Il n'y a pas de bon modèle métier

Devoxx 2012 Gregory Weinbach

https://youtu.be/qN43Dy6fGkk

Souvent application ne répond pas aux utilisateurs:

- parce que besoin mal exprimé
- parce que difficile de traduire ce besoin en software

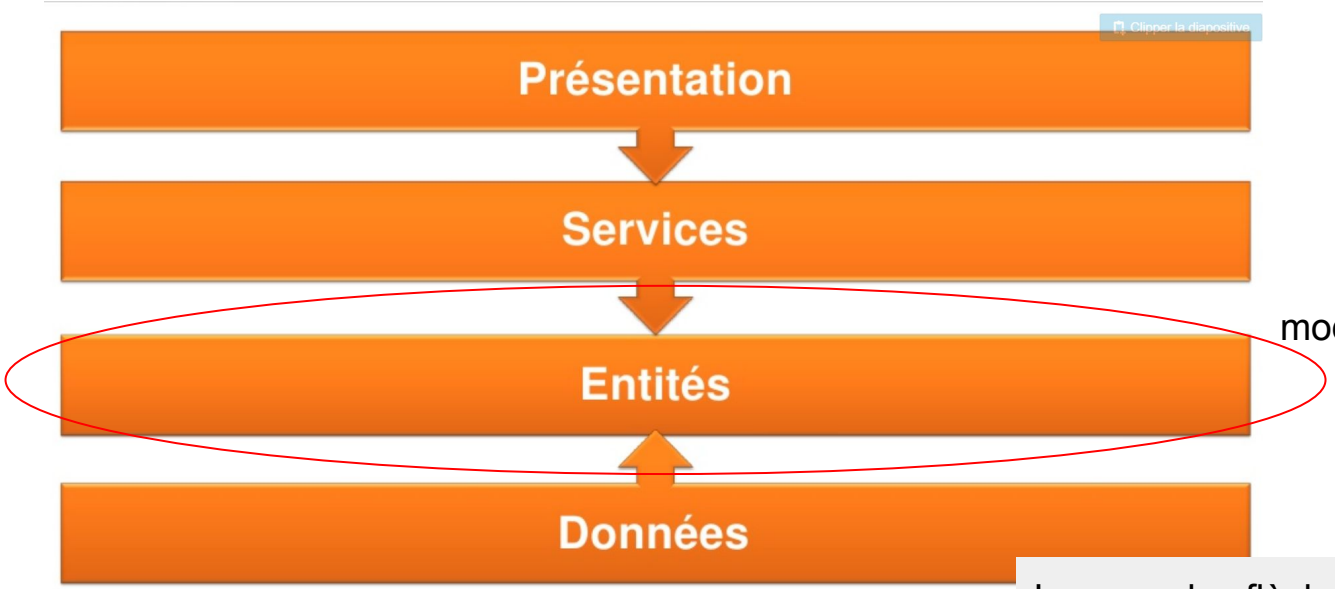
Concevoir pour l’utilisateur. Pourquoi vous voulez un logiciel? Quel sont ses objectifs métier? Agile pour diminuer le téléphone arabe.

DDD Domain Driven Design

DDD, livre de Eric Evans :

- la connaissance du domaine, une vue sur le domaine, le modèle domaine
- l'originalité dans DDD, le modèle est un **modèle de conception** :
  - il vit en parallèle du code, c'est un équivalent du code;
  - les deux ont un cycle de vit commun
  - **quand on modélise, on code**
- Le modèle et le code partagent un langage comme, ubiquitous language, partagé entre les utilisateurs, les concepteurs, les développeurs
- c'est un vrai modèle objet (donnée, structure, comportement)

Architecture en couche et modèle métier



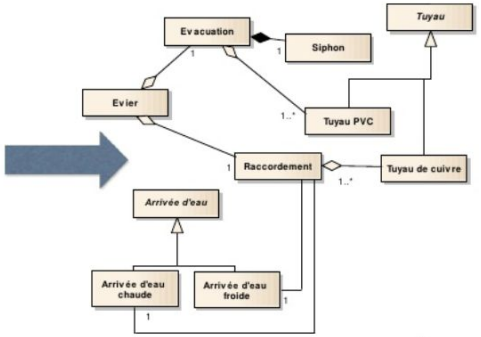
modèle métier se trouve ici

Le sens des flèches est important, c'est la dépendance, pas les flux

Comment bien construire ce modèle métier?

