'un point de vue statique, elle correspond à une relation logique entre éléments. Dans une perspective plus dynamique, elle est vue comme un potentiel de communication, instancié en liens entre objets. Dans tous les cas, elle ne peut représenter directement une interaction. La distinction association / lien est fournie par la nature des éléments reliés (respectivement classificateur et instance).

La relation d'héritage entre deux éléments est une spécialisation ou une extension. Une sous-classe spécialise la sur-classe en jouant éventuellement sur le polymorphisme de certaines opérations. Une sous-classe étend aussi les responsabilités d'une sur-classe en ajoutant des opérations et/ou des attributs.

La relation de dépendance est extrêmement générale. De nombreux stéréotypes UML adaptent cette "subordination". Elle signifie simplement : si la cible évolue, la source est impactée, le contraire n'étant pas nécessairement vrai.

La réalisation, comme les autres relations, couvre plusieurs cas de figure. Elle correspond à la mise en oeuvre ou implémentation. Cas typique : un composant réalise une interface. Dès l'activité d'analyse, la réalisation permet de relier (par raffinement d'une certaine manière) des éléments de modélisation. La collaboration entre objets (et donc classes) d'analyse est une réalisation d'un cas d'utilisation.

Ces relations fondamentales ne sont pas exclusives. Une spécialisation par héritage est aussi une dépendance : l'évolution de la sur-classe impacte les sous-classes. Les modélisations de systèmes comportent souvent des ambiguïtés qu'il faut lever par la connaissance plus approfondie des éléments. Un avion possède un moteur... ou bien un avion est-il un moteur spécialisé (un moteur et beaucoup d'éléments tout autour) ?

N Les relations entre éléments sont quelquefois subtiles. Le choix entre héritage et dépendance de type
 O <<binding>> (pour les classes paramétrées) n'est pas toujours évident. Un container d'entier est-il un
 T container spécialisé ou bien un container généré par binding (liaison) depuis une classe paramétrée ? L'étude
 E précise des classes permet de statuer. Dans ce cas, les algorithmes des méthodes (comme insererElement())
 sont identiques quels que soient les éléments insérés, au type près des éléments. Il s'agit de binding.

2.3 Mécanismes communs

2.3.1 Etendre UML

L'un des points forts d'UML est son adaptabilité. Les trois mécanismes

Stéréotype
 Contrainte
 Propriété

forment une boîte à outil d'extension du metamodelle et des formalismes. L'extension d'UML est à la base de la modélisation de systèmes non logiciels. En particulier, les stéréotypes appliqués aux relations de dépendance offrent de multiples possibilités.

2.3.1.1 Stéréotypes

Ils permettent de créer de nouveaux types au sens éléments de modélisation. Le Unified Process propose par exemple les trois stéréotypes <<boundary>> <<control>> <<entity>>. UML en tant que langage utilise intensivement ce mécanisme. De nombreuses relations entre éléments de modélisation reposent sur les stéréotypes.

Toutefois, les stéréotypes devraient être manipulés avec précaution. Au plus les éléments sont stéréotypés, au plus les modèles s'éloignent du standard UML. Dans cet ordre d'idée, les « standard profiles » sont un moyen terme : ils adaptent UML à des situations classiques de modélisation et font partie de la spécification.

N Pratiquement, comment gérer les stéréotypes ?

O
T Le premier point à prendre en compte est la relation entre le stéréotype et le metamodelle UML. Un stéréotype est nécessairement attaché à une *classe de base* dans le metamodelle.

E Le deuxième point est l'ensemble des stéréotypes qui doit être référencé dans le projet, voire dans l'organisation. Un document des « Stéréotypes applicables » permet d'éviter des doublons entre autres problèmes.

Un nouvel élément de modélisation peut être représenté par une icône spécifique. Ainsi, les diagrammes deviennent plus naturels.

