Se comprendre autour d'un système d'information

Roland Billon, Médiaconstruct et Enseignant à l'ENSA de Marseille
Isabelle Fasse, Médiaconstruct, ex Enseignante à l'ENSA de Marseille
Jacques Zoller, Professeur à l'ENSA de Marseille
Pascal Tonarelli, Maître de conférences à l'UVHC
Hafida Boulekbache, Maître de conférences à l'UVHC
Stéphane Duriez, médiatiseur à l'UVHC

Table des matières

ChapitreI. Résumé et prise de conscience

Résumé :

Introduction aux langages de spécification de données et aux modèles conceptuels, dans l'objectif de dialoguer avec les partenaires des échanges, et de se faire comprendre sans ambiguïté des professionnels étrangers au Bâtiment.

Le *modèle conceptuel* qui nous intéresse ne prend pas seulement en charge la description et l'organisation des locaux, ouvrages et composants, mais aussi les études, la construction et l'exploitation de l'objet construit. L'aspect statique de l'information, préalable structurant, doit être complété par les aspects dynamiques et fonctionnels, intégrant le temps, les hommes, les contraintes.

Prise de conscience :

Dialoguer avec des partenaires par outils informatiques interposés suppose adopter une rigueur dans l'expression de ses idées, et dans la saisie des données des logiciels.

L'étudiant doit s'habituer à des méthodes formelles non seulement pour échanger le projet, mais aussi pour dominer l'évolution des outils propres à son métier. En devenant plus compétent sur les systèmes d'information, il évitera des petites et grandes catastrophes.

ChapitreII. Utiliser un langage formel pour communiquer ses idées !

A. Ecoutez-moi, je vous parle !

Lorsque vous utilisez votre langue naturelle pour vous exprimer dans les évènements courants de la vie, les interlocuteurs arrivent plus ou moins à se comprendre en général.

Encore faut-il que celui qui parle (l'émetteur) sache clairement ce qu'il veut dire (ce qui n'est pas toujours le cas) et que celui à qui est destiné le message (le récepteur) écoute vraiment (ce qui est loin d'être acquis).

Si le message est simple, l'émetteur a bien dormi, et le récepteur attentif, la communication fonctionne correctement.

Quand il s'agit d'un dialogue parlé, au moins l'émetteur peut s'assurer qu'il a été entendu. Cette précaution évite bien des ennuis.

Il en est tout autrement quand les concepts qui constituent le message sont complexes, et qu'en plus, facteur aggravant, les interlocuteurs communiquent par l'intermédiaire d'un texte, d'une image, d'un dessin, d'un écran d'ordinateur. Le récepteur est « passif », car non stimulé par le dialogue.

C'est le cas pour vous, cher lecteur.

Pour que la communication se réalise effectivement dans ce cas, il faut d'abord que l'émetteur domine la compréhension des concepts à transmettre (ce qui se comprend bien s'énonce clairement ...).

Il faut ensuite que le récepteur ait réellement envie de comprendre, sans quoi il ne déploiera pas l'effort préalable nécessaire pour surmonter sa passivité naturelle.

Il faut enfin que le support de communication utilisé (le langage) pour s'exprimer soit non ambiguë : il ne faut pas qu'il puisse exister deux interprétations possibles du même discours.

C'est tout le problème d'une langue naturelle : la complexité de l'idée à transmettre entraîne obligatoirement un allongement du texte, car il existe rarement des mot suffisamment précis pour être concis.

L'allongement du texte dénature le message à transmettre, dilue sa signification, égare l'attention dans des digressions comparatives, fait perdre du temps et le fil de sa pensée, ennuie le lecteur. Le pire est atteint quand il y a multiplication des documents transmis, écrits ou même dessinés, et multiplication des destinataires. C'est le cas des métiers du Bâtiment.

On estime chaque année que dans un pays comme la France plus de 10 milliards d'Euros sont perdus en procès, sinistres, défauts de qualité, dysfonctionnements et gaspillages divers dus à de mauvaises interprétations, et à des non-dits des pièces écrites et plans techniques.

Pertes dues à une communication déficiente entre les intervenants professionnels, et qui contribue à l'appauvrissement d'un pays et du niveau de vie de ses habitants.

Ceci malgré l'existence des outils logiciels dont une des qualités essentielles est pourtant de mieux réaliser le traitement de l'information !

B. Les méthodes de spécification formelles.

Puisque qu'un langue naturelle est « structurellement » ambiguë, sauf à déployer des talents littéraires qui ne sont pas à la portée de la majorité d'entre-nous, il a bien fallu se tourner vers des méthodes plus précises.

Pas pour s'exprimer dans la vie courante, ni développer des idées philosophiques.

Mais simplement dans un cadre professionnel, pour permettre à des intervenants de travailler ensemble sur un ouvrage commun : la conception d'une machine, d'un produit industriel, d'un ordinateur, d'une procédure informatique, d'un système d'information, d'un ouvrage du BTP.

A retenir

L'objectif d'une méthode de *spécification formelle* est de pouvoir décrire des concepts peut-être compliqués, mais finalement rendus très simples, car on peut toujours décomposer un objet complexe en sous-objets élémentaires.
Cette méthode décomposition permet au *modèle conceptuel* de représenter un aspect du réel sans erreurs d'interprétation possibles.

Autrefois le bâtiment était représenté par une ébauche de modèle, constitué de documents graphiques et écrits disjoints, juxtaposés, sans liens systémiques.

Un petit, tout petit progrès, car partiel, a été accompli par l'utilisation récente de logiciels techniques de *DAO*, *CAO* et de calcul, dans la mesure où des informations centralisées propres à chaque logiciel font l'objet d'exploitations par extraction. Par exemple, on obtient les façades automatiquement par calcul des faces cachées d'un modèle géométrique tridimensionnel, et des quantités de la même façon.

Aujourd'hui, un modèle beaucoup plus rigoureux et complet vous est proposé, sous la forme d'une base de données d'informations centralisées, organisées, normalisées, commune à tous les logiciels, donc communicables et bientôt partageables : les *IFC*.

La conception des IFC n'a pu être possible qu'à travers une méthode de *spécification formelle*.

Il existe plusieurs méthodes de spécification formelle, toutes utilisant des conventions graphiques, car chacun sait qu'un dessin est plus facile à comprendre qu'un texte. On peut citer :

- *MERISE*, surtout utilisée par les informaticiens depuis de nombreuses années.

- *NIAM* (Nijssen Information Analysis Method), basée sur des propriétés binaires d'expression de relations.

  Plus récente la méthode dite *EXPRESS-G*, dérivée du langage informatique *EXPRESS*, langage objet spécialisé dans la description des bases de données orientées « objet ». C'est celle qui a été utilisée pour la description des IFC.
- Diverses variantes de la méthode NIAM, dont la méthode Z, par *Henri Habrias*
- D'autres méthodes comme *OMT*, *Booch*, *OOSE*, utilisant toutes les concepts des langages orientés objets.
- Et enfin *UML* (Unified Modeling Language) , tentative de synthèse d'un formalisme de spécification, plus qu'une méthode, qui permet de décrire aussi bien l'aspect statique des données ou d'un modèle, que l'aspect dynamique indispensable pour décrire un traitement (procédure algorithmique) intégrant le temps.

C. Ce que vous pourrez faire avec une méthode de spécification formelle

L'utilisateur doit en connaître une ou deux dans plusieurs objectifs, que nous rappelons constamment dans ce cours :

A retenir

- **Comprendre comment le modèle a été élaboré** et les intentions des auteurs (ceci pour dominer le sujet).
- **Savoir interpréter le modèle,** c'est à dire déchiffrer sans ambiguïté les documents élaborés avec la méthode. C'est le moins que l'on puisse attendre d'un professionnel : devenir un interlocuteur « valable » qui domine le sujet.

Ces deux derniers objectifs sont ceux de tout « généraliste » des métiers de l'AEC.
Si de plus vous êtes chargé de quelques responsabilités informatiques, vous devez impérativement poursuivre les objectifs ci-après :

A retenir

- **Savoir spécifier soi-même des** *interface* **d'exploitation**, c'est à dire décrire vos souhaits à des informaticiens non compétents dans votre métier. Dans les années qui viennent, et en tant qu'utilisateur, vous serez constamment confrontés à ce type de problèmes : comment concilier les outils informatiques, logiciels et données existants dans votre entreprise, avec la nouvelle norme en marche, pour

au moins les rendre compatibles ? Une des solutions consiste à mettre en place des passerelles entre informations et traitements de natures différentes, c'est à dire des *interface*.

- **Savoir décrire des aspects complémentaires du modèle** pour les besoins propres de votre entreprise ou bureau d'étude, car les IFC ne peuvent décrire la totalité des cas. Les IFC sont d'ailleurs « ouvertes » pour des extensions à la charge de l'utilisateur.
- **Devenir administrateur, coordinateur des échanges, responsable des flux d'informations techniques entre les divers métiers,** pour ceux qui se destinent aux **nouveaux métiers de la gestion des systèmes d'information et bases de données des ouvrages et composants du Bâtiment.**

Quelle méthode choisir ?

Il suffit d'en connaître une pour acquérir le réflexe indispensable de précision, comportement de tout professionnel digne de ce nom.

Il suffit d'en pratiquer une seule pour bénéficier d'une retombée méthodologique importante qui dépasse l'objectif initial de décrire pour communiquer : un outil de réflexion efficace et systématique pour mettre de l'ordre dans ses propres idées, et dans la perception que l'on a soi-même d'un problème.

C'est pourquoi dans ce cours nous avons choisi *NIAM* pour s'initier à la spécification formelle. Si ce n'est pas la plus récente, c'est celle qui certainement offre une pédagogie la plus attrayante, et permet d'exprimer certaines nuances, du moins dans l'aspect statique d'un modèle.

Puis nous évoquerons *UML*, plus générale, que vous risquez de rencontrer souvent.

Et pourquoi pas *EXPRESS-G*, puisque c'est celle utilisée par les *IFC* ?

Ce sera à vous de vous adapter, en guise d'exercice.

Enfin, rassurez-vous. Il n'est nul besoin d'être informaticien ou mathématicien pour dominer l'utilisation d'une méthode de spécification.

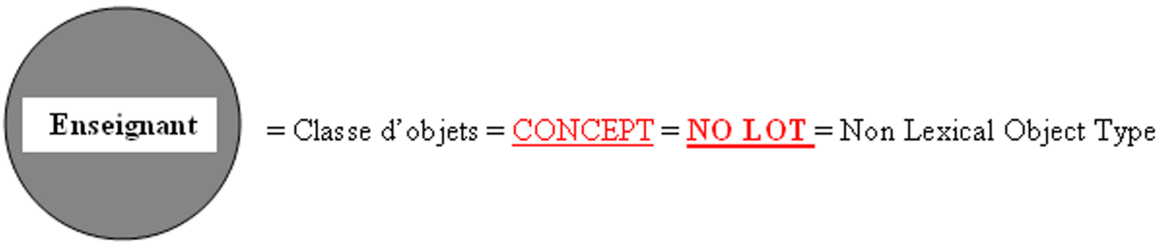
Il faut simplement un peu de bon sens, et ne pas être allergique à la réflexion, qualité qui ne doit pas vous faire défaut.

ChapitreIII. Survol de la méthode de spécification formelle « NIAM ».

