Refonte et modélisation des systèmes d'information

Vers les systèmes d'information agiles

Pierre Bonnet et Dominique Vauquier

Pierre Bonnet, directeur du conseil chez Orchestra Networks, a mené d'importants projets de refonte de systèmes d'information, dont celui de la SMABTP. Il est coutumier des architectures de services (SOA) et des solutions de BRMS (moteurs de règles) et de MDM (gestion des données référentielles). Il apporte son expertise technique et son expérience de directeur de projets à l'initiative pour une méthode publique : Praxeme. Notamment, il anime le « Collège des concepteurs et architectes logiques » au sein du Praxeme Institute et est co-auteur, avec Jean-Michel Detavernier (DSI adjoint de la SMABTP) et Dominique Vauquier, de l'ouvrage « Vers le système d'information durable, La refonte des systèmes d'information par SOA ».
Avec Dominique Vauquier, Pierre Bonnet indique, ici, le lien entre la modélisation et les ambitions pour l'agilité des systèmes.

Le contexte

La maintenance des systèmes d'information est coûteuse. Elle génère des régressions et demande du temps. On évite les risques des modifications en profondeur des logiciels, on préfère ajouter des couches supplémentaires qui finalement augmentent la complexité. Les tentatives d'externalisation de la maintenance dans les pays à bas coûts échouent car l'éloignement du métier et de l'informatique ne fait que dégrader, encore plus, la qualité du logiciel.
Dans le même temps, les besoins des entreprises évoluent et il faudra bien que le système d'information se modifie suffisamment vite : alliance entre entreprises, attente des clients pour une plus grande interactivité, réglementations plus complexes et contraignantes, etc. À cela s'ajoute la question sensible du maintien des savoirs sur les systèmes existants. En particulier, ceux qui font tourner le « cœur des organisations », c'est-à-dire la partie du système construite par les informaticiens qui partiront à la retraite dans les cinq prochaines années. Au moment de leur départ, que se passera-t-il ? La jeune génération n'aura pas eu l'occasion de prendre en main la logique de ces anciens systèmes, la maintenance en sera d'autant plus difficile. Au fil des années, la documentation de ces systèmes n'a pas été entretenue de manière exemplaire, donc il ne faudra pas compter sur un auto-apprentissage par les nouvelles équipes. L'héritage informatique est sombre : impossibilité d'agir efficacement en maintenance, pas de documentation, perte à moyen terme des hommes à l'origine de ces systèmes.

La refonte du SI

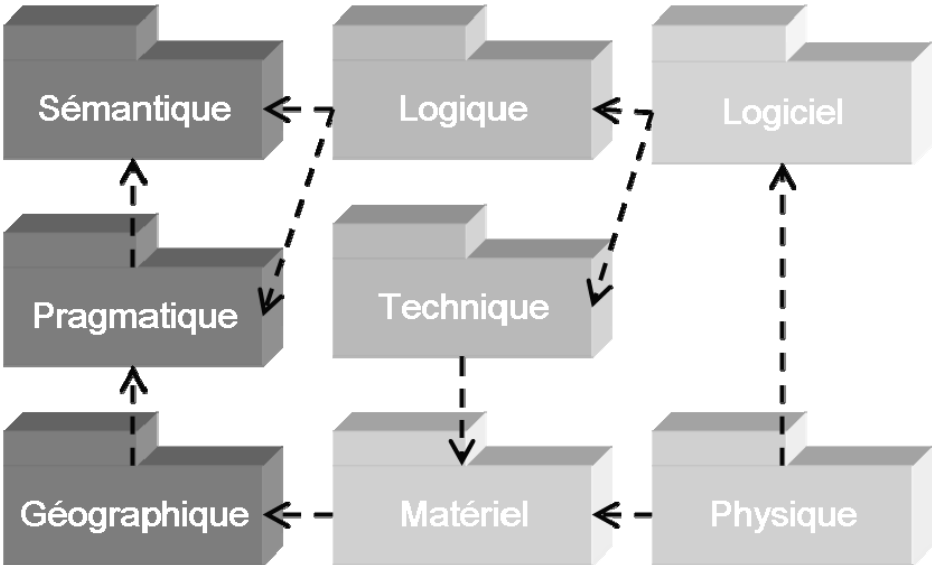
Il est encore temps d'agir, mais le temps presse. L'heure de la refonte progressive des systèmes d'information a sonné. Il faut que les entreprises engagent ces projets afin de tirer profit de la présence des informaticiens à l'origine des systèmes existants. Ils apporteront à la jeune génération leur connaissance fonctionnelle et leur expérience en conduite de projets. Les informaticiens plus jeunes apprendront réellement la conception des systèmes d'information et apporteront leurs connaissances sur l'usage des nouvelles technologies qui s'offrent aujourd'hui à nous. Le système d'information à reconstruire sera d'un nouveau genre. Il devra être capable de s'adapter aux multiples évolutions organisationnelles, « métier » et techniques, ceci sur de longues durées, plusieurs décennies. Ce système d'information sera capable de se recycler plus naturellement que nos anciens systèmes, il sera plus agile et permettra un meilleur alignement du métier avec le logiciel. Ce sera un système d'information plus « durable ».
Mais comment y parvenir ? Comment mettre en ordre de marche l'entreprise pour que ce projet stratégique de la refonte progressive du système d'information réussisse ? Comment en gérer les risques ? Comment ne pas reproduire les erreurs du passé qui nous conduisent à constater, aujourd'hui, la rigidité de nos systèmes, le manque de documentation, l'incapacité à faire plus que les fonctions prévues à l'origine ?

