Vers un cadre unificateur pour l'enseignement des outils et méthodes de gestion de l'information numérique

Yves MARCOUX

GRDS - EBSI - Université de Montréal
C.P. 6128, succ. Centre-ville
Montréal (Québec)
CANADA H3C 3J7

yves.marcoux@umontreal.ca

Problématique et objectif

Trois types d'outils très importants pour la gestion de l'information numérique dans la société—et donc, pour les professionnels de l'information—sont les bases de données textuelles, les bases de données relationnelles et les documents structurés (XML, SGML, XHTML). L'éventail d'applications couvertes par ces trois types d'outils est extrêmement vaste, allant de la gestion de données factuelles à la gestion de documents en texte intégral ou multimédia, à des interfaces d'accès à l'information entièrement dynamiques, collaboratives et personnalisables, en passant par la gestion de métadonnées et de données bibliographiques et catalographiques. Plus de dix ans d'expérience dans l'enseignement de ces outils amènent l'auteur à un triple constat. D'une part:

- Dans leurs caractéristiques générales, les trois types d'outils ont de fortes ressemblances entre eux.
- En conséquence, les démarches méthodologiques professionnelles encadrant l'utilisation correcte et judicieuse de ces outils pour solutionner divers problèmes documentaires devraient avoir des caractéristiques communes fortes.

D'autre part, pourtant:

- Les approches « classiques » à l'enseignement de ces trois types d'outils ne mettent pas en évidence les ressemblances, mais plutôt leur spécificités.

Il s'ensuit que chaque type d'outils est accompagné d'une démarche méthodologique « idiomatique », peu ou pas comparable aux démarches associées aux autres types d'outils. Ceci entraîne que la compétence fondamentale reliée à l'ensemble de ces types d'outils, à savoir la capacité de choisir l'outil le plus approprié aux besoins spécifiques à combler dans une situation donnée, est reléguée aux domaines—flous—de l'expérience ou de l'intuition et à peu près laissée-pour-compte dans les démarches pédagogiques. Comme l'apprenant ne possède pas encore intuition ou expérience, il développe ce que nous appelons un attitude d'« ambivalence » à l'égard des types d'outils.

Une position différente que l'on rencontre aussi à l'occasion est l'« universalisation » d'un outil, c'est-à-dire la proclamation sans autre forme de procès qu'un des outils (par exemple les bases de données relationnelles) est fondamentalement supérieur aux autres et que l'on devrait toujours viser à travailler de préférence avec celui-là. L'apprenant développera attitude de « biais » en faveur d'un type d'outils, attitude qui lui évitera la difficulté inhérente au choix du type d'outils le plus approprié à une situation donnée, mais ne l'amènera pas forcément à de meilleurs choix que l'ambivalence; en fait, elle remplace le choix judicieux par un choix arbitraire *a priori*.

Ces attitudes que peut développer l'apprenant à l'égard des types d'outils ne sont pas les seules conséquences néfastes des approches « classiques »: l'apprentissage même des outils est ralenti par le fait que l'apprenant ne peut pas réutiliser les concepts assimilés en les transposant d'un type d'outils à l'autre. L'apprentissage des méthodologies professionnelles encadrant l'utilisation de chaque type d'outils s'en trouve lui aussi alourdi.

Notre objectif est de développer un modèle unique et général d'outils documentaires dont chacun des types d'outils mentionnés ci-dessus est un cas particulier, accompagné d'une démarche méthodologique du même niveau de généralité par rapport aux méthodologies classiques accompagnant les trois types d'outils. Le but du présent article est de franchir quelques pas vers l'établissement d'un tel modèle et d'une telle méthodologie.

L'approche est élaborée avec en tête les trois types d'outils mentionnés ci-dessus (bases de données textuelles, bases de données relationnelles et documents structurés), mais elle pourrait être *a priori* généralisable à d'autres types d'outils.

Systèmes d'information

La base de notre approche est de considérer les outils des types visés comme des composants d'éventuels *systèmes d'information*. Conséquemment, nous voyons les méthodologies professionnelles gravitant autour des outils comme des méthodologies de *développement de systèmes d'information*. Alors que chaque type d'outils vient traditionellement avec sa propre méthodologie, nous cherchons en quelque sorte une méthodologie générale où chaque type d'outils trouve sa place relative et qui chapeaute l'ensemble des méthodologies individuelles. Évidemment, nous ne voulons pas procéder par simple « addition » des méthodologies existantes, mais en fournissant un cadre général dont chaque méthodologie existante est un cas particulier.