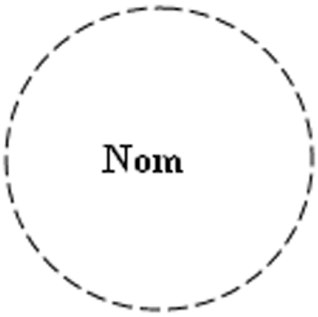
A. Les concepts et identifiants.

NIAM ou Nijssen Information Analysys Method.
On l'appelle aussi IA Method, pour Intelligence Artificielle.
Elle se formalise sous une double forme : des schémas graphiques et une série de « *phrase élémentaire* » équivalentes.
Nous nous contenterons d'expliquer les concepts principaux de ce langage graphique, c'est à dire la syntaxe et quelques règles sémantiques, pour être capable de lire un schéma, et de formaliser des « idées ». Le lecteur pourra compléter sa formation en se reportant de préférence aux ouvrages écrits par *Henri Habrias*, professeur à l'université de Nantes, ou aux ouvrages édités par l'ISO, puisque la méthode est normalisée.

Il existe une forte analogie ou correspondance entre le vocabulaire syntaxique de *NIAM*, les *langages orientés objets* , et l'algèbre ensembliste.
Un *concept* au sens NIAM est une classe d'objets. Les objets qui sont regroupés sous le même concept sont porteurs d'informations communes relatives à leurs propriétés (comportement) et descriptions (attributs) vis à vis du problème (du modèle) examiné.
Par exemple, être humain, enseignant, poteau, prix unitaire ... peuvent être des *concept* pertinents.
On ne fait pas référence à un individu précis ou à une occurrence de l'ensemble. Le concept est dit *NO LOT* (non Lexical Object Type). Il se dessine par un cercle plein, porteur de la dénomination du concept :



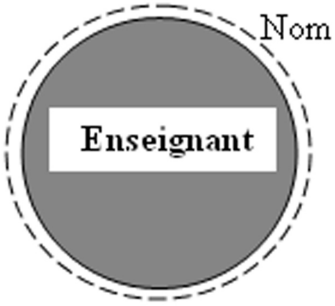
Pour faire comprendre que l'on veut prendre en compte une liste d'enseignants, c'est à dire les objets de l'ensemble « Enseignant », et que l'on a choisi de les identifier par leur nom, alors la syntaxe est un cercle en pointillé, accompagné de la nature de l'**identifiant**.
Chaque objet du concept étant identifié, on parle de Lexical Object Type, soit *LOT* :



= IDENTIFIANT = LOT = Lexical Object Type

Comment dans un schéma mettre en relation un concept et l'identifiant de ses objets ?

Il existe une forme graphique appelée « pont de dénomination ». Cette forme développée est oubliée, au profit du schéma condensé suivant, rapide à dessiner réunissant les deux cercles précédents concentriques :



= Un CONCEPT et son (unique) IDENTIFIANT

A retenir

REGLE
Dans un schéma conceptuel, il n'existe qu'un seul *concept*.
Et chaque concept n'admet qu'un seul identifiant.

Autrement dit, on ne peut pas dessiner deux fois le même concept dans un schéma. Ce qui ennuie souvent le débutant qui ne sait plus comment disposer son dessin qui devient très vite tentaculaire.

Rien n'interdit de décomposer un schéma volumineux en plusieurs schémas indépendants !

C'est même conseillé !

B. Les idées.

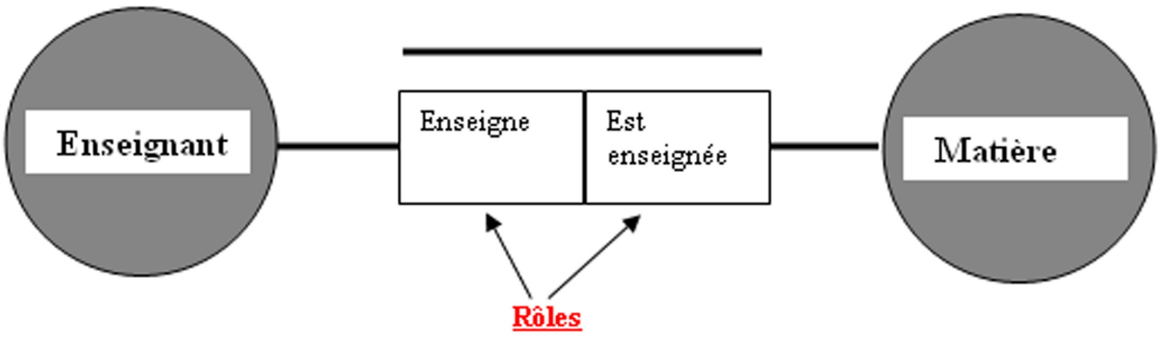
Ce terme est utilisé pour exprimer une *relation* entre deux *concept*, A et B par exemple. N'oublions pas qu'un concept s'identifie à un ensemble.

Une *idée* est donc une relation ensembliste, c'est à dire qu'elle permet de mettre en relation un certain nombre d'objets de l'ensemble A avec ceux de l'ensemble B, et inversement, car une relation entre A et B peut se lire dans les deux sens.

La nature sémantique de la relation n'est pas formalisée, c'est seulement ses propriétés qui le sont, à travers des *contraintes* binaires de cardinalité.

Un exemple, tiré du cours de *Henri Habrias* qui se propose de modéliser les données d'une école. Commençons par exprimer une idée entre les deux concepts « Enseignant » et « Matière ».

Au départ, avant toute réflexion sur les *contraintes*, on met en place le vocabulaire syntaxique composé des concepts, avec ou sans les identifiants, et des *rôles*, pour lesquels on peut formuler un petit texte (prédicat) pour fixer les idées (l'idée !).



Si on n'impose aucune contrainte de cardinalité sur la relation (ce qui est noté par le trait épais couvrant les deux rôles) et qu'on laisse le schéma en l'état, on le traduit en langue naturelle par les deux *phrase élémentaire* suivantes :

Dans le sens Enseignant => Matière :

- **Zéro, un ou plusieurs Enseignants enseignent zéro, une ou plusieurs matières.**

Dans le sens Matière => Enseignant :

- **Zéro, une ou plusieurs matières sont enseignées par zéro, un ou plusieurs enseignants.**

En effet, sans *contraintes* de cardinalité, toutes les éventualités sont possibles !

C. Symbolisme des contraintes entre deux concepts

Evidemment notre idée sur le fonctionnement de l'école est plus précise. Nous avons à notre disposition pour nous exprimer :

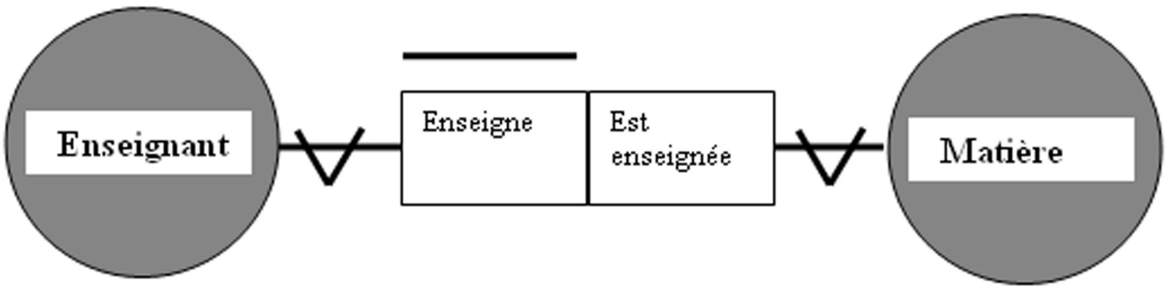
- une contrainte restrictive de « *Totalité* » qui porte sur chacun des ensembles de départ, et qui se note ∨, traduite en langue naturelle par « Tout », ou « Quelque soit »
- une contrainte « d'*Unicité* », qui se rapporte à la relation avec l'ensemble

d'arrivée, qui se note par un trait court tracé sur le rôle concerné du sens de la relation.

Si je désire préciser que :

- **tout enseignant enseigne (obligatoirement) une seule matière**
- **toute matière est enseignée par zéro, un ou plusieurs enseignants,**

le schéma devient :



Mais, confronté à la dure réalité, je m'aperçois que ce schéma décrit un modèle qui va m'attirer des ennuis, si je suis le directeur d'établissement.

D'une part pour des raisons de logistique, il est indispensable que certains enseignants puissent enseigner plus d'une seule matière.

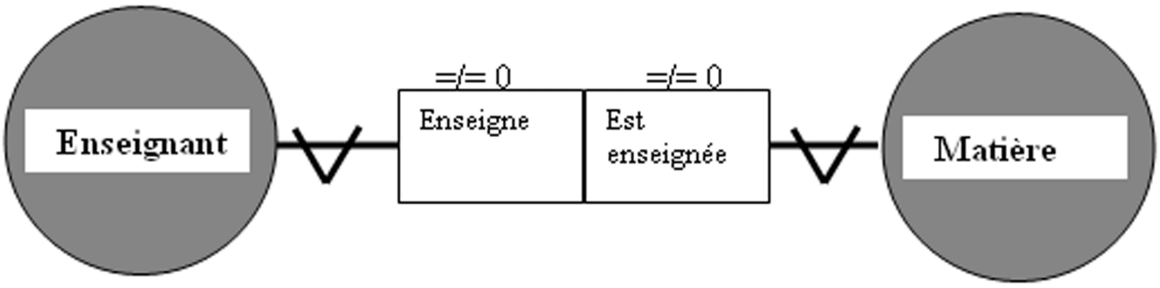
D'autre part le schéma permet qu'une matière, pourtant inscrite dans le programme pédagogique, ne soit pas enseignée. S'il n'existe aucun enseignant compétent dans cette matière, il faut que le système informatique qui exploite la future Base de Données le signale immédiatement pour alerter les responsables du recrutement.

Mon idée s'exprime alors par les deux *phrase élémentaire* remodelées :

- **tout enseignant enseigne une ou plusieurs matières**
- **toute matière est enseignée par au moins un enseignant.**

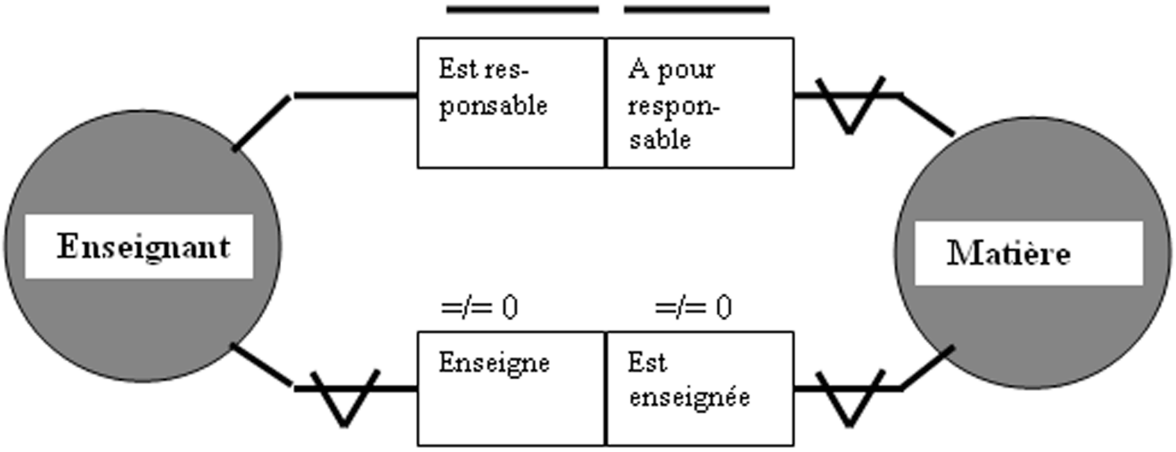
Et pour dessiner mon schéma je suis bien « embêté », il n'y a pas dans NIAM la possibilité d'exprimer des nuances de cardinalité, le modèle étant basé sur les propriétés relationnelles binaires (tout ou rien), pour avoir une correspondance avec l'algèbre ensembliste et ses nombreuses propriétés.

