eProcess

Intégration des processus métiers dans les systèmes d'information d'entreprise

Quand nous parlons de gestion des processus métiers (Business Process Management / BPM) nous considérons la capacité de découvrir, de concevoir, d'utiliser, d'exécuter, d'entrer en interaction, de faire fonctionner, d'optimiser et d'analyser tous les éléments du processus, et de le faire dans le cadre de l'entreprise, non pas au niveau de l'implémentation technique. Pour ce faire, la gestion des processus métiers se base sur une « couche » dite d'automatisation des processus métiers ou Business Process Automation (BPA). Cette couche offre des outils pour modeler, simuler, déployer et faire fonctionner le processus de bout en bout. Ces outils permettent de créer un enchaînement de tâches fusionnant les processus automatisés des systèmes informatiques et manuels impliquant des acteurs humains (workflow). La BPA est lui-même construit sur une autre couche: ***l'intégration des processus métiers****.*

Intégration de processus métiers

L'intégration des processus métiers *(Business Process Integration / BPI)* recouvre la définition et la gestion des échanges d'information d'une entreprise à travers une sémantique processus offrant une meilleure vue d'ensemble qu'une simple analyse du parcours des données échangées. La *BPI* définit comment est gérée une séquence d'événements; ces événements représentent des activités plus ou moins longues devant être complétées pour traiter une tâche faisant partie d'un processus métier. Ces activités consistent surtout à récupérer, à modifier et à actualiser les données des systèmes informatiques.

Afin d'implémenter ces activités, la *BPI* consisterait idéalement à pouvoir définir un modèle graphique du processus métier et à générer automatiquement l'intégration entre modèle, systèmes informatiques et acteurs. Malheureusement, la réalité de la *BPI* est tout autre. La plupart des produits *BPI* sont incapables de générer les parties plus complexes de l'implémentation de l'intégration, telles que la création de connexions aux systèmes informatiques. Néanmoins il est possible de générer à partir d'un modèle de processus métiers des alignements *(mappings)* avec les données que le processus manipulera. Cependant, selon [4], cela ne représente en général pas plus de 50 pour cent du travail nécessaire pour compléter l'intégration. Pour compléter cette intégration, il faut utiliser des technologies telles que le *workflow*, les serveurs d'applications, l'*Enterprise Integration Application (EAI)* et les *Web Services* (Fig. 1).

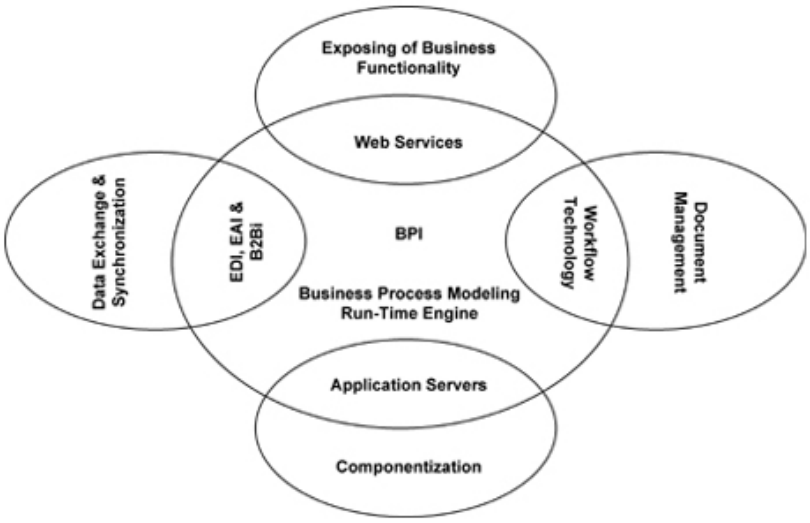

Figure 1 BPI : Relations entre le *workflow,* les serveurs d'applications, l'*EAI* et les *Web Services.*

Dans cet article, nous vous présenterons ces technologies en soulignant la manière dont elles permettent des connexions faciles aux systèmes d'information. Chaque partie débutera par une présentation de la technologie (Par exemple : *EAI*) et ses principaux composants (Par exemple : *Message-Oriented Middleware / MOM*). Ensuite leurs avantages et leurs limites seront présentés. Enfin, nous conclurons par des recommandations concernant leur utilisation pour l'intégration des processus.

Workflow et serveurs d'application

Introduction

Le *workflow* et les serveurs d'applications sont deux composantes importantes pour l'intégration des processus métiers : la première gère la circulation des documents échangés par les différents protagonistes impliqués dans le processus, tandis que les serveurs d'applications fournissent les ressources dont on a besoin pour traiter les processus distribués. Dans cette partie, nous verrons que même s'ils n'ont pas les mêmes buts, le *workflow* et les serveurs d'applications partagent la même évolution.