Réponse :

Une ***Entité*** doit représenter
un Concept du métier



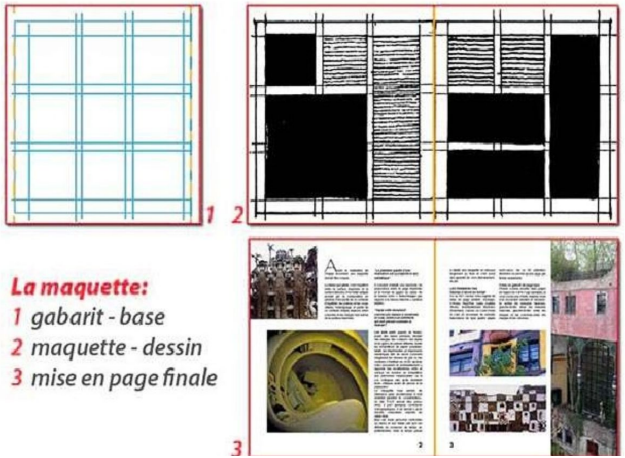
On va voir le métier, on l'interroge, et on distille sa connaissance dans le modèle

Comment bien construire ce modèle métier?

Exemple, système de gestion d'un éditeur : qu'est ce qu'un livre **pour vous** ?

Pour l'éditeur, un livre c'est ça :

Pour la PAO, :



Pour l'auteur : un fichier ms-word

Pour l'informatique:

Comment bien construire ce modèle métier?

Exemple, système de gestion d'un éditeur : qu'est ce qu'un livre **pour vous** ?

Pour la vente : prix, rating des lecteurs, des livres associés

Pour la logistique: un poids, un encombrement

Comment bien construire ce modèle métie

Exemple, système de gestion d'un éditeur : qu'est ce qu'un livre **p**

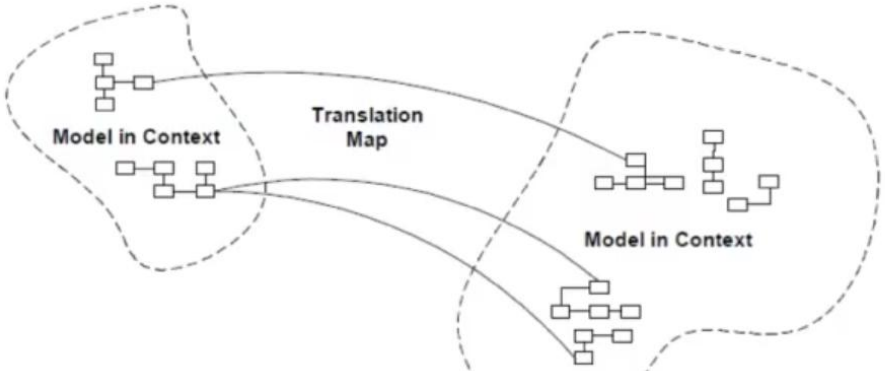
Livre
titre
auteur
éditeur
ISBN
copyright
nombre de pages
hauteur
largeur
épaisseur
/volume
illustration couverture
texte quatrième de couverture
stock
date de disponibilité
date de réimpression
dépôt légal
type de papier
type de reliure
texte brut
texte mis en forme
collection
tome numéro
allée numéro
position
photo de l'auteur
présentation de l'auteur
opérations spéciales
prix de vente public

Comment bien construire ce modèle métier?