Par dérogation, les concepteurs de modèles qui utilisent NIAM prennent des libertés avec la méthode, et traduisent les deux dernières phrases élémentaires par le schéma « adapté » suivant, qui utilise la restriction « différent de zéro » :



Toujours dans un exemple donné par *Henri Habrias*, je veux exprimer une deuxième idée, c'est à dire le concept de responsabilité d'une matière : parmi les enseignants qui enseignent une matière, un seul est responsable de la matière.

Le schéma devient :



La nouvelle idée se lit :

- **zéro, un ou plusieurs enseignants sont responsables chacun d'une seule matière**
- **toute matière a pour responsable un seul enseignant.**

D. Contraintes entre deux idées, appelées aussi relations entre relations.

En toute logique rien ne peut interdire dans ce schéma l'existence du fait suivant :
Un enseignant est responsable d'une matière qu'il n'enseigne pas.
Si vous voulez interdire cette éventualité, par la phrase élémentaire suivante :

- **Tout enseignant qui est responsable d'une matière doit enseigner cette matière**

Alors vous ne pouvez plus exprimer une nouvelle idée au sens NIAM, car il s'agit en fait d'une contrainte entre les deux idées déjà exprimées.

Les contraintes entre deux idées permettent dans NIAM l'expression de nuances quelquefois subtiles. Les étudiants ont souvent du mal à les exprimer, car il faut aller au fond de son idée, la décomposer obligatoirement dans une ou plusieurs des 5 contraintes élémentaires suivantes :

Contrainte de **disjonction ou exclusion** : les relations assurées dans une idée ne doivent pas l'être dans l'autre idée (ou les autres idées). Cette contrainte est notée

Schéma de la contrainte de disjonction ou exclusion

Contrainte de *Totalité* : Il ne peut exister d'autres idées que celles notées entre les couples de concepts liés par les relations.

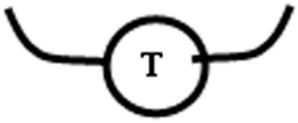


Schéma de la contrainte de totalité

Contrainte d'**égalité** : les deux (ou plusieurs) relations doivent être vérifiées à la fois.

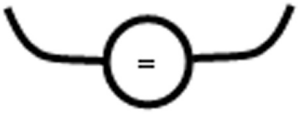


Schéma de la contrainte d'égalité

Contrainte d' '*Unicité* : plus délicat à comprendre : les objets des ensembles qui sont pris par exemple dans deux relations forment un couple unique, qui ne peut à nouveau servir.

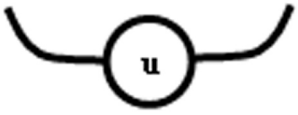


Schéma de la contrainte d'unicité

Contrainte de **sous-ensemble** : Une relation définit dans chacun des concepts reliés un

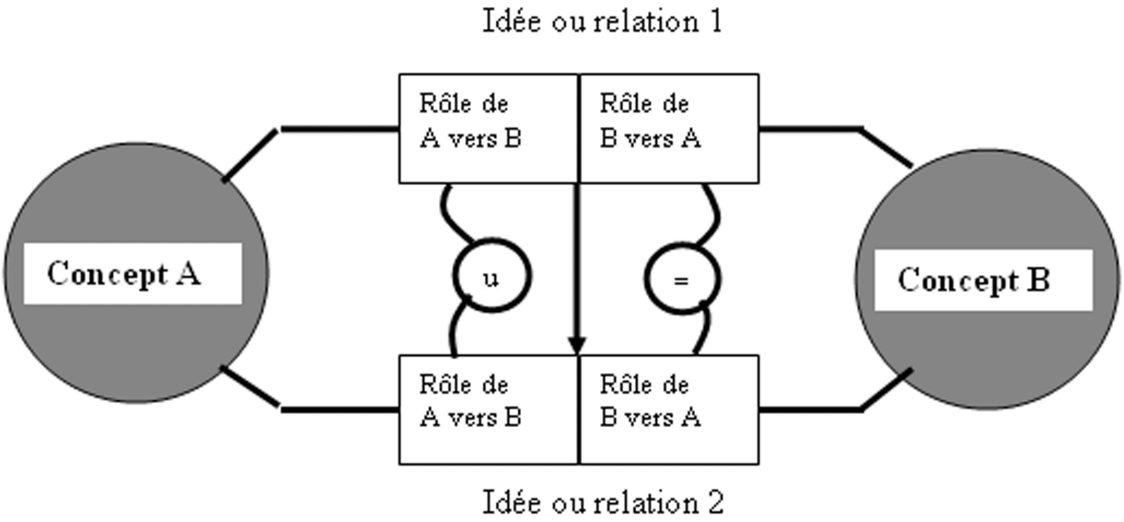
sous ensemble d'objets, par rapport à l'autre relation qui utilise l'ensemble des objets en présence dans les deux concepts (classes).

Notée par une flèche orientée du sous ensemble vers l'ensemble

Schéma de la contrainte de sous-ensemble

Ces *contraintes* peuvent porter soit sur un seul sens des relations, soit sur les deux sens à la fois. Leur emplacement sur les *rôles* sera donc différent.

Dans l'exemple ci-après, la contrainte d'*'Unicité* porte sur le sens A => B des deux relations, la contrainte d'**égalité** sur le sens inverse, et la contrainte de **sous ensemble** définit les objets pris dans la relation 1 comme le sous ensemble des objets de la relation 2.



E. Des exemples pour bien comprendre

Le concept de contraintes sur relations n'est pas toujours simple à exprimer.

Reprenons la phrase élémentaire citée plus haut concernant les deux idées entre enseignant et matière :

- **Tout enseignant qui est responsable d'une matière doit enseigner cette matière**

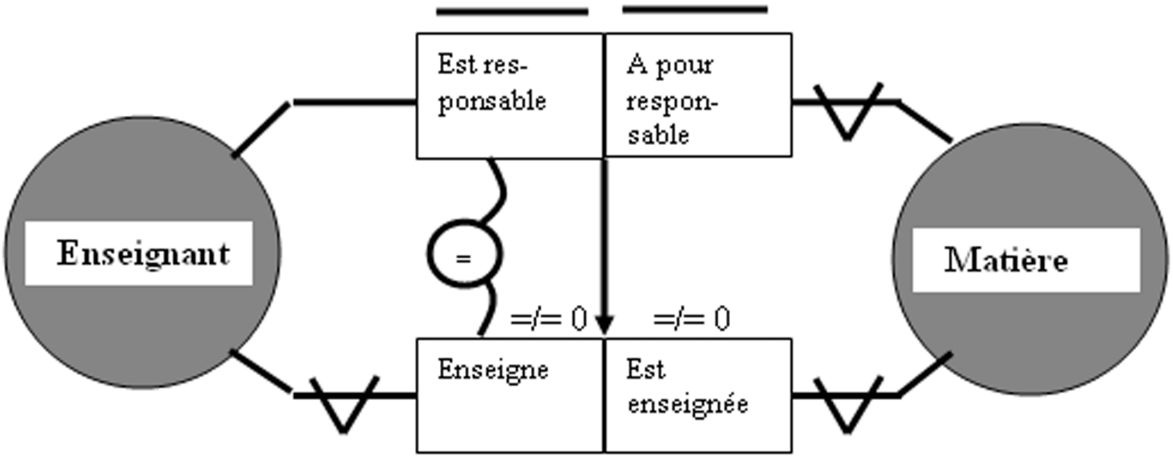
Pour aller au fond de son idée, en utilisant seulement l'une des 5 contraintes à notre disposition, on s'aperçoit que sémantiquement cette phrase concerne une contrainte d'**égalité** :

- si pour un enseignant pris dans l'ensemble des enseignants, la relation dite de « responsabilité » est vérifiée, alors celle dite « d'enseignement » doit l'être. Cette contrainte ne s'applique qu'aux enseignants, pas aux matières. Donc elle qualifie uniquement les relations dans le sens « Enseignant » => « Matière ».

Mais en plus, et c'est une réalité, il se trouve que la relation de responsabilité définit des couples d'objets Enseignants-Matières qui sont des **sous ensembles** de la totalité des couples d'objets Enseignants-Matières définis par la relation d'enseignement. Information précieuse

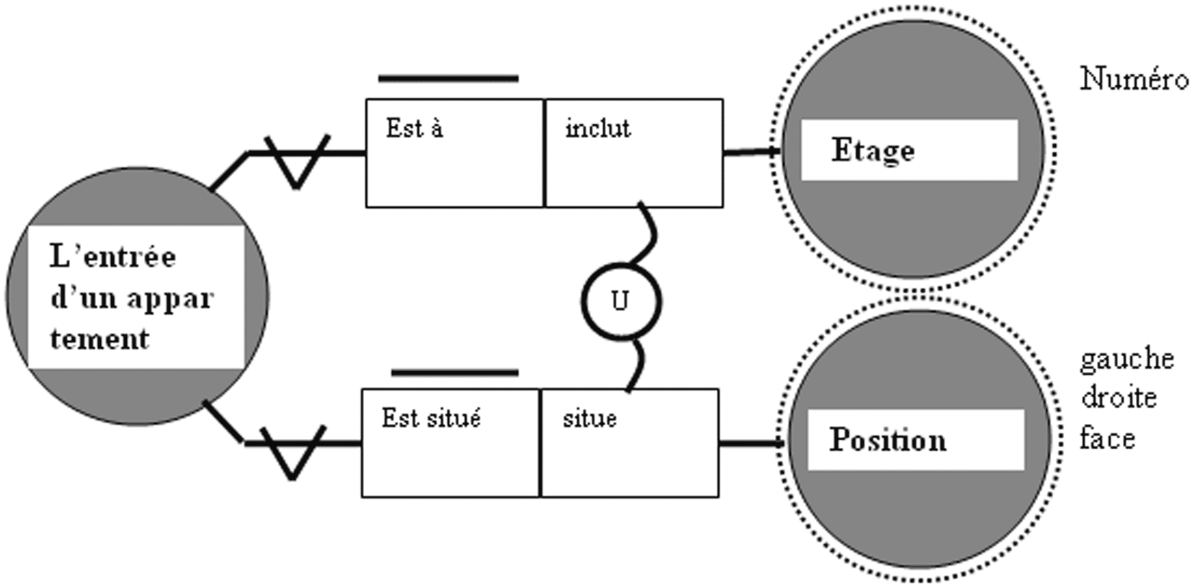
pour celui qui va organiser la base de données de l'Ecole, et qui mérite d'être notée.

Le schéma complet de ces deux idées se note :



Un autre exemple concernant la contrainte d' '*Unicité*.

Je veux exprimer la situation topologique des appartements d'un immeuble, en utilisant les deux critères de localisation d'un appartement : l'étage dans lequel est situé son entrée(ce qui n'interdit pas les duplex), et sa position par rapport au palier de l'ascenseur : face, gauche, droite.. Traduisez en phrases élémentaires le schéma suivant :



Pourquoi la contrainte d' '*Unicité* ne porte que dans le sens Etage/Position => Entrée ?

