



Annexe interopérabilité du SDET

Date : 01 décembre 2006

Réf : interoperabiliteV1.01

Nombre de pages : 48

Références du document

Date	Version	Description des modifications
15/12/2003	0.1	Initialisation, plan du document
12/01/2004	0.2	Intégration des remarques réunion du 15/12
19/01/2004	0.3	Intégration des éléments livrés par les auteurs et rédaction
23/01/2004	0.4	Intégration de nouvelles fiches
26/01/2004	0.5	Restructuration du plan en séance
10/02/2004	0.6	Ajout et arrangement de sous parties
26/02/2004	0.7	Travail en réunion plénière
03/03/2004	0.8	Réorganisation et ajouts de nouvelles parties conformément au plan modifié
12/03/2004	0.9	Finalisation du document
18/03/2004	0.99	Intégration fiche de relecture
08/04/2004	1.0	Relecture et corrections mineures
01/12/2006	1.01	Modification de la forme pour mise en conformité avec le référentiel et mise à jour des liens internet

TABLE DES MATIERES

<u>1</u>	<u>AVANT-PROPOS.....</u>	<u>6</u>
1.1	CONTEXTE ET PORTEE DU DOCUMENT.....	6
1.2	NIVEAUX DE PRECONISATIONS	6
<u>2</u>	<u>URBANISME ET INTEROPERABILITE</u>	<u>8</u>
2.1	URBANISATION DU SYSTEME D'INFORMATION.....	8
2.1.1	OUVERTURE DES SYSTEMES D'INFORMATION ET DEMATERIALISATION.....	8
2.1.2	QU'EST CE QUE L'URBANISATION D'UN SYSTEME D'INFORMATION ?	9
2.1.3	URBANISME ET INTEROPERABILITE.....	11
2.1.4	INTEROPERABILITE ET <i>MIDDLEWARE</i>	12
2.2	ARCHITECTURE D'INTEROPERABILITE	14
2.2.1	FORMATS D'ECHANGE.....	14
2.2.2	SOLUTIONS D'INTEGRATION D'APPLICATIONS (EAI).....	15
2.2.2.1	Le problème de l'intégration.....	16
2.2.2.2	Les moteurs d'intégration	17
2.2.3	ARCHITECTURES DE SERVICE (SOA)	19
2.2.3.1	La notion de service logiciel.....	19
2.2.3.2	Une architecture de services	19
2.2.3.3	Web Service Oriented Architecture (WSOA)	20
<u>3</u>	<u>METHODOLOGIE DE L'INTEROPERABILITE</u>	<u>21</u>
3.1	CONDUITE DE PROJET ET INTEROPERABILITE	21
3.1.1	CYCLE PROJET ET RISQUES ASSOCIES	21
3.1.2	QUELQUES QUESTIONS IMPORTANTES	22
3.2	PRECONISATIONS ET EXIGENCES.....	23
<u>4</u>	<u>UML, MODELES D'OBJETS METIER ET FORMATS D'ECHANGE.....</u>	<u>25</u>

4.1	UML : UNIFIED MODELING LANGUAGE.....	25
4.1.1	APPUI SUR UN STANDARD OU UNE NORME.....	26
4.1.2	CONDITIONS D'EMPLOI	26
4.1.3	OUTILS DISPONIBLES.....	26
4.1.4	COMPETENCES REQUISES.....	27
4.2	XML : EXTENDED MARKUP LANGUAGE.....	27
4.2.1	PRINCIPALES COMPOSANTES TECHNIQUES DE XML.....	27
4.2.2	APPUI SUR UN STANDARD OU UNE NORME.....	28
4.2.3	CONDITIONS D'EMPLOI	28
4.2.3.1	Gestion de contenu.....	28
4.2.3.2	Persistance des documents XML.....	29
4.2.3.3	Personnalisation.....	30
4.2.3.4	Architecture XML XSL.....	30
4.2.4	COMPETENCES REQUISES.....	31
5	<u>LES OUTILS D'INTEROPERABILITE</u>	32
5.1	SERVICES WEB.....	32
5.1.1	APPUI SUR UN STANDARD OU UNE NORME.....	33
5.1.2	CONDITIONS D'EMPLOI	33
5.1.3	IMPLEMENTATIONS DISPONIBLES.....	34
5.1.3.1	WSRP: Web Service for Remote Portals.....	34
5.1.3.2	WSOA : Web Service Oriented Architecture.....	35
5.1.4	METHODOLOGIE SOMMAIRE DE MISE EN ŒUVRE.....	37
5.1.4.1	l'approche « bottom-up » :.....	37
5.1.4.2	l'approche « top-down » :.....	37
5.1.5	COMPETENCES REQUISES.....	38
5.2	AUTRES OUTILS.....	38
5.2.1	LDAP : <i>LIGHTWEIGHT DIRECTORY ACCESS PROTOCOL</i>	38
5.2.1.1	Appui sur un standard ou une norme.....	39
5.2.1.2	Implémentations disponibles	39
5.2.1.3	Recommandations.....	40
5.2.2	WEBDAV.....	40
5.2.2.1	Appui sur un standard ou une norme.....	41

5.2.2.2	Conditions d'emploi	41
5.2.2.3	Implémentations disponibles	41
5.2.2.4	Compétences requises.....	41
5.2.3	<i>SAML : SECURITY ASSERTION MARKUP LANGUAGE</i>	42
5.2.3.1	Appui sur un standard ou une norme	42
5.2.3.2	Conditions d'emploi	42
5.2.3.3	Implémentations disponibles	42
5.3	LES MIDDLEWARES	43
5.3.1	<i>MIDDLEWARES BASES DE DONNEES</i>	43
5.3.1.1	Appui sur un standard ou une norme	44
5.3.1.2	Conditions d'emploi	44
5.3.1.3	Implémentations disponibles	44
5.3.1.4	Compétences requises.....	44
5.3.2	<i>MIDDLEWARE ORIENTE MESSAGE</i>	45
5.3.2.1	Conditions d'emploi	46
5.3.2.2	Implémentations disponibles	46
5.3.2.3	Compétences requises.....	46
6	<u>GLOSSAIRE</u>	47

1 Avant-propos

1.1 Contexte et portée du document

La publication de la version 1 du Schéma directeur des espaces numériques de travail en janvier 2004 a marqué une étape importante de la réflexion sur le déploiement des services en lignes pour les établissements et écoles. Ce schéma directeur fournit des recommandations sur les plans fonctionnels, organisationnels, et technologiques, pour la mise en œuvre des espaces numériques de travail.

Toutefois, cette version 1 ne répondait pas à l'ensemble des préoccupations soulevées à l'occasion de l'appel à commentaire sur l'esquisse du schéma directeur mené entre mars et juin 2003. De nombreux acteurs soulignaient notamment la nécessité à la fois de présenter une vision plus large des problématiques d'interopérabilité et de préciser les préconisations correspondantes.

Après les travaux menés sur les annuaires dans l'enseignement supérieur, et sur les fonctions d'authentification, d'autorisation, et de Single Sign-On, le présent document a pour ambition de compléter la réponse à cette attente, dans les limites de l'état de l'art actuel en matière de standardisation. Il doit donc être considéré comme une annexe du Schéma directeur des espaces numériques de travail.

Sa portée se limite à la communication inter-applicative entre le socle d'un espace numérique de travail et les services logiciels qui viennent s'y connecter. Ces notions sont redéfinies dans le document. Après avoir rattaché la problématique technique de l'interopérabilité à **l'enjeu stratégique de modernisation des organisations et de leurs systèmes d'information**, la présente annexe identifie les principes méthodologiques et technologiques à appliquer et les compétences à mobiliser pour améliorer le niveau d'interopérabilité des espaces numériques de travail.

Le présent document est le fruit d'un groupe de travail composé d'experts techniques de la communauté de l'enseignement scolaire et de l'enseignement supérieur, impliqués dans les projets d'espaces numériques de travail en cours de développement. Ce groupe de travail a été piloté par la SDTICE, qui est en charge de l'élaboration du Schéma directeur des espaces numériques de travail.

La SDTICE a fait appel à la société Cosmosbay~vectis pour renforcer l'expertise technique du groupe et assurer l'assistance à maîtrise d'ouvrage. Cosmosbay~vectis a en outre eu en charge la rédaction de ce document.

1.2 Niveaux de préconisations

Ce document décrira un certain nombre de préconisations et de conseils (bonnes pratiques). Afin de déterminer le niveau d'obligation de respect de ces préconisations, nous utiliserons la terminologie définie dans le RFC 2119 avec les traductions suivantes :

- ✓ *MUST, SHALL* : DOIT
- ✓ *MUST NOT, SHALL NOT* : NE DOIT PAS

- ✓ *REQUIRED* : EXIGE
- ✓ *SHOULD* : DEVRAIT
- ✓ *SHOULD NOT* : NE DEVRAIT PAS
- ✓ *RECOMMENDED* : RECOMMANDE
- ✓ *MAY* : PEUT
- ✓ *OPTIONAL* : FACULTATIF

Voici une traduction de la définition de ces termes dans le RFC 2119 :

1. **DOIT** : ce mot, ou le terme "EXIGÉ", signifie que la définition est une exigence absolue de la spécification.
2. **NE DOIT PAS** : cette expression signifie que la définition est une prohibition absolue de la spécification.
3. **DEVRAIT** : ce mot, ou l'adjectif "RECOMMANDÉ", signifie qu'il peut exister des raisons valables, dans des circonstances particulières, pour ignorer cet item particulier, mais les conséquences doivent être comprises et pesées soigneusement avant de choisir une voie différente.
4. **NE DEVRAIT PAS** : cette expression, ou l'expression "NON RECOMMANDÉ", signifient que la définition est prohibée. Il peut toutefois exister des raisons valables, dans des circonstances particulières, quand le comportement particulier est acceptable ou même utile, de ne pas suivre cette recommandation. Mais les conséquences doivent être comprises et le cas soigneusement pesé.
5. **PEUT** : ce mot, ou l'adjectif "FACULTATIF", signifie qu'un item est vraiment facultatif. Un vendeur peut inclure l'item parce qu'un marché particulier l'exige ou parce qu'il estime qu'il améliore le produit tandis qu'un autre vendeur peut omettre le même item.

2 Urbanisme et interopérabilité

Au-delà de l'aspect technique, l'interopérabilité répond à un enjeu de modernisation des organisations, notamment par l'urbanisation des systèmes d'information. Ce qu'il convient d'affirmer, c'est que tout projet d'ouverture ou de dématérialisation doit se construire au premier chef sur une vision globale du système d'information existant, et aussi sur une vision globale de la cible à atteindre.

2.1 Urbanisation du système d'information

2.1.1 Ouverture des systèmes d'information et dématérialisation

Tant dans la sphère publique que dans le secteur concurrentiel, la tendance aujourd'hui est à l'ouverture des systèmes d'information et à la dématérialisation des échanges.

Les décisions d'orientation fréquemment observées dans le secteur concurrentiel concernent :

- ✓ la mise en place de coopérations et la volonté de rendre plus fluides les échanges nécessaires avec les partenaires et fournisseurs ;
- ✓ la mise en place de nouvelles formes de commerce sur un nouveau canal : Internet ;
- ✓ l'intégration d'activités nouvelles pour optimiser les ressources disponibles, et la diversification pour atténuer les aléas sectoriels ;
- ✓ ou au contraire l'externalisation d'activités existantes pour se concentrer sur les activités à forte valeur ajoutée, et se recentrer sur le cœur de métier pour optimiser la performance.

Dans tous les cas, il y a nécessité de transformer les modes de travail, de transformer les échanges et de les accroître, et de favoriser cela en ouvrant le système d'information (mise à disposition de services aux partenaires, aux clients), et en dématérialisant les échanges.

Les évolutions caractérisant durablement les administrations aujourd'hui sont les suivantes, qu'il s'agisse des services centraux de l'État ou des services déconcentrés, ou encore des collectivités territoriales ou des établissements publics ou parapublics en général :

- ✓ la mise en place de services aux entreprises et aux citoyens ;
- ✓ le développement de réseaux mettant en œuvre des coopérations nouvelles, ou des éléments de mutualisation ;
- ✓ la dématérialisation des échanges (une « administration zéro papier »).

Ce qui implique l'ouverture des systèmes d'information publics, une logique de dématérialisation des échanges (voir à ce propos les documents accessibles sur le site de l'Agence pour le développement de l'administration électronique (ADAE) notamment le « guide des maîtres d'ouvrage des services en ligne »).

Les thèmes de l'ouverture des systèmes d'information et de la dématérialisation sont donc centraux :

- ✓ une ouverture des systèmes d'information :
 - qui nécessite la mise en place de solutions de coopération avec d'autres systèmes dont

- on n'a nullement la maîtrise : une interopérabilité ;
- qui nécessite avant tout une connaissance certaine et une maîtrise de son système d'information : l'urbanisation.
- ✓ la dématérialisation : la mise en œuvre de formats d'échange partagés.

2.1.2 Qu'est ce que l'urbanisation d'un système d'information ?

La modernisation des systèmes d'information des organisations, dont on a vu brièvement plus haut la nécessité, s'incarne dans une vision et une méthode de plus en plus répandue, celle de l'urbanisation des systèmes d'informations. Cette modernisation touche tout autant les systèmes d'information des administrations et des établissements publics que les entreprises.

Le CIGREF (Club informatique des grandes entreprises françaises) propose quelques définitions dans son livre blanc de septembre 2003 :

« Les spécialistes des systèmes d'information ont choisi de dénommer "urbanisation" la démarche qui consiste à rendre un système d'information plus apte à servir la stratégie de l'entreprise et à anticiper les changements dans l'environnement de l'entreprise. »

"L'urbanisation du système d'information" désigne plus précisément la mise en œuvre du plan d'urbanisme du système d'information.

"L'urbanisme du système d'information" désigne la démarche qui consiste à définir un système d'information cible qui puisse s'adapter et anticiper les différents changements (stratégiques, organisationnels, juridiques...) touchant l'entreprise.

