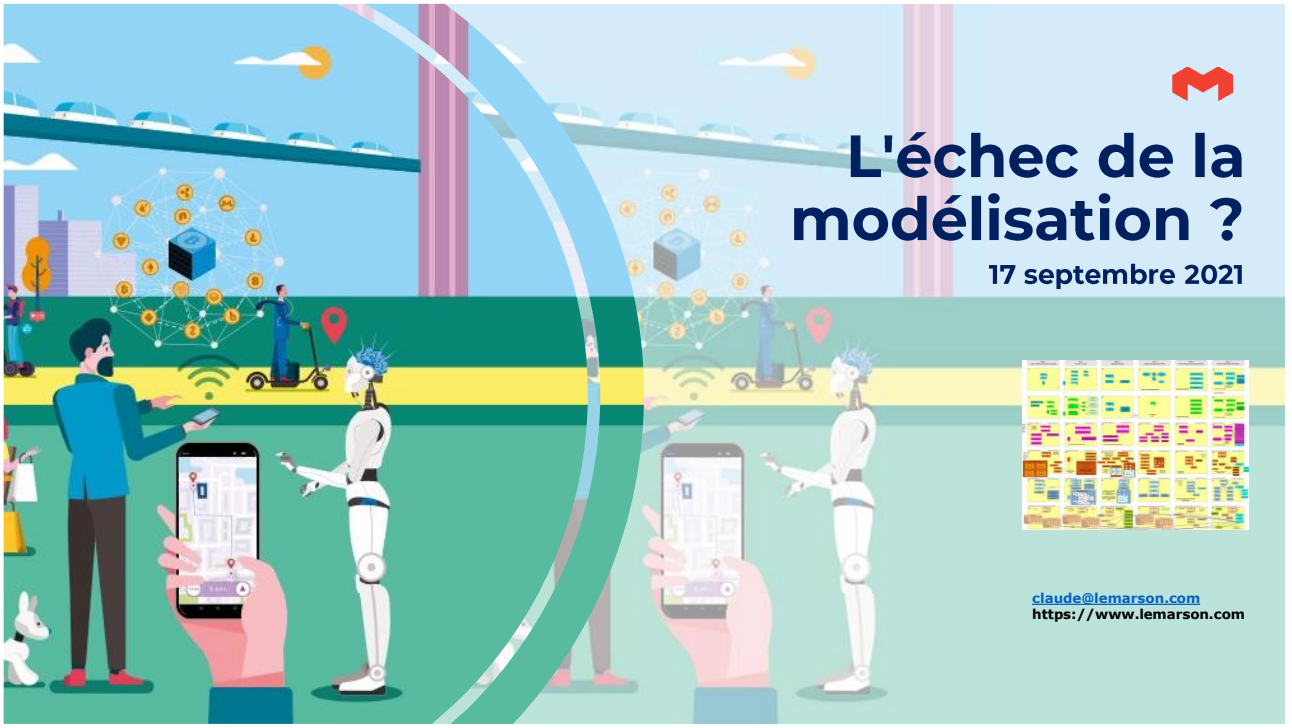


L'échec de la modélisation ?

17 septembre 2021

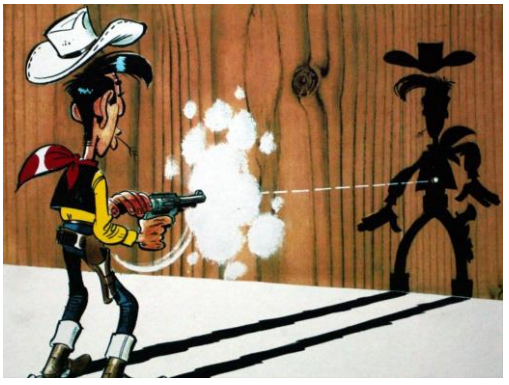
claude@lemarson.com
https://www.lemarson.com

Sommaire

L'échec de la modélisation

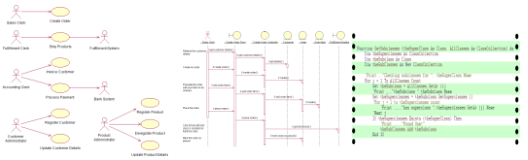
- *Ce que recouvre la modélisation, histoire*
- *Le principe du MDD, développement piloté par les modèles*
- *Réfléchir avant de coder*
- *Distinguer les méthodes, des langages et des modeleurs graphiques*
- *L'approche UML et les raisons de son échec*
- *La modélisation des données*
- *La génération du code et l'apport nouveau de l'Intelligence Artificielle*
- *L'avenir appartient (quand même) à la modélisation*

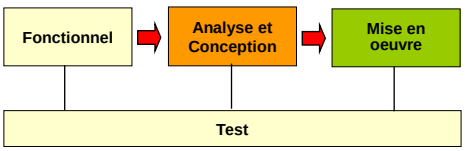
Question : qu'est-ce qui va plus vite que la vitesse de la lumière ?
Réponse : un programmeur qui se lance dans le codage d'un projet…



Le vieux « fantasme » du génie logiciel

- Générer automatiquement un système (ou une grande partie) à partir d'un modèle (modèle « exécutable ») : ça ne date pas d'hier
- Le modèle ne doit pas dépendre de la technologie sous-jacente (architecture technique)
- C'est une autre façon de concevoir les applications
- Le fantasme peut-il et doit-il devenir réalité ?
  - Quelles sont les contraintes technologiques et méthodologiques ?
  - Quelle est l'amélioration de la productivité attendue **?**