La toile de fond de notre discussion est donc le développement de systèmes d'information, tâche structurée globalement comme suit:

1. Identification de la clientèle-cible et du contexte.
2. Identification des besoins à combler.
3. Conception d'un système pour combler les besoins.
4. Construction du système conçu.
5. Mise en service du système construit.
6. Opération du système mis en service.
7. Évaluation périodique du système.

Nous appelerons parfois *analyse* l'application de cette démarche dans un contexte donné, et *analyste* le professionnel qui l'effectue.

Nature d'un type d'outils

Pour fins de concision et de simplicité, nous présenterons ici notre discussion dans le contexte de deux types d'outils seulement: les bases de données relationnelles (que nous appellerons BDR) et les documents XML structurés par DTD (que nous appelerons simplement XML). Nous supposons le lecteur familier avec ces deux types d'outils.

Par *objet numérique*, nous entendons simplement une suite de bits (chiffres binaires) stockés sous forme d'un fichier dans un *référentiel institutionnel*. Un objet numérique pourrait aussi être un dossier informatique, c'est-à-dire d'un ensemble de fichiers/sous-dossiers adressables par nom à l'intérieur du dossier, mais par simplicité, nous supposons qu'il s'agit d'un fichier unique. Conséquemment, pour les bases de données relationnelles, nous supposerons qu'une base de données est stockée physiquement comme un seul fichier (ce qui est le cas notamment avec Microsoft Access).

Par simplicité, nous supposons que le référentiel institutionnel est un système de fichiers. En fait, tout référentiel capable de stocker/fournir des objets numériques à partir d'un nom ferait l'affaire. Concrètement, il pourrait s'agir de l'intranet institutionnel. L'*institution* sous-jacente au *référentiel institutionnel* est la *cliente* de la démarche de développement du système d'information. Il peut s'agir d'une collectivité, d'une organisation ou même d'une personne quelconque.

Un type d'outils est constitué des éléments suivants:

1. Un langage de définition de contraintes sur des objets numériques (ou simplement *langage de contraintes*). Cet élément s'appelle dans le monde des bases de données le *langage de définition de données* ou DDL. Une spécification dans ce langage s'appelle un *schéma d'objets numériques*.
2. Un éditeur d'objets numériques (ou *éditeur d'objets*). Cet éditeur accepte en intrant un schéma d'objets numériques et permet la création d'un nouvel objet numérique respectant les contraintes exprimées dans le schéma, ou la modification d'un tel objet existant déjà. L'éditeur a la capacité de ranger (enregistrer) l'objet numérique à un certain endroit (i.e., sous un certain *nom*) dans le référentiel institutionnel.
3. Un langage de manipulation d'objets numériques. Dans le monde des bases de données, cet élément s'appelle le *langage de manipulation de données* ou DML. Une spécification dans ce langage s'appelle une *requête*. En BDR, il s'agit de la partie DML de SQL; en XML, il s'agit de XSLT ou XQuery.
4. Un moteur d'exécution de requêtes. Ce moteur accepte en intrant un objet numérique et une requête et modifie l'objet/produit un résultat conformément à la requête.
5. Un langage de développement d'applications. Il s'agira en général d'un langage de programmation complet imbriquant d'une façon quelconque le langage de manipulation d'objets numériques. Une fois développée et mise en service, une application possède un identifiant dans le référentiel institutionnel et peut être lancée (sous réserve des politiques d'accès) en « activant » cet identifiant (par exemple, dans un navigateur Web). Nous supposons les applications « non paramétrables », c'est-à-dire qu'une application donnée travaille implicitement sur un certain objet numérique.

Dans la démarche de développement d'un système d'information, c'est l'étape de conception du système qui va en fixer la nature et la forme exacte, mais qui doit également établir les liens entre les objets gérés par le système et le reste du monde. C'est à cette étape que les schémas pour les différents objets gérés par le système seront créés (modélisation des données) et documentés. La logique de localisation des différents objets et du système lui-même (son adresse dans le référentiel institutionnel) doit également être planifiée et documentée, même si elle n'est concrétisée qu'à la mise en service.

La cohérence entre la logique du système développé et l'utilisation qu'en feront les utilisateurs est directement tributaire de leur compréhension des liens entre les objets gérés par le sytème et l'institution (et éventuellement le reste du monde). Pour que ces liens soient compris, ils doivent être présentés aux utilisateurs d'une manière convenable. En principe, cela pourrait se faire par des communications directes avec les développeurs du système, mais il est beaucoup plus pratique et fiable que ce soit fait par *documentation*. Le fait qu'une documentation existe n'empêche pas de procéder à des formations, à des tutoriels, et assure qu'on peut en tout temps retourner à la source au besoin.