Chaque fois que dans les relations, un couple d'objets Etage/Position est pris, celui-ci est unique et ne peut qualifier un deuxième appartement.

D'ailleurs, pour preuve, remplacez gauche, droite, face par les numéros 1,2,3.

Vous pourrez nommer chaque appartement par la concaténation du numéro d'étage et de sa position, sans ambiguïté. Exemple : Appartement 32 : troisième droite.

Les gardiens d'immeuble et les hôteliers sont incollables en NIAM !

Il nous manque un exemple de contrainte d'**exclusion**. Elles sont les plus nombreuses dans les modèles conceptuels, comme dans la dure réalité de la vie.

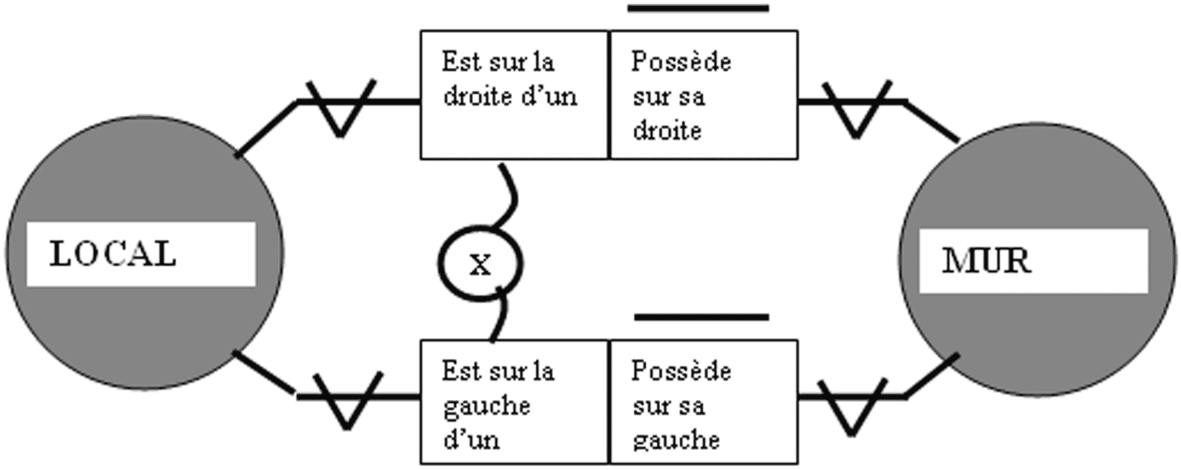
Prenons un exemple dans le Bâtiment, pour exprimer le fait apparemment banal **qu'un mur sépare deux locaux** (l'extérieur étant considéré comme un local particulier).

Cette constatation est toujours vraie dans toutes les dimensions si le mur a été préalablement découpé de manière à ce qu'il ne sépare que deux locaux seulement. On dit que le composant mur, ainsi que ses deux faces verticales latérales, ou « *nus* », sont *homogènes*.

Ce cas topologique exclut les murs « paravent » qui s'avancent dans une pièce sans la séparer. Mais une disposition permet de les prendre en compte.

On utilisera donc les seuls concepts « Mur » (qui regroupe toutes les variétés façades, refend, cloisons ...) et « Local ». On exprimera les relations topologiques des deux types d'objets entre-eux par les idées « Local à gauche d'un Mur », « Local à droite d'un Mur ».

Traduisez les phrases élémentaires du schéma suivant :



L'**exclusion** portant sur les relations topologiques du local est essentielle. Constatant qu'un local ne peut être à la fois sur la droite et la gauche d'un mur, avec les précautions de redécoupage homogène que nous avons prises, nous avons topologiquement organisé la base de données du Bâtiment, défini une orientation à tout composant associé aux murs, et autorisé les relations de continuité de voisinage entre les pleins et les vides. Nous verrons que ces deux relations sont fondamentales dans les *IFC*.

Remarque : Nous n'avons pas précisé la définition exacte du concept *local*.

Les relations partant du local, qui se lisent « Tout local est sur la droite de zéro, un ou plusieurs murs » impliquent qu'un local n'est pas forcément complètement clos par des murs. On peut définir des « murs virtuels » pour que ce soit le cas.

F. Pour mémoire, classification ensembliste des relations binaires entre concepts.

Pour l'instant, nous n'avons pas donné de nom aux relations entre concepts.

Leur nature sémantique, nous avons vu qu'elles étaient « binaires » du point de vue mathématique, dépend de la signification donnée par l'utilisateur. Selon les cas, ce seront des relations de proximité, de voisinage, de décomposition, d'appartenance, d'affinité, de correspondance ... La liste est illimitée, ouverte.

La seule propriété commune de ces relations binaires est de mettre « en correspondance » certains objets de l'ensemble A avec ceux de l'ensemble B.

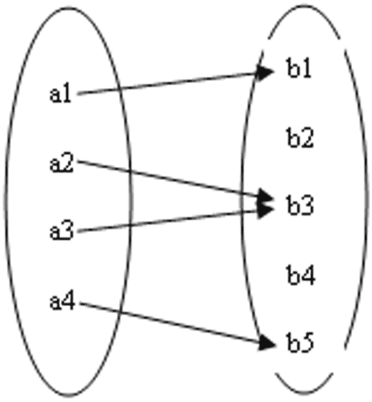
Si l'on veut être plus précis sur une classification de la nature structurelle de ces relations, on peut faire appel aux propriétés identifiées en algèbre ensembliste, qui distingue par exemple 8 cas possibles issus de la combinatoire des contraintes appliquées aux relations binaires entre deux ensembles (en éliminant les symétriques) :

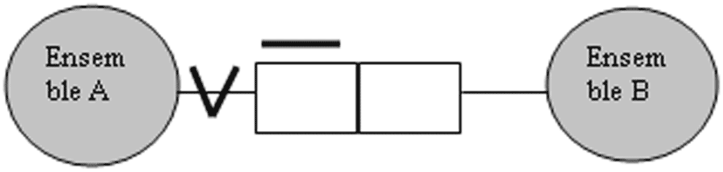
- l'injection
- la bijection
- la surjection
- l'application simple
- la fonction
- la relation 1-N
- la relation 1-1
- la relation N-M

On peut représenter chaque relation ensembliste par un schéma NIAM, et inversement chaque idée d'un schéma NIAM peut être qualifiée par une relation ensembliste.

A titre d'exercice, pour ceux qui n'ont pas oublié leurs cours de mathématiques du primaire ou du secondaire, vous pouvez vous amuser à reproduire les schémas des 8 relations ensemblistes citées plus haut.

Par exemple, le schéma de « l'application simple », accompagné de son diagramme de Venn :





A tout objet de l'ensemble A correspond un seul objet de l'ensemble B.

G. Relations d'héritage, dite aussi de sous-typages entre concepts.

Cependant, il existe une relation particulière dans la méthode NIAM, comme dans

G. Relations d'héritage, dite aussi de sous-typages entre concepts.

Cependant, il existe une relation particulière dans la méthode NIAM, comme dans n'importe quelle méthode de spécification formelle, qui est plus importante que les autres : celle de l'*héritage*.

C'est à dire que les objets de l'ensemble B ont un lien de ressemblance, de parenté avec ceux de A, dont ils sont issus. Bien évidemment, cette relation n'a d'intérêt que si effectivement les deux ensembles sont réellement de même nature pour une opération prévue, et si on peut distinguer une hiérarchie entre les deux ensembles A et B.

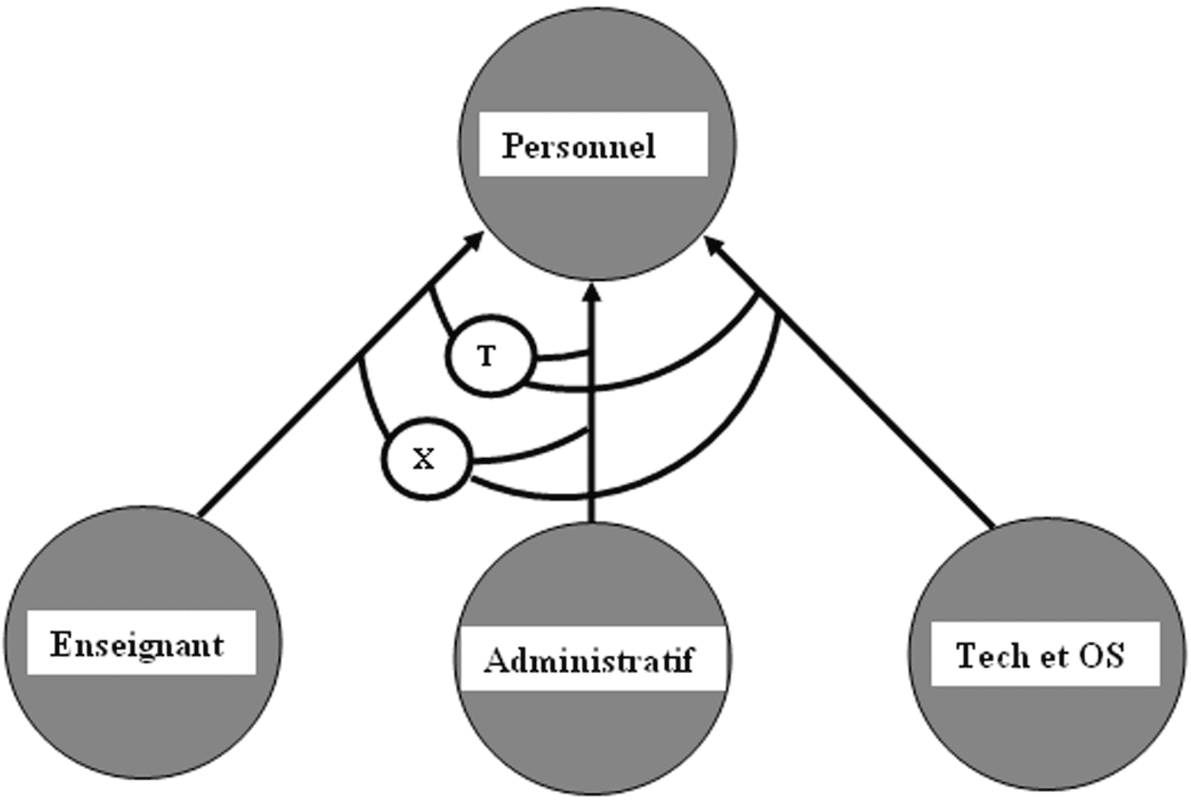
C'est le cas lorsque on a consacré de l'énergie à décrire et renseigner les propriétés et le comportement des objets de la classe (ou concept, ou type) A. Si ensuite on veut définir un sous ensemble B issu de la classe A, pour le spécifier plus en détail, on aimerait bien ne pas être obligé de recommencer la description depuis le début. Cet objectif a donné un nom à cette fonction particulière de sous-ensemble : l'*héritage*, encore nommé *sous-typage*.

On la note par une flèche entre deux concepts dont le sens est obligatoirement du concept qui hérite (sous-type, ou descendant, ici B) vers celui qui donne (le sur-type, ou ascendant, ou parent, ici A).

