

Que nous apporte Réellement UML2?

3ème partie: Choisir UML2 et MDA

© 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'OMG¹ 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 actif² 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 troisième et dernière 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 *UML2 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.*

¹ 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

² 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. UML2/UML1.4 : l'ampleur du changement

Le tableau ci-dessous synthétise l'ampleur des changements intervenus entre UML1.4 et UML2.0. Les éléments nouveaux ou très impactés sont notés "+++", cependant que les moins impactés sont notés "-".

Mis à part quelques points mineurs, relevant plutôt de la cohérence d'ensemble de la notation, l'ensemble des modèles UML1.4 existe dans UML2.0 qui les étend par de nouvelles capacités. Les *flux d'information* sont une nouveauté importante, utile pour modéliser au plus haut niveau les systèmes. Les *diagrammes d'activité* ont été fortement réaménagés, pour apporter trois nouveautés essentielles:

- **Indépendance des activités:** sous UML1.4, une activité appartenait toujours à un contexte, comme typiquement une classe. Dorénavant, une activité existe au plus haut niveau du modèle, au même titre qu'un package ou qu'une classe. Ceci correspond à la notion de processus, identifiée dès les étapes initiales d'une cartographie, et vue comme une notion de premier ordre.
- **Décomposition des activités:** les activités sont dotées de paramètres, qui permettent de séparer l'invocation d'une activité (usage de "pins" pour les paramètres actuels) de la définition d'une activité (qui déclare ses paramètres). Ainsi, les activités peuvent être décomposées en "poupées russes", et une même activité peut être référencée depuis plusieurs activités.
- **Extension de la sémantique des activités:** les mécanismes des diagrammes d'activité se sont considérablement enrichis, apportant plus de capacité d'expression et plus de souplesse. A titre d'exemple, la gestion des événements et exceptions permet de simplifier certains modèles, l'enrichissement des partitions offre la capacité de décrire finement les responsabilités si chères aux MOT (Modèles Organisationnels de Traitement) merisiens.

| Modèle | Degré d'évolution | Commentaire |
|------------------------|-------------------|---|
| Information Flow | +++ | Nouveauté |
| Activity diagram | +++ | Extensions et enrichissement pour la modélisation des processus et "workflow" |
| Interaction diagram | +++ | Structuration et Structures de contrôle sur les diagrammes de séquence. |
| Collaboration diagram | ++ | Unification avec "parts" et "ports" |
| Class diagram | ++ | Nouveautés: parts, ports, structure interne |
| Deployment diagram | + | Refonte et généralisation |
| State diagram | + | Structuration |
| Protocol state diagram | + | Formalisation, State invariant, pré & post conditions |
| Use Case | - | Quasi inchangé pour l'utilisateur final. |

Les *diagrammes de séquence* (*interaction diagram*) ont été totalement refondus, afin de leur donner plus de précision et de capacité de modélisation. Il est maintenant possible de décomposer des diagrammes de séquence en "sous diagrammes de séquence" et d'employer

des structures de contrôle au sein des diagrammes de séquence. Leur extension a surtout été inspirée par les besoins des domaines des télécoms, qui doivent exprimer finement des protocoles.

Les *diagrammes de collaboration* sont peu utilisés par les praticiens de UML1.4. Peu s'intéressent donc à leur évolution, qui du point de vue de la représentation reste stable.

Les *diagrammes de classe* ont bénéficié de quelques extensions. Dans une grande majorité de cas, ceux-ci restent inchangés par rapport à UML1.4. Lorsque des besoins de modélisation architecturale apparaissent, alors de nouvelles notions sont utilisées, comme par exemple les composants, les parts et les ports.

Les *diagrammes de déploiement* corrigent des insuffisances de UML1.4. Les notions d'artifact permettent de modéliser la chaîne de production d'une application, en décidant des éléments gérés et produits, comme par exemple des bibliothèques ou des exécutables.

Les *diagrammes d'état* peuvent être structurés et factorisés pour être réutilisés. On distingue maintenant les diagrammes d'état *de protocole*, qui permet de décrire le cycle de vie des objets.

Les *diagrammes de Use Case* enfin, ne présentent pas d'évolutions notables.