Workflow

Quand les processus métiers (tels l'envoi/réception d'un formulaire signé) font intervenir des acteurs humains, leur complexité devient importante, étant donné qu'on ne peut pas prévoir le comportement humain avec précision. Pour gérer cette complexité il y a le *workflow*.

Le *workflow* peut être défini (cf. [9]) comme étant « l'automatisation, totale ou partielle, d'un processus métier, dans lequel les tâches passent d'un participant à l'autre pour être effectuées, selon un ensemble de règles procédurales ». Il organise les processus métiers de deux façons : il définit l'ordre dans lequel les tâches doivent être accomplies, et stipule les étapes et les points de contrôle où interviennent des êtres humains. Les solutions *workflow* offrent en général les avantages suivants:

- Reporting
  Ils gèrent les processus en mesurant la durée des processus, leur taux de réussite, etc.
- Identification des problèmes
  Ils fournissent en cas de problèmes un mécanisme d'alerte au gestionnaire du processus.
- Simulation
  Ils offrent un outil de simulation pour aider l'entreprise à prendre des mesures afin de continuellement améliorer ces processus

Mais là où les solutions *workflow* excellent réside leur grande faiblesse : elles sont conçus pour travailler avec des humains (c'est-à-dire avec les documents qu'ils ont créés), non pas avec des systèmes d'information de l'entreprise. En d'autres termes, ils n'ont pas étés conçus pour intégrer des applications à un processus. D'autre part, les éditeurs de logiciels de l'*EAI* se sont concentrés (cf. [8]) sur le transport de données entre les applications (comprenant le développement des applications nécessaires pour envoyer l'information, la recevoir et y réagir) en oubliant de tenir compte de l'aspect humain dans la gestion des processus métiers. C'est une erreur puisque les processus métiers sont un ensemble de processus internes exécutés par des applications et de processus mû par des humains. Cette inadéquation a donc forcé les entreprises à utiliser différents systèmes, souvent incompatibles, pour gérer l'intégration des applications et le *workflow*.

Serveurs d'application

Les serveurs d'applications sont utilisés pour centraliser les processus des applications sur des machines plus puissantes. Ils fournissent des outils et un environnement pour développer et faire usage d'applications distribuées. En outre, les serveurs d'applications deviennent de plus en plus importants pour l'intégration des processus métiers des entreprises, car ils présentent les avantages:

- Permet l'intégration par composant

Cela veut dire que les composants écrits dans des langages différents par différents éditeurs de logiciels sont compatibles s'ils ont la même interface standard telle que *EJB* ou *CORBA*.

- Offre des services relatifs au temps d'exécution

  Ces services comprennent la gestion des transactions, celle des objets, le regroupement des connexions, la gestion des sessions, celle des événements et des pannes et l'équilibre de la charge.

- Plateforme fiable

  Cette plateforme offre des services comme la gestion des transactions particulièrement adaptés ä l'EAI et à ses propres services comme le routage intelligent des messages.

Mais les serveurs d'applications ont tendance à être excessivement complexes à utiliser. La plupart des vendeurs, comme *IBM* avec *Web Sphere,* offrent d'ailleurs une palette de services afin d'aider les entreprises à optimiser leur utilisation. Et, comme c'est le cas pour des solutions *workflow,* il manque aux serveurs d'applications, des fonctions nécessaires à l'intégration d'applications comme la transformation de formats et le routage intelligent des données.

Conclusions

L'intégration des processus métiers implique une gestion des processus dans laquelle interviennent à la fois les utilisateurs et les systèmes. Ainsi, le *workflow* ne peut pas être utilisé comme seule et unique solution. Il doit être combiné à des solutions qui traitent l'intégration des applications. Le contraire est également vrai. Nous remarquons que certains éditeurs de logiciels ont déjà commencé à offrir (cf. [8]) un produit fonctionnel pour *EAI* et le *workflow*.

D'autre part, les serveurs d'applications sont en train de devenir des solutions *EAI* à part entières qui offrent des possibilités d'intégration avec les progiciels, la synchronisation des données à travers des sources de données multiples et la gestion des transactions distribuées. Mais ils ne remplaceront pas une solution *EAI* parce qu'ils n'offrent pas (encore) de transformation de formats des données, ni d'outils graphiques servant à définir le transfert de données ou d'acheminement intelligent, qui sont tous des services offrant des solutions d'intégration flexibles et adaptables.