En général, on interdit, pour des raisons de complexité de gestion, ce qu'on appelle l'héritage multiple : une classe n'hérite pas de plus d'un parent. En revanche, un parent peut avoir plus d'un descendant. De cette manière, l'ensemble des relations d'héritage dans un schéma NIAM est représenté par un arbre.

Reprenons l'exemple de *Henri Habrias* décrivant une école. L'ensemble « Personnel » de cette école est composé exclusivement d'Enseignants, d'Administratifs et de Techniciens et Ouvriers Spécialisés.

Ce qui se représente :



La contrainte de totalité portant sur les trois relations indique qu'il n'y a pas d'autres type de personnel dans l'école que les trois sous-types mentionnés, dont la somme des occurrences représente bien la totalité du personnel.

La contrainte d'exclusion indique qu'un membre du personnel ne peut être présent dans plus d'une classe. Par exemple un enseignant ne peut aussi avoir une fonction administrative. A titre d'exercice, réfléchissez sur la fonction de chercheur et d'enseignant-chercheur, et complétez le schéma avec ces deux concepts.

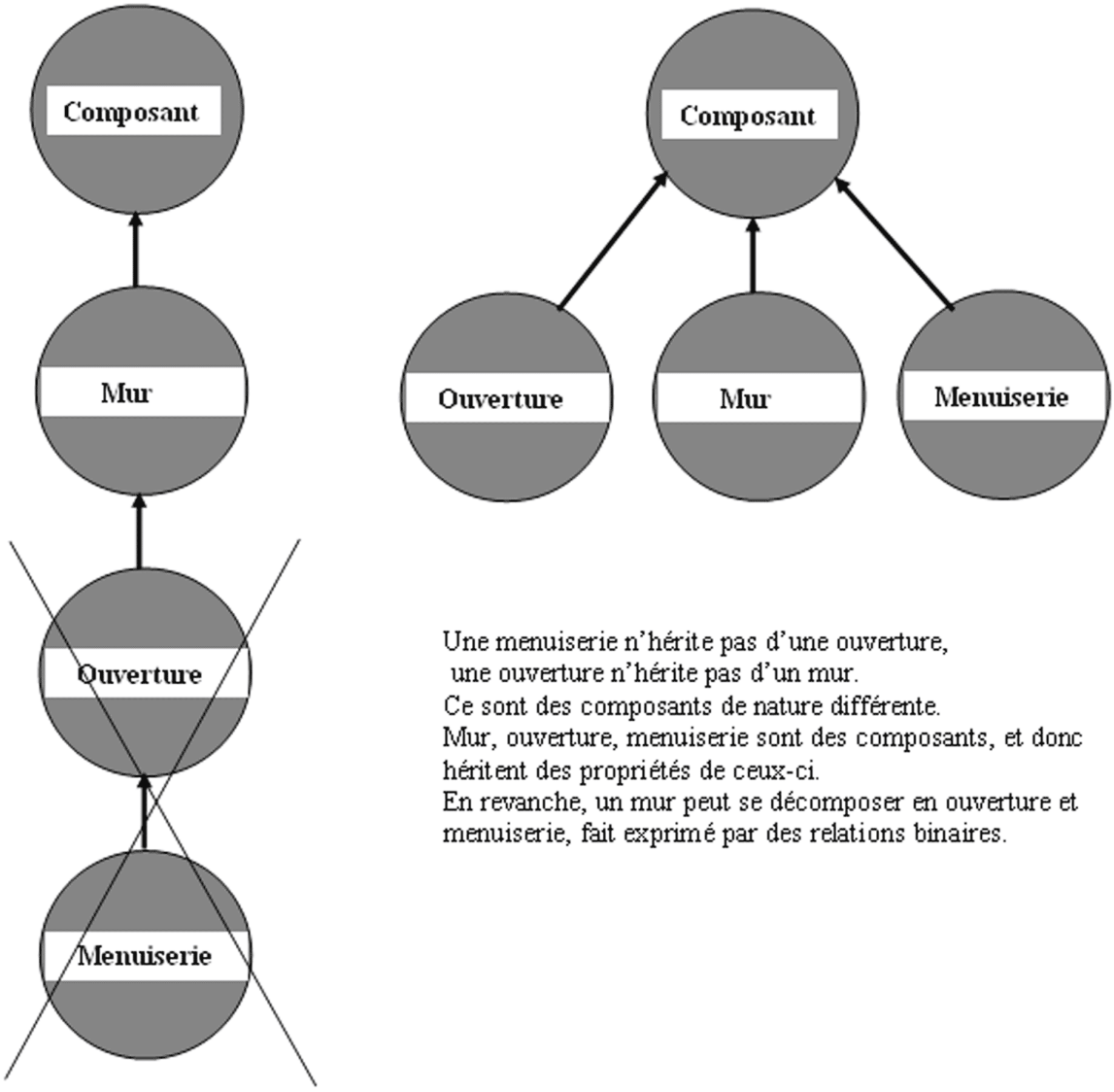
Cette relation d'héritage est tellement importante dans les systèmes d'information qu'elle en est devenue structurante. Dans un modèle conceptuel, on commence d'abord par décrire l'ensemble des relations d'héritage entre les concepts. Ensuite on complète le schéma par les relations binaires, les idées.

H. Ne pas confondre Héritage et Agrégation.

Le débutant, quand il conçoit un schéma, a souvent du mal à faire la différence entre une relation d'*héritage* et une relation binaire de composition (d'appartenance, dite aussi d'agrégation).

Exemple : développez l'arbre d'héritage et les idées intéressantes entre les concepts COMPOSANT, MUR, OUVERTURE et MENUISERIE.

Surtout, ne construisez pas l'arbre d'héritage du schéma de gauche suivant :



Si vous hésitez entre une relation d'héritage et une relation de composition, posez-vous la question suivante :

Le concept B est-il « **une sorte de** » du concept A (Par exemple, une ouverture est-elle une sorte de mur ?). Quelquefois, l'héritage et l'appartenance sont superposés.

Par exemple, un étage, une pièce, un appartement sont des sortes de locaux.

Mais également, une pièce est dans un appartement qui est dans un étage, chacun d'eux étant aussi dans un local (plus vaste).

I. Les propriétés descriptives des concepts (aussi appelées attributs).

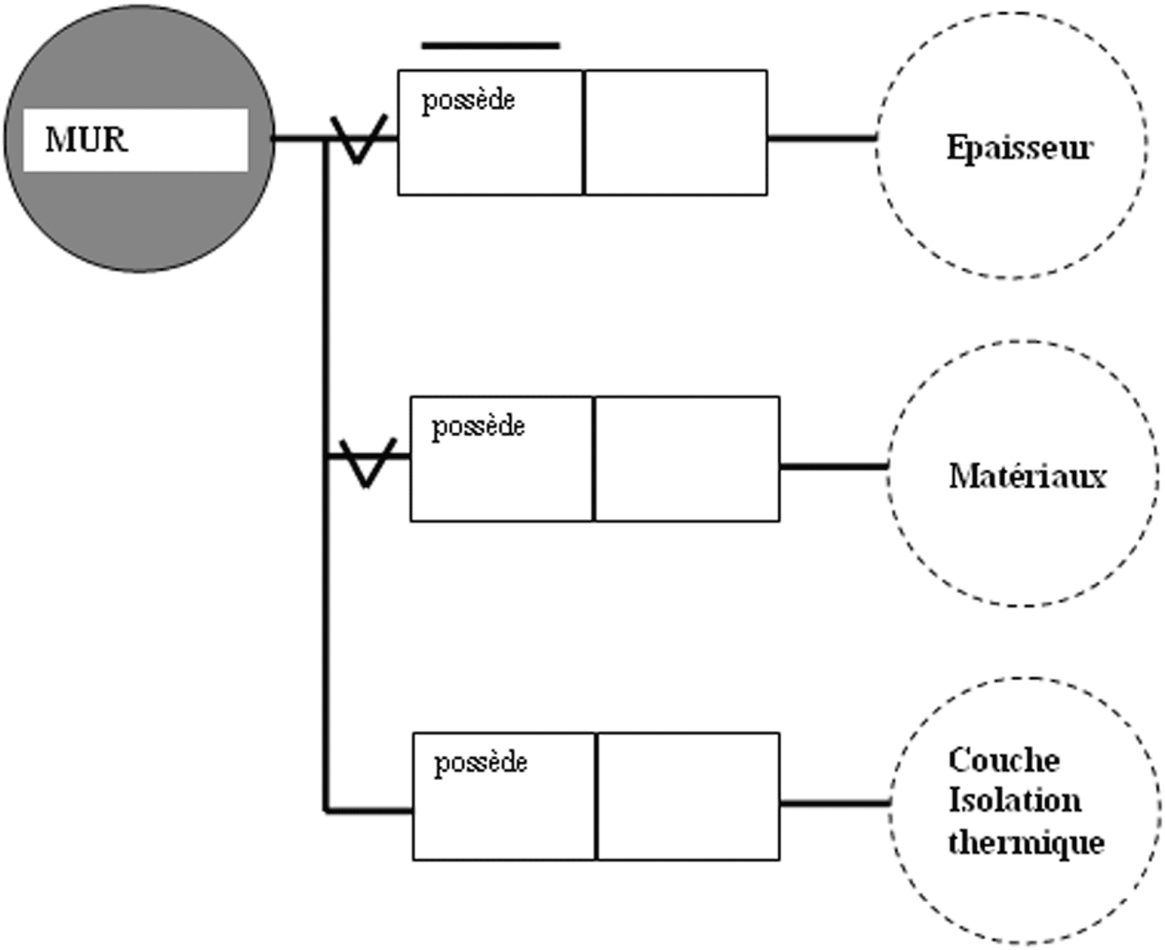
Nous en avons déjà examinée une : l'**identifiant**, qui est une propriété particulière. Par exemple, pour le concept « enseignant », c'est son « nom ».

Mais on peut associer autant d'informations qu'il est nécessaire à une classe, pour les besoins d'une procédure.

Par exemple, pour l'enseignant, il faut savoir son numéro de sécurité sociale, son adresse, sa date de naissance, sa nationalité, ses diplômes, s'il est marié, s'il a des enfants, s'il a d'autres activités, la puissance de sa voiture ... toute information qui sera exploitée pour des tâches administratives (fiche de paye, remboursement de frais), pédagogiques (affectation), et peut être d'autres encore ...

En général, on donne le statut de « *propriété* » à toute information qui ne fera sans doute pas l'objet d'un concept ou d'une relation. C'est une information « terminale » et son statut dépend du contexte. Par exemple, si le fait d'être marié implique que l'on doive connaître avec qui, alors on est obligé de développer une idée, ou relation, dont le rôle devient « est marié (e) avec ».

On représente une propriété (ou plusieurs) de la manière suivante :



Ce qui se lit dans le sens *concept => propriété* :

Tout mur (sous-entendu morceau de mur homogène) **possède**

- **une seule épaisseur**
- **zéro, un ou plusieurs matériaux** (zéro pour une paroi virtuelle)

Un mur peut avoir une isolation thermique (autre façon de dire un mur a zéro, une ou plusieurs couches d'isolation thermique)

Dans l'autre sens, on ne dit rien, car il n'y a pas d'ambiguïté possible.

