UML et la modélisation

Antoine Clave
Consultant indépendant en gestion de projet et en modélisation fonctionnelle

L'article paru dans la Lettre n°52 d'ADELI intitulé « UML : vers un Monde Lisible », de Dominique Vauquier, explique pourquoi « UML est un événement historique dans la communauté des informaticiens ! » et ce faisant, dresse un bref retour d'expériences sur la mise en œuvre d'UML dans les entreprises.

Dans sa conclusion, l'auteur reprend deux points développés : la grande richesse d'applications de ce langage et sa difficulté de mise en œuvre.

> *« En conclusion, UML est un outil précieux, mais, pour bien l'utiliser et en faire un instrument de lisibilité, il nous faut l'accompagner d'un mode d'emploi. Pour l'élaborer, il nous faut reprendre les questions dans la tradition du génie logiciel. »*

Ayant une certaine pratique professionnelle de la modélisation et de la gestion de projets, et d'UML depuis sa naissance, je pense intéressant de considérer ce bref mais riche article comme l'amorce d'un débat, débat dans lequel bien modestement je me permets de prendre place.

UML : c'est quoi ?

UML est un langage de modélisation (Unified Modeling Language), dont la mise au point est supervisée par un consortium de plusieurs entreprises (essentiellement américaines) : l'Object Management Group (OMG). Ce consortium comprend à la fois des éditeurs informatiques, des SSII, et des entreprises utilisatrices. Ce dernier point est à mon sens important. À noter que l'OMG n'a pas pour seul souci la maintenance d'UML.

À l'origine, UML est le résultat de la fusion de trois méthodes objets préexistantes :
- OMT de James Rumbaugh ;
- OOSE de Ivar Jacobson ;
- BOOCH de Grady Booch.

Cette initiative de fusionner ces méthodes (à une époque où les méthodes se déversaient sur le marché de manière quasi hémorragique) revient à Rational : éditeur indépendant de l'outil Rose.

UML propose 9 diagrammes :
- Cas d'Utilisation
- Classes
- Objets
- Séquences
- Collaboration
- États et Transitions
- Activité
- Composants
- Déploiement

Les auteurs d'UML ont parfois inventé ex nihilo, mais ont également emprunté. Par exemple, le diagramme des États et Transitions, inventé par Harel, a été repris pratiquement sans modifications. Enfin, certains acteurs ont essayé d'introduire des extensions plus ou moins fructueuses : IBM par exemple qui a proposé avec un maigre succès de modéliser les contraintes entre les entités du langage à l'aide d'OCL (Object Constraint Language).

Pour une plus ample description on se reportera au site : **http://www.omg.org**, sur lequel la version 2.0 d'UML sera disponible gratuitement sous peu.

Enfin insistons sur le fait qu'UML n'est pas une méthode mais un langage. De la même manière que connaître la syntaxe du C++ ne confère pas automatiquement une bonne compétence pour maîtriser un projet, ne connaître qu'UML ne permet pas de conduire, dans de bonnes conditions, une modélisation. Ceci dit, tout utilisateur d'UML est en droit d'attendre de cet outil les qualités requises pour sa finalité : offrir toutes les facilités pour fournir un modèle clair et productif.

Dans un premier temps, je propose de passer rapidement en revue les qualités et défauts attribués par Dominique Vauquier à ce langage, en y apportant une appréciation toute personnelle basée sur mon expérience professionnelle, et dans un deuxième temps j'essaierai d'entamer une réflexion sur la notion de modèle, question centrale de cet article.

Je souhaiterais insérer cette contribution dans un débat plus général, et, pourquoi pas, faire émerger une commission autour des questions de la modélisation.

Qualités et défauts reconnus à UML

Commençons par les bonnes choses …

Qualités d'UML

Langage standard de fait
UML est reconnu parfois comme un des standards du marché. Deux facteurs ont permis à UML d'accéder à ce statut :

- il résulte de la fusion de trois outils préexistant complémentaires qui déjà avaient une audience non négligeable (voir ci-dessus, ainsi que le site de l'OMG donné ci-dessus également) ;
- il a bénéficié de la force commerciale d'un éditeur particulièrement offensif : Rational, éditeur de l'outil Rose basé sur UML. Aujourd'hui, Rational est une entité membre d'IBM (depuis début 2003).
  Rappelons que Rational n'est pas le seul éditeur à œuvrer dans le monde UML et que certains des outils sont gratuits dans leur version personnelle.

Enfin l'utilisation d'UML est libre de droits.

