

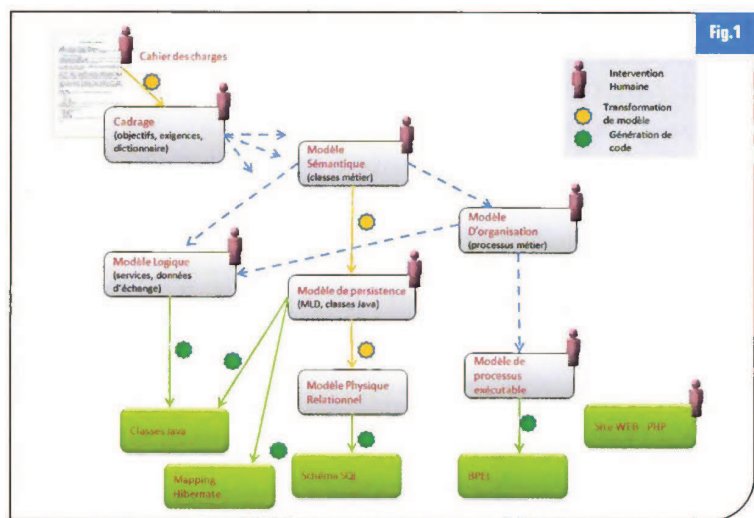


Modélisation de bout en bout d'un projet : de la vision de l'entreprise à l'implémentation dans son SI

Les avancées des techniques de modélisation, MDA (Model Driven Architecture) et les outils de modélisation permettent de supporter l'ensemble de la démarche, depuis la définition des orientations de l'entreprise, en passant par la formalisation du métier, par celle de l'architecture technique, puis par le développement, pour aboutir à la réalisation guidée par le modèle de tout ou partie d'un SI.

Pour assurer une continuité de la démarche sur toute la portée d'une entreprise, faire en sorte que les différents intervenants communiquent et soient en phase, et garantir que le résultat final corresponde aux attentes, comment faut-il procéder ? Il est nécessaire de s'appuyer sur une démarche générale d'entreprise, par exemple la méthode ouverte Praxeme (www.praxeme.org) ou d'autres pratiques s'appuyant sur le cadre méthodologique TOGAF. Il est également recommandé de mettre en œuvre un atelier de modélisation unique couvrant toute la portée : besoin, objectifs, terminologie, règles métier, architecture d'entreprise, logiciel. Dans cet article, nous allons nous intéresser à un cas d'étude illustrant la modélisation de bout en bout, en s'appuyant sur un cahier des charges publié par un organisme indépendant - le Ceisar (www.ceisar.org) dédié à l'évangélisation de l'architecture d'entreprise -. Ce cas d'étude s'appuie sur l'atelier de modélisation Modelio (www.modelio.fr), du fait qu'il supporte toute la portée de modélisation, qu'il intègre la technologie MDA et fournit un ensemble de générateurs permettant d'automatiser le codage et la documentation.

Le cas d'étude porte sur le support de réservations de voyages via internet pour une agence de voyage factice appelée « DiscountVoyages ». Un white paper détaillé, le cahier des charges Ceisar, une base de modélisation complète Modelio, ainsi que l'application résultante peuvent être téléchargés sur (<http://www.modelio-soft.fr/tutorials-fr/model-realise-soa-application-fr.html>).



Différents types de modèles, liaisons, interventions automatiques ou manuelles

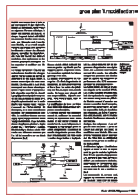
La solution technologique s'appuie sur une architecture SOA, une application Java, l'utilisation de Hibernate et MySQL, la mise en place de services WEB, la mise en œuvre de BPEL, et l'utilisation du serveur d'application open source Glassfish (NetBeans).