En d'autres termes, le *workflow* et les serveurs d'applications deviennent de plus en plus les éléments essentiels d'une solution *EAI* à long terme. Dans le paragraphe suivant, nous présenterons les autres éléments essentiels *EAI* nécessaires à une intégration des processus métiers.

Enterprise Application Integration

Introduction

L' Enterprise Application Integration *(EAI)* représente un ensemble de services intégrés qui associent *workflow et middleware* au système de messagerie. Le but de l'*EAI* est d'automatiser le mouvement des données et les processus entre des applications qui n'ont pas été explicitement conçues pour communiquer entre eux.

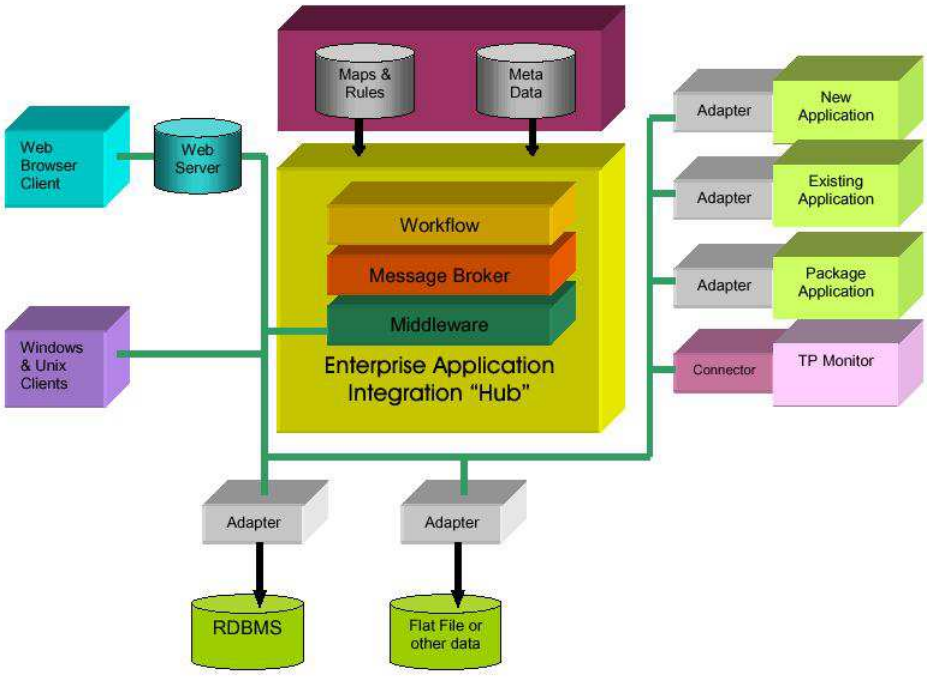

Figure 2 Solution EAI [13]

Au centre d'une solution EAI se trouve le middleware (Fig.2). Les prochains paragraphes présenteront donc les différents composants du middleware- système de messagerie, adaptateurs et connecteurs Ces différents éléments seront analysés dans le cadre de la *BPI,* c'est-à-dire que nous soulignerons la manière dont ils permettent des connexions faciles aux systèmes.

Middleware

Par comparaison aux systèmes d'exploitation et aux services réseaux, les services des middleware permettent à une application, via une API, de localiser de manière transparente à travers le réseau, une autre application. Cela permet aux développeurs de gérer plus facilement la différence entre les plateformes et les bases de données en écrivant des applications qui se connectent à la couche middleware, laissant cette dernière se charger du transport des messages et de leur traduction.

Il existe différents types de middleware:

- Services applicatifs
- Serveurs d'application (cf. section précédente)
- Services distribués
  - Remote Procedure Call (RPCs)
  - Object Request Brokers (ORBs)
  - Message-Oriented Middleware (MOM)

Dans le prochain paragraphe nous allons nous intéresser sur le MOM offrant un couplage moindre que RPC ou ORB (Plus de détails dans [45]).

Message-oriented middleware

Introduction

Définition

Le message-oriented middleware (MOM) fait partie de l'infrastructure client/serveur d'une application. Il permet à cette dernière application d'être distribuée sur des plateformes hétérogènes en cachant au développeur de l'application les détails des différents systèmes d'exploitation et interfaces réseaux. En général, les solutions *MOM* fonctionnent en faisant circuler les messages d'une manière asynchrone (c'est-à-dire que celui qui envoie n'est pas obligé d'attendre la réponse pour le message précédent) d'un programme à un autre ou à plusieurs programmes. Contrairement à *ORB* et *RPC,*

MOM considère que la couche de transport n'est pas fiable, comme cela se produit lorsque les programmes doivent communiquer via un *WAN* ou Internet.