Large spectre d'utilisation
Ce langage peut avoir un domaine d'application très étendu et les exemples cités par Dominique Vauquier prouvent qu'UML apporte des bénéfices à toute l'entreprise et pas uniquement à ceux qui s'occupent de son SI.

On pourra à ce sujet se reporter avec profit à l'article « l'Utilisation du langage UML dans les différentes phases d'un projet », paru dans IT-Expert, n° 44 de juillet 2003, et notamment le chapitre original traitant du dialogue entre les maîtrises d'ouvrage et d'œuvre du projet.

Capacité à décrire en emboîtant les niveaux d'abstraction
Les exemples cités ne seraient pas crédibles si UML ne savait pas gérer les niveaux d'abstraction habituellement reconnus et hiérarchiser les représentations. Ainsi, une entité du modèle (que ce soit un processus, une instance, un événement, un flux) pourra être située dans son niveau hiérarchique :

- le métier de l'Entreprise (niveau stratégique) ;
- les blocs fonctionnels, concourant à l'accomplissement de ce métier (niveau tactique) ;
- les fonctions (processus), chargées d'une mission identifiée (niveau opérationnel I) ;
- les activités unitaires (niveau opérationnel II).

Méta modèle accessible
Le langage UML donne accès à son méta modèle. Dominique Vauquier ne cite pas cette qualité non négligeable ; je me permets de la signaler. Un méta modèle est une représentation (une modélisation donc) des concepts utilisés par le langage lui-même. Cette représentation permet de théoriser sur l'outil, permet de l'enrichir facilement, permet de dégager des règles de bonne utilisation.

Par exemple, l'un des principes recommandés pour assurer la cohérence entre les diagrammes du modèle se formule ainsi :

- toute instance citée dans un diagramme des séquences doit être membre d'une classe citée dans un diagramme des classes ;
- toute classe doit avoir une entrée dans le dictionnaire des concepts métier et être citée dans au moins un Cas d'Utilisation.

Dès lors qu'on se préoccupe des relations entre les éléments constitutifs du langage, on attaque le méta modèle. Les promoteurs d'UML nous fournissent beaucoup de descriptions sur ses éléments constitutifs, nous permettant de formuler quelques critères qualité.

Défauts d'UML : les difficultés de mise en œuvre

Absence de « critères clairs pour garantir la qualité des modèles »
UML est destiné à modéliser. Il faut donc se préoccuper des objectifs de la modélisation et savoir apprécier la qualité des résultats atteints à l'aide d'UML.

Mais il faut se méfier des évidences … On n'a pas attendu UML pour modéliser. Warnier, Merise, OMT et bien d'autres se sont donnés pour but la représentation graphique des données et/ou des traitements intervenant dans le métier de l'entreprise.

De ce fait, la question à se poser serait plutôt : comment se fait-il que les modèles élaborés à l'aide d'UML laissent désormais apparaître ce manque de qualité ? Comment se fait-il qu'on exprime aujourd'hui ces exigences alors que jusqu'à maintenant elles ne prêtaient pas à débat ? Les modèles, depuis qu'ils ont été construits à l'aide d'UML sont-ils donc devenus des horreurs ?

Ce qui signifie, qu'aujourd'hui, il est interdit à un éditeur de fournir un outil sans qu'il se préoccupe de la qualité de ce qu'il permet de construire. Mais peut-on affirmer que l'impression de livres néfastes ait gêné la diffusion de l'invention de Gutenberg, à qui, de plus, devrait être reproché de ne pas avoir prévu les mauvaises utilisations de l'imprimerie ? Le mauvais comportement des conducteurs est-il un frein à la commercialisation des automobiles et les constructeurs de voitures se sentent-ils coupables des accidents routiers ?

Tout est question de responsabilité …

Ceci étant posé on admettra bien volontiers l'importance de cette question, même si UML n'y est pas pour grand chose. On posera comme postulat (parfaitement discutable) que si les modèles ne sont plus satisfaisants, ce n'est pas parce qu'ils se sont dégradés, c'est parce qu'ils sont devenus à la fois indispensables et beaucoup plus ambitieux. C'est

pourquoi on a choisi de décaler la discussion de l'avenir d'UML vers l'avenir de la modélisation.

L'invérifiable pertinence des diagrammes
« Il n'est pas rare de voir réaliser de nombreux diagrammes qui, finalement, ne servent à rien ».

Cette critique soulève deux questions :
- d'une part, comment choisir les diagrammes adaptés à ce qu'on veut modéliser, plutôt que noyer le lecteur sous une masse de schémas dont la cohérence n'apparaît pas (à supposer qu'on sache expliquer la cohérence entre des diagrammes) ?
- d'autre part, comment rendre ces diagrammes productifs ?