- Quand une mise à jour doit être faite, à la suite d'une modification fonctionnelle, il faut, dans la mesure du possible, l'implémenter dans le modèle et regénérer le code correspondant
- On sait que dans la pratique ce n'est pas le cas et que pour parer au plus pressé, les développeurs portent les modifications directement dans le code…car ils n'ont pas le temps de passer par les modèles

Ce que recouvre la modélisation. histoire

- La tendance est à la « formalisation » des développements, pour les rendre plus rigoureux
- Les méthodes formelles existent depuis cinquante ans : elles viennent des milieux universitaires
- Il s'agit de donner un cadre mathématique (formel) à la description des applications à construire, de réduire les incohérences dues à l'expression en langage naturel des cahier des charges, de décrire précisément les modules techniques
- Une méthode formelle (Hinchey) est un ensemble d'outils et de notations, dotés d'une sémantique formelle, utilisée pour caractériser de manière non ambigüe, les spécifications d'un système et **supportant les preuves sur ces spécifications et les preuves de l'adéquation des corrections**
- Langage formel : syntaxe et sémantique pour décrire un système et son comportement
- Nécessité de maîtriser la formulation mathématique et l'outil informatique : **c'est là où ça coince…**
- Les avantages :
  - Améliorent la qualité et la rigueur du processus de développement
  - Améliorent la fiabilité des applications produites
  - Réduisent les erreurs de spécifications
  - Bon support pour la maintenance et l'évolutivité
  - Limitent l'importance des tests unitaires et des tests d'intégration

Peaucelle: CORIG, histoire d'une méthode en informatique de gestion
CORIG, HISTOIRE D'UNE MÉTHODE EN INFORMATIQUE DE GESTION

V. LES SUCCÈS DE CORIG

V.1. Le succès industriel de la CGI

La méthode CORIG est la raison du succès de l'entreprise CGI. Robert Mallet est l'inventeur de la méthode. Il est le propriétaire de l'entreprise, fondée en 1969. En 1966, le gouvernement lance le « Plan Calcul ». Robert Mallet est furieux parce que « la CGO n'est pas dans le coup ». Pour avoir plus d'autonomie stratégique, il propose de filialiser l'activité informatique. C'est fait en 1967. Mais la CGI-Informatique, filiale de la CGO, n'est pas dirigée par Mallet. Ce n'est pas assez net. La Compagnie Générale d'Informatique (CGI) prend son autonomie en 1969, après une rupture amiable. Elle est dirigée par Robert Mallet[6]. Elle ne compte alors que quelques dizaines de personnes.

Sa croissance est rapide (voir figure 2). À partir de 1986, elle est cotée en bourse. Elle rachète alors d'autres SSII, en France et à l'étranger[7]. Les bénéfices, après impôt, sont montés, en 1990, jusqu'à 10 % du chiffre d'affaires. À partir de 1991, le secteur, en France, entre dans une période de crise. La CGI est moins affectée que d'autres. Néanmoins, les résultats se dégradent. En 1992, elle compte 4 000 personnes et réalise un chiffre d'affaires de 2 milliards de francs. En avril 1993, après de longues négociations, elle est rachetée par IBM[8].

Figure 2 : Évolution des résultats de la CGI d'après les chiffres des rapports annuels[9].

[6] Robert Mallet et Jacques Debaussin à eux deux possèdent 75 % du capital (20 000 F). Le capital fut augmenté plusieurs fois par incorporation de réserves, sauf en 1972 où 600 000 F d'espèces furent apportés. En 1986, l'introduction en bourse apporte 24 millions pour les deux tiers du capital

[7] Parmi les plus célèbres, CRZA (France) en 1987, Euréquip (France) en 1988, Productions systèmes (France) en 1989, SRS-Network (USA) en 1989, Youridon (USA) en 1990, Préd-Art (France) en 1991, Interprogram (Allemagne) en 1991

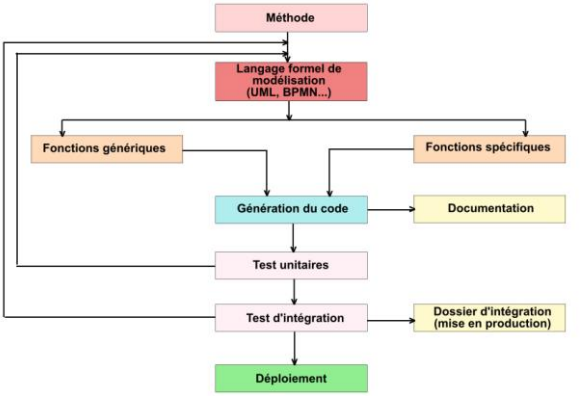
[8] OPE annulée où la CGI fut valorisée à 2,5 milliards de francs, soit 20 fois le bénéfice de l'année précédente. Robert Mallet et Jacques Debaussin possédaient alors 25,4 % du capital à eux deux (01 Informatique du 9 avril 1993). L'opération devait permettre à CGI «de conserver son indépendance et tous ses caractères originaux» (Bernard Chaput, président du directoire, dans le rapport 1993). En janvier 1995, IBM commence à restructurer les activités de la CGI en les intégrant dans celles de ses structures qui exercent le même métier. Beaucoup d'ingénieurs sous de la CGI partent. L'entreprise CGI disparaît en 1999