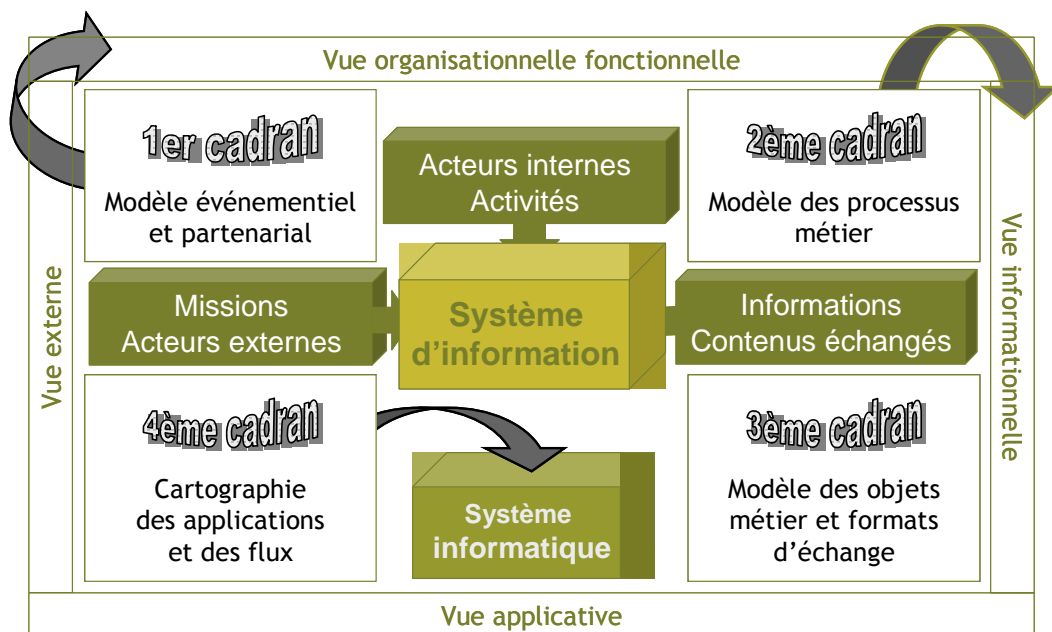
Le "plan d'urbanisme du système d'information" désigne l'agrégation de la définition du système d'information cible et des règles d'urbanisme avec la trajectoire à suivre pour atteindre ce système d'information cible. »

De manière lapidaire, une organisation (administration, collectivité, entreprise, ...) :

- ✓ est construite pour satisfaire un objectif (de service public, de vente, de production, ...) ;
- ✓ elle définit ainsi un intérieur (les agents de l'organisation) et un extérieur (partenaires, administrés, clients, ...) ;
- ✓ les agents de l'organisation réalisent des activités pour satisfaire cet objectif ;
- ✓ dans ces activités de l'information est manipulée, en mettant en œuvre des supports divers : un système d'information.

Pour rendre lisible et maîtrisable un système d'information, il est nécessaire d'en avoir plusieurs vues, par la mise en œuvre d'une démarche d'urbanisme. Ces vues, composantes d'un plan d'urbanisme, sont le support pour pouvoir faire évoluer et pour moderniser le système d'information :

- ✓ une vue externe (missions, administrés, citoyens, clients, partenaires, échanges, ...) ;
- ✓ une vue interne fonctionnelle (activités, agents, ...) ;
- ✓ une vue informationnelle (documents, informations gérées, ...) ;
- ✓ une vue applicative (outils informatiques mis au service de la réalisation des activités).



1

Vue externe

La vue externe est celle des partenaires, administrés, usagers, clients, ... Elle met en évidence et formalise les missions de l'organisation : le « modèle événementiel et partenarial ».

Il s'agit ici de considérer une organisation sous l'angle de ses interlocuteurs privilégiés, ceux à qui elle destine les résultats de sa « production » (citoyens, administrés, consommateurs, clients, ...), ceux dont elle dépend pour les produire (partenaires, fournisseurs, ...). La vue externe est centrée sur les échanges, qui peuvent prendre la forme d'échanges oraux (accueil à un guichet, téléphone, ...), d'échanges écrits (lettre, formulaire, contrat, ...)..., échanges qu'il est aujourd'hui de plus en plus question de dématérialiser (authentification / droits / échanges).

Vue interne

La vue interne est celle des agents de l'organisation qui collaborent à la réalisation des objectifs, c'est une vue mettant en évidence la façon dont l'organisation prend en charge ses missions : le « modèle des processus métier ».

L'organisation a pour objectif d'assurer la coopération de différents types d'acteurs pour permettre la réalisation d'activités nécessaires pour répondre aux sollicitations externes dont elle est l'objet, conformément aux services qu'elle est destinée à rendre, ce que nous appellerons « événements métier » (par exemple, une « demande de certificat de non gage » à une préfecture).

Un processus est un ensemble d'activités reliées entre elles, réalisées par des acteurs et concourant à atteindre un objectif fixé. L'étude des processus consiste en l'analyse du fonctionnement interne de l'organisation avec une vision transversale, en décrivant et analysant les activités des acteurs, en mettant en évidence le franchissement des frontières entre les différentes unités fonctionnelles.

¹ Emprunté à « *Urbanisation et modernisation du système d'information* » - éditions Hermès Lavoisier - 2004

Vue informationnelle

Une troisième vue est celle de l'information manipulée, qui constitue un micro univers, la représentation restreinte du monde dont s'est dotée une organisation pour satisfaire ses objectifs : le « modèle des objets métier et formats d'échange ».

Cette vue est de toute première importance dans un contexte de coopération et d'échanges (donc d'interopérabilité et de dématérialisation), elle sera largement précisée dans le chapitre suivant consacré aux formats d'échange.

Si l'on résume les vues présentées ci-dessus, il est possible de dire de la vue externe qu'elle constitue la réponse à la question « Pourquoi ? », de la vue interne la réponse aux questions « Qui et comment ? », de la vue informationnelle la réponse à la question « Quoi ? ».

De manière générale l'information manipulée dans le contexte d'un système d'information peut être distinguée selon son niveau de formalisation :

- ✓ informations structurées, c'est ce qui est couramment stocké dans les référentiels de données (les bases de données) d'une organisation ; le modèle des informations structurées, ou modèle des objets métier, permet de créer, pour chaque type d'interlocuteur d'une organisation, des vues particulières de ces informations adaptées aux modalités d'échanges avec ce type d'interlocuteur. L'ensemble de ces vues particulières constitue le référentiel des formats d'échange d'une organisation, d'une importance particulière dans un contexte de dématérialisation. ;
- ✓ informations non structurées (lire non structurées comme une base de données), c'est l'ensemble du référentiel documentaire d'une organisation, que l'on tente aujourd'hui, de plus en plus, de rendre facilement accessible (la généralisation de l'usage de XML permettant d'offrir une structure à des informations documentaires – cf. les référentiels de Légifrance, par exemple les codes (civil, pénal, du travail, ...) en ligne en XML).

Vue applicative

Une quatrième vue est celle des outils informatiques mis à la disposition des utilisateurs, et sensés supporter l'exercice opérationnel du métier, la « cartographie des applications et des flux ».

Les trois premières vues fournissent à elles seules la compréhension métier de ce qu'est le système d'information. La quatrième vue présente le système informatique, sous deux aspects :

- ✓ les applications et les flux inter applicatifs (l'aspect le plus important dans le cadre de ce document) ;
- ✓ l'architecture technique, c'est à dire à la fois les supports (systèmes, réseaux), la localisation physique, et les choix techniques.

L'ensemble de ces vues articulées du système d'information est nécessaire à la maîtrise de la complexité du système d'information, donc à son urbanisation, qui seule est à même de permettre de le moderniser (ouverture, dématérialisation, ...).

2.1.3 Urbanisme et interopérabilité

Les quatre vues brièvement présentées ci-dessus permettent de donner un éclairage particulier des enjeux de la modernisation des organisations et de leurs systèmes d'informations :

- ✓ au « modèle événementiel et partenarial » correspond l'enjeu d'ouverture du système d'information aux citoyens, aux usagers, aux administrés, aux partenaires ;
- ✓ au « modèle des processus métier » correspond l'enjeu de la transformation du métier, de la modification des façons de faire, souvent, mais pas seulement, induites par la modernisation du rapport aux usagers, administrés ou partenaires ;
- ✓ au « modèle des objets métier et des formats d'échange » correspondent les enjeux de valorisation du patrimoine informationnel des organisations d'une part, et d'autre part, de la dématérialisation des échanges ;
- ✓ à la « cartographie applicative » correspond l'enjeu de modernisation de l'outil informatique, visant notamment toujours plus d'évolutivité et de performance pour plus de réactivité.

En passant de l'urbanisation à la technologie, nous sommes amenés à noter qu'à la question stratégique et métier de l'urbanisation correspond la question technique de l'interopérabilité. En effet, ouvrir son système d'information, multiplier les échanges externes et internes et les dématérialiser, nécessite la mise en œuvre d'une informatique centrée sur le réseau qui va utiliser un modèle d'architecture où les applications ont entre elles des relations d'échange de données et/ou d'appel de services.

Ces applications sont distantes, diverses, disparates, et pourtant elles coopèrent, elles interopèrent. **L'interopérabilité correspond à la capacité, plus ou moins grande, qu'ont les applications, quelle que soit leur origine, appartenance, diversité, à coopérer entre elles.** Dans le cas présent, il s'agit de la capacité qu'ont les applications informatiques du système éducatif français, voire européen, à coopérer entre elles et avec les partenaires de ce système éducatif, avec un regard appuyé sur les applications destinées aux établissements d'enseignement.

2.1.4 Interopérabilité et *middleware*

La gestion de ces relations entre applications permettant leur coopération est prise en charge par une couche logicielle jouant un rôle d'intégration et assurant la communication entre des composants applicatifs, leur « inter opération » : c'est le rôle des solutions d'interopérabilité, qui incorporent nécessairement un *middleware* (le *hardware*, c'est le matériel, le *software*, c'est le logiciel, et le *middleware*, c'est le logiciel du milieu, celui qui est entre des logiciels différents et disparates pour les faire coopérer).

Synthétiquement, on distingue deux modes de communication entre applications :

- ✓ sans connexion (asynchrone) : chaque échange entre applications est mono-directionnel et n'a pas connaissance d'autre contexte que le sien. Les deux applications ne sont pas nécessairement actives en même temps. On parle de couplage faible entre applications.
- ✓ avec connexion (synchrone) : une session d'échanges bidirectionnels est maintenue entre les applications. Les deux applications sont nécessairement actives en même temps et en permanence durant tout le dialogue. On parle de couplage fort entre applications.

Il existe sur le marché une grande diversité de solutions d'interopération, des *middleware* que, par commodité et sans souci d'exhaustivité, nous classons en quatre catégories :

Accès aux bases de données

Ce type de *middleware* est fourni sous forme de pilotes (*drivers*) permettant à une application de se connecter à une base de données et de lui transmettre les requêtes à traiter. Ces pilotes encapsulent directement l'interface cliente du SGBD particulier considéré (de marque X ou Y) et sont fournis par les éditeurs de SGBD (X ou Y).

En s'adressant au *middleware* et non directement à la base de données, l'application s'est rendue en quelque sorte indépendante physiquement du fournisseur de la base de données (non d'un point de vue fonctionnel bien sûr, mais d'un point de vue technique : on peut passer d'Oracle à Sybase ou à PostgreSQL...) : un progrès de modularité et d'évolutivité significatifs.

Appel de procédures distantes

L'appel de procédures à distance et l'invocation d'objets à distance : ce mécanisme permet d'exécuter une fonction située dans un autre exécutable pouvant être sur une machine distante, ou permet à des objets distribués de communiquer par appel de méthodes. Son but est de masquer à l'appelant qu'une procédure appelée s'exécute sur une machine distante comme si elle était locale.

Dans ces deux cas, la dépendance vis-à-vis de la localisation physique d'une fonction est limitée, la présence du *middleware* invite à se concentrer sur la forme de l'appel, la structure des paramètres, autrement dit l'interface d'une fonction, son contrat d'utilisation. Un haut degré de modularité peut ainsi être atteint.

File d'attente de messages

La file d'attente de messages : des messages sont placés dans des files d'attente auxquelles des applications sont abonnées. Ce type de *middleware* orienté message découple les applications communicantes. Il permet une communication asynchrone fiabilisée et autorise des échanges dans des environnements distribués très hétérogènes.

La dépendance physique est ici minimale, il n'y a nul besoin que l'application destinataire du message (ou de la demande de service) soit active au moment de l'envoi du message (contrairement à l'appel de procédure), elle recevra le message au moment où elle redeviendra active et disponible et le traitera. Le niveau élevé de découplage entre applications tend à maximiser la modularité et à minimiser l'interdépendance.

Moniteur transactionnel

Le contrôle de transactions est assuré par des moniteurs transactionnels dont le rôle est de contrôler, dans un environnement distribué complexe, l'exécution de traitements transactionnels ACID (Atomique, Isolé, Cohérent, Durable) (il s'agit ici de solutions telles que CICS ou Tuxedo). Le *middleware* assure ici la capacité des parties communicantes à évoluer séparément.

Tous ces types de solution servent à la fois d'intermédiaire entre les applications diverses et de tampon entre elles ainsi qu'avec l'architecture du réseau qu'elles essaient de rendre transparente. Leur mise en œuvre aboutit à :

- ✓ des gains de modularité : chaque élément est rendu autonome et indépendant des autres, il se contente de respecter un contrat de service clairement défini ;
- ✓ une plus grande évolutivité technique : l'usage généralisé de *middleware* isole les couches techniques des éléments communicants, donc permet de substituer une solution

- technique à une autre équivalente ;
- ✓ une minimisation de l'adhérence (dépendance technologique) entre sous-systèmes, la communication via des *middleware* limite la dépendance entre sous-systèmes au besoin fonctionnel (contrat de service)..