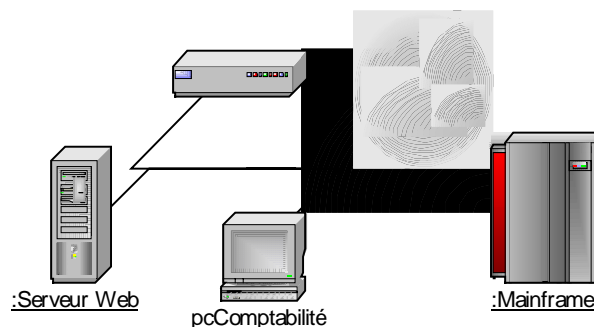


Figure 2-9 : Diagramme de déploiement avec stéréotypes

2.3.1.2 Contraintes et propriétés

Les stéréotypes enrichissent le metamodelle par ajout d'éléments. Les contraintes et propriétés (ou valeurs marquées, nommées) enrichissent les propriétés des éléments de modélisation.

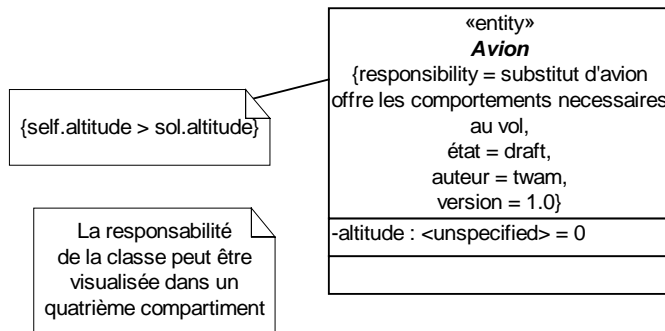


Figure 2-10 : Contraintes et valeurs marquées sur un élément de modélisation

N Les ateliers de génie logiciel offrent plus ou moins de possibilités : création de stéréotypes d'icônes, gestion
O des contraintes et valeurs marquées des éléments, etc. Un guide d'utilisation de l'outil, défini en début de
T projet, permet d'harmoniser les diagrammes.
E

2.3.2 Organiser

Organiser est essentiel. De même que les fichiers d'une machine sont regroupés, structurés en arborescence de répertoire, de même les éléments de modélisation sont organisés en paquetages. Un paquetage est une unité d'organisation. Pratiquement, il regroupe éléments de modélisation et diagrammes UML.

2.3.2.1 Paquetage

Un paquetage a un rôle prédestiné : organiser. Les répertoires contiennent des fichiers de plusieurs types : traitement de texte, images, feuilles de calcul, sous-répertoires, etc. Cet inventaire éclectique est en harmonie avec une certaine rigueur d'organisation. La gestion d'un projet, par exemple, se concrétise en terme de documents, de tableaux de coûts, de sous-répertoire par équipe de développement, etc. Ainsi un répertoire correctement géré contient malgré tout des fichiers de types différents.

Plus précisément, un paquetage contient

- Des éléments de modélisation

- Des diagrammes

- Des sous-paquetages.

La précocité de la structuration en paquetages garantit une bonne organisation des modèles.

2.3.2.2 Sous-système

Un sous-système est un paquetage. C'est aussi un classificateur. Autrement dit, il peut être instanciable. Contrairement au paquetage qui a pour objectif unique l'organisation, le sous-système représente aussi des *macro objets* du système. La relation de composition relie les sous-systèmes et les éléments, physiques ou logiques, qui le constituent. De même qu'une classe est intrinsèquement composée à partir de datatypes (entier, long...), de même un

sous-système est composé de classes et plus généralement d'éléments de modélisation. C'est la même relation, mais à un niveau différent.