[9] Ces rapports m'ont été fournis par Jean-Louis Bernardin, qui était chargé de la rédaction des rapports annuels. Je le remercie spécialement

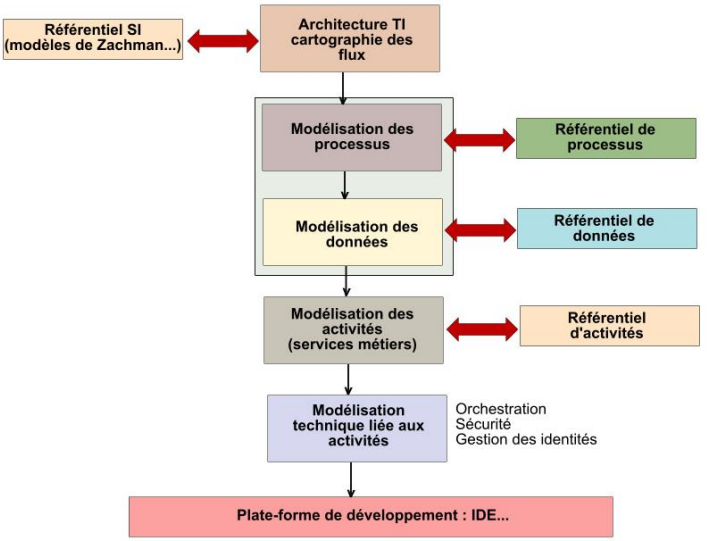
Le français Robert Mallet, créateur de CORIG, l'ancêtre de tous les langages de modélisation

Le principe du MDD : Développement piloté par les modèles

- Les techniques modernes de développement tendent à éviter au développeur de se perdre dans les arcanes techniques
- Il vaut mieux passer du temps sur la conception, que sur le codage
  - La réflexion dans la phase codage sera limitée
  - On réutilisera au maximum les solutions qui ont fait leurs preuves : API, patterns
- L'idée était de passer par un langage de modélisation, avec deux familles d'éléments
  - Génériques, reproduits à l'identique
  - Particuliers aux applications
- Pour aboutir à une phase de tests automatisés
- À la génération du code
- À la génération de la documentation
- Ce n'est pas (du tout) ce qui s'est passé
  - On fait toujours du spécifique
  - La documentation est encore trop souvent bâclée
  - Les tests sont insuffisants
- Contrairement à ce que dit la pensée unique, l'avenir appartient à MDD