La mise en œuvre systématique de *middleware* peut être généralement considérée comme un progrès d'urbanisation d'un système et facilite son ouverture.

2.2 Architecture d'interopérabilité

La généralisation du besoin d'ouverture des systèmes d'information met en valeur des solutions d'architectures visant à accroître l'interopérabilité et qu'il convient donc d'aborder ici :

- ✓ les formats d'échange ;
- ✓ les problématiques d'intégration de systèmes complexes : EAI ;
- ✓ le modèle d'architecture dont il convient de se fixer les principes comme cible : SOA (*Service Oriented Architecture*).

2.2.1 Formats d'échange

Un format d'échange est la définition partagée que donnent des partenaires des éléments d'information qu'ils sont amenés à échanger dans le contexte de leur partenariat.

Un format d'échange, comme nous le verrons dans le chapitre consacré à ce point, est la définition d'un ensemble d'informations structurées :

- ✓ le plus souvent modélisées avec UML (diagramme de classes), pour des raisons de lisibilité et des besoins de validation par les maîtrises d'ouvrage concernées ;
- ✓ toujours produites sous la forme de Schémas XML .

La constitution des formats d'échange vise à répondre à de multiples besoins :

- ✓ la multiplication des échanges entre les administrations ;
- ✓ l'accès des citoyens à des informations personnelles dans des conditions de sécurité acceptables ;
- ✓ la possibilité pour les citoyens d'effectuer un certain nombre de démarches plus simplement ;
- ✓ la capacité des citoyens d'accéder à ces services de chez eux ou en se déplaçant dans de multiples guichets polyvalents ;
- ✓ et de multiples autres services envisagés.

Ces besoins ne pourront être satisfaits sans une réflexion importante et un travail de définition de formats d'échange.

La circulaire du Premier Ministre du 21 janvier 2002 relative à la mise en oeuvre d'un cadre commun d'interopérabilité pour les échanges et la compatibilité des systèmes d'information des administrations stipule, à propos des schémas XML, qui constitue la forme que doivent prendre les formats d'échange :

« Il sera bon que chaque nouveau projet de système comportant des échanges d'informations (au sein de l'administration ou avec les tiers) soit l'occasion de poursuivre, et même d'intensifier, l'élaboration de schémas XML, dont on connaît l'importance pour faciliter les échanges. Ils seront conçus de manière à faire clairement apparaître leur définition, ainsi que celle des éléments qui les composent, par application de la méthode dite des "espaces nominatifs", conforme aux standards de l'Internet. Ils seront publiés, d'abord à l'état de projet, puis sous leur forme définitive, dans le répertoire des schémas XML de l'administration, dont la création a été décidée lors du comité interministériel pour la réforme de l'État en date du 12 octobre 2000 ».

Le répertoire des schémas XML de l'administration contribue à favoriser :

- ✓ l'interopérabilité et la dématérialisation des échanges au sein de l'administration, et des échanges qui lient l'administration à ses partenaires, prestataires et usagers ;
- ✓ la compatibilité des logiciels de traitement.

L'ADAE (www.adae.gouv.fr) est responsable de la publication de ce répertoire de schémas, et il est nécessaire de prendre connaissance de ses missions sur ce plan :

- ✓ http://www.adele.gouv.fr/sdae/IMG/rtf/repertoire_schemas_xml_version_1_juin.rtf

La démarche de définition de formats d'échange est générale aujourd'hui et concerne tous les secteurs d'activité.

Des langages XML « verticaux », c'est-à-dire couvrant des domaines métier bien délimités, adressant les domaines des sciences et de l'économie, se développent, comblant par là même un objectif majeur d'XML, celui de fournir un support à tous les échanges. Outre le répertoire des schémas XML de l'administration en France dont nous avons déjà parlé, tous les domaines métier se munissent aujourd'hui de schémas XML propres :

- ✓ mathématiques : MathML (*Mathematical Markup Language*) ;
- ✓ astronomie : AIML (*Astronomical Instrument Markup Language*) ;
- ✓ santé : HL7 (*Health Level Seven*) ;
- ✓ banque : IFX (*Interactive Financial eXchange*) ;
- ✓ commerce : cXML, xCBL, ebXML ;
- ✓ ...

DEVRAIT : Un des enjeux du déploiement des ENT est donc la définition, par les acteurs de l'éducation, éventuellement dans un contexte normatif européen, des formats d'échange propres à leurs métiers (la définition de ce qu'est une formation, un cursus, un support pédagogique donné, etc.).

2.2.2 Solutions d'intégration d'applications (EAI)

Dans de nombreuses administrations le parc applicatif est le reflet de l'évolution informatique des dix voire vingt dernières années. Il en résulte un système d'information composé de différents sous-systèmes disparates et hétérogènes n'offrant pas une vision et encore moins un fonctionnement intégré.

Or, l'ouverture des administrations vers l'extérieur (usagers, administrés ou partenaires) suite à l'avènement de l'Internet, les oblige à offrir un visage homogène de leur système d'information et à gérer de façon efficace les événements qui leur arrivent sous forme électronique (demande de formulaire, commande de document, déclaration en ligne, etc.).

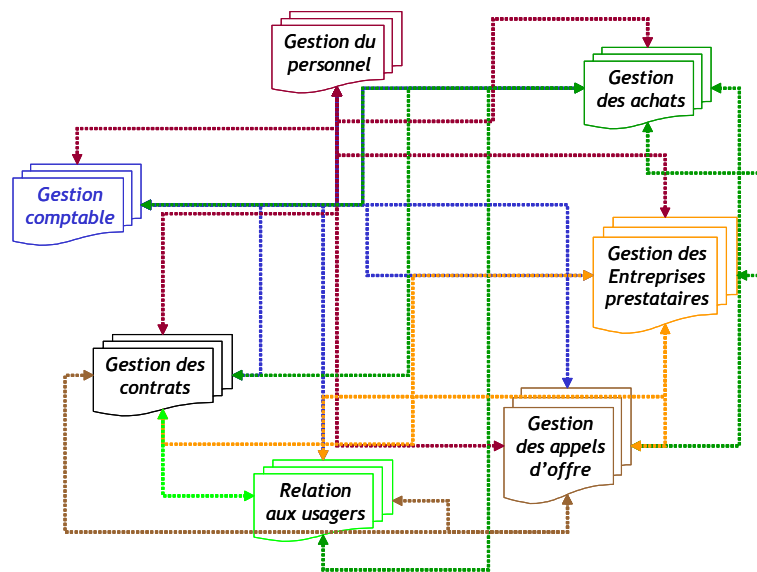
2.2.2.1 Le problème de l'intégration

Dans ce cadre, trois logiques peuvent guider la démarche menant au système d'information intégré :

- ✓ recréer un système de toute pièce afin qu'il soit intégré : c'est généralement impossible ;
- ✓ généraliser des passerelles inter-applications au coup par coup : c'est à terme ingérable ;
- ✓ fédérer des applications existantes et à venir via un moteur d'intégration.

Architecture « plat de spaghetti »

Les multiples interconnexions nécessaires entre les applications du système d'information sont souvent développées dans une vision point à point.



Il s'agit d'interconnecter chaque application avec les sources de données qu'elle utilise (les référentiels) ainsi qu'avec les autres applications qui fournissent des événements sur les données (changement d'état, création d'un nouvel objet ...).

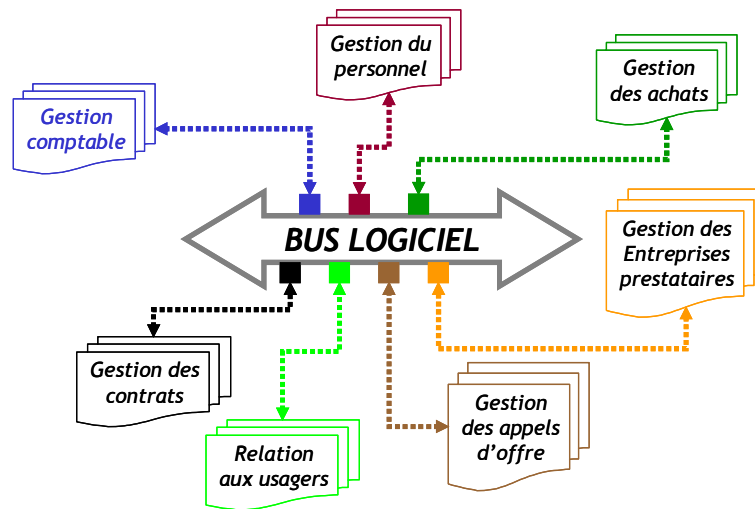
Ce modèle aboutit à un développement qui rend redondantes de nombreuses fonctionnalités d'interconnexion d'applications, ce que le Gartner Group appelle « Système plat de spaghetti ».

Les principaux écueils rencontrés dans ce modèle de développement sont tout d'abord le coût du développement (renouvelé pour chaque nouvelle application) puis le coût de la maintenance de ces nombreuses interfaces durant le cycle de vie de l'application (n applications communicantes $2 \times 2 = n(n-1)/2$ interfaces ; 10 applications=45 interfaces, 100 applications=4950 interfaces, une application change et c'est 99 interfaces qui changent !).

L'écueil majeur est surtout lié à la grande complexité de toute vision globale, urbanisée, cartographique, des échanges entre les applications, ce qui s'oppose à toute capacité d'évolution et d'ouverture.

Architecture « moyeu et rayons »

Les solutions d'intégration visent donc à rompre avec cette vision point à point des communications inter-applicatives en fournissant une plateforme unique et homogène de gestion des échanges, le modèle centralisé, généralement appelé « *hub & spoke* » (moyeux et rayons).

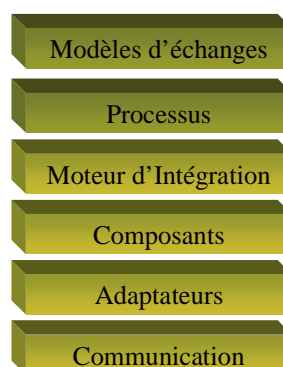


Du point de vue de l'exploitation, la plate-forme d'intégration étant le point de passage obligé de toutes les communications inter-applicatives, elle fournit un outil homogène d'administration des flux qui permet d'en fédérer l'utilisation, le nombre d'interfaces se réduit strictement au nombre des applications.

On retrouve sur ce plan la nécessité du format d'échange « en interne », toute application, quelle que soit la structuration de l'information qui lui est propre, communiquera dans un langage commun et banalisé avec toutes les autres applications, le langage pivot du système d'information, celui qui supporte l'intégration.

2.2.2.2 Les moteurs d'intégration

Les outils d'intégration distinguent six couches jouant chacune un rôle distinct dans l'intégration :



- ✓ communication : la communication entre les applications d'entreprise et l'outil d'EAI repose sur un *middleware*. Une plate-forme d'intégration doit savoir se connecter à tout *middleware* existant et doit proposer des passerelles entre *middleware* hétérogènes. Cette communication peut s'effectuer de manière asynchrone ou synchrone ;
- ✓ connecteurs / adaptateurs : permettent à la plate-forme d'EAI d'accéder aux informations détenues par les applications patrimoniales :

- un connecteur offre en général un simple accès technique au système à intégrer en s'appuyant sur une API propre à ce système,
- un adaptateur est un connecteur qui masque la spécificité de l'API (*application programming interface*) d'accès et des données natives en offrant une vue métier des données. Les adaptateurs peuvent être plus ou moins intrusifs en ce sens qu'ils nécessitent ou non une modification du système d'information à intégrer.
- ✓ composants : supportent la logique fonctionnelle de traitement de l'information manipulée par la plate-forme d'intégration et constituent l'extension fonctionnelle de la plate-forme d'intégration. En effet, le moteur d'intégration a beau être le cœur de la plate-forme d'EAI, c'est un organe essentiellement technique, qui ne dispose pas d'un référentiel de logique métier adapté ;
- ✓ moteur d'intégration : le cœur de la plate-forme d'EAI. Il joue deux rôles essentiels que sont le routage des données vers la ou les applications concernées et la transformation de ces données pour les adapter au format attendu par l'application destinataire :
 - transformation : La syntaxe et la sémantique des informations gérées par chaque application sont connues de la plate-forme d'EAI qui peut donc effectuer les transformations (appelées aussi *mapping*) nécessaires pour passer d'un modèle de données à un autre,
 - routage : permet d'effectuer correctement l'adressage des informations vers l'application adéquate. Ce routage n'est pas nécessairement point à point. La réception d'une commande peut, lors de la phase de transformation, générer plusieurs formats de données destinés à des applications différentes.
- ✓ processus : déclenché par un événement métier (par exemple la réception d'une demande de formulaire), il décrit les enchaînements d'actions à effectuer pour réaliser le traitement de l'événement. Il peut nécessiter l'intervention humaine pour fournir des données complémentaires, prendre une décision, ... Un moteur de *workflow* permet de dérouler ces processus métier en offrant des possibilités supplémentaires au simple moteur d'intégration : interruption / reprise de processus, envoi de notification à un utilisateur, ... Cette couche Processus permet de séparer la modélisation métier de l'implémentation technique des processus. La couche processus d'une offre d'EAI permet de créer le lien entre les processus internes de l'entreprise et les processus partagés entre partenaires. C'est pour cette raison que l'on parle d'intégration de processus (BPI – *Business Process Integration*) ;
- ✓ modèles d'échanges : On regroupe sous le terme de modèles d'échanges, les formes normalisées prises par les relations aboutissant à une transaction au moyen de protocoles de communication standards, légers et conçus pour Internet. Ces relations se font sur la base de définitions communes de modèles de données pour tous les échanges concourant à ces relations.

Les solutions d'EAI tendent à converger aujourd'hui vers un type d'architecture de service appelé *Enterprise Service Bus*, conjuguant les principes de l'EAI tels que présentés ci-dessus avec les techniques des architectures de services basées sur les Services Web.

