Developpez.com

Club des Développeurs

Pourquoi modéliser ?

Quel que soit le langage de modélisation ou la méthode utilisé, vous vous êtes peut être demandé pourquoi réaliser des modèles, à quoi cela peut-il vraiment servir à part me faire perdre du temps alors qu'il serait plus rapide de passer directement au codage. Vous avez peut être aussi travaillé à partir de modèles réalisés par les autres et vous n'avez pas vraiment saisie l'intérêt, voire vous vous êtes dit « mes ces modèles sont faux, je ne fais pas comme cela dans mon code ».

Bon, quelle que soit la raison pour laquelle vous avez eu envie de lire cet article, je vais tenter de vous convaincre que la modélisation, Merise ou UML, peut être utile à plusieurs points de vue. Allez, c'est parti…

Qu'est ce qu'un modèle ?

Un modèle est avant tout une représentation abstraite du monde réel. A ce titre, ce n'est pas le monde réel et donc ce n'est pas, exactement le monde réel. Un modèle offre donc une vision schématique d'un certains nombre d'éléments que l'on veut décrire ; un dessin quoi !
On a coutume de dire : *« Un dessin vaut mieux qu'un beau discours »* ; et bien écoutons et tentons de comprendre nos ancêtres, nos parents ou tout ceux qui ont pu nous répéter cette phrase. Réaliser un modèle c'est avant tout dessiner ce que l'on a compris d'un problème dans une syntaxe précise (la syntaxe UML ou Merise).

Un modèle va donc nous servir à **communiquer** et **échanger** des points de vue afin d'avoir une **compréhension** commune et précise d'un même problème.

Les niveaux d'abstraction

Lorsque l'on a à traiter un problème complexe, il est conceptuellement impossible de l'appréhender d'un seul bloc, dans son ensemble. Notre esprit à besoin de « dégrossir » le problème afin de pouvoir le comprendre pas à pas, petit à petit. Une fois le problème découpé en sous-problèmes de taille plus petite, l'analyse de chacun de ces sous-problèmes nécessite de commencer à en comprendre les grandes lignes puis d'affiner sa compréhension pour enfin comprendre tous les détails.
Ce mode de résonnement est à la base même des niveaux d'abstraction que l'on retrouve dans les méthodes comme Merise (niveau conceptuel, niveau logique, niveau physique) ou RUP[1] (niveau fonctionnel, niveau analyse, niveau conception).
Il faut donc bien comprendre la signification de ces niveaux et le nécessaire enrichissement d'un modèle de niveau N par l'analyse de niveau N+1. J'ai parfois entendu des concepteurs/développeurs critiquer vivement les modèles UML réalisés par des analystes. Outre que tout modèle est discutable, il est normal qu'un modèle d'analyse ne dise pas tout, qu'il soit incomplet du point de vue du concepteur. Le modèle d'analyse n'est qu'une vision du monde réel à un niveau de détail « analyse » et à ce titre il prépare la compréhension du monde réel pour les concepteurs (eh les concepteurs, y a quand même du boulot !)

[1] Rational Unified Process

Un outil pour documenter

« Bon, ok tout cela est bien gentil mais moi, dans l'équipe on a que des développeurs expérimentés et à partir des spécifications, ils savent faire la conception dans leur tête et écrire le code dans la foulée ; on va donc plus vite et on livrera le logiciel plus rapidement »

Ce type d'argument est malheureusement classique et parfois justifié sauf que pas mal de soucis peuvent en découler. Si un modèle, comme on l'a dit précédemment, sert à comprendre le monde réel, le problème de votre maîtrise d'ouvrage quoi, il est aussi un moyen de poser ses idées « sur la table », de ne pas les oublier et de les partager avec les autres membres de l'équipe. Un modèle est un langage commun, précis, qui est connu par tous les membres de l'équipe et il est donc à ce type un vecteur privilégié pour communiquer. Imaginez en plus le cas d'équipes géographiquement distribuées et de nationalités différentes (n'en déduisez pas que la modélisation ne sert que dans ce cas de figure☺).

« Bon, ok pour la communication mais notre équipe travaille main dans la main avec le client / la maîtrise d'ouvrage et on fait comme dans eXtreme Programming, on fait les spécifications en temps réel avec le client et on code 'en live'. La modélisation, pas besoin »

Là encore, pourquoi pas mais la communication ne s'arrête pas aux membres de l'équipe projet. Une fois mise en production, l'application va devoir être maintenue, probablement par une autre équipe et pas nécessairement de la même société que la société ayant créée l'application.

- Comment fournir une documentation suffisante à cette équipe de maintenance ?
- Comment assurer l'homogénéité de cette documentation entre plusieurs projets pour faciliter les passages d'un projets à un autre (c'est la capitalisation ça ?!) ?

Générer le code

Si les modèles servent donc à documenter le besoin de notre maîtrise d'ouvrage, à se mettre d'accord sur ce besoin, à affiner ce besoin,… les modèles ne sont pas que de la documentation ou de beaux dessins. Les outils actuels savent maintenant exploiter ces modèles pour faire de la génération de code voire des allers-retours entre le code et le modèle sans perte d'information.
Ouf, nous voila donc rassurés, les modèles vont être en plus utiles pour l'écriture du code. Pas de miracle à attendre non plus[2], mais les outils de modélisation vont quand même nous aider à générer la partie fastidieuse de ce code ; et nous permettre de nous concentrer sur les algorithmes et sur une analyse haut niveau du problème.
Mais attention à ne pas tomber dans le travers qui consisterait à « coder en UML » (avec Merise c'est plus dur). Rappelons nous les niveaux d'abstraction, c'est niveaux ont une réelle utilité. Certaines initiatives comme celle de Microsoft autour de la modélisation et de sa plate-forme .Net ont une petite tendance à pencher vers ce travers et à « détourner » les équipes de la vraie modélisation en proposant des outils de modélisation un peu trop proches du code. C'est tentant et probablement utile quand ont en arrive à un niveau d'abstraction proche du code comme la conception mais n'oublions pas les niveaux qui précèdent.

Bon, j'espère avoir convaincu au moins une ou deux personnes ☺
N'oublions pas que la modélisation est utile pour soi-même mais avant tout pour communiquer avec les autres, qu'ils soient son client, les autres membres de l'équipe ou encore l'équipe de maintenance.

---

[2] On verra dans un article futur que des techniques « miraculeuses » se sont quand même développées (programmation par aspects, génération à base de tags,…)