À la première question on aura répondu lorsqu'on aura examiné la notion même de modèle, et comment un modèle se bâtit. Quel est le point de départ ? Que doit-on fournir en amont ? Qu'est-ce qui justifie chaque composant d'un modèle ? Comment conduit-on la modélisation ?

La deuxième question évoque l'utilisation du modèle et ce qu'on en attend. Qui sont les utilisateurs ? Qu'en font-ils ? Quel est le but de la modélisation ?

Modélisation, j'écris ton nom

Nous essaierons très modestement de décrire :
- la notion de modèle ;
- ce qui est habituellement modélisé ;
- comment on modélise ;
- et enfin l'utilisation qui est faite d'un modèle.

Ce chapitre est exclusivement basé sur des expériences professionnelles personnelles et n'a absolument pas l'ambition de brosser une théorie des modèles. Il se place dans la perspective dégagée ci-dessus : un échange autour de cette activité.

Qu'est-ce qu'un modèle ?

Un modèle est une représentation exacte, partielle, actuelle, passée ou future, de la réalité. Mais d'abord :

Que représente-t-on ?
Tous les modèles présentent des acteurs entre lesquels s'écoulent des flux. Cet écoulement est provoqué par des événements et est soumis à des règles. Ces flux peuvent être matériels (exportation de voitures, évacuation des distillats d'une raffinerie) ou informationnels (flux financiers, rapports de police, factures, …). Un acteur est origine ou cible d'un flux dont la production et la réception est conditionnée par l'état de cet acteur. L'état d'un acteur est déterminé par des événements, internes ou externes.

Notons que chaque flux informationnel doit avoir une origine, qui elle-même est également de nature informationnelle. Autrement dit, l'information se déplace et le système la recombine ; elle n'est pas créée par ce système à modéliser.

C'est une représentation
On utilise donc un média fait de signes conventionnels (son vocabulaire) et de leur signification (sa sémantique) ainsi que de règles d'utilisation (sa syntaxe).

Ce média peut-être graphique, ou essentiellement textuel, ou combiner les deux. Dans le cas où on utilise du texte, il convient de suivre des règles de description (phrases courtes, sans adverbes, ponctuées, au registre du vocabulaire strictement maîtrisé et limité aux termes du métier des lecteurs) qui appliquées au langage naturel, font de celui-ci un outil bien éloigné du langage littéraire et poétique … Un modèle textuel tient plus du rapport de police que du discours de politique générale (une phrase = sujet + verbe + complément, et un point. Pour le reste OK mais d'abord, passez me voir[1]).

Si la représentation est graphique, on remarquera que la sémantique surgit des icônes (unités graphiques élémentaires), mais aussi de leur combinaison. Par exemple, de la combinaison des flèches représentant les messages circulant entre des acteurs, on en déduira celui ou ceux qui ont l'initiative du dialogue, donc celui ou ceux qui a préemption dans le cadre de ce processus. Ce qui permettra de faire l'économie de l'icône particulière dédiée aux acteurs préemptifs.

Donc quel que soit le média de modélisation, de l'utilisation réfléchie et économique de l'outil peut survenir un sens non prévu par les propres promoteurs du langage de modélisation. Et remarquons que cela va de pair avec une représentation simplifiée, sobre et concise.

La tâche d'un consultant assistant une équipe de modélisation sera donc de dégager, d'expliquer et de faire appliquer les règles de ce média.

Cette représentation est exacte
Cette représentation ne doit pas révéler de contradictions ni de manques. Elle ne doit pas être ambiguë.

Contradictions
Quelques exemples qu'un modélisateur cherchera prioritairement à débusquer :
- contradiction entre les rôles attribués à un acteur et les services qu'il peut invoquer (règle de cohérence entre profil et services accessibles) ;

1 « Faites des phrases courtes. Un sujet, un verbe, un complément. Quand vous voudrez ajouter un adjectif, vous viendrez me voir. »[Consignes aux journalistes de L'Aurore.] Georges Clemenceau

- contradiction entre les transitions d'états d'un acteur et les événements dont il peut être la cible (règle de cohérence entre les états d'une instance et les événements qu'elle peut interpréter) ;
- contradiction entre les cibles des messages qu'un acteur peut envoyer et les cibles qu'il a l'autorisation de voir (règle de visibilité entre les classes) ;
- contradiction entre les données consommées par les traitements et les données accessibles à ce moment (règle de cohérence entre les informations disponibles et les informations exigées).