DEVRAIT : Dans le cas d'un existant informatique substantiel dont il faudrait rendre disponibles certains services à l'extérieur de l'entité qui l'utilise habituellement (modification des habitudes de gestion, partenariat, ouverture du système d'information, ...), les solutions d'EAI sont nécessairement à examiner.

2.2.3 Architectures de service (SOA)

L'ouverture des systèmes d'information consiste précisément à mettre à la disposition d'utilisateurs externes à une organisation des services du système d'information de cette organisation. Cela nécessite bien sûr de gérer les droits et habilitations afin d'assurer la sécurité nécessaire, ce point a été abordé dans l'annexe AAS du SDET. Cela invite surtout à mettre en place des architectures de systèmes d'information orientées service.

2.2.3.1 La notion de service logiciel

La notion de service logiciel n'est pas liée à une technologie ou à un outil en particulier mais plutôt à un modèle de conception des applications. Les principales caractéristiques d'un service logiciel sont les suivantes :

- ✓ un service logiciel est d'abord un module logiciel utilisable par programmation : il n'est pas directement destiné à l'utilisateur, et il est nécessaire de bâtir une interface homme-machine, dans un autre module logiciel, pour une utilisation humaine. Le service logiciel implémente donc une des règles des architectures client/serveur multi-niveaux : la séparation des traitements et des interfaces ;
- ✓ un service logiciel est autonome, complet et cohérent : il fournit plusieurs fonctions liées au même macro objet métier, par exemple la création, la modification et la consultation d'un contrat. C'est un service de caractère métier (macroscopique et signifiant) et non technique (microscopique et dénué de sens métier) ;
- ✓ un service logiciel est autodéscriptif et permet la réutilisation : il fournit une description des fonctions proposées de manière à n'imposer aucun prérequis à leur utilisation : le format de description est commun à tous les services et manipulable par tous les outils de développement qui utilisent les services. Cette description constitue un véritable contrat de service, le service logiciel est indépendant des plates-formes et outils de développement.

Le service logiciel se distingue du composant logiciel en ce qu'il est un service métier de haut niveau et non une fonction de bas niveau ou technique

2.2.3.2 Une architecture de services

Construire une architecture de services est une stratégie qui consiste à structurer le système d'information comme un ensemble d'applications ayant la capacité d'exposer leur interface fonctionnelle.

Ce modèle d'intégration orientée « services » permet de satisfaire des besoins de collaboration plus forte entre les applications d'un système ou avec celles des systèmes de partenaires. Il est ainsi envisageable de satisfaire des besoins d'automatisation de processus transversaux (inter-établissement et inter-sites), ou bien ceux d'applications de type « portail » fédérant l'accès à différentes applications de différents établissements.

Ce modèle induit de fait un annuaire des services, accompagné de mécanismes d'agrément et de définitions de processus d'échanges avec les partenaires, ainsi que tous les services techniques de sûreté de fonctionnement, d'administration et de supervision des applications et de leurs échanges.

Cette cible architecturale (aussi appelée SOA, pour *Service Oriented Architecture*) est une infrastructure logicielle qui englobe des composants applicatifs relativement autonomes, les met à disposition au travers du réseau en les répertoriant dans un annuaire, et permet leur utilisation au travers d'un modèle de communication indépendant des technologies d'implémentation. Elle conduit ainsi à une réutilisation maximale et systématique des données et des applications.

Au niveau de l'interopérabilité, les avantages procurés par ce modèle architectural sont :

- ✓ des interfaces indépendantes des plates-formes technologiques ;
- ✓ une granularité des échanges de niveau « métier » ;
- ✓ une fluidité des échanges, pouvant se faire en temps réel (synchrone) ou en temps différé (asynchrone) ;
- ✓ des communications qui peuvent s'appuyer sur des couches de transport hétérogènes (HTTP, SMTP, FTP, JMS...) ;
- ✓ une intégration avec d'autres systèmes qui peut se paramétrer rapidement.

2.2.3.3 *Web Service Oriented Architecture (WSOA)*

Le concept d'architecture orientée services, après avoir été l'objet d'implémentations initiales, revient aujourd'hui sous les feux de l'actualité, du fait de l'apparition d'un ensemble de nouvelles technologies – les Services Web – directement dérivées du langage XML.

Il convient de distinguer les termes « Services Web » et « architecture orientée services » qui recouvrent des concepts différents, mais interdépendants.

L'expression « Services Web », au sens où nous l'entendons aujourd'hui (et non pas au sens plus général qui désigne tout service au sens large offert par un prestataire d'accès Internet), désigne « *un système logiciel, identifié par un URI, dont les interfaces publiques et les liaisons sont définies et décrites en format XML. Cette définition peut être découverte par d'autres systèmes logiciels. Ces systèmes peuvent interagir avec le service Web, selon la manière précisée dans la définition, par l'utilisation de messages en format XML véhiculés sur des protocoles Internet.* » (traduction de la définition donnée dans le glossaire du W3C).

Ce qui est important dans cette définition est l'omniprésence du langage XML : la définition de l'interface publique, la description de la liaison entre le service et les protocoles de communication qui peuvent être utilisés pour y accéder, le format des messages échangés, l'instance de service Web et ses processus clients sont tous décrits en format XML.

DEVRAIT : Une solution majeure de mise en place d'une architecture de service repose aujourd'hui sur la mise en œuvre des Services Web, nous revenons sur le détail de cela dans le chapitre consacré aux Services Web. C'est une orientation forte aujourd'hui pour la modernisation et l'ouverture des systèmes d'information publics.

DEVRAIT : La définition de Services Web implique de les décrire au format WSDL. La possibilité offerte par le répertoire de service UDDI est celle de rendre accessibles à la communauté de l'éducation nationale les services publiés par des membres de cette communauté : la mise en place d'un annuaire UDDI au ministère permettrait cette publication.

3 Méthodologie de l'interopérabilité

3.1 Conduite de projet et interopérabilité

Les projets mettant en jeu des problèmes d'interopérabilité et des problèmes d'ouverture de système d'information doivent donner lieu à une gestion particulièrement vigilante. Il convient de respecter attentivement les points suivants :

- ✓ tout projet d'interopérabilité et d'ouverture pose un problème de sécurité, cette question appelle la plus grande attention ;
- ✓ tout projet d'interopérabilité et d'ouverture pose un problème de destinataire des services : ce n'est pas le maître d'ouvrage lui-même qui est le destinataire principal du service, mais c'est plutôt un administré, un partenaire, une personne ou organisme extérieur, ..., c'est donc quelqu'un qu'il va falloir consulter pour savoir ce qu'il souhaite en évitant de penser qu'on sait à sa place très exactement ce qu'il veut, et ce dont il a besoin ;
- ✓ tout projet d'interopérabilité et d'ouverture pose un problème de disponibilité de service : on est là dans une logique inhabituelle, celle de la disponibilité 7j/7 et 24h/24, il importe d'en tenir compte très tôt dans le projet pour prévoir un déploiement correct ;
- ✓ tout projet d'interopérabilité et d'ouverture peut viser à offrir un service à un acteur externe, celui-ci peut-être assez facilement convaincu d'utiliser le service en ligne, mais il est très volatile, tout échec peut le détourner durablement de ce type de solution, il ne faut donc mettre en ligne que sur la base d'une bonne qualité de service : attention donc à la précipitation et à l'excès d'ambition, mieux vaut moins mais mieux ;
- ✓ tout projet d'interopérabilité et d'ouverture pose un problème d'accès pour tous, il faut donc toujours penser aux publics particuliers que sont par exemple les handicapés (notamment les mal voyants) ;
- ✓ tout projet d'interopérabilité et d'ouverture a nécessairement un impact sur la façon de travailler et donc sur la réactivité attendue des services, il faut donc se pencher attentivement sur ce problème et faire en sorte que les agents soient préparés et motivés à mettre en œuvre ces nouvelles manières de travailler, communication et conduite du changement s'imposent donc ;
- ✓ tout projet d'interopérabilité et d'ouverture met en jeu des technologies dont la maîtrise n'est pas nécessairement partagée, il convient donc de gérer avec la plus grande attention le risque technologique.

3.1.1 Cycle projet et risques associés

Tout projet informatique suit un cycle standard suivant (dans le cas d'un appel à prestation externe), nous allons souligner à chaque phase les risques majeurs associés :

- ✓ étape de préparation de projet :
 - recueil et analyse critique du besoin ;
 - attention à intégrer les destinataires du service,
 - attention à fixer des objectifs atteignables,
 - attention prévoir les conséquences de la mise en place du service sur les modes de travail,

- prise de connaissances des solutions disponibles et état de l'art ;
 - attention aux effets d'annonce des éditeurs, vérifier la réelle disponibilité d'une solution,
 - attention aux solutions propriétaires et non standards qui risquent de nous détourner durablement de notre cible,
 - attention aux standards exprimés sous forme de spécification, il faut vérifier la disponibilité réelle des implémentations,
- rédaction d'un cahier des charges ;
 - attention aux cahiers des charges qui se limitent à du texte (donc trop verbeux),
 - utiliser autant que possible les formalismes reconnus pour exprimer les besoins, ceux-ci permettent de lever bien des ambiguïtés (modèles de processus, cartographies applicatives, UML, ...),
 - réfléchir dès la rédaction du cahier des charges à la grille d'évaluation des réponses, celle-ci permet en fait de vérifier la bonne hiérarchisation des thèmes dans le CdC,
- ✓ étape de réalisation :
 - spécifications détaillées ;
 - prévoir les ressources nécessaires pour exprimer le besoin dans son niveau de détail le plus fin,
 - se poser systématiquement sur tout point de spécification les questions : sécurité, disponibilité, impact sur les modes de travail,
 - prévoir les ressources nécessaires à une validation formelle des spécifications détaillées réalisées par la maîtrise d'œuvre,
 - conception technique ;
 - vérifier que la maîtrise d'œuvre met effectivement en œuvre les bonnes pratiques en terme d'interopérabilité,
 - développement ;
 - préparation de la validation ;
 - prévoir des éléments de validation spécifiques liés aux questions de sécurité, interopérabilité, satisfaction des destinataires quand c'est possible,
 - validation et gestion des retours ;
 - ne procéder à une validation formelle qu'à partir du moment où le livrable est conforme au spécifié,
 - déploiement ;
 - tout livrable logiciel doit nécessairement inclure la documentation et les supports nécessaires au déploiement,
 - mise en service ;
 - anticiper les impacts éventuels sur les agents,
 - ne pas décevoir le destinataire au premier contact, car alors on peut le détourner durablement des solutions électroniques, c'est là qu'on reboucle avec l'expression du besoin et son ambition : mieux vaut offrir un service moindre mais qui marche qu'un service prétendument extraordinaire mais qui ne marche pas.

3.1.2 Quelques questions importantes

Les caractéristiques attendues d'un socle

- ✓ le socle est un portail d'information et de service ;

- ✓ les accès en sont gérés par un annuaire qui assure la gestion des droits ;
- ✓ il permet une présentation et un accès unifié aux services (il dispose et met en œuvre une charte graphique formalisée) ;
- ✓ il assure la gestion de sessions utilisateurs ;
- ✓ il est compatible avec les protocoles standards du marché ;
- ✓ il respecte une architecture modulaire ;
- ✓ il permet le SSO.

Comment décrire un service ?

- ✓ une service se décrit du point de vue de son usage, c'est-à-dire d'un point de vue externe ;
- ✓ la définition d'un service est donc celle d'un contrat d'utilisation de ce service ;
- ✓ la définition d'un service est donc celle des interfaces de ce service :
 - interfaces applicatives,
 - interfaces homme machine.

Compétences requises pour les projets ?

- ✓ compétence importante en gestion et management de projet ;
- ✓ compétence importante en expression de besoin et modélisation (notamment UML) ;
- ✓ compétence essentielle en connaissance des technologies et capacité de veille technologique et connaissance de l'état de l'art.

3.2 Préconisations et exigences

- 1) Les liaisons inter-applicatives NE DOIVENT PAS être de type point à point. L'architecture globale DEVRAIT être de type bus logiciel.
- 2) Nous avons distingué plus haut la notion de « service logiciel » identifié au début de la partie sur les SOA des « composants logiciels ». Les « services logiciels » DOIVENT proposer une interface d'accès de la famille de protocoles Services Web et XML, l'accès aux « composants logiciels » DOIT avoir lieu à travers des protocoles comme WebDAV et des outils de *middleware* (base de données,...).
- 3) La publication de ressources numériques DOIT utiliser XML. Le fond et la forme du document DOIVENT être séparés, par exemple à l'aide de feuille de style ou de moteur de transformation XSL ; la transformation XML -> XHTML DOIT avoir lieu sur les serveurs, en différé ou à la volée. Dans le cas d'intranets où le poste client est maîtrisé, la transformation PEUT avoir lieu sur le poste client ;
- 4) Dans le cadre des Services Web, les protocoles SOAP, WSDL et UDDI DOIVENT être utilisés. Les spécifications des protocoles WS-Security et BPEL4WS, qui seront disponibles à une date encore indéterminée, POURRAIENT fournir un cadre d'interopérabilité pour la sécurisation de l'accès au service et l'orchestration des services si elles répondent aux besoins de l'Éducation nationale.

- 5) L'annuaire LDAP de l'établissement ou de l'école DOIT être utilisé pour gérer des ressources, leurs profils et leurs droits. L'annuaire LDAP ne DOIT PAS être une base de données pour gérer des informations autres que celles servant à l'authentification et à la gestion des habilitations. L'usage d'un LDAP est en lecture seulement en exploitation et en écriture en administration.
- 6) L'accès aux bases de données DEVRAIT avoir lieu à travers un *middleware* base de données.
- 7) Pour assurer la réception asynchrone de message, un *middleware* orienté message PEUT être utilisé.
- 8) Pour des raisons de lisibilité, de possibilité de partage et de communication sur des modèles d'information, les projets DEVRAIENT utiliser la modélisation UML à la fois pour les développements applicatifs et la définition de formats d'échange.
- 9) Les services destinés à être publiés sur le portail DOIVENT utiliser le SSO fourni par le socle.
- 10) Les services destinés à être publiés sur un portail DEVRAIENT proposer une interface conforme à la recommandation WSRP du consortium OASIS.