- En voulant répondre à tous les besoins, on a répondu à aucun besoin
- Il n’y a pas un modèle, il y a des modèles
- Il n’y a pas un livre, il y a des livres

Eric Evans, sur son bouquin insiste sur un point, **la notion de contexte**

Un Modèle Métier = Un Contexte



= Un Domaine

- Un modèle n’est valable que dans un contexte donné
- C’est un point de vue métier
- C’est qui est difficile et important, c’est identifier ces contextes
- Mais aussi de bien les cerner les frontière (context boundaries)
- Un livre qui passe de l’éditeur à l’imprimeur, il doit y avoir quelques informations qui passent d’un contexte à l’autre (souvent, à part l’identifiant, il y en a très peu)
- On doit donc être capable de traduire une partie des informations qui sont visible dans un contexte vers un autre contexte, grâce à une translation map
- translation map = transcodification basé sur un partage d’identité
- Donc première chose à faire identifier les contexte et définir les limites et comment on va passer de l’un à l’autre
- Pour simplifier, un contexte = 1 domaine

Exemple d’un contexte : La vente aux librairies

Use case :

1. rendre visite à un libraire (client) pour lui présenter le catalogue
2. Prendre un commande pour un client

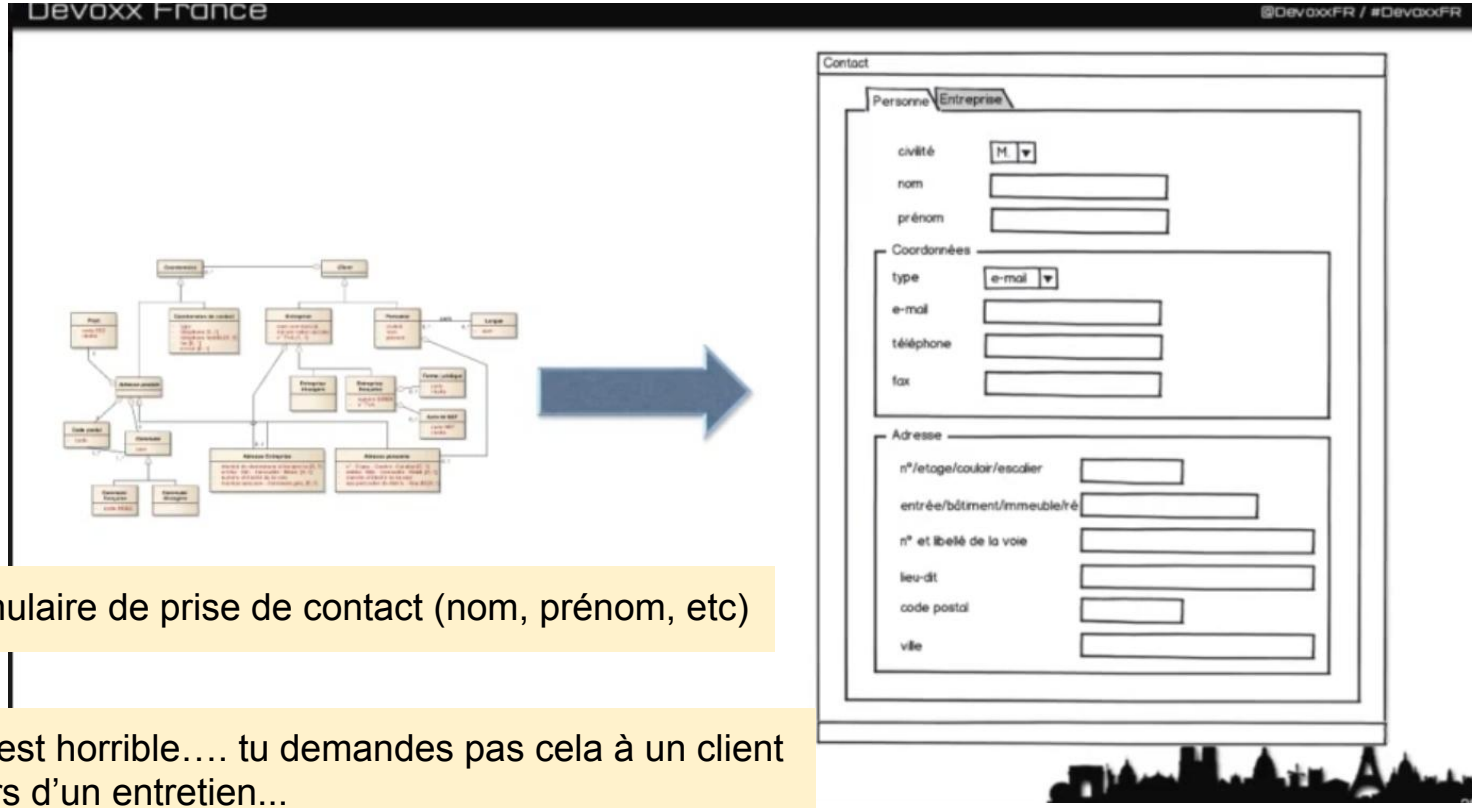
Comme le recommande Eric Evans, on s'intéresse au core domain : ici le **client**.