Manques

- Pasteur a réfuté en 1855 la théorie de la génération spontanée : cette théorie qui s'applique aux êtres vivants peut être étendue aux événements et circulations que nous modélisons. Ainsi :
  - chaque changement (changement d'état d'une instance, ou déplacement d'une information) doit être initié par un événement ;
  - chaque événement doit être provoqué par un acteur ;
  - chaque acteur doit être une instance de classe douée d'une autonomie d'action qui doit être décrite dans ses droits.

- Un modèle n'est pas seul ; il fait suite à un problème à résoudre, une mission donnée, un périmètre défini. Deux conditions doivent être vérifiées impérativement :
  - que le problème, la mission, le périmètre soient clairement exposés (le cahier des charges, le schéma prévisionnel ou directeur) ;
  - comme rien n'est fixe ni immuable, et pour respecter un certain droit à l'erreur, que les conditions de changements (du périmètre, des fonctionnalités) soient prévues et donc que le processus de modélisation lui-même soit capable d'encaisser les volte-face des donneurs d'ordre.

- Chaque entité modélisable (acteur, message informationnel, événement) présente trois dimensions :
  - dimension fonctionnelle : quel est sa raison d'être ?
  - dimension statique : quelle est sa nature ?
  - dimension dynamique : quelle est son pouvoir d'action ?

Le modélisateur vérifiera la présence de ces trois dimensions (parfois squelettiques, parfois diarrhéiques) dans la description de chaque entité.

Cette représentation est partielle

Des limites sont imposées ; ces limites font parties de la mission. Mais une modélisation est toujours relative à un point de vue : celui du modélisateur, celui d'une tierce personne.

Le point de vue depuis lequel on modélise doit être clairement désigné, sous peine de faire passer cette partialité du point de vue pour une incomplétude du modèle.

Un modèle peut s'articuler autour de plusieurs points de vues ; ainsi on peut modéliser le même processus vu depuis plusieurs secteurs fonctionnels de l'entreprise. En particulier il est bon de clairement dissocier les processus de management, des processus de production et des processus de soutien (support).

La situation à représenter n'est pas forcément actuelle

Bien entendu, la modélisation peut être destinée à représenter ce qui doit être réalisé (projet). Dans ce cas le modèle est non seulement résultat de la représentation mais aussi un outil de réflexion pour la mise au point d'une solution à déployer.

De ce fait, le modèle est un support de dialogue et d'échanges de propositions. Cette pratique est bien intégrée par les informaticiens qui ont besoin de mettre en évidence ce qui doit être automatisé (pris en charge par le système) et ce qui doit rester manuel. Dans ce but, il est bon de situer les exigences fonctionnelles du système au sein des exigences fonctionnelles du métier de l'entreprise.

Qu'est-ce qu'un bon modèle ?

Des Sociétés de Service mettent sur le marché leur compétence à mener correctement une modélisation, sous forme de « bonnes pratiques » (best practices), relatives au but recherché. Cette bonne pratique visera à dégager la bonne adéquation entre l'outil mis en œuvre et l'objet de la modélisation. Certaines (Softeam, Progress and Software, …) se posent explicitement la question de la bonne qualité de la modélisation.

Combinée avec les propositions dégagées ci-dessus nous pourrions décrire quelques qualités présentées par un bon modèle. Là encore cette tentative n'a pas d'autre ambition que de s'inscrire dans un débat, est le résultat d'une certaine pratique professionnelle et ne prétend pas à l'exhaustivité.

Lisibilité : critère qui semble évident mais qui passe par le respect de règles de nommage, de construction, de vocabulaire … trop dépendant du métier de l'entreprise et donc sur lesquelles je ne m'étendrai pas.

Pertinence : ce qui est représenté répond-il bien à la question ? Cas classique d'un modèle non pertinent : le cahier des charges destiné à poser un problème à résoudre (et donc à le modéliser) qui est souvent rédigé en termes de solution. Dans ce cas précis du cahier des charges, analyser le registre du vocabulaire employé est un bon moyen de se prémunir contre la non pertinence : pas de termes techniques mais uniquement du vocabulaire utilisateur. D'une manière plus générale il convient d'assurer l'adéquation entre le niveau de description et le niveau hiérarchique.