III. UML2 et MDA

Parler de UML pour les phases amont, pour l'architecture et pour la programmation, nous conduit inévitablement à parler de l'approche MDA³. En effet, l'approche MDA nous permet de définir le support de ces différents aspects de la modélisation en les outillant spécifiquement, puis de faire le lien entre eux en automatisant ou accompagnant les transformations à effectuer, et donc d'assurer la maîtrise complète du cycle de développement d'application en définissant chaque étape et chaque transition entre étapes.

UML1.4 fournissait déjà les moyens de supporter une approche MDA: ainsi, l'atelier Objecteering avec son outil "Profile Builder" permettait de spécialiser UML pour chaque aspect devant être couvert, et d'automatiser les transformations de modèle nécessaires.

L'apport de UML2 s'effectue à travers l'ensemble des standards qui l'accompagnent: MOF2.0, OCL2.0, XMI1.2 sont en effet difficilement dissociables. Deux éléments sont à noter particulièrement:

- La mise en cohérence de ces standards, et leur uniformisation, qui permettent de simplifier les outils associés, ainsi que d'augmenter l'interopérabilité entre ces ateliers,
- Les profils UML2.0. Ceux ci ont été consolidés sous UML2.0. Ils permettent eux-mêmes d'être modélisés, et d'avoir le support d'outils qui assistent le travail de construction de paramétrage MDA. Ainsi, l'atelier Objecteering/MDA Modeler permet de définir des générateurs de documentation ou de code très rapidement sans programmation spécifique, et sans connaissance nécessaire des "métamodèles" sous jacent à l'outillage MDA.⁴

Il est certain que les bénéfices de MDA sont optimisés par le support du modèle UML2, riche, complet et standardisé. L'introduction d'une approche MDA passe par une réflexion de fond sur la démarche de développement: il faut définir quels sont les points de vues supportés pour modéliser une application, comment UML supporte ces points de vue avec des profils spécifiques le cas échéant, et comment ces modèles dédiés sont exploités à des fins de représentation, vérification, et productions automatisées. Un white paper dédié décrira ces aspects nécessaires à prendre en compte pour bénéficier des gains de l'approche MDA. Il déterminera en quoi un outillage dédié comme Objecteering/MDA Modeler facilite et optimise la mise en œuvre d'une démarche MDA basée sur UML2.

³ Model Driven Architecture: Standardisé à l'OMG, l'approche MDA permet de séparer les niveaux de modélisation (dependants ou non des plateformes), et d'automatiser les transformations entre chaque niveau. Pour plus de détails, voir "<http://www.omg.org/mda/>"

⁴ voir "http://www.objecteering.com/products_mda_modeler.php"

IV. Conclusion: pourquoi et Comment passer de UML1.4 à UML2

Beaucoup d'utilisateurs de UML basculeront vers UML2.0 simplement parce que les nouvelles versions des outils UML supporteront UML2.0. Il est évident que ce passage doit être contrôlé, car UML2 introduit de nouveaux concepts de modélisation qu'il faut maîtriser méthodologiquement.

A quelques exceptions près, les diagrammes UML1.4 sont des constructions valides sous UML2. Cette propriété permet ainsi de s'appuyer sur les pratiques actuelles, pour progressivement introduire les apports UML2.

Le basculement doit être guidé par le besoin. Le présent article décrit certains aspects de la modélisation, où UML apporte des avantages importants. Ceux-ci guideront donc les points de passages intéressants vers UML2. Notamment, les aspects de problèmes qui ne pouvaient pas être représentés sous UML1.4 et le peuvent sous UML2, ainsi que les concepts UML2 exploités par les ateliers pour rendre de nouveaux services seront privilégiés lors du basculement.

UML2 n'est pas une révolution. Les méthodologies développées sous UML1.4, utilisant notamment les diagrammes de Use Case, et les diagrammes de classe restent valides. Nous pensons que UML2 permet de compléter certains aspects:

- Les diagrammes de flux sont très utiles pour les phases amont,
- Les classes structurées, parts et port sont des outils puissants pour modéliser les architectures,
- Les machines à état "protocole" (Protocol State Machines) peuvent être utilisées extensivement, comme un complément utile à apporter aux diagrammes conceptuels.

V. 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 « MDA⁵ », 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.

⁵ « Model Driven Architecture » : démarche et technologie sous tendant les nouveaux standards de l'OMG