Qui est votre client ?

- Des entreprise, des librairies, de libraires
- il a des coordonnées de contact, pour faire la prise de commande
- Des adresse pour livrer les livres commander
- Langue des libraires (car international)

Exemple d’un contexte : La vente aux librairies

21:28

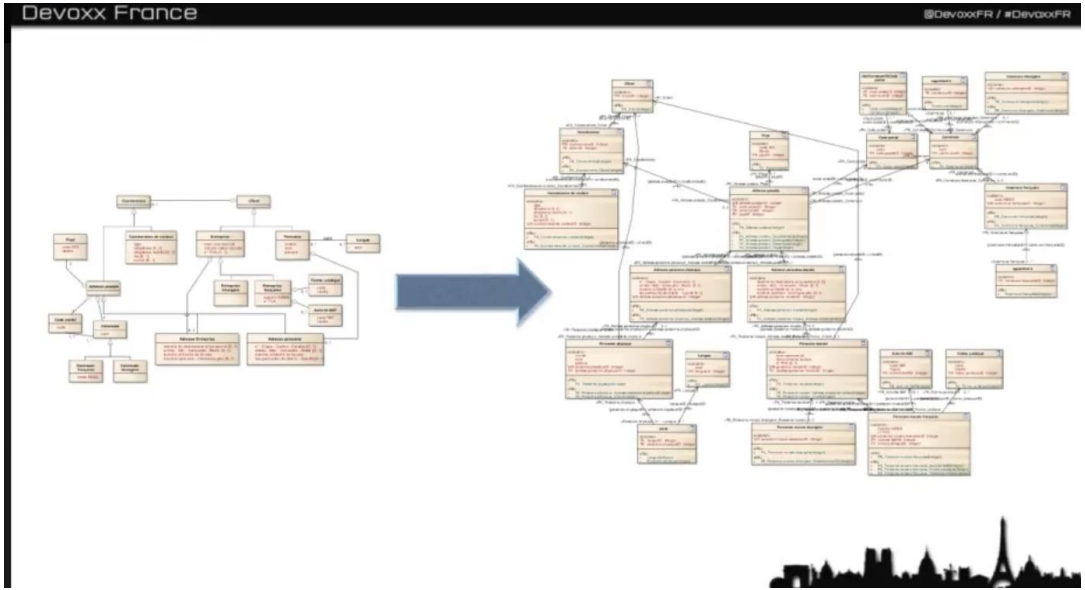


Formulaire de prise de contact (nom, prénom, etc)

C’est horrible…. tu demandes pas cela à un client lors d’un entretien...