4 UML, modèles d'objets métier et formats d'échange

4.1 UML : *Unified Modeling Language*

UML est né de la fusion de trois des plus célèbres méthodes de conception « objet » du début des années 90 :

- ✓ OOAD (*Object Oriented Analysis and Design*) de Grady Booch (Rational)
- ✓ OMT (*Object Modeling Technique*) de James Rumbaugh (General Electric)
- ✓ OOSE (*Object Oriented Software Engineering*) de Ivar Jacobson (Ericsson).

Au début des années 90, il existe en effet plus de 70 méthodes de modélisation recensées pour le paradigme objet. En 1994, Grady Booch et James Rumbaugh décident de travailler ensemble pour rapprocher leurs méthodes (OOAD & OMT). De nombreux partenaires sponsorisent ces travaux (HP, Oracle, Microsoft, IBM, etc.) sur *Unified Method*.

Fin 1996, l'OMG (*Object Management Group* – consortium éditant des spécifications de standards regroupant les 800 plus grosses entreprises mondiales utilisatrices et productrices de logiciel) lance un appel d'offre pour la standardisation des langages de modélisation, le projet *Unified Modeling Language* est né, il a pour objectifs :

- ✓ la recherche d'un consensus sur un langage de modélisation commun ;
- ✓ l'utilisabilité par toutes les méthodes, l'adaptation à tous les besoins d'un projet, la compatibilité avec toutes les techniques de réalisation ;
- ✓ la convergence des méthodes à partir du partage d'une notation unique, facilitant la traçabilité entre les phases d'un projet, permettant l'échange de modèles entre les outils ;
- ✓ la standardisation des supports du développement (Modèles, Notations, Diagrammes).

UML est un langage permettant de spécifier, de construire, de visualiser et de documenter l'ensemble des aspects d'un système :

- ✓ il dispose de symboles, d'un lexique, d'une grammaire définissant une syntaxe, d'une sémantique favorisant la communication entre personnes (utilisateurs, consultants, programmeurs...);
- ✓ il permet de construire le modèle du métier (le besoin à satisfaire) ou celui du système (la solution informatique à réaliser) ;
- ✓ il permet d'obtenir un point de vue formel et synthétique d'un système afin de dominer la complexité d'un problème et d'une solution.

UML est construit sur la base d'un métamodèle, écrit en UML. Le métamodèle d'UML définit la sémantique des concepts de base :

- ✓ classement des concepts ;
- ✓ signification de ces concepts ;
- ✓ représentation graphique.

UML 1.4, qui est le standard en cours, propose 9 types de diagrammes (UML version 2.0 en cours de validation en propose 13). Tous ne sont pas utiles pour ce qui nous concerne ici, c'est à dire la modélisation d'objets métier et de formats d'échange.

4.1.1 Appui sur un standard ou une norme

UML version 1.4 est le standard international de modélisation objet.

UML 2.0 est en cours de validation.

Il est standardisé par l'OMG (Object Management Group) : www.omg.org

L'ensemble des ressources est disponible à : www.omg.org/uml

4.1.2 Conditions d'emploi

UML est destiné à modéliser la structure de l'information, que cette information soit celle manipulée par une application (modèle d'objets métier), qu'elle soit celle stockée dans une base de données, ou encore qu'elle soit celle utilisée pour réaliser des échanges avec des partenaires (référentiels de schémas XML).

UML couvrant aussi bien les besoins de modélisation dans une phase d'analyse que de conception détaillée, il convient de bien déterminer les usages à des fins d'analyse, et ceux à des fins de conception.

UML dispose d'une grande expressivité du fait des concepts mis en œuvre (héritage, association, ...). Le passage du modèle UML au schéma XML (modèle hiérarchique) nécessite une transformation (dérivation) qu'il convient de définir et maîtriser. De la même manière, le passage du modèle UML à un modèle relationnel (modèle tabulaire) nécessite une transformation (dérivation) qu'il convient de définir et maîtriser.

4.1.3 Outils disponibles

Il existe de très nombreux outils de modélisation implémentant la spécification UML. La plupart sont à un très haut degré conformes à cette spécification.

Les outils mettent généralement en œuvre des formats internes qui les rendent incompatibles (un modèle construit dans un outil et conforme à UML n'est pas spontanément migrable vers un autre outil). Il existe un standard d'interopérabilité des outils de modélisation UML qui s'appelle XMI (et qui met en œuvre XML), ce standard est cependant encore à ce jour incomplet, et ne supporte pas le transfert de la disposition graphique des modèles (organisation visuelle et graphique des diagrammes). Il convient néanmoins de vérifier qu'un outil choisi supporte les imports et exports XMI.

Pour plus d'information sur les nombreux outils disponibles :

- ✓ http://www.objectsbydesign.com/tools/umltools_byCompany.html

et aussi :

- ✓ http://www.cetus-links.org/oo_oaa_ood_tools.html#oo_oaad_tools_start_here

4.1.4 Compétences requises

UML est un outil puissant de modélisation, et donc complexe. Il convient que ses utilisateurs aient pu disposer d'une formation spécifique.

4.2 XML : eXtended Markup Language

Après cinq années d'existence, XML a apporté un profond courant d'évolution dans la manière dont l'information est gérée au sein des organisations : le patrimoine informationnel tend à s'affranchir des technologies propriétaires. L'utilisation d'XML a largement dépassé le cadre documentaire, le cœur du système d'information est concerné :

- ✓ sa capacité d'ouverture et d'échanges avec les partenaires ;
- ✓ les fondations de son architecture.

XML a pour objectif de fournir :

- ✓ une approche simple utilisant des balises pour structurer un document ;
- ✓ un langage lisible, mais conçu pour les programmes et indépendant de la technologie ;
- ✓ un langage flexible et extensible ;
- ✓ un langage séparant la structure du contenu et de sa présentation.

XML n'est pas un langage mais un « meta-langage », il permet de construire des langages partageant la même syntaxe et la même grammaire, ainsi il atteint les objectifs de permettre une vérification syntaxique et sémantique des documents (cf. la partie format d'échanges) :

- ✓ syntaxique : un document est bien formé s'il se conforme aux règles syntaxiques du langage XML, un seul nœud racine, une imbrication correcte des éléments ;
- ✓ sémantique : un document est valide s'il se conforme au vocabulaire et à la grammaire définie par la DTD ou le schéma XML.

4.2.1 Principales composantes techniques de XML

XMLSchema

Les Schémas XML visent à palier aux insuffisances des DTD (technique précédente de définition de document – *Document Type Definition*). Les schémas XML sont des modèles exprimés en XML, ils gèrent des types de données et des contraintes étendus.

Les espaces de nommage (namespaces)

Les vocabulaires XML étant libres, deux applications XML peuvent utiliser le même vocabulaire, ce qui peut amener des problèmes de communication et/ou de compatibilité. Les espaces de nommage visent à éviter l'ambiguïté des vocabulaires, afin de garantir la réutilisation des applications XML.

XSL

XSL permet de formater les données contenues dans un document XML en un document exploitable, au sens où il est mis en forme (HTML ou PDF par exemple). XSL est à la fois un

langage de transformation d'un document XML (XSLT) et un vocabulaire XML pour formater les données (*Formating Objects* - FO).

XML Path Language (XPath) Version 1.0

XPath permet de localiser une partie d'un document XML par chemin d'accès ou expression de sélection. XPath est commun à XSLT et Xpointer.

4.2.2 Appui sur un standard ou une norme

Les travaux de normalisation sur XML sont gérés par le consortium W3C. Depuis le 10 février 1998, XML 1.0 est une recommandation du W3C (www.w3.org):

- ✓ dernière recommandation (celle du 6/10/2000) : <http://www.w3.org/TR/REC-xml>
 - en cours : XML 1.1 avec support Unicode 3.1 (*Candidate Recommendation* du 15/10/2002) : <http://www.w3.org/TR/xml11/>
- ✓ les travaux du W3C sur XSLT ont le statut de recommandation depuis le 16 novembre 1999 <http://www.w3.org/TR/xslt> (Version 1.1 en *Working Draft* (Ajout de langages de script)) ;
- ✓ les travaux du W3C sur FO ont le statut de recommandation depuis le 15 octobre 2001 <http://www.w3.org/TR/xsl> (FO Version 1.0) ;
- ✓ les travaux du W3C sur XML Path ont le statut de recommandation depuis le 16 novembre 1999 <http://www.w3.org/TR/xpath>.

Autres standards ou candidats standards liés à XML :

- ✓ DOM (*Document Object Model*), SAX (*Simple API for XML*) : interface de programmation pour les messages et documents XML échangés ;
- ✓ XHTML : reformulation d'HTML comme langage XML sous forme modulaire afin de permettre sa généralisation à tous les terminaux ;
- ✓ RDF (*Ressource Description Framework*) : définition de méta-données associées au Web, pierre angulaire du « Web sémantique » ;
- ✓ SMIL (*Synchronised Media Integration Language*) : langage de synchronisation des objets multimédia (animations audio, vidéo...) ;
- ✓ SVG (*Scalable Vector Graphics*) : langage de description des graphiques vectoriels en 2D ;
- ✓ XForms (XML Forms) : formulaires XML ;
- ✓ XML Signature, XML Encryption : sécurité basée sur une IGC (infrastructure de gestion de clés).

4.2.3 Conditions d'emploi

Nous n'aborderons pas ici l'utilisation d'XML dans les Services Web, dans les formats d'échanges, voir à ce propos les chapitres concernés.

4.2.3.1 Gestion de contenu

XML est un support idéal pour gérer de manière organisée des contenus documentaires. C'est ainsi, par exemple, que Légifrance dispose d'un référentiel de tous les codes (code du travail, code

civil, code pénal, code de la sécurité sociale, ...) en XML, les documents sont ainsi structurés, indexables, à même de supporter une recherche documentaire efficace, ...

Dans le contexte des espaces numériques de travail, il convient de favoriser au maximum la gestion de contenus au format XML, dont le schéma XML (la structure) est correctement défini et diffusable. Cela est vrai pour les contenus :

- ✓ fournis par les fournisseurs de contenus (éditeurs de dictionnaires, encyclopédies, manuels, ...);
- ✓ créés par les éditorialistes de l'Éducation nationale (élèves, étudiants, enseignants, chercheurs, ...).

L'utilisation d'XML pour la gestion de contenu vise à permettre :

- ✓ la publication multi-supports ;
- ✓ la réutilisation du contenu au sein du système d'information ;
- ✓ une séparation entre le contenu et la présentation.

La création de contenu doit être généralisée dans le cadre des espaces numériques de travail. Cette création de contenu doit tendre à s'organiser selon un schéma qui enchaîne :

- ✓ la définition de modèles de contenus (modèles de documents - fonction d'administration) - Technologies RDF et Schéma XML ;
- ✓ la création de contenus structurés selon ces modèles (fonction utilisateur éditorial) - Technologies XML ;
- ✓ la catégorisation et la recherche sur des contenus (fonction basée sur la structure pour les gens qui consultent les contenus) - Technologies Méta-données en XML - Les métadonnées XML permettent de spécifier les catégories auxquelles les documents sont rattachés, ce qui facilite la localisation de l'information : recherche sur les catégories, navigation dans les catégories ;
- ✓ la personnalisation et la spécialisation de contenu (fonction basée sur la structure pour les gens qui consultent les contenus) - Technologies XSL, XSLT et XSL-FO.

Il existe deux possibilités de création de contenu structuré XML :

- ✓ à partir d'éditeurs de texte classiques ou de formulaires WEB (créés avec Web forms par exemple), pour les données structurées « simples », pour une saisie occasionnelle, lorsque tout utilisateur est appelé à être contributeur ;
- ✓ à partir d'un éditeur XML, pour les données structurées « complexes », lorsque la création de contenu est le travail de production, lorsque l'on désire mettre en place une solution de WCM (Web Content Management – gestion de contenu).

4.2.3.2 Persistance des documents XML

Plusieurs niveaux d'intégration d'XML aux bases de données existent :

- ✓ niveau 1 : La base de données retourne un document XML à partir d'une requête SQL classique, langages de requêtes SQL avec extensions éventuelles ; cela concerne donc tous les SGBDR ;
- ✓ niveau 2 : La base de données accepte les requêtes sous forme de documents XML, langages de requêtes basés sur XPath et XQuery (W3C) ;
- ✓ niveau 3 : La base de données accepte le stockage des documents XML dans leur format

natif, langages de requêtes basés sur XPath et XQuery (W3C).

Solutions « libre » :

- ✓ Apache Xindice : niveau 3 ;
- ✓ Projet XML Apache - xml.apache.org/xindice ; évolution Open Source du projet dbXML basé sur l'initiative XML:DB (www.xmldb.org).

Solutions commerciales :

- ✓ Microsoft SQLServer 2000 – XMLSQL : solution de Niveau 2 ;
- ✓ Oracle 9i : solution de niveau 2/3 ;
- ✓ Tamino (Transaction Architecture for the Management of Internet Objects) : solution de niveau 3 ;
- ✓ eXcelon XIS (eXtensible Information Server) – ObjectStore - serveur de données XML : solution de niveau 3 ;
- ✓ X-Hive/DB : solution de niveau 3 ;
- ✓ ...

4.2.3.3 Personnalisation

Le contenu structuré ou non structuré formaté avec XML dispose de capacités de transformation via XSL qui vont être mises à profit pour formater tout contenu en fonction du périphérique de destination ou de l'utilisateur de destination.

La personnalisation s'effectue ainsi en terme de :

- ✓ contenu - Les transformations XSL peuvent également effectuer un filtrage des données ou contextualiser le contenu et la navigation en fonction du profil de l'utilisateur ;
- ✓ présentation - Les transformations XSL mettent en forme un contenu en fonction de préférences associées à un profil ;
- ✓ support - Les transformations XSL adaptent le contenu au support cible : navigateur Web, PDA, WAP, imprimante (PDF)...