Les chantiers de la modélisation



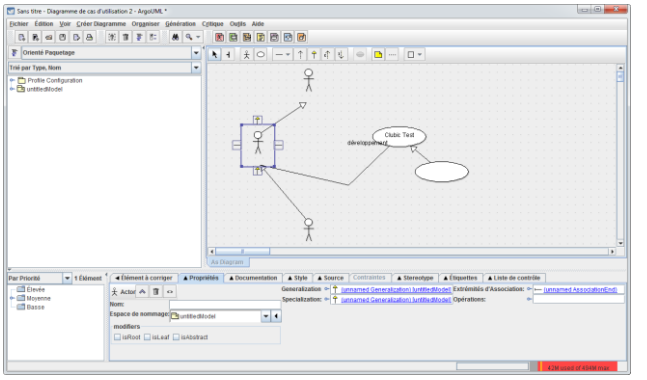
Réfléchir avant de coder

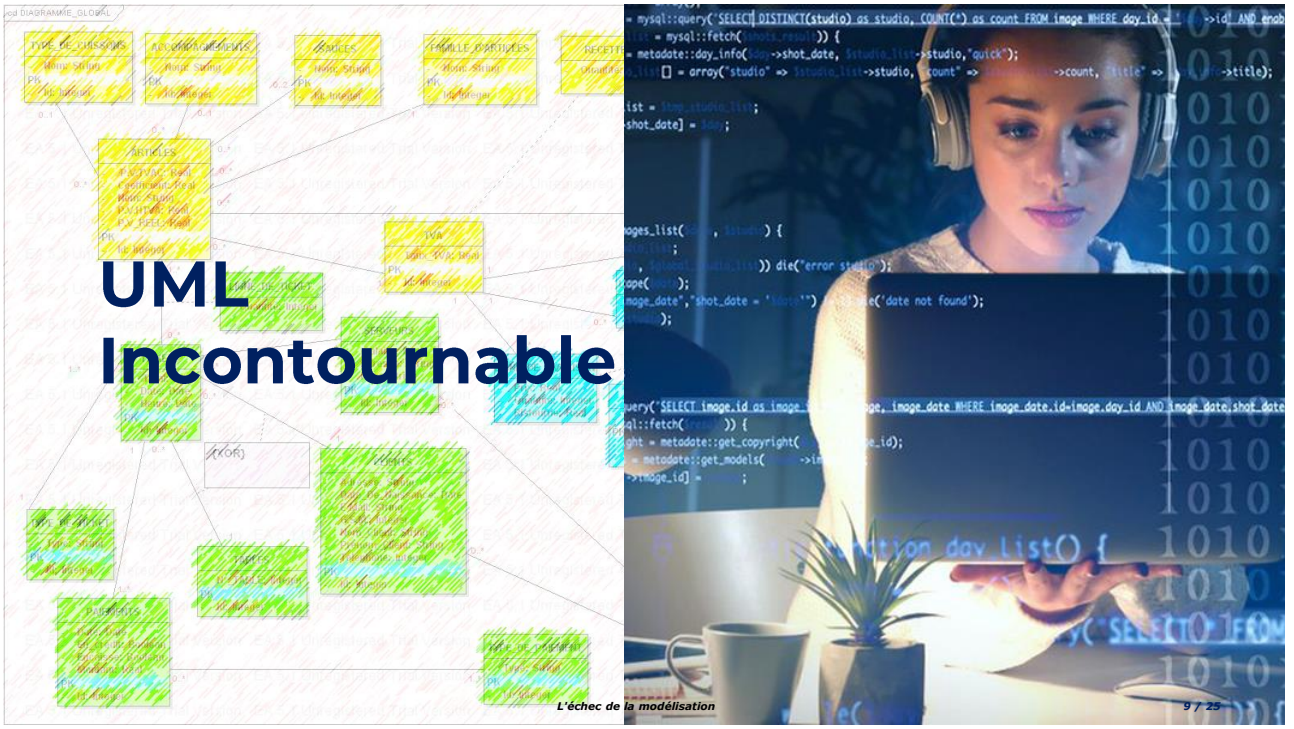
- Le codage procède plus d'un mécanisme de répétition, alors que l'analyse (la modélisation en fait partie) est un processus de création plus complexe, qui nécessite plus de neurones en activité…
- L'avantage de la modélisation est de réfléchir à une organisation, à une architecture fonctionnelle, sans être (trop) perturbés par des considérations techniques, liées aux plates-formes, aux langages et API
- Maîtrise de la complexité
- On peut se placer selon divers points de vue, avec des niveaux d'abstraction différents
- Avantages
  - Qualité et précision des descriptions : un process ou une donnée, sont ou ne sont pas là, il n'y a pas d'ambiguïté
  - Réduction du temps de conception
  - En fin de projets : réduction des coûts
  - Maintenance plus efficace et moins coûteuse
  - Un bon dessin vaut mieux qu'un long discours (Napoléon Bonaparte)
  - Excellent outil pour communiquer avec les utilisateurs
- Les domaines applicatifs privilégiés : services réutilisables, SOA, MSA, données
  - Leur caractère générique oblige à peaufiner la conception

Le penseur de Rodin

Les modeleurs graphiques

- Ne pas confondre le langage de modélisation, type UML ou BPMN avec le modeleur graphique.
- Il existe aujourd'hui de nombreux modeleurs de qualité, qui permettent de représenter graphiquement les concepts du langage de modélisation.
- L'Open Source est très représentée qui n'a rien à envier aux plates-formes majeures.
- BPMN et UML sont les langages les plus représentés dans les modeleurs modernes
  - Les produits récents intègrent les approches modernes, telles que les SOA
  - Exportation des processus en BPEL
  - Support de la description WSDL pour les Web Services
- De nombreux produits existent: Mega, Telelogic, IDS Scheer, Casewise, mais aussi Proforma, iGrafX, IBM Rational, EMC, ArgoUML…

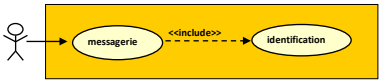



UML
Incontournable

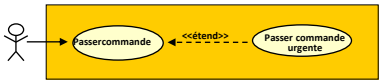

2016
UML 2.4.1 béta
2010
UML 2.2
UML 2.0
2005
UML 1.5
UML 1.4
2000
UML 1.1
UML 0.9
1995
Jacobson
Booch
Use Case
Diagrammes dynamiques
Relations entre un acteur et une fonctionnalité du système (côté utilisateur). Chaque processus est un cas d'utilisation (« Use Case »).
Diagrammes statiques
Séquence
Représente les échanges de messages entre objets. L'axe vertical étant le temps t et l'axe horizontal les objets qui collaborent et se parlent.
Classe
Description des classes et leurs associations.
Les diagrammes UML
Collaboration
Mettent en évidence les interactions entre les objets du système et les messages qu'ils échangent
Objet
Liens entre objets, les instances de classes.
Etat
Représente les traitements d'un domaine, associés à une classe pour laquelle différents états sont gérés
Comportement
Conditions de production des éléments physiques à partir de leurs spécifications et de l'environnement de développement.
Activité
Sert à représenter les traitements à effectuer et les acteurs impliqués.
Déploiement
Décrit l'installation des composants physiques en termes d'exploitation.
Rumbaugh

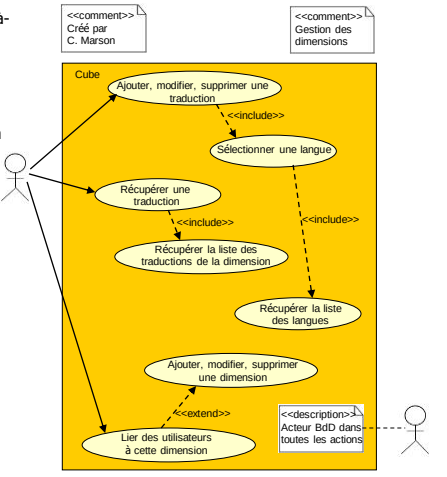
Les Use Case (cas d'usage)