Complétude : ce qui est représenté est-il un panorama complet du point de vue de l'observateur ? Pour le vérifier, montrer que les trois niveaux sont traités (fonctionnel, dynamique, statique), qu'à chaque fonctionnalité sont affectées les unités chargées de la mettre en œuvre et que ces entités vérifient les règles d'exactitude citées plus haut. On ne perdra pas de vue que la complétude est relative au but recherché, c'est-à-dire à l'utilisation ultérieure de ce modèle.

Productivité : un modèle a une finalité ; cette finalité est-elle atteinte ? Tout modélisateur / analyste doit savoir répondre à cette question : quelle équipe attend mon travail ? que va-t-elle en faire ? À ce titre, les processus du projet méritent d'être considérés comme n'importe quel autre processus opérant au sein d'une entreprise.

Unicité des descriptions : ne pas confondre : multiplicité des points de vue et donc multiplicité des descriptions, avec redondance (dire la même chose une deuxième fois, sans apport nouveau). Cette qualité est délicate à évaluer.

Criticabilité : on se permettra ce néologisme pour exprimer qu'un modèle doit être la cible de commentaires de tous les acteurs concernés, donc inclut la qualité de lisibilité, de clarté, de non ambiguïté. Le silence comme réponse à une proposition n'est pas un bon signe …

Que fait-on d'un modèle ?

Je parlerai dans cette section de la seule utilisation que je connaisse : l'utilisation des modèles dans le cadre d'un projet informatique.

Les informaticiens ont la possibilité de faire du modèle un outil de production du code compilable. Au risque de paraître surprenant, nous considérerons le code informatique comme un modèle en lui-même. De ce fait, il en résulte que programmer, c'est modéliser, et qu'il y a un continuum entre la modélisation et la production du code exécutable. Le langage du modèle serait un langage interprétable.

L'exécution de ce code doit fournir un résultat, conforme à ce qui est attendu. Cette conformité évoque bien évidemment la dimension test. Il est évidemment possible (mais pas facile) de tester un code exécutable (exécutons-le), mais la question : « un modèle est-il testable ? » est cruciale.

Autant que je sache, tester un produit informatique revient, du moins à l'heure actuelle, à tester son exécution. Et donc les critères d'acceptation sont des critères se rapportant à un ensemble de résultats d'exécution et de conditions d'obtention (les performances).

Ce qui semblerait rejeter pour longtemps encore les modèles UML dans la catégorie des modèles non testables. Désespoir dans le camp de ceux qui réclament l'évaluation du niveau de qualité des modèles UML …
D'où le problème qui se pose à nous maintenant : comment mettre au point une série de critères qualité qui permettent de se passer du résultat d'exécution ?

Ces critères seraient d'ordre anticipatif (prédictif : ce que ce modèle va vraisemblablement donner …), descriptif (respect de normes ébauchées ci-dessus, auxquelles la nature même d'UML permet d'ajouter plusieurs principes), et donc ressembleraient à un état des lieux. Mais quelle garantie offre-t-il au responsable payeur final ?

Enfin examinons la nature de la description du fameux résultat qui est attendu : il est décrit dans le cahier des charges, qui lui-même est un modèle, et donc qui devrait être testé.

Tester un modèle revient donc à tester la conformité de ce modèle à ce qui lui a donné naissance, donc au modèle antérieur, antécédent. Ne reculons pas devant la question qui en découle : un cahier des charges est-il testable ? Au moins cet angle d'attaque permet d'éviter les sempiternelles récriminations de la maîtrise d'œuvre vis-à-vis de la maîtrise d'ouvrage (et son assistance), chargée de la rédaction de ce document.

Conclusion : débattons autour d'UML, langage de modélisation

Plusieurs questions mériteraient confrontation d'expériences et alimenteraient d'autres suites à l'article de Dominique Vauquier ; je cite en vrac :

- UML est une caisse à 9 outils (9 diagrammes) : quand et comment utiliser chacun d'eux ?
- Critères qualités d'un Modèle : qu'est-ce qu'un bon modèle ? Dans quelle mesure la connaissance du méta modèle d'UML permet d'améliorer considérablement la qualité des modèles écrits ? Peut-on trouver dans ce méta modèle de quoi vérifier la cohérence, la fermeture sémantique, la complétude des cycles de vie, … ?
- Est-il raisonnable de se servir d'un modèle UML pour générer du code informatique ?
- Comment tester un modèle ?
- Puisque UML a un point d'entrée dans le cahier des charges, dans quelle mesure la qualité de ce livrable conditionne-t-elle la suite des productions ?
- Le processus de modélisation avec UML est-il différent des processus de modélisation mettant en œuvre d'autres langages ? ▲

antoine.clave@wanadoo.fr