La personnalisation donne ainsi à un contenu unique une utilisation multiple.

4.2.3.4 Architecture XML XSL

XML permet de garder une plus grande indépendance entre les données et les applications. Il s'insère facilement dans un environnement hétérogène. Les informations sont transmises, présentées et manipulées différemment en fonction des utilisateurs et de leurs besoins. La présentation est gérée par XSL. Différentes solutions d'architecture sont possibles pour cette mise en forme XSL. Les architectures sont complémentaires et non concurrentes. Elles doivent être adaptées au besoin.

Transformation XSL sur le serveur

Cette solution elle permet de garantir la maîtrise de la transformation, et par conséquent celle de la présentation des données aux clients. Elle oblige en contrepartie le serveur à supporter les coûts de transformation.

Les transformations XSL peuvent être effectuées à la demande des clients. La génération « à la volée » offre la possibilité de personnaliser le rendu des données en fonction du client, mais induit des coûts importants de transformation.

Les transformations peuvent également être anticipées, c'est-à-dire être effectuées en batch, a priori. Si ce mode de fonctionnement peut garantir les montées en charge, il est en revanche incompatible avec la personnalisation du rendu, et induit une désynchronisation entre les données présentées et les données existantes au moment de la consultation.

Transformation XSL en *batch* :

- ✓ les documents XML sont mis en forme en différé et rendus disponibles au format XHTML ;
 - avantages : efficacité de traitement ; performance à la restitution ; maîtrise de la transformation,
 - inconvénients : nombre de modèles limités ; pas de personnalisation,
 - recommandé pour un Internet : permet de garantir les montées en charge.

Transformation XSL sur serveur WEB :

- ✓ les documents XML sont mis en forme au format XHTML à la demande sur le serveur web ;
 - avantages : personnalisation et agrégation ; modification uniforme des modèles et des styles ; maîtrise de la présentation,
 - inconvénients : coût de traitement ; charge du serveur,
 - recommandé pour un extranet : une montée en charge maîtrisée ; permet d'offrir de la personnalisation.

Transformation XSL sur navigateur client

- ✓ les documents XML sont mis en forme au format (XHTML, ...) à la demande sur le navigateur client (retour au client « riche ») ;
 - avantages : coût de transformation déporté sur poste client ; personnalisation et agrégation ; modification uniforme des modèles et des styles,
 - inconvénients : nécessite un client supportant XML et XSL (ou CSS2) ; limite les cas d'utilisation (Web),
 - recommandé pour un intranet : les postes clients sont susceptibles d'être maîtrisés (masterisés) ; libère la charge coté serveur.

4.2.4 Compétences requises

Pour la création de contenus sous forme de fichiers XML, seule la connaissance de l'outil utilisé s'impose.

Pour le reste : connaissance des technologies du Web, connaissance notamment de XHTML, connaissance de XML, connaissance des schémas, connaissance des outils, et cf. chapitre consacré aux formats d'échange : connaissance de UML (*Unified Modelling Language*).

5 Les outils d'interopérabilité

5.1 Services Web

La première utilisation des Services Web consiste à fournir une interface SOAP à des applications existantes (CCM, EJB ...). Ceci est extrêmement facile à mettre en œuvre parce que la plupart des environnements de développement permettent d'exposer les logiciels qu'ils fabriquent sous forme de Services Web :

- ✓ **Les Services Web s'affirment donc comme le protocole universel de dialogue entre composants hétérogènes.**

Les Services Web peuvent également servir pour bâtir des composants, sans restriction quant aux technologies ou aux outils qui permettent de les implémenter. Ainsi le composant n'est plus COM (Microsoft), CCM (CORBA) ou EJB (J2EE) mais un composant SOAP, accessible par toutes les applications du système d'information.

Le concept de « Services Web » se concrétise autour de trois standards complémentaires que sont : SOAP, UDDI et WSDL :

- ✓ SOAP (*Simple Object Access Protocol*) est un protocole léger, permettant d'échanger des informations dans un environnement décentralisé et distribué. Son objectif est de faire communiquer ensemble des services distribués réalisés à l'aide de technologies différentes.
- ✓ WSDL (*Web Service Definition Language*) est le langage de description de services. C'est une syntaxe au format XML qui, au même titre qu'IDL (cf. CORBA) décrit les interfaces de services.
- ✓ UDDI (*Universal Description, Discovery and Integration*) est une spécification d'annuaire des Services Web mis à disposition par une ou plusieurs entreprises ; il permet la découverte d'un service, sa sélection, et la mise à disposition de la description des interfaces (WSDL) nécessaires à son invocation.

Les Services Web font l'objet d'un consensus indéniable entre les éditeurs de solutions logicielles et les organismes de normalisation tels que le W3C et l'OASIS.

Ils se démarquent des autres *middlewares* sur trois principaux aspects :

- ✓ l'utilisation d'un protocole de communication – SOAP – basé sur XML ;
- ✓ le recours aux protocoles de transport universels comme HTTP, SMTP, FTP très peu contraignants lors du déploiement d'un réseau et aujourd'hui bien maîtrisés et disponibles pour toute entreprise utilisatrice d'Internet ;
- ✓ l'utilisation conjointe de XML, aujourd'hui supporté par tous les outils de développement, et de protocoles largement répandus et déjà déployés au sein des entreprises, permet de réduire considérablement les difficultés liées à l'apprentissage des technologies, à leur déploiement et leur sécurisation.

5.1.1 Appui sur un standard ou une norme

Les technologies de Services Web sont édifiées sur un très important socle de spécifications, dont les plus importantes sont déjà en cours de normalisation. Il faut remarquer qu'elles dérivent toutes du langage standard XML, normalisé en 1998, et font largement appel aux spécifications associées XML Namespaces et Schéma XML.

La standardisation de ces technologies est placée sous le contrôle de trois organisations :

- ✓ le W3C (*World Wide Web Consortium*) ;
- ✓ l'OASIS (*Organization for the Advancement of Structured Information Standards*) ;
- ✓ le WS-I (*Services Web Interoperability Organization*).

Cette dernière organisation n'est pas, à proprement parler, chargée de normalisation de ces technologies, son rôle consiste à faire des recommandations sur l'usage des standards édictés par les deux premières organisations de manière à assurer l'interopérabilité des différentes implémentations de ces standards et spécifications.

5.1.2 Conditions d'emploi

Le volume de spécifications associées aux technologies de Services Web est conséquent. Cependant, certaines sont très récentes et n'ont pas encore atteint le niveau de maturité des spécifications de base.

Que faut-il en penser ? Que peut-on déjà utiliser ?

D'une manière générale, on peut classer les problématiques posées par les spécifications de la manière qui suit :

- ✓ problématiques résolues (spécifications du noyau des technologies de Services Web) :
 - transport des messages et invocation des services : **SOAP** ;
 - description des modèles de services et des messages échangés : **WSDL** ;
 - publication des modèles et localisation des points d'accès : **UDDI**.
 - nombreuses implémentations complètes – Interopérabilité sous contrôle du WS-I (*Basic Profile*)
- ✓ problématiques presque résolues (spécifications d'infrastructures) :
 - échanges de messages sécurisés : **WS-Security** (implémentations publiques incomplètes) ;
 - gestion des processus métier : **BPEL4WS** (implémentations publiques incomplètes).
 - quelques implémentations plus ou moins complètes – Interopérabilité non contrôlée par le WS-I (groupe de travail Basic Security Profile créé – Pas de groupe de travail créé pour les processus métier)
- ✓ problématiques partiellement résolues (spécifications d'infrastructures) :
 - échanges de messages fiables : **WS-ReliableMessaging** ou **WS-Reliability** ? (implémentations privées indisponibles) ;
 - gestion des transactions : **WS-Coordination/WS-Transaction** ou **WS-CAF** ? (implémentations privées indisponibles).
 - rares implémentations – Pas d'accord sur les spécifications – Interopérabilité non

- ✓ contrôlée par le WS-I (aucun groupe de travail créé)
- ✓ problématiques non résolues (spécifications d'infrastructures) :
 - gestion des infrastructures : **OMI, WSMF** ou **WS-Manageability** ?
 - rares implémentations de niveaux très divers – Pas d'accord sur les spécifications, mais toutes sous le contrôle d'une seule entité (OASIS) – Interopérabilité non contrôlée par le WS-I (aucun groupe de travail créé).

Les principales pierres d'achoppement sont donc la gestion de la sécurité, celle de la messagerie fiable et celle de la gestion des transactions, du fait notamment que les principaux éditeurs disposent tous de produits commerciaux qui couvrent ces domaines : problèmes de protection de bases installées, de rapprochements difficiles entre acteurs concurrents, ...

DEVRAIT : Sur ces différents points, il est donc préférable de temporiser, car les problématiques ne sont pas encore mûres et surtout, il n'existe pratiquement pas d'implémentations utilisables.

PEUT : Si cependant, les besoins ne peuvent être différés, il faut garder à l'esprit que des solutions propriétaires sont disponibles auprès des grands éditeurs, tout en sachant que ces solutions seront à court terme obsolètes et remplacées par des solutions réellement interopérables.

5.1.3 Implémentations disponibles

D'une manière générale, pratiquement tous les compartiments de l'offre logicielle sont touchés par l'apparition des technologies de Services Web et sont contraints d'évoluer pour prendre en compte les besoins des nouvelles architectures qui en découlent.

5.1.3.1 WSRP: Web Service for Remote Portals

WSRP est une initiative de l'OASIS et s'appuie sur le travail déjà réalisé du point de vue des liens avec la couche de présentation, notamment sur le plan de l'invocation de Services Web à partir d'un portail :

- ✓ OASIS WSIA *Technical Comitee (Services Web for Interactive Application)*.

Un Service Web « WSRP » s'intègre dans un portail sans programmation et constitue ainsi une forme de *Portlet* distant et standardisé.

Les objectifs sont :

- ✓ l'alignement sur les standards existants ;
- ✓ l'interopérabilité au delà du monde Java (Microsoft .NET, Open Source) ;
- ✓ une intégration la plus « *plug & play* » possible ;
- ✓ définit la notion de « fragment d'information » basées sur les langages HTML, XHTML, VoiceXML ;
- ✓ gère la notion de session et de persistance ;
- ✓ présente un enjeu important pour l'interopérabilité des portails et la standardisation de la mise à disposition d'informations ou de services entre partenaires.

DEVRAIT : La publication d'un service dans un portail ENT est facilitée par la mise en œuvre de ce service sous la forme d'un Service Web « WSRP ». A minima, il convient qu'un service réutilise les fonctions SSO et authentification du portail.

5.1.3.2 WSOA : *Web Service Oriented Architecture*

Une première génération de Services Web est apparue à partir de 1999, utilisée essentiellement en intranet à des fins d'amélioration de l'intégration d'applications patrimoniales (applications du système d'information de gestion 'traditionnel : GRH, paie, ...). D'un point de vue technique, cette génération de Services Web s'est appuyée sur l'utilisation des implémentations des trois spécifications de base que sont SOAP, WSDL et UDDI.

Les environnements capables de supporter ce trio de spécifications de base sont maintenant nombreux et disponibles sur de nombreuses plates-formes technologiques.

Depuis 2001, les éditeurs de ces plates-formes d'exécution et de développement de Services Web, ont commencé à étudier la possibilité de combiner l'usage de plusieurs Services Web entre eux afin de donner la possibilité de créer des processus métier partiels ou complets et d'offrir ainsi des services de plus forte valeur ajoutée.

Ces processus métier, exposés sous forme de Services Web, doivent pouvoir être combinés entre eux ou avec d'autres Services Web, et ainsi de suite... Cette faculté de combinaison récursive est désignée « composition (agrégation) de services ».

Cela permet la création de nouvelles architectures de systèmes d'information orientées services, c'est-à-dire constituées progressivement par le développement et l'agrégation de Services Web jusqu'à implémenter des processus métier complets, capables de prendre en compte l'activité des acteurs internes des entreprises, mais également d'autoriser la mise en place de processus inter-entreprises (concept d'entreprise étendue) ou même à destination du grand public : C'est de l'apparition de cette deuxième génération de Services Web que date la résurgence soudaine de l'acronyme SOA.

Nous utiliserons par la suite l'acronyme WSOA pour désigner les architectures de services implémentées via l'utilisation des technologies de Services Web.

L'agrégation de Services Web sous forme de processus métier s'appuie sur l'utilisation d'implémentations de la spécification BPEL4WS (*Business Process Execution Language for Web Services*).

Quelles sont les nouvelles problématiques apparues avec les architectures WSOA ?

L'apparition des architectures WSOA a fait émerger une série de problématiques nouvelles qui relèvent toutes de la gestion d'infrastructure :

- ✓ comment assurer la sécurité de bout en bout : l'agrégation de services susceptibles d'échanger entre eux des messages, éventuellement à travers l'utilisation de différents protocoles de communication et avec des nœuds de communication qui peuvent être situés à l'extérieur du pare-feu de l'entreprise, nécessite d'utiliser des solutions adaptées. Cette problématique est prise en charge par l'architecture de sécurité proposée par IBM et Microsoft via la spécification WS-Security ;
- ✓ comment assurer la fiabilité des échanges de messages entre services : cette

problématique, pratiquement ignorée par les architectures fondées sur l'utilisation de Services Web de première génération, est devenue critique pour les architectures WSOA. La possibilité de créer des processus métier qui invoquent des Services Web, soit de manière synchrone, soit de manière asynchrone, nécessite de contrôler le bon acheminement des messages émis par les services clients à destination des services prestataires. Pour l'heure, cette problématique est adressée par deux spécifications concurrentes : WS-Reliability et WS-ReliableMessaging ;

- ✓ comment gérer la coordination des Services Web et les transactions : la nécessité de coordonner les échanges entre les Services Web agrégés et, éventuellement, de transactionner les changements d'états effectués durant le déroulement du processus métier a introduit le besoin de disposer de nouveaux outils capables de gérer des transactions distribuées courtes (au sens classique ACID), mais aussi longues. Ces transactions longues ne peuvent plus être traitées de la manière classique (protocole de confirmation/annulation assuré au niveau système), mais doivent être prises en charge au niveau application : l'annulation d'une transaction en échec est traitée via le déroulement d'une transaction dite de « compensation ». A l'heure actuelle, cette problématique est prise en charge par deux spécifications concurrentes : WS-Coordination/WS-Transaction (complémentaires de BPEL4WS) ou WS-CAF (Services Web Composite Application Framework) ;
- ✓ comment gérer l'administration des processus métier : le déploiement d'architectures qui implémentent des processus métier et qui agrègent de multiples Services Web, hébergés sur différentes plates-formes technologiques, et susceptibles d'échanger des messages à travers l'utilisation de divers protocoles de communication (HTTP, FTP, SMTP, ...), a introduit la nécessité de disposer d'une nouvelle génération de systèmes de gestion d'infrastructures distribuées. Là encore, deux spécifications se disputent la standardisation de cette nouvelle catégorie d'outils d'administration : WS-Manageability et WSMF (Services Web Management Framework).