- Un use case définit une exigence fonctionnelle, représentée sous forme d'une séquence d'étapes, qui comprend les actions réalisées par le système et les interactions entre le système et les acteurs.
- Le lien entre l'acteur et le cas d'utilisation représente le type d'interaction et la dépendance de l'acteur vis-à-vis du système.
- La dépendance est représentée par une association, schématisée par une flèche.
- Les Use Case permettent de concevoir les cahiers des charges. Ils leur apportent la rigueur mathématique que n'a pas le texte.
- Les liens entre cases
- Une dépendance d'inclusion relie deux cas d'utilisation : le cas de base et le cas inclus. Elle s'affiche sous la forme d'une ligne discontinue qui va du cas de base vers le cas inclus, étiquetée par le mot clé "include".



- Une dépendance d'extension relie un use case étendu et un use case de base, pour indiquer que le use case étendu étend (ou est inséré) dans le use case de base.
  - Le lien d'extension signifie qu'un cas d'utilisation de base englobe le comportement d'un autre cas d'utilisation.
  - On utilise le lien d'extension pour modéliser la partie d'un cas d'utilisation facultatif ou alternatif par rapport au cas de base.
  - Cette dépendance s'affiche sous la forme d'une ligne discontinue définie par le mot clé <<étend>> qui va du cas étendu vers le cas de base.

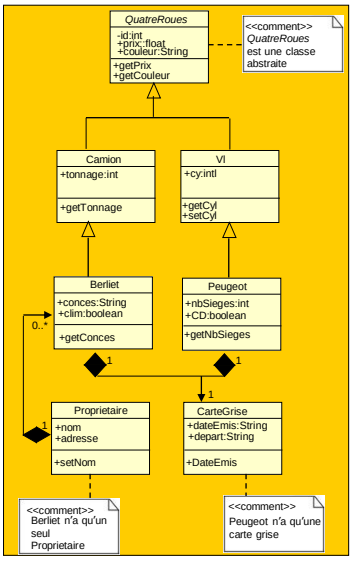




Un use case

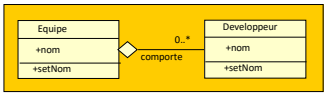
Diagramme de classes

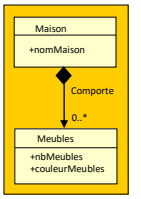
- Décrit la structure statique du système à l'aide de classes, packages et relations
- Décrit des concepts théoriques alors que les diagrammes d'objets décrivent des instances de ces concepts, la réalité
- Outre le nom, comportent une description des attributs et une description des méthodes
- Les classes sont reliées par des associations, qui définissent des types de liens et représentent des relations générales entre classes
  - **Accesseurs** : une opération qui renvoie une information sur l'état de l'objet
  - **Modifieur** : une opération qui modifie l'état de l'objet (méthode qui agit sur une propriété)
  - **Constructeur** : une opération de la classe qui permet d'initialiser une nouvelle instance
  - **Héritage**
  - **Agrégation**
  - **Composition**
  - **Polymorphisme**



Agrégation et composition

- Une association représente une relation générale entre classes et définit un type de lien
- Une association binaire établit une relation entre 2 classes.
- Une agrégation décrit une relation entre un agrégat, le tout et ses parties
- Elle est représentée par un losange vide attaché à la classe représentant le tout
- Une agrégation ne s'utilise qu'avec une association binaire





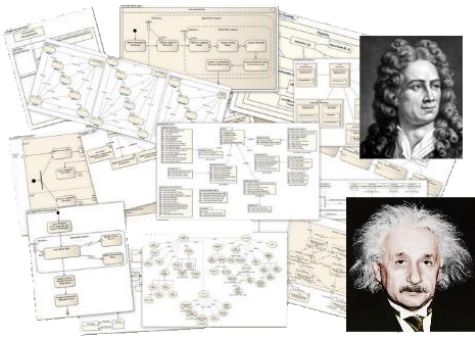
- Une composition, dite aussi agrégation composite, décrit une relation entre un composite (le tout) et ses parties, les parties ne pouvant appartenir qu'à un seul tout et où le tout est responsable de la création et de la destruction de ses parties lors de sa propre création ou destruction.
  - Ex : la classe Maison comporte des Meubles. Si Maison cesse d'exister, les meubles cesseront d'exister au sens parties de Maison.
  - Ce qui ne les empêchera pas d'exister par ailleurs.
- La composition ne s'utilise qu'avec des associations binaires.