À notre avis, c'est dans la façon de concevoir et de déployer la documentation des différents aspects d'un système d'information que se caractérise une méthodologie professionnelle de développement. C'est grâce à la qualité de cette documentation et à sa *disponibilité dans le système lui-même* que l'utilisateur comprendra la part de travail qui lui incombe et pourra « collaborer » efficacement avec le système.

Nous avons évoqué ci-dessus la logique de localisation (entre autres, de nommage) du système et des différents objets numériques qu'il manipule. Il est clair que cette logique est externe au système. Elle n'en qualifie pas moins le lien entre le système et le reste du monde (par exemple, on ne s'attend pas à ce qu'un catalogue de bibliothèque situé dans la zone « Test » de l'intranet soit représentatif de la collection réelle de l'institution). Cette logique doit faire l'objet d'une documentation propre, qui constitue en quelque sorte le contexte d'interprétation du système et de sa documentation.

Reconnaître que le travail du professionnel comprend une part non seulement importante, mais essentielle, de déploiement de documentation, met en perspective des aspects souvent ignorés ou négligés des types d'outils disponibles et amène à la fois des critères de sélection nouveaux et un cadre unificateur à la démarche de développement.

Nous allons illustrer ce phénomène en regardant plus en détail les langages de contraintes.

Langages de contraintes

En XML (avec DTD), le langage de définition de contraintes est le formalisme des DTD. Dans ce contexte, un schéma d'objets numériques est simplement une DTD. En BDR, ce langage est la partie de SQL qui concerne la définition des données (énoncés CREATE). Un jeu de contraintes exprimées dans ce langage permettra de déterminer exactement si telle ou telle chaîne de bits constitue un objet numérique admissible ou non.

Notons que les schémas n'existent pas nécessairement de façon indépendante des objets numériques. Par exemple, en BDR (avec Access), les définitions des tables d'une BD sont stockées dans la BD elle-même. Dans tous les cas, au minimum, un élément de la logique de localisation du référentiel institutionnel précisera quel schéma est applicable à quel objet.

Le langage de contraintes doit bien sûr être interprétable par machine, puisqu'une des première étapes que doivent effectuer l'éditeur d'objets et le moteur de requêtes lorsqu'on leur demande de travailler sur un objet existant est de vérifier que l'objet est conforme aux contraintes. Mais il doit aussi être interprétable ou compréhensible par l'humain, pour que les utilisateurs du système développé l'utilisent à bon escient. Les contraintes expriment une logique interne que doit posséder un objet numérique, mais cette logique doit pouvoir être mise en lien avec la logique de l'institution. C'est pourquoi chaque contrainte doit être justifiée et expliquée en des termes significatifs pour l'institution. De plus, ces justifications et explications devraient être accessibles directement à partir du système en opération.

Ici, une tendance populaire est d'introduire des langages intermédiaires qui permettent de définir une logique institutionnelle (langages du genre « business rules specification »). Même avec un tel langage, nous en sommes toujours à une logique interne, qui doit être rattachée au réel de l'institution pour avoir un sens pour les utilisateurs. Que l'on ait recours à des langages intermédiaires ou non, nous prétendons que le travail n'est pas complété tant qu'on n'en arrive pas à des justifications/explications s'appuyant ultimement sur rien d'autre que des *compétences communicationnelles générales*, par exemple un certain niveau de compétence dans la maîtrise d'une langue naturelle. Nous basons cette position sur l'observation suivante: à moins de problèmes majeurs qui devraient de toutes façons être corrigés, un individu nouveau-venu dans l'institution, qui ne connaît rien de plus sur l'institution que sa « logique de base » (mission, etc.), réussit pourtant à devenir compétent et fonctionnel avec les systèmes d'information institutionnels. C'est donc dire que, forcément, le nouveau-venu est exposé à une succession de communications qui l'amènent de son état pré-embauche à un état de compétence adéquate. Or, dans son état pré-embauche, l'individu ne peut baser sa compréhension des communications que sur ses compétences communicationnelles générales, par exemple, sa maîtrise d'une langue naturelle. Les communications qui lui confèrent la compétence voulue ne peuvent donc pas s'appuyer sur autre chose que les compétences communicationnelles générales; ces communications sont donc forcément disponibles sous une forme ne s'appuyant que sur les compétences communicationnelles générales.

Cette forme peut consister en formations, en tutorat, en mentorat ou en *ressources d'information*. Notre position est que la disponibilité sour forme de ressources d'information est un minimum qui doit être assuré. C'est peut-être la forme la plus flexible, pouvant être consultée de façon autonome, en tout temps et sur demande. Nous suggérons en fait que les contraintes exprimées dans un schéma d'objets numériques devraient être interprétables dans une *langue naturelle* (éventuellement enrichie de sons et d'images) qui ne suppose de connue et comprise que la logique institutionnelle de base.