La séquence des modèles aboutissant à la solution

La Figure 1 représente schématiquement les différents modèles manipulés, et les types d'intervention et liaisons existant entre ceux-ci. Le Cadrage et le modèle Sémantique permettent de guider totalement le reste de la modélisation. Plus on affine le modèle et on s'approche de la solution, et plus l'automatisation devient conséquente. Ainsi, des parties importantes du modèle logique (données d'échange) et du modèle de persistance sont automatiquement déduites. Le modèle physique relationnel est intégralement généré par

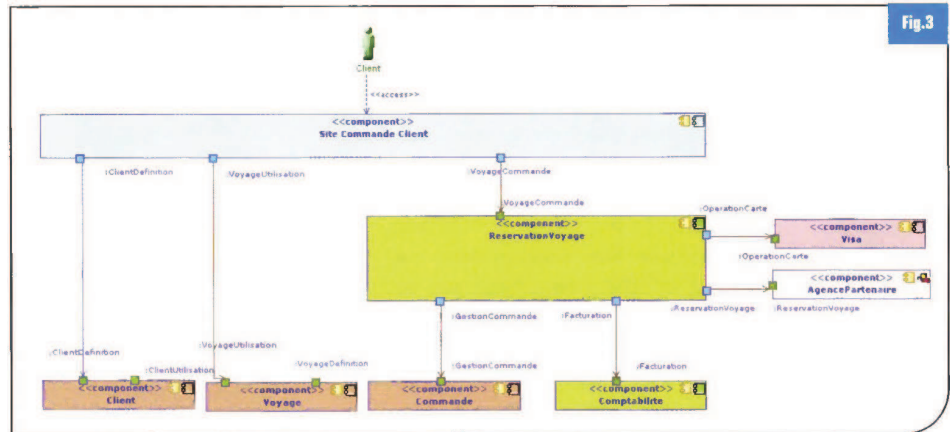
transformation de modèle. De même, les classes Java pour les services, la manipulation des données d'échange, et l'accès à la base de données sont intégralement déduits par génération.

Le cadrage : Les modèles de l'entreprise et du système d'information doivent « coller » au cahier des charges. Une première phase dite de « cadrage » va permettre à partir de ce document initial d'identifier les objectifs de l'entreprise, les termes du métier révélant les notions principales de celui-ci, les règles métier devant être suivies par l'entreprise et son SI, ainsi que les exigences de l'entreprise sur son SI. Le modèle sera ainsi construit sur ces éléments préliminaires de cadrage, en se référant à eux grâce à des liens de « traçabilité » supportés par Modelio. On commence tout simplement par éditer le cahier des charges sous Word, en utilisant des commandes (macros) spécifiques



Modelio pour annoter dans le texte ce qui correspond à des *objectifs*, *termes du dictionnaire*, *règles métier* ou *exigences*. Par cette technique, le cahier des charges est analysé, et chaque fragment textuel intéressant est annoté en fonction de sa nature, puis, ces éléments sont importés sous Modelio, où un travail complémentaire est engagé pour compléter les éléments : ajouter des éléments absents, compléter les descriptions, etc. Modelio fournit des éditeurs tabulaires, des explorateurs et des éditeurs graphiques dédiés pour éditer, organiser et modéliser ces éléments.

Le modèle Sémantique : À partir des notions issues du cahier des charges, que l'on retrouve dans les termes du dictionnaire et des règles métier, le modèle sémantique est créé sous Modelio. On détermine à quel terme correspond une classe sémantique, quelles règles métier s'appliquent à telle classe (ou propriété de classe). Le modèle sémantique (voir www.praxeme.org) est un modèle de classe UML simplifié, qui est focalisé sur la modélisation des notions métier, indépendamment de l'organisation de l'entreprise, et de toute considération technique sur l'implémentation possible. Le modèle ci-dessous exprime que la classe Voyage dérive du terme Voyage (dictionnaire), et que sur cette classe s'appliquent plusieurs règles métier (ex : localisation voyage). Modelio intègre dans un même référentiel ces différents éléments que l'on fait ici apparaître sur un même diagramme. [Fig.2]



Architecture logique – partie liée au site de réservation de voyages

Chaque classe métier possède des états de gestion significatifs que l'on modélise par un diagramme d'état. Les transitions explicitent les liaisons possibles entre états.

Le modèle organisationnel de l'entreprise : On décrit l'organisation de l'entreprise, en représentant les rôles et leurs liens, les unités d'organisation, les flux échangés entre ces éléments. On trouve pour chaque rôle ses responsabilités, objectifs assignés, liens hiérarchiques, liens de communication.

La modélisation de leurs participations dans les processus métier complètera cette description.