L'ensemble de ces nouvelles problématiques est couvert par des spécifications dont la stabilité est plus ou moins grande selon les domaines et qui ont fait, pour certaines d'entre elles, l'objet d'une soumission à un organisme de standardisation.

Le repositionnement de l'offre logicielle

L'émergence des technologies de Services Web de première génération, et plus encore l'apparition des architectures WSOA a contraint les éditeurs de logiciels à repositionner leurs offres traditionnelles. Cette évolution a été encore accélérée du fait de l'apparition de nouveaux acteurs spécialisés qui ont développé une offre dédiée à ces nouveaux marchés (*pure players*) et se posent ainsi en concurrents directs des acteurs établis.

Les réponses apportées aux nouvelles problématiques par les éditeurs traditionnels diffèrent selon leur positionnement d'origine : serveurs d'applications, logiciels d'EAI, logiciels ERP ou CRM, ... On observe cependant une convergence d'ensemble entre les fonctionnalités apportées par les nouvelles offres quel que soit le positionnement d'origine :

- ✓ on peut noter l'évolution des offres Serveurs d'applications vers des offres Serveurs de Services Web (ou de processus). Les prochaines étapes de l'évolution des offres Serveurs d'applications prendront en charge : le support complet de la « roadmap WS-Security », la généralisation du support des processus métiers transactionnés, le support intégré de la messagerie fiable, l'intégration avec les solutions standardisées de gestion

d'infrastructure.

- ✓ le marché des logiciels EAI/B2B est également fortement impacté. On assiste à l'heure actuelle à l'évolution des offres EAI/B2B traditionnelles vers des architectures ESB (*Enterprise Service Bus* = couplage d'une problématique d'EAI avec une architecture WSOA), notamment sous la pression de nouveaux acteurs, qui converge avec l'évolution des offres serveurs d'application.
- ✓ les ressources des applications patrimoniales (*legacy systems*), généralement servies par des machines de type mainframes (systèmes transactionnels de type IMS, CICS, Tuxedo, ... ou bases de données centrales) sont maintenant exposées, grâce à des solutions d'ores et déjà disponibles sur le marché, sous forme de Services Web à travers des offres propriétaires ou de tierces parties.

5.1.4 Méthodologie sommaire de mise en œuvre

Deux approches sont généralement mises en œuvre selon la situation.

5.1.4.1 l'approche « bottom-up » :

Cette approche consiste en la rétro-conception d'éléments ou composants du patrimoine applicatif de l'entreprise (*legacy systems*), qui sont :

- ✓ des classes (ou interfaces) Java, C# ou C++ ;
- ✓ des composants COM, CORBA ou EJB.

L'obtention de la description WSDL se fait par :

- ✓ inspection du code exécutable ;
- ✓ inspection des descripteurs de déploiement.

Utilisée à des fins d'intégration et/ou restructuration de SI : production de Services Web de première génération

5.1.4.2 l'approche « top-down » :

Cette approche consiste en la production directe de la description WSDL à partir :

- ✓ d'un éditeur WSDL indépendant ;
- ✓ d'un éditeur intégré à un environnement de développement (IDE) ;
- ✓ de la génération de la description WSDL à partir d'un environnement de conception.

Elle peut relever aussi de la reprise d'une description WSDL standardisée par une organisation :

- ✓ sectorielle (ex. : RosettaNet) ;
- ✓ commerciale (ex. : Xignite U.S. *Economical Statistics and Charts*) ;
- ✓ ...

Utilisée à des fins de développement de nouvelles fonctions de SI : production de Services Web de deuxième génération.

5.1.5 Compétences requises

Les compétences requises pour concevoir, développer et déployer des architectures de services (SOA) basées sur les technologies de Services Web sont relativement étendues :

- ✓ connaissances de XML et des technologies dérivées : notamment XML *Namespaces* et Schéma XML qui sont pratiquement utilisées dans toutes les spécifications de Services Web, ainsi que XML Encryption et XML Signature qui sont nécessaires pour la compréhension des spécifications liées à la sécurité ;
- ✓ connaissances bien entendu des spécifications de Services Web : SOAP et WSDL sont les plus utilisées, mais aussi UDDI en cas d'utilisation d'annuaires de services, BPEL pour pouvoir construire des processus métier et les spécifications liées à la sécurité (notamment WS-Security).

La manipulation directe des documents XML utilisés dans le cadre de la mise en œuvre de ces spécifications se trouve de plus en plus occultée, au fur et à mesure de l'apparition d'outils de plus haut niveau et pourrait autoriser un degré plus faible de connaissance de ces spécifications. Cependant, en de nombreuses occasions, il reste encore souvent nécessaire de visualiser les documents XML échangés afin de comprendre un mauvais fonctionnement ou des difficultés de mise au point d'une application.

La connaissance de ces spécifications doit bien sûr être associée à celle de plates-formes logicielles qui les implémentent et qui doivent être choisies en fonction des besoins des projets :

- ✓ moteurs d'exécution SOAP ;
- ✓ moteurs d'exécution BPEL ;
- ✓ annuaires de services UDDI ;
- ✓ logiciels ESB ;
- ✓ ...

Enfin, des connaissances plus classiques restent bien entendu nécessaires pour concevoir, développer et déployer les implémentations des Services Web et de leurs clients. Ces compétences relèvent essentiellement des technologies actuellement utilisées.

5.2 Autres outils

5.2.1 LDAP : *Lightweight Directory Access Protocol*

Autrefois basés sur la norme X500, les annuaires électroniques suivent en général aujourd'hui la norme LDAP (*Lightweight Directory Access Protocol*). LDAP est un protocole basé sur TCP/IP pour dialoguer avec un annuaire électronique. Il permet :

- ✓ de récupérer, modifier et ajouter des entrées dans un annuaire ;
- ✓ de connaître et modifier la structure hiérarchique d'un annuaire.

Le protocole LDAP (v3) utilise un encodage UTF-8 pour tous les caractères non ASCII. Par extension, LDAP qualifie également les annuaires respectant le protocole LDAP.

Les annuaires LDAP sont aujourd'hui largement utilisés pour :

- ✓ l'authentification des utilisateurs, dans les applications et/ou les postes clients ;
- ✓ le stockage d'informations sur les utilisateurs, sous forme d'attributs ;
- ✓ la gestion de groupes d'utilisateurs, sous forme d'attributs multivalués ou de groupes LDAP ;
- ✓ le stockage de tout autre type d'information, sous une forme non relationnelle (par opposition aux bases de données), telles :
 - la téléphonie d'un établissement,
 - les machines clientes d'un réseau,
 - ...

Pivot fonctionnel des ENT, les annuaires électroniques sont en général implémentés de manière redondante pour assurer répartition de charge et/ou tolérance aux fautes. Cette redondance peut être mise en œuvre à l'aide :

- ✓ d'échanges entre réplicas, à l'aide du protocole LDUP (*LDAP Duplication/replication/Update Protocol*), ou bien directement en utilisant le protocole LDAP, ou bien encore à l'aide d'un protocole propriétaire ;
- ✓ d'un *middleware* dédié, interface entre les clients et les annuaires, tels MIIS (Microsoft Identity Integration Server, <http://www.microsoft.com/miis/>).

Dans les annuaires LDAP, les importations et exportations de données sont en général réalisées en utilisant le format LDIF (*LDAP Data Interchange Format*), format textuel utilisant l'encodage Base64 pour les caractères non ASCII.

5.2.1.1 Appui sur un standard ou une norme

En 1988, l'Union Internationale des Télécommunications (UIT) met au point la norme X.500 qui donnera le protocole DAP (*Directory Access Protocol*) :

- ✓ en décembre 1993, l'IETF (*Internet Engineering Task Force*, <http://www.ietf.org/>) donne la structure des chaînes de requête des filtres de recherche LDAP (RFC 1558, remplacé par le RFC 1960 en juin 1996) ;
- ✓ en 1995, l'Université du Michigan crée le premier serveur basé sur LDAP, une version simplifiée de DAP ;
- ✓ en mars 1995, l'IETF normalise le protocole LDAP v1 (RFC 1777, RFC 1823) ;
- ✓ en juin 1996, LDAP v2 (RFC 1959) ;
- ✓ en décembre 1997, LDAP v3 (RFC 2251, RFC 2255) ;
- ✓ les spécifications techniques du format LDIF sont définies dans la RFC 2849.

Un groupe de travail au sein de l'IETF prépare actuellement la standardisation de LDUP.

5.2.1.2 Implémentations disponibles

En open source

Les logiciels OpenLDAP (<http://www.openldap.org/>) sont développés par la communauté Internet sous la licence OPL (*OpenLDAP Public Licence*). OpenLDAP propose une suite d'outils libres permettant de manipuler les annuaires LDAP (serveur, clients, bibliothèques de

programmation ...). OpenLDAP fédère les initiatives du monde libre autour du protocole LDAP, parmi lesquelles on peut citer :

- ✓ JLDAP : une bibliothèque d'accès au annuaire LDAP depuis Java, développée par Novell (<http://www.openldap.org/jldap/>) ;
- ✓ JDBC-LDAP : un connecteur JDBC pour les annuaires LDAP, développé par Octet String (<http://www.openldap/jdbclldap/>).

Sur le marché

La plupart des grands acteurs du marché informatique propose des implémentations de serveurs LDAP. Citons par exemple :

- ✓ IBM Tivoli Directory Server
(<http://www-306.ibm.com/software/tivoli/products/directory-server/>) ;
- ✓ Microsoft Active Directory
(<http://www.microsoft.com/windows2000/technologies/directory/ad/default.asp>) ;
- ✓ Netscape Directory Server ;
- ✓ Novell NDS Directory (<http://www.novell.com/products/edirectory>) ;
- ✓ Oracle Internet Directory (<http://otn.oracle.com/products/oid/content.html>) ;
- ✓ Sun ONE Directory Server
(http://www.sun.com/software/products/directory_srvr/home_directory.html) ;

Notons que la compatibilité de ces solutions commerciales avec le protocole LDAP n'est pas toujours complètement assurée. Il convient, pour s'en assurer, de s'adresser à leurs distributeurs.

5.2.1.3 Recommandations

Comme indiqué dans les recommandations de l'annexe « SupAnn » du SDET (<http://www.cru.fr/ldap/supann/>), les annuaires électroniques DOIVENT être compatibles LDAP (v3).

Les importations/exportations de données vers/depuis les annuaires LDAP DOIVENT utiliser le format LDIF.

Les annuaires LDAP sont nativement optimisés pour la lecture. Pour cette raison, les annuaires LDAP NE DEVRAIENT PAS être utilisés pour stocker des informations non pérennes.

Enfin, pour des raisons de performance, les annuaires LDAP NE DOIVENT PAS être utilisés pour stocker des informations sous forme relationnelle, cet usage doit être réservé aux bases de données.

5.2.2 WEBDAV

WebDAV (*Web-based Distributed Authoring and Versioning*) est un ensemble d'extensions au protocole HTTP permettant d'éditer collaborativement et de gérer des ressources distantes sur un serveur web, il vise à fournir un cadre standard pour la mise à jour de ressources web en utilisant un protocole largement utilisé.

Les extensions de base (RFC 2518) sont implémentées par plusieurs serveurs et clients. D'autres extensions sont d'ores et déjà standardisées (*Versioning*, ACP), ou en cours de définition.

5.2.2.1 Appui sur un standard ou une norme

- ✓ Hypertext Transfer Protocol - HTTP 1.1 IETF standard draft (révision de la RFC 2068) ;
- ✓ RFC 2518 : HTTP Extensions for Distributed Authoring – WEBDAV ;
- ✓ RFC 3253 : Versioning Extensions to WebDAV ;
- ✓ DAV Searching and Locating (DASL) draft ;
- ✓ WebDAV Bindings IETF Internet Draft (rev 2) ;
- ✓ WebDAV Redirect Reference Resources IETF Internet Draft (rev 2) ;
- ✓ WebDAV Ordered Collections Protocol IETF Internet Draft (rev 2) ;
- ✓ Access Control Extensions to WebDAV IETF Internet Draft (rev 7) ;
- ✓ Advanced ACL Extensions to WebDAV IETF Internet Draft (rev 0) ;
- ✓ UUIDs and GUIDs IETF Internet Draft (rev 1).

5.2.2.2 Conditions d'emploi

Utiliser sans modération pour l'accès aux systèmes de stockage.

WebDAV n'a pas (n'aura certainement jamais) les performances observées avec des protocoles standards d'accès au systèmes de fichiers tels NFS ou CIFS, qui resteront certainement prédominants pour les accès au disque depuis un LAN. Il est néanmoins en passe de les remplacer pour tout ce qui concerne l'accès à distance et semble la seule réponse crédible au problème du stockage nomade.

5.2.2.3 Implémentations disponibles

En open source

De nombreuses implémentations libres de serveurs WebDAV existent (<http://www.webdav.org/projects/#opensource>), les deux serveurs les plus communément utilisés sont mod_dav (module d'Apache) et Slide (projet Jakarta).