Mettre de l'ordre

Tout d'abord, il faut remettre de l'ordre dans les métiers de l'informatique car ceux-ci se sont brouillés ces dernières années. Le rôle de l'informaticien n'est pas clair : le développeur s'occupe de spécification fonctionnelle, le concepteur logique prend position sur des questions d'architecture fonctionnelle, l'expert Java décide de l'ergonomie du poste de travail, le DBA a la charge de la modélisation métier des données. À cela s'ajoute les difficultés de communication entre les utilisateurs et l'informatique. Pour clarifier cette situation, l'entreprise doit retenir un *cadre de référence* qui va préciser les disciplines clefs nécessaires pour penser l'entreprise et ses transformations. Quand il s'agit de reconstruire le SI, la première urgence est de clarifier les produits

(livrables), les liens entre ces produits, les procédés à mettre en œuvre pour les obtenir, les compétences à mobiliser côté utilisateur et informaticien. La Topologie du Système Entreprise ordonne les représentations selon huit aspects.

La Topologie du Système Entreprise a déjà été présentée dans les Lettres n°61 et n°67, ainsi que dans les actes des Assises d'ADELI en 2002.



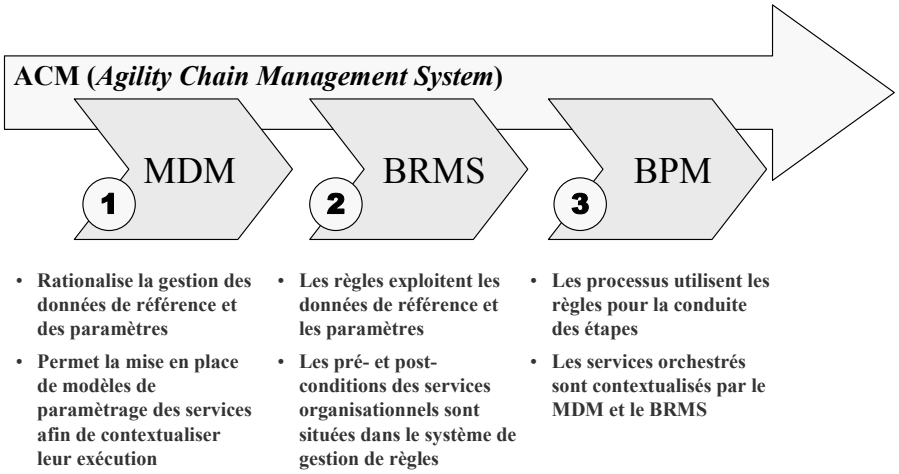
Représentation de la Topologie du Système Entreprise

Ce schéma a été mis au point par Dominique Vauquier et appliqué sur de nombreux projets, depuis une dizaine d'années (on l'appelle aussi le « framework de Vauquier » !). Sa bonne compréhension est essentielle pour la construction des systèmes agiles. Pour une explication détaillée, voir le « Guide général » de Praxeme (référence : PxM-02).

Des dispositifs pour assurer l'agilité

Une fois cette topologie acquise, il faut préciser l'approche qui permettra de construire le système d'information d'un nouveau genre, celui que nous avons qualifié de durable. Il remplacera progressivement les anciens systèmes. La question qui nous intéresse, en priorité ici, est celle de l'agilité du système. En vue d'augmenter cette fameuse agilité, la première chose à faire est de séparer les règles « métier » (ou règles de gestion), d'un côté, et les règles d'organisation, de l'autre. Les premières, imposées au métier par la réalité ou la réglementation, sont plus stables et universelles ; les secondes, au contraire, doivent pouvoir évoluer puisque l'organisation est une variable d'ajustement à la disposition entière des entreprises. Praxeme s'appuie sur la modélisation sémantique pour la première partie, et sur la modélisation pragmatique pour la seconde. Ensuite, on conçoit une architecture logique qui va accueillir, en dur, la logique immuable (les fondamentaux du métier) et expulser dans des « dispositifs d'agilité » les parties plus variables (l'organisation). Les dispositifs d'agilité sont des solutions techniques qui réifient et prennent en charge certaines catégories, par exemple les règles ou les processus.