J. Conclusion sur cette première approche de la spécification formelle.

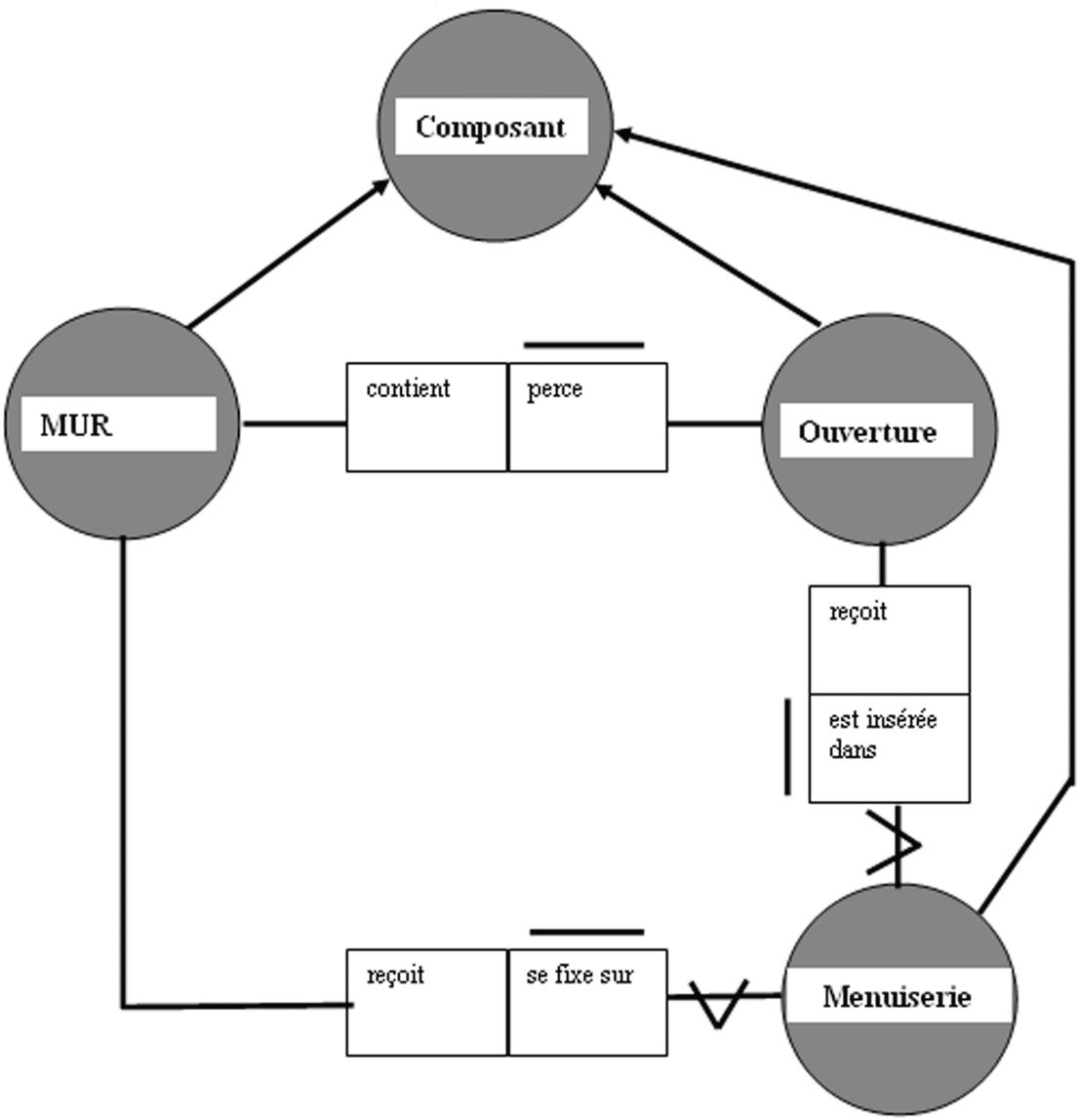
A retenir

Bien évidemment, nous avons seulement survolé une méthode de spécification formelle, et il ne faut surtout pas vous prendre déjà pour un spécialiste.
Vous êtes capables de lire un schéma NIAM, de l'interpréter, de le modifier ou de le compléter, chaque fois que vous participerez à un débat sur la formalisation d'un modèle, l'exploitation d'une base de données, la définition des données à saisir, le renseignement d'un projet.
Vous vous trouvez en face d'une autre méthode ?
Cela n'a pas d'importance. Vous vous adapterez facilement, car vous avez acquis la connaissance des principaux concepts des méthodes de spécification, qui se ressemblent.

Mais surtout vous avez développé votre capacité à analyser les problèmes, à les décomposer en problèmes élémentaires, à les formaliser avec assez de précision pour qu'ils soient compréhensibles par tout professionnel du même métier, ou d'un métier différent.

Exercice

Le schéma NIAM suivant exprime quelques idées entre les constituants possibles d'un mur.



En résumé, une menuiserie se localise dans une ouverture, mais se fixe dans le mur.

Ce schéma est un début de modèle. Il faudrait le « spécialiser » encore, en distinguant par exemple les murs façades des murs intérieurs, en y ajoutant les composants tableaux et sous-face, appuis, seuil (d'une ouverture), volets roulants, et en distinguant les relations de composition et d'accrochage (jonction mécanique) nécessaires, en décrivant quelques propriétés relatives aux couches d'un mur composite. Ceci par exemple pour faire comprendre qu'une menuiserie se fixe certes dans le mur, mais pas dans la contre-cloison ou dans l'isolation ...

A titre d'exercice, à vous de le compléter.

ChapitreIV. Modèles Statique, dynamique et fonctionnel.

A. Les aspects inséparables d'un système d'information.

Nous venons d'examiner à travers la méthode *NIAM* un seul aspect de l'information contenue dans un modèle conceptuel : sa structure, son aspect « statique ».

Cette information, stockée dans une base de données, est destinée

- à être exploitée par des acteurs à travers des logiciels (l'aspect fonctionnel),
- à évoluer dans les différentes étapes de la vie de l'objet modélisé (l'aspect dynamique).

Il est obligatoire de compléter l'information du modèle conceptuel statique d'un Bâtiment, car vous savez quelle importance représentent les fonctions et le temps dans les activités complexes de conception et surtout de construction.

Dans la procédure de conception du modèle lui même, on choisit de commencer par modéliser la structure de l'information. C'est l'aspect le plus important, car il précise la sémantique des objets et leurs relations. C'est aussi le plus difficile.

C'est pourquoi de nombreuses méthodes se concentrent sur ce seul aspect statique, dans le but de décrire une base de données, comme *NIAM* et *EXPRESS-G*.

Cette dernière méthode sera évoquée plus loin, car elle est utilisée par les concepteurs du modèle des *IFC*.

D'autres méthodes permettent de modéliser à la fois l'aspect statique, et l'aspect dynamique, comme par exemple *UML*.

Cher lecteur, votre rôle professionnel n'est pas seulement de contrôler la cohérence des informations techniques qui seront stockées dans les bases de données.

Votre rôle est surtout de les exploiter dans votre métier, lequel est en relation avec les autres métiers de l'AEC.

Pour continuer d'être un interlocuteur valable, vous devez acquérir des notions sur les nouveaux concepts qui décrivent les deux autres aspects d'un *système d'information*.

Vous pourrez ainsi en tant qu'utilisateur, client d'un système informatique, participer à

- l'élaboration d'un cahier des charges (spécifications) avec votre vue « métier »,
- valider des développements, les tester aux différentes étapes,
- évaluer des performances,

- proposer des modifications, envisager de nouvelles fonctions,de nouveaux logiciels,
- vous serez le mieux placé pour intégrer ces nouveaux outils dans la vie de votre entreprise, élaborer des méthodes, former le personnel technique.
- Et enfin peut être vous spécialiser dans les opérations de synthèse des études d'une opération de construction, et en devenir l'administrateur du système d'information interopérable, nouveau métier né de cette méthode de travail.

A retenir

Trois aspects décrivent le modèle conceptuel d'un objet complexe :

- **l'organisation statique** (sa structure, l'organisation d'une base de données)
- **les fonctions** (son utilisation par des acteurs à travers des logiciels)
- **l'organisation dynamique** (l'évolution dans le temps des informations modélisées, les divers états des éléments du modèle).

Le schéma suivant illustre ces trois aspects inséparables d'un modèle :

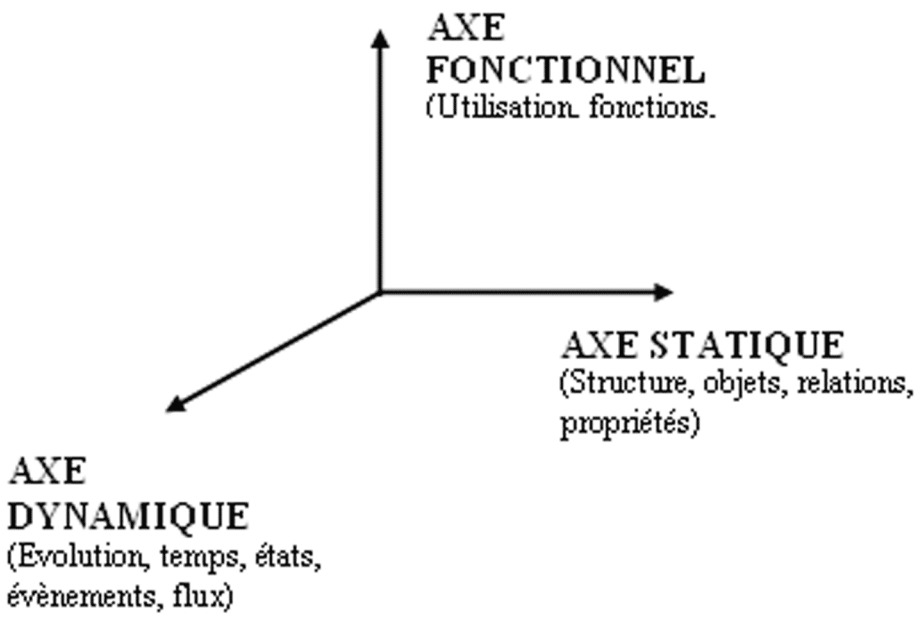


Schéma illustrant les trois aspects inséparables d'un modèle

B. Le cycle de vie d'un logiciel.

L'utilisateur est un *acteur* particulier, dont l'action est prévue à la fois à l'extérieur du système d'information, et à l'intérieur.

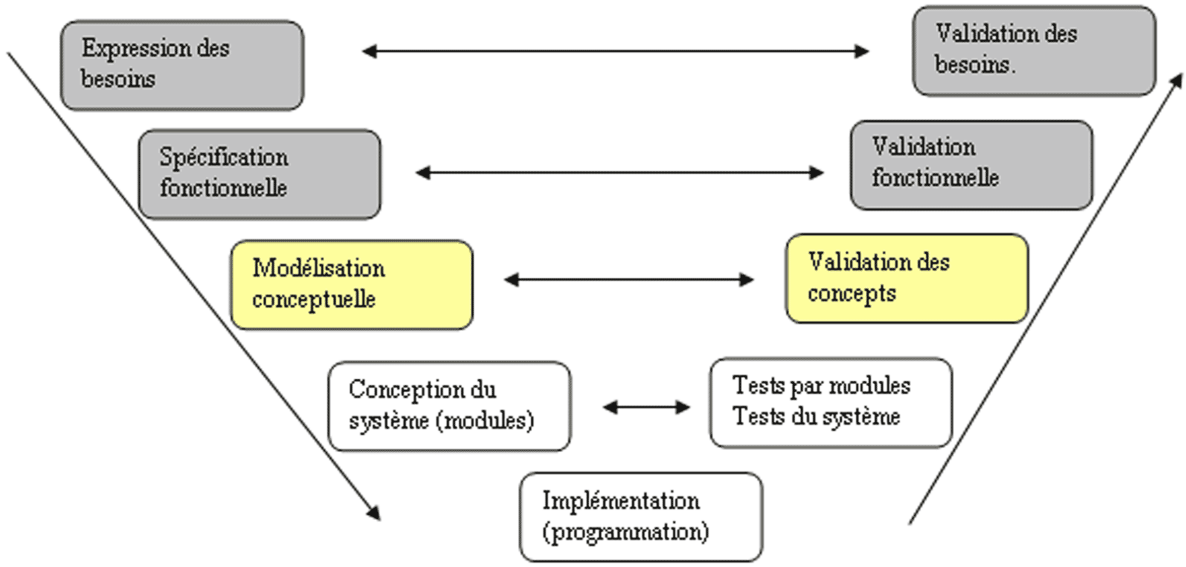
En tant que destinataire du système, il intervient obligatoirement dans certaines phases de son élaboration. Cette participation est évidente quand il s'agit de modéliser et développer un logiciel nouveau, de modifier un logiciel existant, d'ajouter une interface.