Il y a quelque chose qui ne va pas pourtant on a suive les guidelines:

- On a interrogé le métier
- On a distillé son savoir dans le modèle
- On en a dérivé une IHM et un data modèle



Pourquoi ?

Parce qu'il y a une grosse différence entre le monde réel et le monde des logiciels

Ceci n'est pas un livre mais une projection d'un livre, une photo, une représentation

Le livre qu'on manipule dans un logiciel, c'est une représentation avec un pont de vue donnée, sans un contexte donné

Pour un **usage** donné

Ça a l'air simple

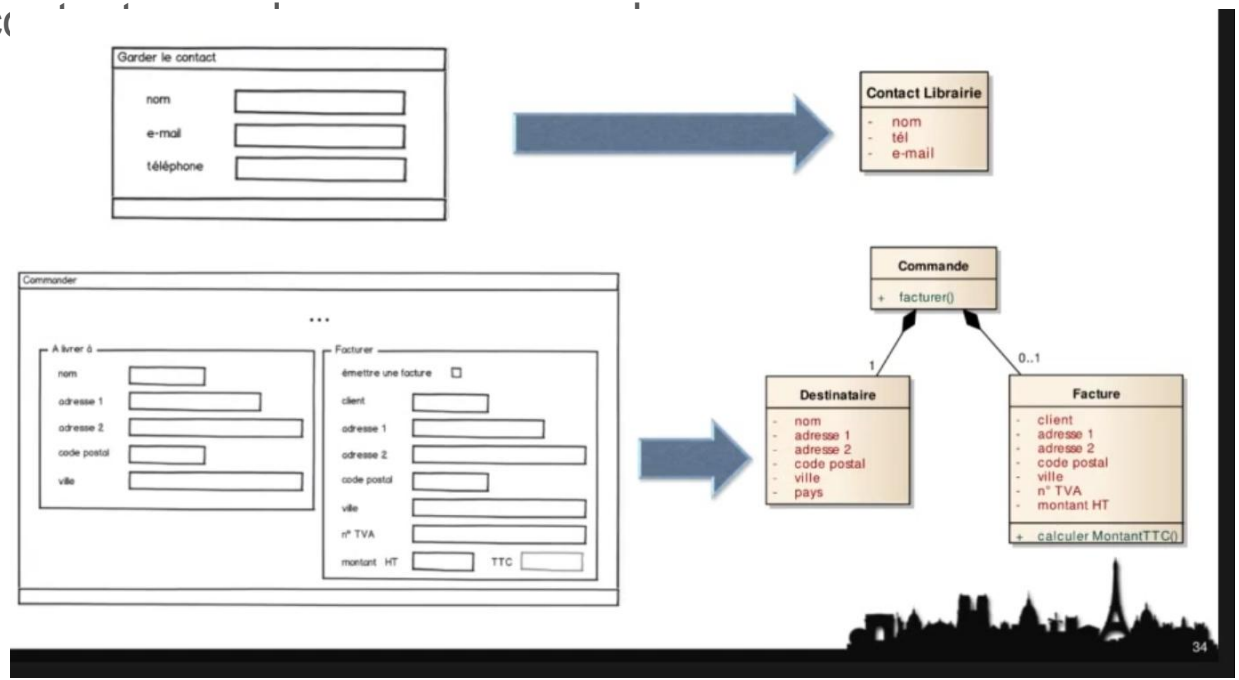
Ce n'est malheureusement pas ce qu'on voit

La vraie question ce n'est pas si le modèle représente bien la réalité ?

La bonne question c'est a quoi sert le modèle ?