Mais concrètement à quoi ressemble cette partie variable ? Quand nous voulons appréhender la variabilité dans les entreprises, nous faisons face à trois catégories de représentation ou descripteurs : les processus, les règles et les données. Le point original se situe dans l'idée de prévoir l'agilité sur chacune de ces trois composantes, en même temps : « pas de processus sans règles et pas de règles sans données de référence et paramètres ». La perte d'agilité dans les systèmes vient généralement d'une faille d'agilité sur l'une de ces composantes. Par exemple, l'entreprise met en place un *workflow* intégré à un moteur de règles pour augmenter la flexibilité des processus, mais les règles sont écrites « en dur » car elles manipulent des données de référence et des paramètres qui ne sont pas administrés dans un référentiel central. À chaque modification d'une règle, il faut revoir l'intégration du moteur avec le reste du système d'information, pour aller chercher les nouvelles données de référence et paramètres nécessaires. L'agilité est donc une chaîne dont les maillons sont les trois composantes : a) processus, b) règles, c) données de référence et paramètres. Le maillon le plus faible donnera le niveau de force de cette chaîne.



La chaîne des dispositifs assurant l'agilité du système

Pour réussir le système agile, il faut donc à la fois agir sur les processus avec un BPM (*Business Process Management*), sur les règles avec un BRMS (*Business Rules Management System*) et sur les données de référence et paramètres avec un MDM (*Master Data Management*). L'architecture logique devra prévoir l'accueil de ces « dispositifs d'agilité » et la modélisation des besoins devra bien distinguer le moins variable (métier) du reste (organisation).

L'exigence de modélisation

Une fois ce dispositif en place, on peut agir sur le comportement du système par une configuration des processus, des règles, des données de référence et des paramètres. Cette activité n'appartient pas au processus de conception, c'est un nouveau processus « de gestion du changement » qui s'appliquera tant que la structure fixe du système restera valable face aux évolutions à prendre en compte. Ce processus de changement est beaucoup plus flexible que celui de l'ingénierie qui nécessite de modifier le logiciel. Cependant, au moment où la partie fixe du logiciel doit évoluer, ce qui arrive obligatoirement, il faudra être capable de modifier rapidement le système en mettant à jour les modélisations amont (sémantique, pragmatique), en mettant à niveau l'architecture logique et enfin en modifiant le logiciel qui traite l'aspect fixe du système (le reste est situé dans le BPM, BRMS et MDM). Ici également, il faut que l'agilité soit maximum et surtout que le logiciel qui traite la partie fixe soit aligné avec les besoins, faute de quoi on risquerait de perdre le contrôle sur le système, par exemple en ne sachant plus ce qui doit être dans le logiciel « en dur » et ce qui doit être situé dans les dispositifs d'agilité.

Le standard MDA (*Model Driven Architecture*) permet un meilleur alignement des modèles amont (sémantique, pragmatique) avec les modèles de l'architecture logique puis du logiciel. Il se concrétise par des règles de dérivation qui permettent de passer d'un modèle à un autre :

- règles de dérivation des modèles sémantiques et pragmatiques vers le modèle logique (services, flux et données) ;
- règles de dérivation du modèle logique en fonction de l'architecture technique.

Grâce à cette chaîne d'activité fondée sur MDA, la partie fixe du logiciel peut être remise à jour rapidement et de manière parfaitement alignée sur les besoins. Ensuite, le processus de « gestion du changement », c'est-à-dire la configuration du système par les dispositifs d'agilité, reprend son cours. Au final, le système est documenté par les modèles. Grâce à la synchronisation modèle/code (*round-trip development*), les changements restent maîtrisés ; du moins, nous sommes équipés pour maintenir à niveau la description du système. En associant ces nouveaux moyens de modélisation et les dispositifs d'agilité, nous pouvons mieux faire face aux changements sur les processus, les règles et les données de référence et paramètres. Dès lors, nous sommes en mesure de donner un sens concret à l'agilité et de construire un « système durable », au terme d'une refonte progressive. ▲

pierre.bonnet@orchestranettworks.com
dominique.vauquier@praxeme.org

Sigles

SOA : Service Oriented Architecture – Architecture orientée service
BRMS : Busines Rule Management System - Système de Gestion des Règles Métiers
MDM : Master Data Management – Gestion des données de référence