Mais indirectement, il participe également à l'évolution d'un logiciel standard, car l'éditeur s'appuie sur ses critiques pour améliorer le produit, tout au long de son existence. En principe.

Pour savoir à quel moment vous devez intervenir dans l'élaboration d'un modèle, il faut connaître les grandes étapes de la procédure de développement appelée « *cycle de vie* ».

Il n'est nul besoin d'être informaticien pour comprendre la nécessité de ces étapes lorsque les partenaires de développement sont nombreux et les fonctionnalités du logiciel ambitieuses.

Elles sont résumées par le schéma suivant, dit « cycle en V » :



Cycle de vie d'un logiciel - Cycle en V

La flèche descendante aligne d'une façon linéaire les étapes de la conception du système d'information, la flèche montante celles de la validation. Les doubles flèches horizontales mettent en regard les types d'information à valider, dans des procédures qui peuvent définir des boucles (des retours en arrière). Les trois dernières étapes restées en blanc appartiennent exclusivement aux informaticiens.

Les *acteur* extérieurs au système d'information, c'est à dire les professionnels des métiers, sont les seuls compétents pour intervenir dans les étapes grisées. Ils doivent également intervenir dans les deux étapes en jaune qui concernent la modélisation conceptuelle, en coopération avec les informaticiens.

La responsabilité du professionnel non informaticien, détenant le savoir-faire métier, est donc écrasante, puisque sa présence est obligatoire dans six étapes sur neuf. Il faut savoir que pour un gros développement, le cycle de vie du logiciel, de sa conception à son exploitation satisfaisante, peut durer de 5 à 10 ans.

Tout d'abord l'acteur responsable des fonctions du système doit recenser les besoins d'une façon exhaustive, puis ensuite définir les fonctions du futur système dans un document contractuel sous la forme de spécifications.

Peu de méthodes existent pour l'aider dans cette tâche, si ce n'est une expérience professionnelle précise sur les thèmes abordés.

C. Quand le modèle conceptuel tient lieu de contrat !

Les documents traditionnels rédigés en langue naturelle sont peu précis, difficiles à comprendre par des informaticiens étrangers au métier de l'utilisateur, et donc sources d'erreurs d'interprétation, d'allongement des délais de développement, de dépassement des coûts, et au pire, mais c'est très fréquent, de graves dysfonctionnements du système d'information, décelés quelquefois bien après la livraison, en pleine exploitation du système.

C'est pourquoi la tendance est à l'adoption comme pièce contractuelle d'un *modèle conceptuel*, effectuée à l'étape suivante (en jaune dans le cycle de vie).

Un premier avantage est que ce nouveau type de document est rédigé dans une forme normalisée, qui évite toute ambiguïté. Le modèle conceptuel constitue l'interface contractuelle entre les personnes compétentes dans le métier, et celles compétentes en informatique.

Un deuxième avantage réside dans la suppression des discontinuités de représentation de l'information lors du passage d'une étape à l'autre. Lorsque les informaticiens prendront en charge le modèle conceptuel pour concevoir le système informatique, les *concept*, *propriété relation* et « **méthodes** » définies dans le *modèle conceptuel* seront intégralement reprises dans les étapes suivantes.

Cette performance remarquable se nomme la « *traçabilité* ».

Mieux, il existe des automatismes pour traduire le modèle conceptuel en instructions de **langages** *objet*, dans un atelier de génie logiciel (*AGL*). Et pas seulement pour définir une structure de base de données, mais aussi pour définir les fonctions internes du logiciel (voir les « *use case* » de *Jacobson*, évoquées plus loin).

A retenir

Lors d'un développement informatique, la tendance est de remplacer l'ancien cahier des spécifications externes par le *modèle conceptuel* comme description contractuelle du système à réaliser.
Ce document offre plusieurs avantages : il est normalisé, non ambiguë, compréhensible quelque soit le métier des partenaires, permet la « *traçabilité* », un contrôle permanent

du développement face aux objectifs, une meilleure intégration de l'outil dans l'entreprise, et permet de réaliser des économies tout en améliorant la qualité du produit.

Pour que ces progrès techniques et économiques dans la conception des systèmes d'information deviennent une réalité banale, un préalable est à réaliser : les professionnels des métiers de l'AEC doivent également maîtriser l'élaboration des modèles conceptuels !

Ils doivent acquérir la pratique d'une ou plusieurs méthodes de spécification formelle normalisée, comme *UML*, une des plus générales.

Cet enseignement doit faire partie de la culture de base d'un architecte et d'un ingénieur en AEC.

Cette matière supplémentaire à inscrire dans les programmes pédagogiques est également justifiée pour d'autres raisons que les coopérations entre partenaires pour l'élaboration de systèmes d'information.

Le principal mérite d'une modélisation conceptuelle est une auto-formation sur les aspects complexes de n'importe quel problème de gestion, technique ou méthodologique. Elle aide à formaliser, expliquer un mécanisme, en vue de la formation du personnel d'entreprise, en vue de la mise en place de nouvelles techniques et méthodes.

Mais elle aide aussi à mettre de l'ordre dans ses propres idées, à rendre explicite ce qu'on perçoit confusément. Quel bénéfice pédagogique pour le décideur, le chercheur, le formateur et tout professionnel !

D. Le statut d'acteur dans un modèle dynamique

Dans la méthode NIAM, nous avons vu qu'un acteur est un objet comme un autre, modélisé à travers un concept (ensemble). C'est normal, dans un aspect limité aux propriétés statiques d'un modèle conceptuel.

Au contraire, dès qu'il s'agit de préciser des fonctions, et de les utiliser dans des procédures qui font intervenir le temps, l'utilisateur d'un système d'information devient un concept prépondérant : non seulement il est le destinataire du logiciel (le client), mais en plus il est actif vis à vis des procédures internes de ce logiciel :

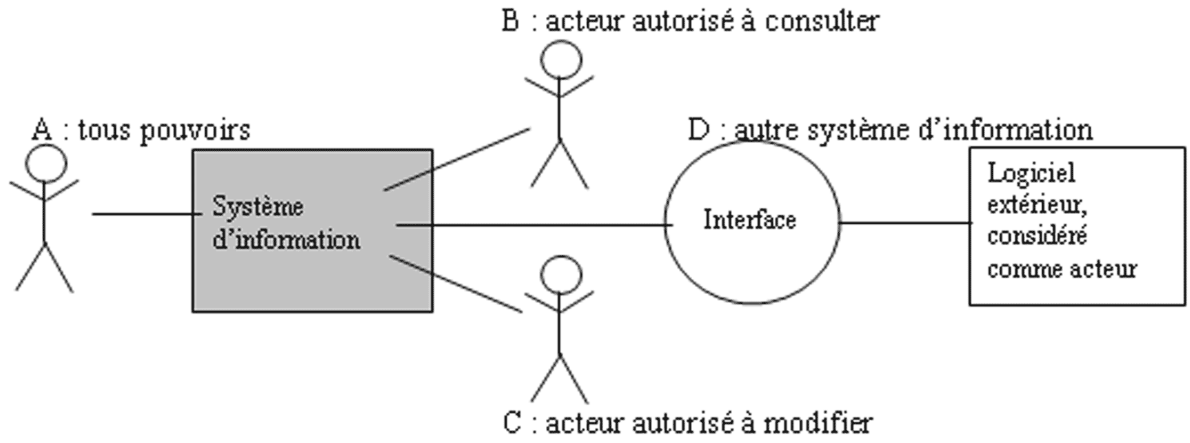
- il est en interaction permanente avec lui,
- à travers les interfaces matérielles (l'écran, le curseur-souris, le clavier)
- les boîtes de dialogue,
- il contrôle la cohérence des données externes,
- examine la pertinence des résultats,
- et découvre souvent des erreurs du logiciel !

Il intervient pour déclencher ce qu'on appelle des « *évènement* » dans la terminologie des langages à objets. Un événement intérieur à un logiciel peut en déclencher d'autres en cascade.

Dans un système d'information, l'acteur n'est pas seulement un être humain. Ce peut être toute cause de déclenchement d'une procédure, comme un autre logiciel, relié au système par une interface, ou le résultat de l'action d'un automate.

Les *acteur* peuvent se différencier selon une typologie, par exemple associée au degré de liberté d'action qu'ils ont sur le système : ceux qui ont seulement le droit d'extraire et d'afficher certaines informations, ceux qui peuvent renseigner certains objets, ceux qui ont tous les droits d'accès (l'administrateur). Le lecteur est sans doute familiarisé avec ce concept d'acteur largement utilisé dans une *SGBD*, ou dans un système d'exploitation comme Windows, qui sont des systèmes d'information.

Dans *UML*, un *acteur* extérieur est représenté par un petit bonhomme, et le système formel, dont le contenu est supposé inconnu par lui, par un rectangle.



E. Les cas d'utilisation (Use Cases).

Avant d'élaborer un modèle conceptuel, nous avons évoqué deux étapes préalables à la charge des utilisateurs du système : le recensement des besoins, les spécifications fonctionnelles.

Nous avons précisé qu'il existait peu de méthodes pour aider l'utilisateur dans ces deux tâches. C'est vrai.

En revanche, il existe une formalisation très simple, plus précise qu'un texte, qui consiste pour chaque fonction envisagée, à dessiner à l'intérieur du rectangle qui symbolise le système, la collection des actions à exécuter, avec éventuellement une relation d'ordre.

On appelle ces fonctions ainsi modélisées « *use case* » (Nom donné par *Jacobson* à cette technique de modélisation).

