

# Que nous apporte Réellement UML2?

## *1<sup>ère</sup> partie : Cible et portée de UML2*

© copyright Objecteering Software 2005

Philippe Desfray (voir à [propos de l'auteur](#))

jeudi 24 novembre 2005

Version 0.1

## I. Présentation

Décembre 2006. La commission "UML RTF" de l'OMG<sup>1</sup> va terminer ses travaux sur UML2, consacrant ainsi l'aboutissement de ce standard. Adopté en Juin 2004, UML2.0 qui est une révision majeure de UML1.4, est déjà supporté par un nombre important d'ateliers UML, d'autres devant annoncer prochainement leur support. Il reste que UML1.4 demeure très majoritairement utilisé, les nouveaux concepts UML2 n'étant pas encore bien compris, mis en œuvre et méthodologiquement supportés.

Cependant que la poussière des volutes marketing retombe, il est temps de faire un point sur les vrais apports du standard UML2.0. Ce "white paper" rédigé par un contributeur actif<sup>2</sup> à l'élaboration de ce standard décrit ses vrais apports et les fausses nouveautés, en fournissant un éclairage pratique sur ce standard.

Ce document est la première partie d'un White Paper décomposé en trois articles publiés séparément:

- 1 Cible et portée de UML2: Nous décrivons à qui UML2 s'adresse, et son degré d'universalité,
- 2 *UML pour chaque phase du cycle de vie: L'intérêt et la pratique de UML pour chaque étape du cycle de vie sont discutés, avec des exemples pratiques,*
- 3 *Choisir UML2 et MDA: La mesure de l'ampleur du changement de UML1.4 vers UML2 est établie, et l'intérêt de l'adoption conjointe d'une démarche MDA est évalué.*

---

<sup>1</sup> Revision Task Force de l'organisme de standardization Object Management Group, chargé de consolider le standard UML2.0, en prenant en compte les remarques émises sur le standard de part le monde

<sup>2</sup> Les soumissionnaires initiaux (core team) ayant construit le standard sont: Computer Associates, Ericsson, I-Logix, IBM, IONA, Kabira Technologies, Motorola, Oracle, Rational Software, SOFTEAM - Objecteering Software, Telelogic, Unisys, et WebGain

## II. A qui s'adresse UML2

UML2 adresse fondamentalement la même cible que UML1.4. Ses extensions permettent cependant d'élargir la gamme de ses utilisateurs en offrant des outils de modélisation plus adaptés.

Une forte composante liée à l'informatique "technique" (télécoms, temps réel) est intervenue dans les évolutions de ce standard. On peut par exemple citer des sociétés comme MOTOROLA ou Ericson, ou des outilleurs de ce domaine comme Telelogic. L'influence de ces sociétés a renforcé UML dans la précision de certains types de diagrammes (séquence notamment), et les capacités de UML relativement à la modélisation des architectures logicielles.

Des influences de natures différentes ont été exercées sur d'autres aspects du standard UML. Ainsi, les diagrammes d'activité ont-ils été refondus, afin d'apporter un meilleur support à la modélisation des processus métiers. Le mécanisme, encore trop peu exploité, des flux d'information (information flow) permet par ailleurs de construire des modèles de haut niveau lors des phases préliminaires, pour par exemple soutenir une approche systémique.

UML a évolué pour offrir un support plus complet et plus compatible à la modélisation liée à des langages de programmation. On peut citer les packages et la sémantique d'import, qui ressemblent beaucoup à la sémantique de Java, les notions d'interface plus proches de celle des langages, le support des "templates", maintenant bien adapté à ce qu'offrent des langages comme C++, C# ou Java.

UML2 a donc un large spectre d'application, s'adressant au programmeur comme à l'architecte, au monde de l'informatique technique comme à l'informatique de gestion, ainsi qu'aux étapes plus amont, comme la maîtrise d'ouvrage ou comme l'ingénierie des systèmes.

## III. UML2: Langage universel?

Cependant que UML, fort du succès de sa version précédente, consolide ses capacités de modélisation, apparaît un mouvement de contradicteurs prônant la définition de DSL: Domain Specific Languages. L'un des participants actifs de ce mouvement n'est autre que Microsoft... Les tenants de l'approche DSL arguent que pour chaque domaine et cible, il est plus efficace de produire un langage de modélisation adapté.

Il est certainement inapproprié de prétendre que UML supprime tous les langages de modélisation. Citons ainsi deux exemples qui ne sont pas directement couverts par UML: la représentation tabulaire, qui est sans doute le modèle le plus populaire et qui se révèle très puissant et souple lors de son utilisation avec des tableurs; Les synoptiques de supervision de processus (réseaux, trafic, etc.) qui adoptent une représentation collée à la réalité en représentant directement des capteurs, des véhicules, des stations, des vannes, et tout autre équipement réel supervisé. Il y a ainsi un grand nombre d'exemples de modèles "DSL" que l'on peut imaginer, effectivement par domaine comme les modèles de chimie, de mathématique, de plans de bâtiments, pour lesquels on n'imagine pas que UML soit une alternative crédible.