Un modèle général des processus permet de positionner les processus métier, en représentant leurs événements et flux métier majeurs en entrée/sortie, et en indiquant leurs initiateurs et intervenants (rôles ou unités d'organisation). L'existence du modèle d'organisation permet d'identifier immédiatement les rôles et les « unités d'organisation » impliqués.

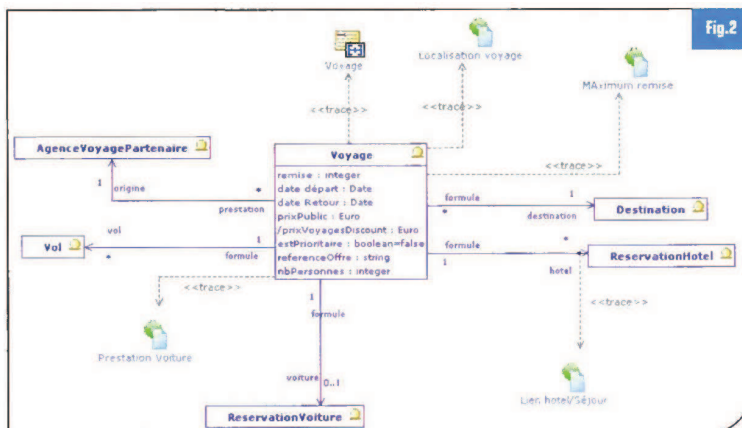
Les flux métier identifiés sur les diagrammes d'organisation sont également des éléments intéressants pouvant être repris. Les objectifs assignés aux processus sont détaillés pour en déduire les indicateurs de performance clés (KPI) correspondant pour les processus retenus.

Le modèle détaillé de chaque processus s'appuie sur le standard BPMN. Les « Lanes » (lignes) BPMN permettent de représenter la responsabilité des tâches. L'intégration UML/BPMN de Modelio permet d'associer ces « Lanes » aux rôles dans l'entreprise, ou aux unités d'organisation, qui sont les responsables habituels des tâches des processus. Les tâches à automatiser sont marquées, afin de préparer l'étude de l'architecture nécessaire à leur support.

Analyse des exigences : Les exigences sont recueillies à partir du cahier des charges du Ceisar, et modélisées sous Modelio. Elles sont ensuite modélisées plus en détail par des Use Case UML, ou directement tracées sur les parties de modèle qui les implémentent.

Architecture logique : L'architecture logique va définir la cible en organisant les grands modules logiciels du futur SI. Une approche s'appuyant sur une démarche SOA nous permet de définir une architecture évolutive, décomposée en composants autonomes et interchangeable, connectés entre eux via des « services ».

[Fig.3] Les composants de services sont répartis en différentes couches (ici, les couleurs : Bleu pour les composants « présentation », vert pour les



Vue détaillée de la classe Voyage



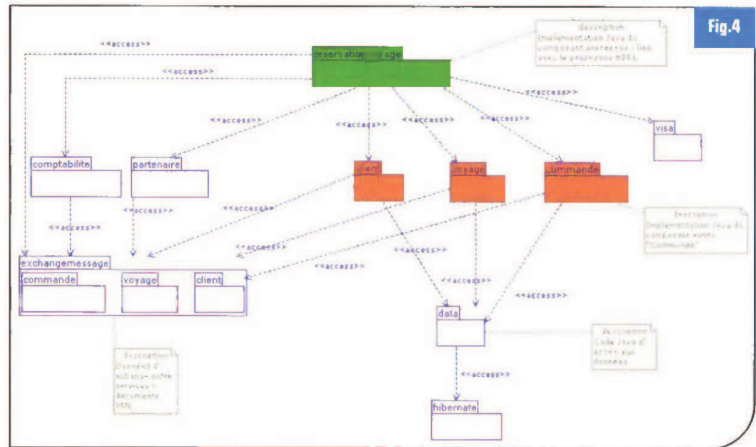
composants « processus » et marron pour les composants « entité »). Ils sont assemblés via les services requis ou les services fournis. Leur déduction est guidée par la démarche : les classes sémantiques saillantes, les plus importantes qui souvent fédèrent des classes satellites, sont traduites par des composants de service « Entité » dont l'objet est de gérer les accès et les modifications des données du référentiel métier. Il s'agit des composants Client, Voyage et Commande, en marron. Les processus métier devant être supportés par le SI seront traduits en composants processus (ici « Réservation Voyage »), et s'appuieront sur les composants entité pour réaliser leur traitement.