Des clients existent pour tous les systèmes d'exploitation et tous les langages applicatifs (<http://www.webdav.org/projects/#opensource>). Citons par exemple DAVfs (montage des ressources WebDAV sous Unix),

Sur le marché

La plupart des éditeurs logiciels proposent aujourd'hui l'accès aux ressources WebDAV.

5.2.2.4 Compétences requises

Solide connaissance du protocole http.

Liens utiles : <http://www.webdav.org>

5.2.3 SAML : *Security Assertion Markup Language*

SAML doit permettre la propagation de l'authentification utilisateur entre applications (Single Sign-on) et l'échange d'attributs utilisateur pour des besoins de contrôle d'accès.

SAML définit un format pour échanger des données liées à la sécurité, de 3 types : authentification, attributs et autorisation.

5.2.3.1 *Appui sur un standard ou une norme*

SAML est un standard défini dans le cadre du groupe OASIS (<http://www.oasis-open.org>). La syntaxe de SAML est basée sur XML.

- ✓ SAML V2.0, en cours d'élaboration ;
- ✓ SAML V1.1, octobre 2003 ;
- ✓ SAML V1.0, novembre 2002.

OASIS Security Services TC

- ✓ http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security

Le groupe Liberty Alliance émet des recommandations pour la propagation d'identités entre partenaires qui se base sur SAML.

5.2.3.2 *Conditions d'emploi*

RECOMMANDE : SAML est conçu pour transmettre des données de sécurité au sein d'un organisme (entre les services d'authentification, d'accès aux attributs et de contrôle d'accès) ou avec l'extérieur (utilisateurs venant de l'extérieur ou accès à des ressources distantes).

NON RECOMMANDE : Du point de vue de l'authentification, SAML n'est pas conçu pour transmettre des éléments d'identification (mot de passe,...) mais des assertions d'authentification.

5.2.3.3 *Implémentations disponibles*

En « libre »

OpenSAML est une librairie (Java+C++) SAML, notamment utilisée par Shibboleth. <http://www.opensaml.org/>

Sun IPL (Interoperability Prototype for Liberty), librairie compatible Liberty Alliance. <http://java.sun.com/developer/codesamples/liberty.html>

SourceID est une librairie compatible Liberty Alliance <http://www.sourceid.org>

Shibboleth permet de gérer l'accès à des ressources web entre organisme. Shibboleth délègue l'authentification des utilisateurs à leur organisme d'origine qui fournit à l'organisme gérant la ressource les attributs utilisateur nécessaire au processus de contrôle d'accès. <http://shibboleth.internet2.edu/>

Sur le marché

Sun ONE Identity Server

http://www.sun.com/software/products/identity_srvr/home_identity.html

+ supporte de nombreuses méthodes d'authentification

- nécessite Sun ONE Directory Server

Netegrity SAML Affiliate Agent <http://www.netegrity.com/>

La liste des produits compatibles Liberty Alliance :

http://www.projectliberty.org/index.php/liberty/resource_center

5.3 Les Middlewares

L'informatique centrée sur le réseau utilise un modèle d'architecture client-serveur à plusieurs niveaux où les applications ont entre elles des relations d'échange de données et/ou d'invocation de services. La gestion de ces relations est prise en charge par le *middleware*, c'est à dire une couche logicielle jouant un rôle d'intégration et assurant la communication entre des composants applicatifs.

Le *middleware*, et plus largement l'intégration logicielle, sont donc apparus avec le modèle d'architecture client-serveur et se sont sophistiqués avec lui. Par rapport au modèle OSI (*Open Systems Interconnection*), le *middleware* concerne les couches 5 à 7 (session, présentation et application) et s'appuie sur les couches de communication de bas niveau (transport, réseau, liaison et physique) dont il masque la technicité. Il s'agit d'une couche logicielle jouant le rôle d'intermédiaire entre une application (le client) et une autre (le serveur).

Bien sûr, ces différentes technologies ne répondent pas de la même manière aux besoins de flexibilité et d'évolutivité du système d'information centré sur le réseau. Nous allons donc, dans les chapitres suivants, caractériser ces technologies d'intégration au travers de deux grandes familles – échanges de données d'une part, et invocation de services d'autre part – puis adresser la problématique architecturale qui en découle pour le système d'information pris dans sa globalité.

5.3.1 Middlewares Bases de données

Le problème : chaque SGBD impose sa propre interface, son mode propre de gestion et de stockage des données, sa version de SQL, son éventail spécifique de fonctionnalités.

Toute application cliente devant accéder à deux ou plus SGBD doit :

- ✓ traduire les requêtes et les transferts de données à l'usage de toutes les interfaces ;
- ✓ ajouter un accès à un nouvel SGBD nécessite l'apprentissage d'une nouvelle interface de programmation.

La plupart des SGBD supportent le langage SQL, presque toujours le SQL ANSI ou ISO, mais chaque SGBD propose une mise en œuvre « autochtone ». Ainsi l'ANSI a normé ses vues système mais la majorité des éditeurs de SGBD n'applique pas ces définitions : chaque SGBD est incompatible avec le SGBD voisin. Dans la pratique, le SQL utilisé par chaque SGBD est

incompatible avec celui des autres SGBD (par exemple, les entiers codés sur quatre octets peuvent s'appeler INT, INTEGER ou encore SMALL INT).

5.3.1.1 Appui sur un standard ou une norme

Standard SQL : Il y a deux versions du standard SQL qui toutes deux sont équivalentes et valables pour l'ANSI (*American National Standard Institute*) :

- ✓ ISO/IEC 9075:1992, « Information Technology - Database Languages - SQL » ;
- ✓ ANSI X3.135-1992, « Database Language SQL ».

5.3.1.2 Conditions d'emploi

Il est recommandé d'accéder à une base de données relationnelle au travers d'un *middleware* afin de rendre le code applicatif indépendant de la base de données utilisée : seules les usages standards de SQL seront ainsi autorisés et l'application restera portable.

5.3.1.3 Implémentations disponibles

JDBC (solutions J2SE et J2EE éditeur ou open source) est une API de connexion à des bases de données relationnelles (norme ANSI SQL92). Cette API est un ensemble d'interfaces Java, chaque vendeur de SGBDR va ensuite fournir une implémentation d'un pilote. Un Pilote permet d'assurer la connexion à une SGBDR spécifique. Dans J2SE, cette API est dans le package `java.sql`, elle permet d'ouvrir des connexions, de construire des requêtes SQL, d'appeler des procédures stockées, d'exécuter les requêtes, de gérer les transactions, de manipuler les résultats de ces requêtes.

JDBC répond à la nécessité d'un standard d'ouverture pour les applications Java accédant à des bases de données relationnelles. JDBC est basé sur X/Open SQL CLI (*Call Level Interface*), c'est une API de bas niveau supportant les fonctions de base SQL qui est utilisable avec des pilotes (drivers) supportant au moins la norme ANSI SQL92. JDBC permet de rendre le code applicatif indépendant d'un choix technique de base de données. Il permet de changer de base de données, ou d'utiliser le même code applicatif dans différents contextes avec différents SGBDR, sans impact sur le code applicatif.

ODBC (*Open DataBase Connectivity*) est une interface de programmation applicative (api) unique pour l'accès à des données stockées dans un SGBDR. C'est une architecture articulée autour d'interfaces de bases de données et de normes qui s'attachent à niveler les différences entre les différents SGBD. C'est une solution mono-plate-forme (Microsoft).

5.3.1.4 Compétences requises

Pour la mise en œuvre de ces solutions, quelles qu'elles soient, de fortes compétences informatiques techniques sont nécessaires, une connaissance des produits utilisés est fortement conseillée.

5.3.2 *Middleware* orienté message

Le rôle d'un *middleware* orienté message est de gérer au fil de l'eau des flux de messages synchrones ou asynchrones entre applications. L'utilisation du MOM répond au besoin d'amélioration de la fiabilité et de l'homogénéité des échanges entre applications.

Généralement, il propose différents modèles d'échanges de messages :

- ✓ Le *message passing* (envoi de message) consiste à envoyer un message vers le gestionnaire de messages qui le dépose à la « porte d'entrée » (ou file d'attente en mémoire) de l'application cible. Cette file d'attente est lue par l'application cible qui y recueille les messages qui lui sont destinés.
- ✓ Le *message queuing* (file d'attente) est très proche du modèle précédent sauf qu'il y ajoute la persistance : les messages sont sauvegardés dans une base de données ou sur disque (véritable boîte aux lettres intermédiaire) tant qu'il n'ont pas été lus par l'application cible, y compris après l'arrêt volontaire ou accidentel du serveur de message.
- ✓ Le mode *request-reply* (requête-réponse) où l'application source émet une requête vers une file d'attente *request queue*, puis l'application cible après avoir lu le message émet une réponse sur la file d'attente *reply queue* (ou queue temporaire). Si l'application source et l'application cible sont disponibles au même moment, ce modèle permet des communications synchrones.
- ✓ Enfin, le modèle *publish&subscribe* (abonnement) est destiné à diffuser des informations sur des files de messages (*channels* ou *topics*) vers un ou plusieurs destinataires : les applications abonnées à la file de messages. Dans ce cas, les applications cibles ne sont pas connues de l'application source et le message est lu par toutes les applications cibles (abonnées au *channel*).

Outre ces modèles d'échange de messages, les MOM fournissent aussi une garantie d'intégrité et de délivrance des messages.

Le MOM peut permettre de tracer les messages en leur donnant un identifiant unique, de regrouper les messages dans des groupes de messages ayant un identifiant de groupe, mais aussi de fixer des priorités sur les messages afin de modifier leur ordre de transmission.

Les applications cibles peuvent bien sûr être déclenchées à heure fixe, fonctionner constamment et lire les messages régulièrement, mais elles peuvent aussi être déclenchées par le MOM en fonction de l'arrivée des messages : à l'arrivée du premier message, à l'arrivée de chaque message, à l'arrivée d'un groupe de message, voire même, à la vérification d'une expression logique sur les paramètres du message (c'est la notion de filtrage).

Au niveau de la sécurité, le MOM gère des droits d'accès sur les files d'attentes : les applications sources et cibles doivent s'identifier pour publier ou lire des messages sur une file d'attente. De cette manière, il est possible de garantir qu'aucune application ne puisse accéder à des files dont elle ne connaît pas les autorisations. De manière générale, il n'est pas possible d'accéder aux données à l'intérieur des files d'attentes du MOM sans passer par ce système d'autorisation. Le MOM permet également de gérer la sécurisation des échanges en affectant des autorisations différentes aux applications et en faisant appels à des logiciels tiers pour implémenter le chiffrement et l'authentification à l'émission et à la réception du message.

5.3.2.1 Conditions d'emploi

RECOMMANDE : Particulièrement adapté aux communications asynchrones inter applicatives où on veut garantir la réception du message par son destinataire (ou éventuellement être prévenu qu'il ne l'a pas reçu et en tirer des conséquences).

NON RECOMMANDE : L'emploi dans un contexte de besoin de communication synchrone et transactionnel.

5.3.2.2 Implémentations disponibles

En « libre »

Les implémentations open source de JMS (*Java Messaging Service*), la spécification MOM des plateformes J2EE (L'API JMS permet aux applications Java d'utiliser les services des *middlewares* orientés messages (MOM) en utilisant les modèles *Message queuing et Publish/Subscribe*).

Sur le marché

Le plus célèbre et leader du marché : IBM MQSeries

Ses challengers sont : MSMQ de Microsoft, Rendez-vous de Tibco, ...

Les autres solutions les implémentations éditeurs de JMS (*Java Messaging Service*), la spécification MOM des plateformes J2EE.

5.3.2.3 Compétences requises

Pour la mise en œuvre de ces solutions, quelles qu'elles soient, de fortes compétences informatiques techniques sont nécessaires, une connaissance des produits utilisés est fortement conseillée.

6 Glossaire

Terme	Définition
couplage (fort faible)	Utilisé pour caractériser la relation de deux applications, couplage faible signifie une interdépendance faible...
dématérialisation	Objectif de remplacement des supports d'information physiques par un support électronique
EAI	Enterprise Application Integration, solutions du marché pour intégrer les systèmes d'information
format d'échange	Modèle des informations échangées par deux partenaires dans le contexte d'une coopération
FTP	File Transfer Protocol, protocole de transfert de fichier
HTML	Hyper Text Markup Language, langage de base de communication 'client serveur' d'Internet
HTTP	Hyper Text Transport Protocol, protocole de base de l'Internet
interopérabilité	Capacité de deux applications distinctes, et éventuellement hétérogènes et distantes, à coopérer
LDAP	Lightweight Directory Access Protocol, protocole standard d'annuaire
middleware	Logiciel d'intermédiation pour permettre une interopérabilité applicative
OASIS	Organization for the Advancement of Structured Information Standards
objet métier	Élément conceptuel représentatif et caractéristique d'un métier donné
OMG	Object Management Group, organisme de standardisation pour ce qui concerne la technologie objet (UML par exemple)
plan d'urbanisme	Représentation de la cible système d'information à atteindre
schéma directeur	Programme opérationnel définissant actions et projets permettant d'atteindre la cible définie par le plan d'urbanisme
SGBD (SGBDR)	Système de Gestion de Base de données (Relationnelle)
SMTP	Simple Mail Transfer Protocol, protocole du courrier électronique
SOA	Service Oriented Architecture, architecture de service

UML	Unified Modelling Language, langage de modélisation pour les objets métier et les formats d'échange
urbanisation	Mise en œuvre d'un plan d'urbanisme
urbanisme	Méthodologie de construction d'un plan d'urbanisme
W3C	World Wide Web Consortium, organisme de gestion de l'Internet
WSOA	Service Web Oriented Architecture, architecture de service mettant en œuvre la technologie des Services Web
XML	eXtended Markup Language, langage de la nouvelle génération de l'Internet
XSL	eXtended Stylesheet Language, langage gérant la représentation d'un contenu XML