Cela peut se faire de différentes façons:

- Les justifications et explications sont rédigées directement sous la forme voulue.
- Elles sont rédigées en couches successives, chacune établissant des définitions de plus en plus spécialisées ne référant qu'aux couches inférieures, et la plus basse ne référant qu'à une compétence linguistique de base et à la logique institutionnelle de base.

Du point de vue efficacité communicationnelle et ergonomie intellectuelle, la première solution est probablement préférable. Mais observons que la seconde est un minimum en ce sens que, même si cela peut demander beaucoup de travail, elle permet d'arriver au même résultat que la première. Observons aussi que la seconde solution est presque réalisée dans bien des systèmes. Par exemple, l'explication d'une contrainte exprimée par un modèle de contenu dans une DTD XML peut être directement donnée en langue naturelle; par contre, elle peut aussi être présentée telle quelle (le modèle de contenu), accompagnée d'une *référence* explicite à un manuel d'introduction aux DTD XML. Dans bien des systèmes, il ne manque que les références pour que la seconde solution soit réalisée.

Quelle que soit la forme des justifications et explications, un élément important est aussi qu'elles soient accessibles sur demande directement à partir du système en opération. Ceci est de moins en moins problématique avec l'essor du Web en tant qu'interface universelle. Néanmoins, ce sont deux points auxquels il faut porter attention: que les références aux documents explicatifs soient données explicitement (éventuellement, en cascade, si une rédaction en plusieurs couches est utilisée) et que ces références soient « activables » (déréférençables) en direct à partir du système en opération.

Nous considérons que cette possibilité d'obtenir en forme adéquate (directement ou en plusieurs couches) toutes les informations nécessaires à l'interprétation des interactions qu'un utilisateur a avec un système d'information est un minimum et que le professionnel impliqué dans le développement du système doit s'en assurer.

Un exemple

Voici une illustration de comment une définition de contrainte peut être convertie en énoncé de langue naturelle. Supposons la table relationnelle suivante:

Table relationnelle avec sa définition

Contacts			
Numéro	**Prénom**	**Nom**	**Téléphone**
`integer`	`text(50)`	`text(50)`	`text(11)`
1	Jeanne	Tremblay	555-1212
2	Roger	Drapeau	555-1313
3	Bernard	Larue	555-1214

On peut considérer que le contenu d'une fiche de cette table est en fait une forme abrégée d'une phrase complète, du genre « Un des contacts porte le numéro *m*; son nom est *xyz* et son prénom *zyx*; son numéro de téléphone est *nnn*. » On a ici remplacé la structuration tabulaire et les entêtes par une formulation linguistique des mêmes informations. On se trouve par le fait même à préciser la correspondance (évidente ici) entre les noms formels de champs (les entêtes) et des concepts d'une « logique institutionnelle » de base très simple. Le fait de penser en termes d'énoncés langagiers peut mener à des décisions de nomenclature qui facilitent ensuite l'interprétation des données par les utilisateurs sous forme d'énoncés ne prenant pour acquis qu'une logique institutionnelle de base.

La phrase avec des mots manquants donnée ci-dessus peut donc constituer une « explication » satisfaisante du schéma de la table (non pas une justification des contraintes spécifiques comme les longueurs de champs, mais du moins une explication) pour fins de formulation d'énoncés en lien avec la « logique institutionnelle ». Notons que cette phrase avec mots manquants a une forte ressemblance avec un gabarit de formatage que l'on pourrait associer à la table (par exemple, par le truchement d'un formulaire en BDR, ou d'un gabarit XSLT en XML).

Conclusion

Nous avons présenté une amorce de réflexion sur un cadre unificateur pour l'enseignement des outils et méthodes de gestion de

l'information numérique. Ce cadre est basé sur la nécessité pour les utilisateurs de pouvoir donner un sens aux interactions qu'ils entretiennent avec un système d'information. Ce dernier doit être conçu de façon à permettre aux utilisateurs de formuler des *énoncés* qui ont un sens directement dans la logique institutionnelle de base (supposée connue comprise des utilisateurs), ou qui s'appuient sur des définitions/explications *accessibles directement dans le système* et formulées en des termes directement rattachables à la logique institutionnelle. Il est suggéré que ces énoncés puissent être formulés dans une *langue naturelle* (éventuellement enrichie de sons et d'images).

La réflexion n'en est manifestement qu'à ses débuts et nous comptons la pousser plus avant dans un futur rapproché.