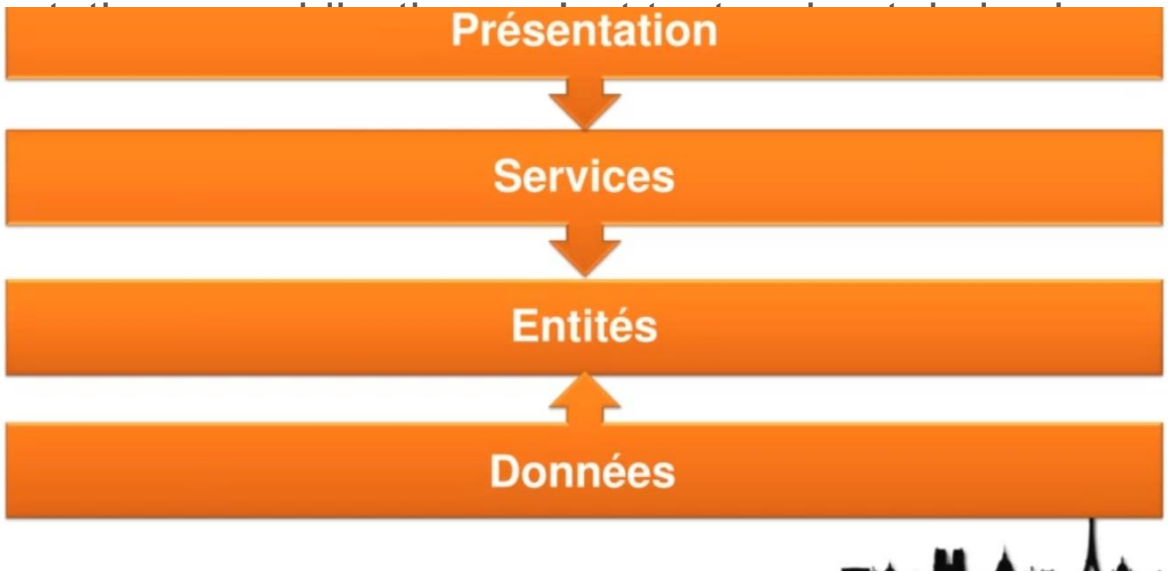
On va passer par les uses cases, les cas d'interaction avec le user.

Exemple : garder le c



On retrouve les même mots, on n'a juste pris ce qui nous intéresse, et on a réorganisé… c'est logique

On est parti de la représ
de dépendance.



Qu'est-ce que vous voulez faire avec vue logiciel

Pas besoin de modéliser un évier.. vous savez déjà ce que c'est…

Vous voulez gérer votre stock de consommables ?

Un bon processus de création du modèle c'est :

- On part de la présentation ou publication
- On en déduit des services nécessaires
- On en déduit des entités qui supportent l'état du logiciel

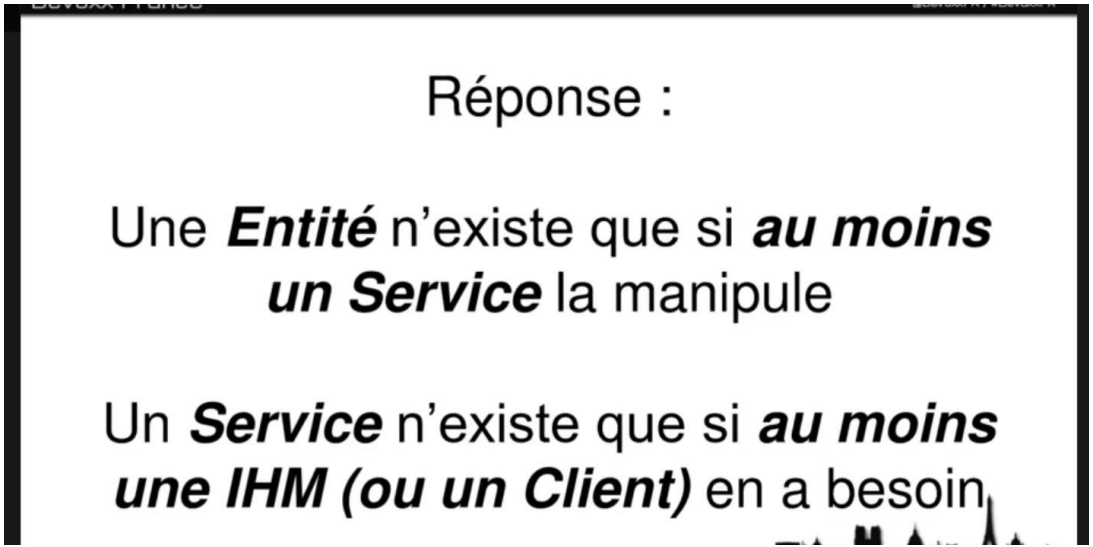
Pour expression du besoin orienté use case TDR Test-Driven Requirements