De fait, UML "langage universel" cohabite donc avec les "DSL". Il y a une zone de compétition lorsqu'en adaptant UML par le mécanisme des profils UML, on rend celui-ci spécifique d'un domaine. Par exemple, des profils UML existent pour modéliser les composants et architectures hardware et la répartition des logiciels sur ces composants. On pourrait également faire un profil UML pour modéliser un synoptique de supervision. Il est vain d'opposer les deux approches, mais utile de rappeler les avantages à rechercher dans chacune des deux:

- UML est LE langage de modélisation standard. Ceci apporte l'avantage d'un support riche et varié par de nombreux outils, avec de nombreux services, et d'une connaissance du langage par une très large partie des informaticiens.
- UML peut être adapté facilement par la technique des profils, et peut combiner plusieurs adaptations pour par exemple mixer des domaines d'application et des cibles techniques. En ce sens, il sera plus flexible que les DSL. Marier deux DSLs dans un même modèle n'est pas tâche aisée - de fait cela conduit à créer un troisième DSL – cependant que plusieurs profils se combinent naturellement sur un même modèle UML.
- UML ne couvre pas tout, et n'a pas une reconnaissance forte dans tous les domaines. Là où des langages spécifiques sont largement pratiqués, il n'est pas opportun d'appliquer dogmatiquement UML.

Le débat s'est étendu, car d'une part UML a augmenté ses capacités de représentation, et donc ses possibilités d'utilisation pour différents domaines et techniques, et d'autre part Microsoft fournit des solutions en amont de ses outils et architectures sous forme de DSL. Dans le cas d'une cible purement Microsoft, l'option est de préserver l'indépendance des modèles en s'appuyant sur UML ou de privilégier l'efficacité et l'adéquation à l'environnement en s'appuyant sur ses DSLs. UML est bien souvent choisi pour les parties amont de la modélisation, de l'analyse préliminaire à la conception technique, pour ensuite basculer dans le monde Microsoft, où le DSL agit comme un langage graphique équivalent au langage source associé (comme par exemple C#).

A noter par ailleurs, que de nombreux profils standards UML apparaissent pour différentes cibles, comme par exemple des profils pour Java/EJB, des profils pour SQL, pour XML, pour l'ingénierie des systèmes, pour le test, pour la qualité de services, etc.

Pour les besoins de modélisation logiciel ou de systèmes au sens large, UML a un bon niveau d'universalité, très renforcé par UML2. Son utilisation se généralise davantage encore chaque jour. UML cohabitera cependant avec d'autres formalismes:

- Les formalismes imposés par certains éditeurs de plateforme, comme le cas typique de Microsoft, ou d'atelier Workflow. Cependant, même ces formalismes s'inspirent de UML.
- Les tableaux, qui complètent UML par exemple pour la gestion des dictionnaires, des besoins, ou des objectifs (goals).
- Les langages spécialisés, comme les réseaux de pétri ou des diagrammes d'états dédiés à certaines approches.

## IV. A Propos de l'auteur

**Philippe DESFRAY**, co-fondateur et directeur technique de SOFTEAM - la maison mère d'Objecteering Software - est un expert internationalement reconnu des modèles et méthodes, en particulier centrés sur UML. Créateur de la méthode objet "Classe Relation" dans les années 1990, il a publié trois livres, en particulier "Object Engineering - The fourth dimension - ADDISON WESLEY - 1994.", et a piloté le développement de l'atelier UML « Objecteering ». Précurseur des technologies « MDA<sup>3</sup> », il a introduit dès 1994 des outils supports de cette approche. Depuis 1994, Philippe Desfray représente de SOFTEAM au sein de l'OMG où il participe à l'élaboration de plusieurs standards, dont notamment UML. Ses travaux continus sur le développement guidé par le modèle l'ont conduit à participer, dès l'origine, à la définition du standard UML, en y dirigeant la définition de nouvelles notions comme par exemple les « profil UML », et à faire développer son support outillé au sein de l'atelier Objecteering. Dirigeant une forte activité R&D au sein de SOFTEAM et en coopération avec de grands organismes européens, Philippe Desfray œuvre pour une application directe des résultats au sein des diverses activités de SOFTEAM : conseil, formation et support outillé par l'atelier Objecteering.

---

<sup>3</sup> « Model Driven Architecture » : démarche et technologie sous tendant les nouveaux standards de l'OMG