UML n'a pas atteint ses objectifs

- Les DSI n'ont pas mesuré à sa juste valeur, le saut quantique qu'ils demandaient à leurs chefs de projets.
- L'approche mathématique de ces langages a rebuté un grand nombre de chefs de projets qui n'on plus la capacité d'abstraction nécessaire.
  - Ex du diagramme de classes : "arborescence quasi-fortement connexe inférieurement".
- A vouloir tout modéliser, depuis le cahier des charges, jusqu'aux échanges de messages à l'exécution (diagramme de séquences), UML est devenu un "monstre".
- Les équipes n'ont pas joué le jeu : les utilisateurs n'ont pas fait l'effort minimum pour rédiger les cahiers des charges avec les « Use Cases » et les développeurs n'ont pas voulu sortir de leur code.
- Les responsables de TI, plutôt que de procéder par touches homéopathiques, ont cherché la rupture brutale.
- Résultat : pour « faire plaisir » au "patron", les chefs de projets ont élaboré des modèles « a posteriori », après le codage. Ce qui ne sert à rien.
- Les inconvénients d'UML :
  - De plus en plus compliqué.
  - De nombreux développeurs font un blocage.
  - On parle aussi de "Unwanted Modeling Language".
  - UML est souvent dévoyé : Microsoft avec Visio, ce n'est pas un outil de dessin.
  - Difficulté pour valider un modèle avant de générer l'implémentation.
  - Rétro-ingénierie très difficile à réaliser.

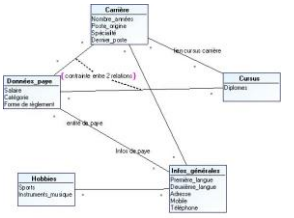
Modélisation des données

La modélisation conceptuelle des données



"Ce qui se conçoit bien s'énonce clairement, et les mots pour le dire arrivent aisément"
(Boileau)

"If you can't explain it simple, you don't understand it well enough"
(Albert Einstein)

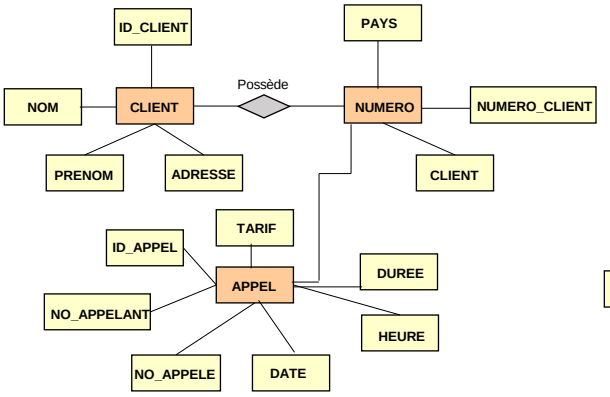


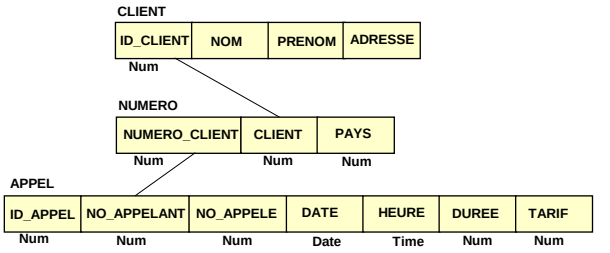
- Les modeleurs de données traitent deux types de besoins :
  - La modélisation des données proprement dites, indépendamment de la structure des bases de données qui les utiliseront : le modèle conceptuel
  - La modélisation des bases de données, qui doivent être générées automatiquement à partir de scripts créés par l'outil de modélisation
- L'idée de base est que toute modification sur les données doit entraîner un nouveau modèle conceptuel, qui générera un nouveau schéma logique lié à une base de données déterminée et les requêtes SQL de création de tables correspondantes
- Dans la pratique il en est très autrement…
  - Les concepteurs « ne font pas l'effort de conceptualisation », ce qui se traduit par des incohérences de données

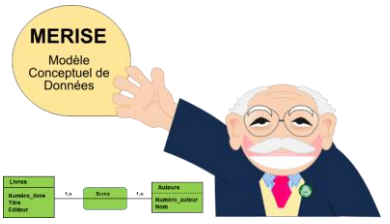
Données conceptuelles et logiques

Les données induisent trois types d'interventions :

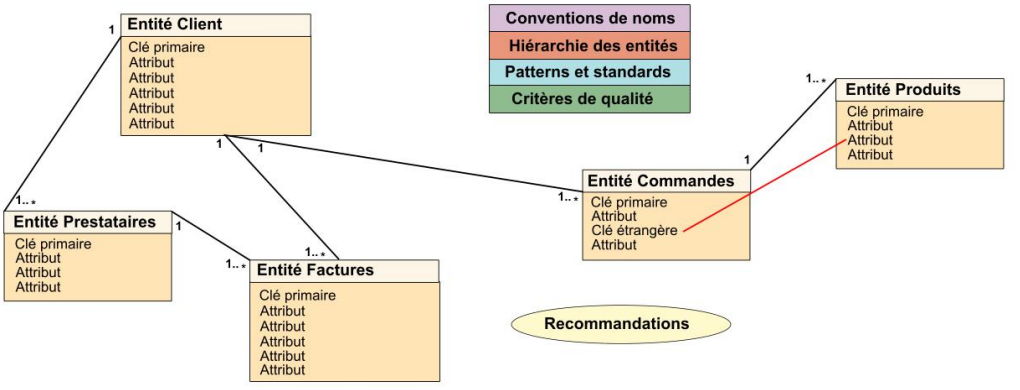
- Description fonctionnelle des données avec un schéma conceptuel représentatif (MOA) : on décrit chaque entité de données avec ses caractéristiques (attributs) et les liens entre les données
- Description logique des données avec un schéma (MOA et MOE) : on établit un schéma dans lequel chaque donnée est représentée selon un format précis : chaîne de 25 caractères, valeur numérique avec 2 décimales, etc
- Stockage physique des données (MOE) : on prévoit la nature du stockage avec la capacité adéquate et les performances attendues, mais aussi les aspects sécurité