Expression de besoin en s'appuyant sur des scénario de test

BDD en DDD) behavior driven development. Description de scénarios

Un modèle écrit a priori ne répond à aucun besoin

Pourtant beaucoup de nos.models sont écrire a priori par une projection du.monde réel dans le logiciel



What about ubiquitous language?

Be l’a t on pas perdu ? Je pense que non.i

Le vocabulaire est le même. Les mots utilisés dans le modèle et donc dans le code sont ceux du métier. La seule chose c'est que ne sont pas organisés pareil

Le modèle du livre qui possède tout est intéressant, il représente un glossaire et un modèle du monde réel. Mais ce n'est pas un modèle de domaine dont on a besoin dans le DDD

Ce n'est pas un modèle issus d'un besoin logiciel

Questions réponses 36:09

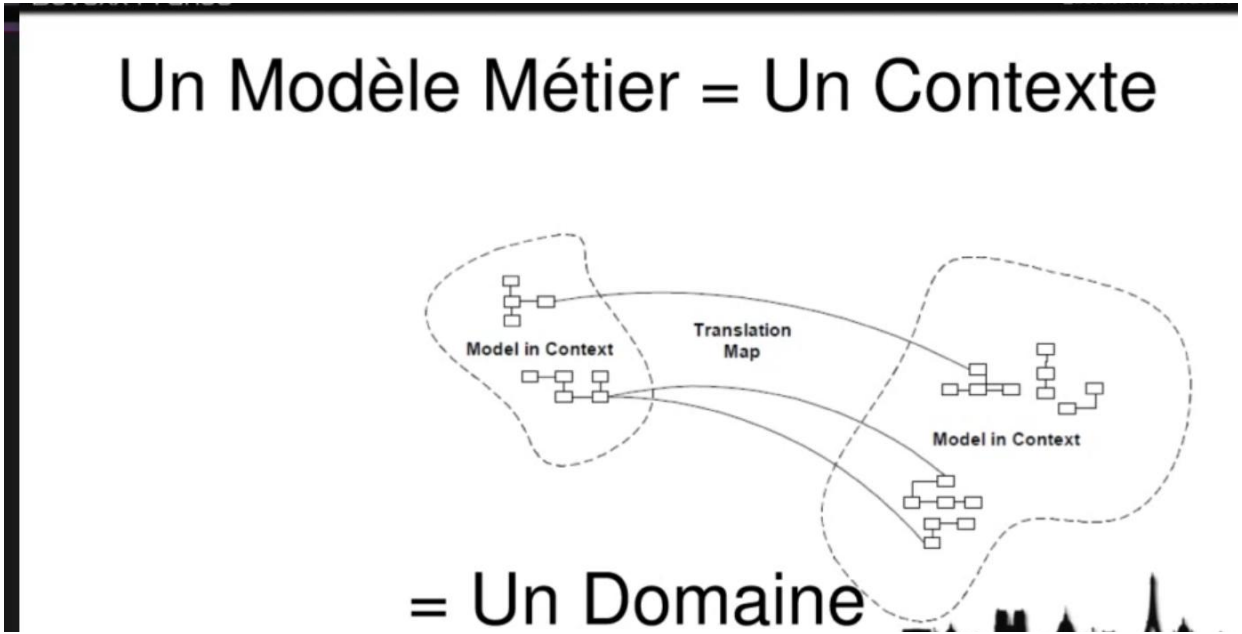
Le DSL ce n'est ni plus ni moins que un couche de présentation.sinom,