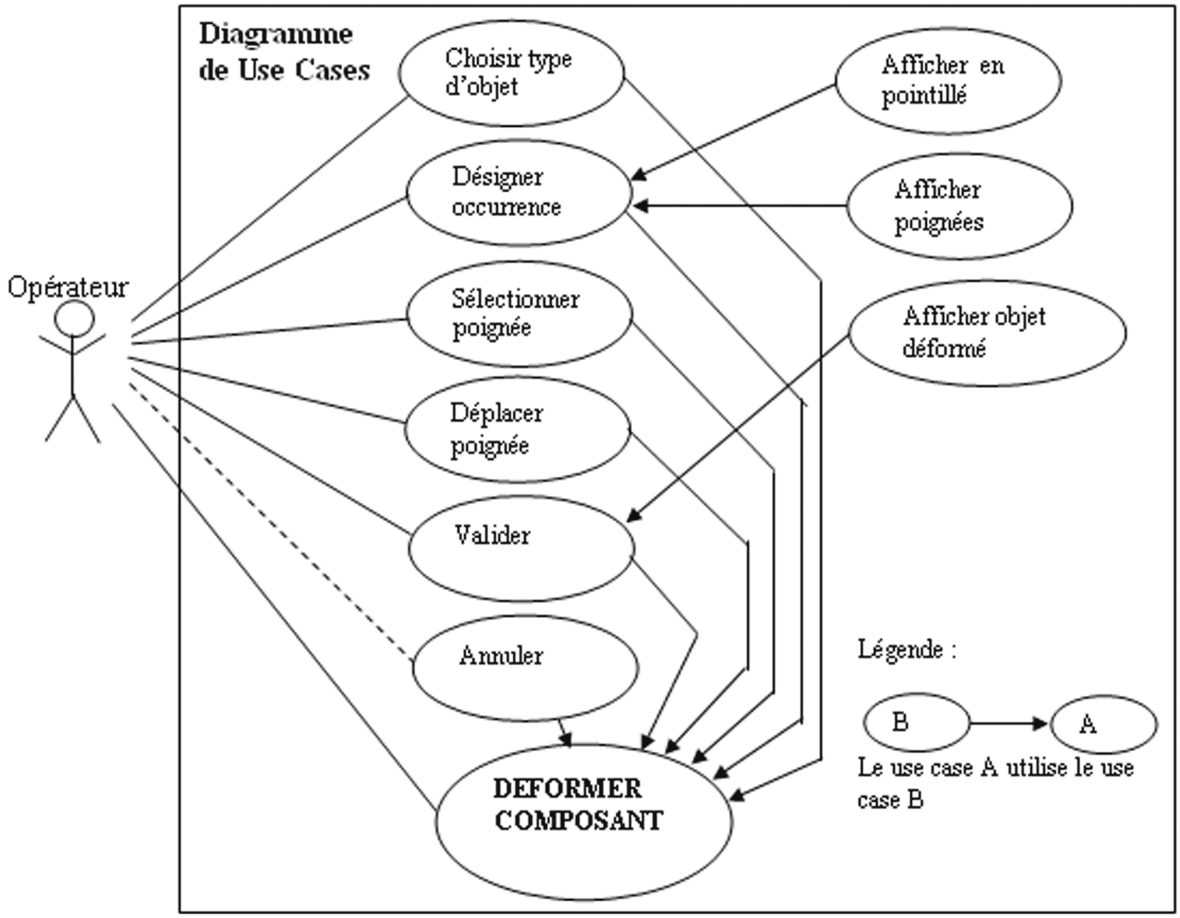
Un *use case* peut faire appel à d'autres, plus fins. Dans un système de CAO, par exemple, il peut en exister des centaines, qui portent souvent le nom d'une commande, d'une icône.

Exemple, le *use case* « Déformer un composant par les poignées», qui fait appel à d'autres use-cases plus spécialisés, dans un « scénario » que l'on peut imaginer :

« Choisir un type d'objet »
« Désigner un objet occurrence »
« L'afficher en pointillé »

« Afficher les poignées »
« Sélectionner une poignée »
« Déplacer une poignée »
« afficher l'objet déformé »
« Valider »

Ne figurent dans cette séquence que les *use case* déclenchées et perceptibles par l'opérateur. Une multitude d'actions seront implémentées à la charge des informaticiens, à l'insu de l'utilisateur, comme par exemple la vérification de la possibilité de déformation de l'objet choisi. En pointillé figure l'option qui permet d'annuler à tout moment son action.



Dictionnaire

acteur :

Voir unité "Quelques exemples d'élaboration de modèles conceptuels".

Utilisateur d'un système d'information, mais qui peut aussi être modélisé comme un objet dans un modèle conceptuel

AGL :

Atelier de Génie Logiciel

Booch :

Auteur de méthodes de spécification formelle, qui a donné son nom à cette méthode. Co-auteur de la méthode UML

CAO :

Conception Assistée par Ordinateur

concept :

Voir l'unité "Le concept d'objets dans les logiciels de DAO, CAO et de calcul".

Dans un modèle conceptuel, c'est une classe (unique) d'objets pertinente pour décrire le système d'information et son organisation sémantique.

contraintes :

Voir unité "Se comprendre autour d'un système d'information".

En spécification formelle, une contrainte s'applique pour filtrer la mise en relation entre les objets des deux concepts en présence.

cycle de vie :

Voir l'unité "Quelques exemples d'élaboration de modèles conceptuels".

Pour un développement d'un logiciel, ensemble des étapes de transformation, de traitement et d'exploitation du système

DAO :

Dessin Assisté par Ordinateur

évènement :

Voir l'unité "Quelques exemples d'élaboration de modèles conceptuels".

Action brusque qui n'a pas de durée, déclenchée par toute cause intérieure ou extérieure au système d'information.

EXPRESS :

Langage formel normalisé, pour décrire la structure de bases de données orientées objets. EXPRESS est un outil de STEP.

Le C.S.T.B. a développé un traducteur de schémas NIAM, qui produit des instructions EXPRESS.

EXPRESS-G :

Formalisme graphique pour décrire une structure de base de données, comme pour NIAM. L'avantage d'EXPRESS-G sur NIAM est qu'il constitue un outil de STEP, qui permet immédiatement d'obtenir une traduction de la base de données, en langage EXPRESS.

Inconvénient : il est moins pédagogique que NIAM. EXPRESS-G est donc réservé aux informaticiens

Henri Habrias :

Professeur à l'université de Nantes, auteur de nombreux ouvrages sur la spécification formelle et la méthode NIAM.

héritage :

Voir l'unité "Le concept d'objet dans les logiciels de DAO, CAO et de calcul".

Dans un LOO, et dans les IFC, mécanisme qui permet à des objets d'une sous-classe de bénéficier des propriétés des classes "parents"

homogènes :

Voir l'unité "Comprendre et échanger les vues métiers du bâtiment".

Dans un modèle, propriété d'une paroi : en tout point de sa surface, les propriétés géométriques, fonctionnelles et physiques sont constantes.

idée :

Voir l'unité "L'IAI et les IFC : introduction".

Terme utilisé dans la méthode de spécification NIAM : information mettant en jeu une seule relation (ensembliste) entre deux concepts.

IFC :

Industry Foundation Classes : Classes d'objets fondamentaux dans le domaine de l'AEC, utilisés dans le modèle conceptuel et le modèle physique des données pour les échanges EDI proposés par IAI.

interface :

Elément intermédiaire entre deux logiciels pour permettre le transfert de données. Pose des problèmes de sémantique et de formats

Jacobson :

Auteur de méthodes de spécification formelle, inventeur des « Case-Use », co-auteur de la méthode UML.

local :

Voir unité "Comprendre et échanger les vues métiers du Bâtiment".

Espace mesurable délimité par des murs, planchers, plafonds ou toitures qui peuvent être matériels ou virtuels. Pièce et ses regroupements.

langages orientés objets :

Langages Orientés Objets : C++, EXPRESS ...

LOT :

Voir unité "Se comprendre autour d'un système d'information".

Dans NIAM, abréviation de Lexical Object Type, c'est à dire désigne les occurrences, ou ensemble d'objets identifiés chacun par leur nom

MERISE :

Méthode (Française) graphique de spécification formelle

modèle conceptuel :

Ce terme est développé dans l'unité "Le concept d'objet dans les logiciels de DAO, CAO et de calcul".

Description formelle des concepts véhiculés dans un modèle de base de données, focalisée sur l'aspect sémantique (statique) du système d'information.

NIAM :

"Nijssen Information Analysis Method". Méthode de spécification formelle de données, utilisable dans n'importe quel domaine pour décrire sans ambiguïté une organisation de concepts, d'objets, y compris les relations et attributs associés. Cette méthode est normalisée (STEP et ISO en 1983)

NO LOT :

Voir unité "Se comprendre autour d'un système d'information".

Dans NIAM, abréviation de Non Lexical Object Type, c'est à dir désigne un Concept, ou une classe d'objets sans préciser ses occurrences.

nus :

Voir unité "Se comprendre autour d'un système d'information".

Objet abstrait servant à modéliser l'enveloppe d'un local. Les nus de locaux s'appuient sur les composants séparatifs.

objet :

Voir l'unité "Le concept d'objet dans les logiciels de DAO, CAO et de calcul".

Nomme indifféremment un type d'objet, ou une occurrence de la classe.

Voir "Orienté Objet" et "Occurence (ou instance)".

OMT :

Object Modeling Technique : Méthode de specification formelle de James Rumbaugh.

OOSE :

Object Oriented Software Engineering : Méthode de spécification formelle de Ivar Jacobson

phrase élémentaire :

Voir unité "Se comprendre autour d'un système d'information"

Dans NIAM, traduction en langue naturelle d'une idée. Une idée se décompose en deux phrases élémentaires, chacune pour un sens de la relation.

propriété :

Voir l'unité "L'IAI et les IFC : introduction".

Dans un LOO, et dans les IFC, une propriété qualifie un objet d'une classe : propriété propre, ou propriété contextuelle.

relation :

Voir l'unité "L'IA et les IFC : introduction".

Dans un LOO, et dans les IFC, une relation est un lien formel entre deux objets de même classe ou de classe différentes, ou avec une autre relation.

rôles :

En spécification formelle, et surtout dans NIAM, un rôle formalise l'expression d'une relation orientée entre deux concepts.

SGBD :

Système de Gestion de Base de Données, en général relationnelles.

sous-typage :

Voir l'unité "Se comprendre autour d'un système d'information".

Autre nom pour désigner la décomposition des concepts en une arborescence de relations d'héritage. Encore appelée "Spécialisation".

- La définition de "Spécialisation" dans le dictionnaire est :
  Voir unité "L'IAI et les IFC : introduction".
  Définir dessous-classes d'objets à partir d'une classe (dite Parent). Des exemples sont fréquents dans une arborescence d'héritage.

spécification formelle :

Voir l'unité "Se comprendre autour d'un système d'information".

Décrire sans ambiguïté un modèle conceptuel, en général au moyen d'une méthode ou d'un langage, comme NIAM, UML, EXPRESS-G, ...

système d'information :

Voir unité "Pourquoi inventer des modèles pour la construction ?".

Ensemble des logiciels et des acteurs qui exploitent la connaissance modélisée d'un objet complexe, réduit à différents points de vue.

Totalité :

Voir unité "Se comprendre autour d'un système d'information".

Indique que dans une relation orientée tous les objets de l'ensemble de départ sont concernés. Se traduit par "Tout objet" ou "Chaque objet".

traçabilité :

Voir unité "Quelques exemples d'élaboration de modèles conceptuels".

Conserver la continuité du cheminement de la transformation d'une information dans le développement d'un modèle, à toutes ses étapes

UML :

Unified Modeling Language, méthode de spécification formelle résultat d'une synthèse entre les trois méthodes OMT, Booch et OOSE

'Unicité :

Indique que dans une relation orientée les objets de l'ensemble de départ ne sont liés qu'à un seul objet de l'ensemble d'arrivée.

use case :

Voir unité "Quelques exemples d'élaboration de modèles conceptuels".

Cas d'utilisation, en français : Liste des actions à exécuter par le système d'information, imaginées par les acteurs pour définir une fonction.