Le modèle entité-relations



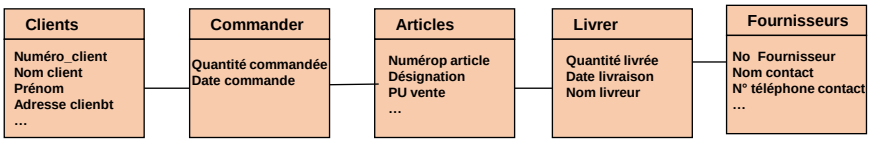
Le MCD : Modèle Conceptuel de Données

- Une entité est une population homogène d'individus
  - Les produits ou les articles vendus par une entreprise peuvent être regroupés dans une même entité « Articles », car d'un article à l'autre, les informations ne changent pas : il s'agit toujours d'identification, de prix, de lieu de stockage…

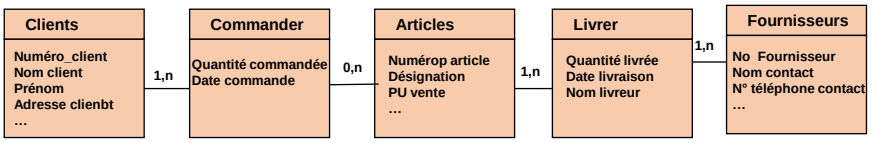
- Une association est une relation qui a une signification précise entre deux entités :
  - L'association « commander » est une liaison entre les entités « articles » et « clients », alors que l'association « livrer » établit le lien sémantique entre les entités articles et fournisseurs.

Le MCD : Modèle Conceptuel de Données

- Un attribut (caractéristique) est une propriété d'une entité ou d'une association.
  - Le prix unitaire est un attribut de l'entité articles, le nom de famille est un attribut de l'entité clients, la quantité commandée est un attribut de l'association commander et la date de livraison est un attribut de l'association livrer.



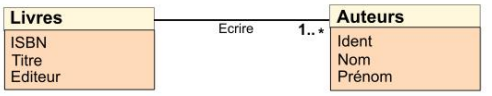
- La cardinalité d'un lien entre une entité et une association précise le minimum et le maximum de fois qu'un individu de l'entité peut être concerné par l'association.



Les règles normales

Exemple d'une règle normale

Livres
ISBN
Titre
Auteur_1_Ident
Auteur_1_Nom
Auteur_1_Prénom
Auteur_2_Ident
Auteur_2_Nom
Auteur_2_Prénom
Auteur_3_Ident
Auteur_3_Nom
Auteur_3_Prénom
Editeur



Méthodes et langages de modélisation des données

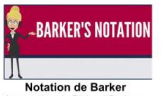
RM/T(Relational Model/ Tasmania)

Extension en 1979 du modèle relationnel de Codd, pour traiter différents aspects de la modélisation des données : atomique et moléculaire

Méthode créé en France dans les années 70 par René Colletti, Arnold Rochfeld et Hubert Tardieu, a été le creuset où se sont exprimés la plupart des tenants de la modélisation des données. Impossible à négliger.

Diagrammes de Bachman

Un peu l'ancêtre du modèle entité-relation actuel. Les relations sont représentées par des boîtes dans lesquelles sont reportées les contraintes.



Notation de Barker

La notation de Richard Barker, conçue en 1981, est encore très utilisée dans les outils CASE d' Oracle. Elle est très différente des autres, avec les pieds de corbeaux et pointillés pour les éléments facultatifs, les structures en boucles pour la récursivité, les boîtes imbriquées, etc

Object-Role Modeling

ORM a été imaginée en 1970 par le néerlandais G.M Nijssen. Elle est fondée sur des objets qui jouent des rôles. Les attributs sont remplacés par des relations. Encore utilisée pour modéliser les règles de gestion, les schémas XML, etc

Solution personnalisée : un compromis

Data Vault Modeling

Méthode proposée en 2000 pr Dan Linstedt, très orientée vers l'historisation et l'audit des données. Occupe une place à part dans l'échiquier des méthodes.

Notation de Chen

Méthode ER (Entité Relation) créée en 1976, pour mettre au moint les bases de données de l'époque. Reste une référence.

EBNF (Extended Backus-Naur Form)

Niklaus Wirth a amélioré le formalisme initial BNF, que l'ISO a standardisé en 1996. EBNF est un code qui exprime la grammaire d'un langage formel. Peu de succès, y compris à l'ISO...

u

UML

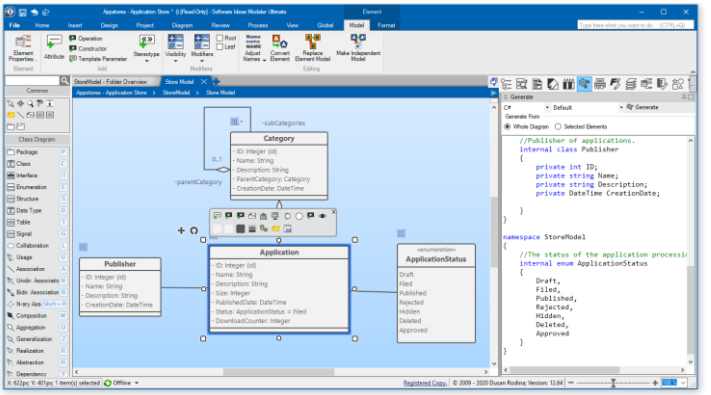
Rumbaugh, Booch et Jacobson, les créateurs d'UML. Parmi les 13 diagrammes UML, celui consacré aux objets peut servir à la modémisation des données. UML souffre toujours d'un formalisme trop mathématique.