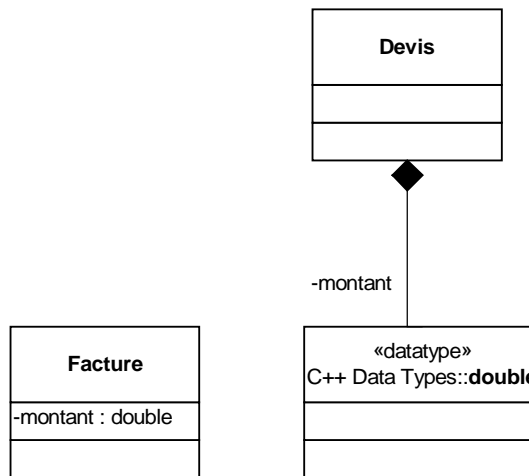


Figure 2-11 : Relations de composition entre classificateurs

La relation entre une classe et un datatype (souvent prédéfini dans la mise en œuvre) n'est pas une relation de composition entre deux types abstraits. Elle est donc généralement visualisée sous forme d'attribut. Toutefois, la nature des liens entre les éléments est la même dans le cas d'une composition par valeur.

Le sous-système constitue un niveau organisationnel supérieur, d'où la notion de macro-objet.

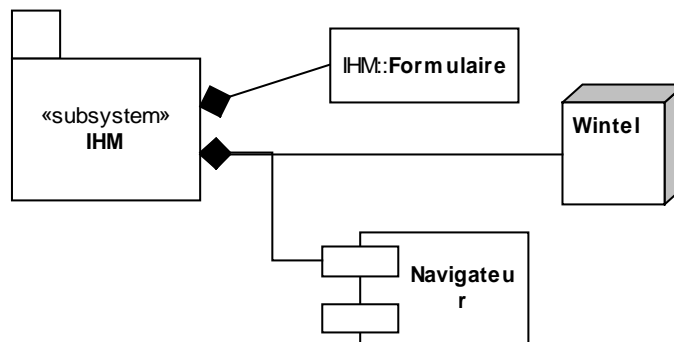


Figure 2-12 : Sous-système et éléments qui le composent

2.3.3 Annoter

De même qu'un fichier source *doit* être commenté, un diagramme doit être annoté. Peut-on imaginer un fichier source sans commentaire ?⁴ Certains commentaires peuvent être gérés

⁴ Je crains que... oui

par un outil : auteur, version, date, etc. La difficulté du « bien commenter » est réelle : le commentaire est-il simplement un bruit ? Est-il adapté au lecteur ? Est-il obsolète ?

Par ailleurs, les notes de UML pallient à une méconnaissance du formalisme ou un manque de l'outil.

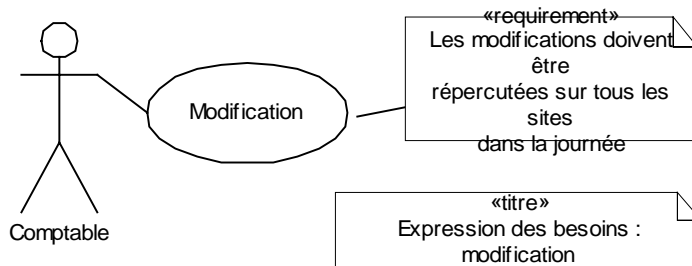


Figure 2-13 : Notes stéréotypées

Certaines situations se prêtent particulièrement à des commentaires.

Un diagramme d'interaction (séquence, collaboration) modélise un scénario qui peut être décrit dans une note

Une question en suspens est facilement injectée dans le diagramme sous forme de note

Un guide de lecture du diagramme ("pour bien comprendre, commencer par la classe orbite et dérouler les relations à partir de là").

UNIFIED MODELING LANGUAGE	2
2.1 Historique	2
2.2 Diagrammes UML.....	3
2.2.1 Modélisation des situations d'utilisation.....	3
2.2.2 Modélisation de la dynamique d'un système	4
2.2.3 Modélisation de la structure d'un système	5
2.2.4 Autres diagrammes	8
2.2.5 Relations entre éléments	8
2.3 Mécanismes communs	9
2.3.1 Etendre UML.....	9
2.3.2 Organiser	11
2.3.3 Annoter	12