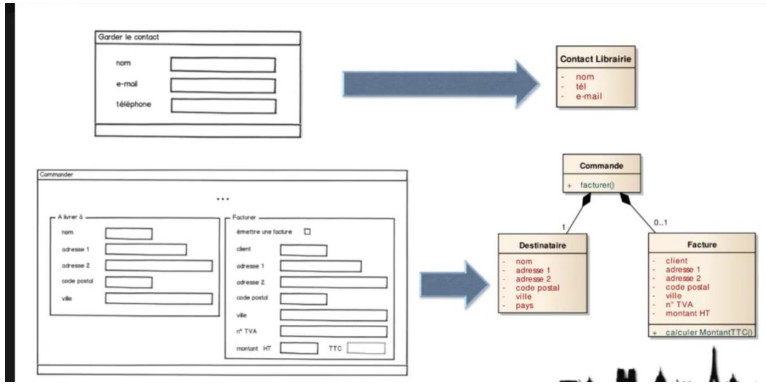
Dans une entreprise, le logiciel va mourir. On s'en fout du logiciel. Comment dire qu'il fait modéliser depuis une IHM alors qu'elle va disparaitre ? Ce qui est important c'est la data dans une entreprise ? On ne connait pas en avance les futurs besoins. N'est il pas important d'avoir la data complète pour pouvoir répondre à ces futurs besoins ?

Je ne dis pas qu'il ne faut pas canonisée les données, les faire rentrer dans un data warehouse. Mais c'est un autre usage, un autre besoin. Ces données qui seront plus pérennes parce qu'elle vont par exemple intéressé le pilotage de l'entreprise. Ce modèle ne doit surtout pas être le modèle de mon logiciel opérationnel. Il y a deux usages. Deux modèles. Un est sans doute plus pérenne que l'autre

Questions réponses 40:05

On aura plusieurs modèles. Et entre les deux, un ETL par exemple.





Modèle adaptatif?

Web sémantique ?

Derrière cette séparation en domaine, vous avez le coût caché de la translation. Pour passer d'un contexte à l'autre. On ne le voit pas sur votre slide?

Mettez ça en regard du coût d'utilisation du modèle générique...

Au lieu d'avoir une forte cohésion et un faible couplage, c'est l'inverse

En plus en terme d'évolution, plus rien ne peut bouge car tout le monde dépend du modèle global et centralisé

Il ne faut pas cacher le coût. Il faut montrer que le modèle qu'on a construit là, il est pour un domaine particulier, pour un ensemble de cas d'utilisation donné

Les données sont des conséquences des services qui sont eux même des conséquences des processus

Le cas de l'ETL qui va synchronise se la data vers autre chose est un cas particulier de use case. Pas pour servir un user, pour servir un autre service. Il faut l'envisager comme tel.

Les projets qui consiste aujourd'hui a mettre en place un gros référentiel centralisé, qui fait fois, et avec ça on est sauvé, on aura plus d'erreur. Parce que dupliquer la data c'est mal… 48:00 grâce a ça les commerciaux vont retrouver leurs contacts qu'il ont eu plus tôt, et leur revendre quelques choses qui ressemble a ce qu'ils ont déjà… le résultat est une usine a gaz totalement inutilisable la plupart du temps. Et gris modèle est le symptôme même du mauvaise système d'information fortement couplé, très peu évolutif

Plus grande difficulté

50:04 expliquer a ceux qui pensent que dupliquer une donnée c'est mal.

Ce qui est mal c'est de dupliquer une information !

Le nom du client a le droit d'être 10 fois dans le système d'information, si c'est pour des usages différént

L'orthographe du nom d'un prospect, c'est pas grave. Sur un client, ça l'est ! pourtant c'est la même personne physique