IDEF1X (Integration DEFinition)

Typique des langages de modélisation, avec un référentiel graphique très avancé. A été développé à l'US Air Force et reste très utilisé aujourd'hui. Une valeur sûre

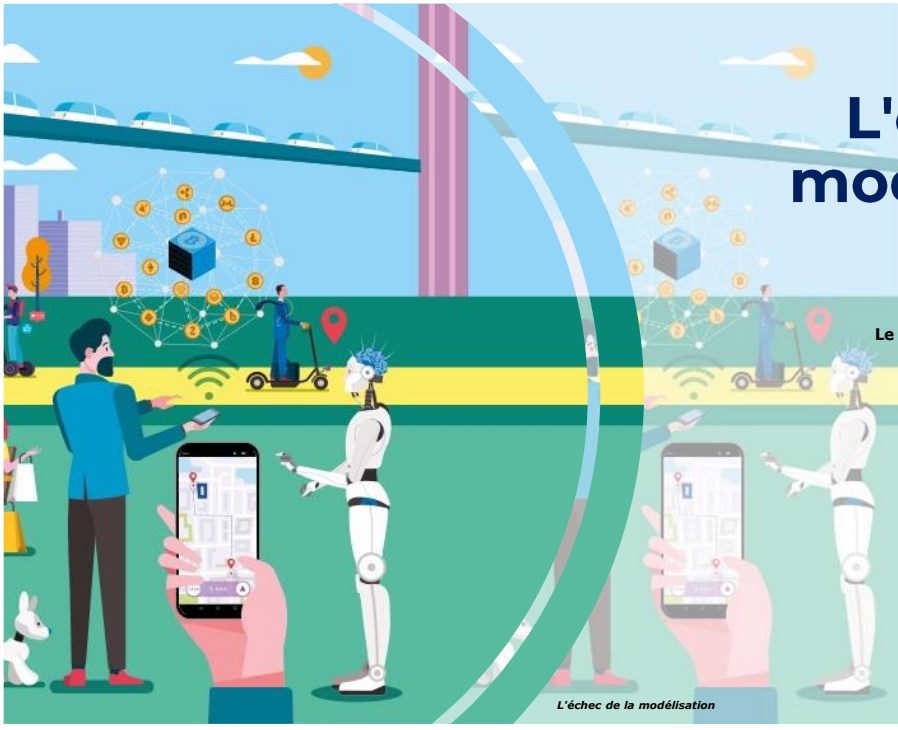
La génération de code

- Nombreux exemples de générateurs de code
- Depuis 30 ans, il y a un divorce entre les chefs de projets/développeurs et les générateurs de code
- Il n'y a aucune valeur ajoutée dans cette phase de projet
- Le code produit est souvent laid, peu optimisé et en manque de qualité et d'esthétisme
- Les programmes doivent-il être des œuvres d'art
- Pour certains d'entre eux, oui, mais pour leur grande majorité, non
- Il ne faut pas se tromper de finalité : l'objectif du codage c'est de traiter une fonction métier, pas le NFT de la Joconde…
- Généralement, les développeurs sont d'un avis contraire…
- La grande question sera l'apport de l'IA…



Les recommandations

- L'approche doit être naturelle.
- Ne rien imposer : analogie avec les mathématiques.
- Impliquer les bonnes personnes dans le soutien de l'approche modélisation
  - Les projets de modélisation aboutissent souvent à des changements d'organisation. Il faut donc avoir le soutien de la DG et des directions impactées. Pendant la vie du projet, il faut avoir accès aux décideurs.
- Lisibilité par rapport à l'exploitabilité des modèles.
  - Il est très difficile de faire un choix dans les techniques de modélisation de manière à construire des modèles qui soient à la fois compréhensibles par les métiers et transmissibles à l'informatique. Il n'y a pas de solution miracle.
- Homogénéité des modèles d'un même niveau.
  - Il faut tendre à ce que le niveau de détail soit le même.
- Un modèle doit avoir un but.
  - Il faut que le but de chacun des modèles soit clairement défini : utilisation et destinataires. Et donc choisir une technique de modélisation adaptée.
- Un modèle doit être lisible.
  - Si possible tenir sur une page. Principe de généralisation/spécialisation.
  - Il faut éviter au maximum les croisements de flux dans un modèle. Il faut que l'aspect général soit attractif et ne pas sembler surchargé. Même si un logigramme avec beaucoup de flèches, ça « fait sérieux »
- Un processus doit avoir un objectif.
  - Si un processus n'a pas d'objectif, c'est qu'il n'en est pas un ou est constitué d'un agglomérat de processus.



L'échec de la modélisation ?

17 septembre 2021

Le sommaire est sur le site www.lemarson.com

claude@lemarson.com
https://www.lemarson.com