L'architecture logique est modélisée de manière détaillée, notamment pour définir précisément les services et les données d'échange. La signature des services est fournie complètement. Les données d'échange sont déduites du modèle sémantique, qui est retravaillé pour fournir les données attendues par les services, mises à plat lors d'échanges typiquement de documents XML.

Production de la base de données : Le modèle logique des données a été obtenu automatiquement sous Modelio en transformant le modèle sémantique. Il est retravaillé manuellement dans un but technique, pour respecter les formes normales, pour définir des identifiants et des clés primaires sur les éléments et pour optimiser la gestion du modèle en base.

Ce modèle est ensuite automatiquement transformé en modèle physique des données faisant apparaître des mécanismes relationnels, comme typiquement des clés étrangères.

Ce modèle peut être retouché pour des raisons physiques, d'optimisations spécifiques d'une base. Modelio maintient la cohérence entre le modèle



Modèle logiciel de l'implémentation Java – Diagramme de Packages UML

de logique et le modèle physique lors de modifications. Puis le code SQL est automatiquement généré. Le choix de la plate-forme Hibernate, permet à Modelio de générer automatiquement les classes Java d'accès aux données.

Architecture logicielle – traduction du modèle logique : L'architecture logicielle reprend le modèle logique, en réalisant un modèle dédié à la cible technique. Ici, le choix des techniques XML, WSDL, BPEL (Glassfish), SQL (MySQL), Hibernate et Java, va conditionner la forme du modèle.

La figure 4 montre la structure de l'implémentation Java. On y retrouve l'implémentation des services pour les composants de service, avec les couches processus et entité.

Les données d'échange sont manipulées par des classes Java et sérialisées en XML. La couche « Data » gère l'accès à la base de données relationnelle.

Le code Java d'accès aux données est intégralement déduit du modèle logique de données. Il produit des classes Java et un « mapping Hibernate » d'accès à la base. Le code Java de manipulation des données d'échange (XML) est intégralement déduit du modèle des données d'échange réalisé avec l'architecture logique. Il permet de sérialiser/désérialiser les données d'échange, et de les manipuler depuis le code Java. En ce qui concerne l'implémentation des services WEB, le squelette du code, avec la signature des opérations de service est déduit du modèle logique des services. Il faut ajouter des com-

pléments de programmation, afin de décrire quels traitements réalisent ces services. Dans ce cas d'étude, il s'agit essentiellement d'accéder aux données en base. Avec Modelio, il est possible d'éditer directement sous l'atelier le code Java, ou d'utiliser un environnement comme Eclipse, le code étant toujours maintenu de manière synchrone avec le modèle.

Le composant « RéservationVoyage » est un composant processus développé avec le langage BPEL. Il a été modélisé en BPMN sous Modelio, pour le générer en BPEL.

La partie IHM WEB a été réalisée manuellement en PHP, et a utilisé les services WEB.

Bilan

Ce cas d'étude complet montre comment modéliser de bout en bout, depuis la définition des objectifs d'une entreprise, en passant par l'architecture d'entreprise, les besoins, l'architecture SOA, le logiciel pour aboutir au code résultant. Le tableau ci-contre résume la volumétrie de code. Ce cas d'étude peut être consulté en détail sur (<http://www.modeliosoft.fr/tutorials-fr/model-realise-soa-application-fr.html>). Des exemples de transformations de modèle et de capacités MDA sont visibles sur la vidéo <http://www.modeliosoft.fr/modules-fr/modelio-mds-designer-fr.html>. Nous remercions le Ceisar pour nous avoir autorisés à publier leur cahier des charges.

■ Philippe Desfray
 Directeur R & D SOFTEAM
 philippe.desfray@softeam.fr

Nature de l'élément	Nombre dans l'application	Nombre généré depuis le modèle	% automatization
Classes Java	40	38	95 %
Lignes de code Java	4035	2987	74 %
Lignes SQL	148	148	100 %
Lignes BPEL	251	251	100 %