Fonctionnement

MOM est un logiciel qui se trouve dans les deux parties de l'architecture client/serveur et qui supporte les appels asynchrones entre le client et les applications du serveur. Des files d'attente de messages permettent une sauvegarde temporaire quand le programme de destination est occupé ou n'est pas connecté.

MOM et BPI

Nous avons déterminé dans [9] qu'un type particulier de *MOM,* connu sous le nom de *pub/sub* (publier/souscrire), convient le mieux à la *BPI*. Avec *pub/sub,* les programmes souscrivent à un sujet défini par le développeur ([Fig. 3]). Les programmes peuvent aussi publier aussi des messages relatifs à un sujet donné. Une fois qu'un programme a souscrit à un sujet, le programme recevra, en temps réel, tous les messages publiés sur le sujet en question.

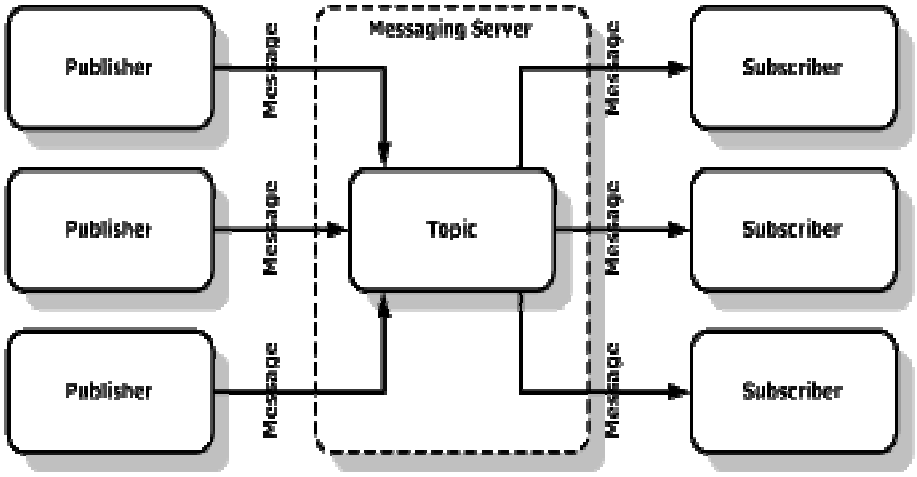

Figure 3 Système d'envoi des messages de type *pub/sub* [15]

Ce système est particulièrement adapté à la *BPI* étant donné qu'il permet la livraison de messages en temps réel à un grand nombre de clients (par exemple: le passage des messages s'est avéré populaire pour créer de grandes applications distribuées). Cela est particulièrement utile dans un cadre de collaboration où l'automatisation du mouvement des données entre les applications et des processus métiers demande une livraison immédiate des messages.

Mais *pub/sub* n'est pas toujours la solution idéale: dans le contexte *EAI* il existe certaines doutes (cf. [21]) sur son efficacité en tant que solution au système d'envoi des messages. Pour cette raison, nous recommandions (dans [45]) que face à une approche 100% *pub/sub* soit aussi étudié d'autres approches (comme le modèle *"hub-and-spoke"*). Un autre problème avec *MOM* est celui de l'interopérabilité. Un *MOM* est généralement implémenté comme un protocole propriétaire de bas niveau, ce qui signifie que l'implémentation est souvent incompatible avec d'autres implémentations *MOM*.

- Pour une meilleure interopérabilité des *MOMs*…

  Une solution intéressante est le Java Message Service (cf. [45]), une API qui permet aux systèmes de messagerie d'interagir avec d'autres systèmes de manière consistante. Néanmoins si les messages échangés en utilisant JMS sont de formats différents, le problème reste entier.

Adaptateurs et Connecteurs

Introduction

Il y a deux façons d'intégrer les applications: une intégration invasive et une intégration non invasive. Une intégration invasive (c'est-à-dire une modification des interfaces) n'est ni souhaitable ni possible avec un grand nombre de systèmes d'origine et de progiciels. Pour cette raison, une intégration non invasive constitue la façon la plus courante d'aborder le problème. Néanmoins, lorsqu'on veut

connecter deux systèmes, l'API du premier système doit au moins être adapté pour convenir au modèle de programmation de l'autre système, et les données doivent être transformées pour être échangées entre les deux modèles de données. Par conséquent, nous avons besoin d'adaptateurs et de connecteurs.

On confond souvent adaptateurs et connecteurs. Le tableau ci-dessous donne au lecteur une idée de leurs différences marquées.

	Adaptateurs	**Connecteurs**
Ce qu'ils sont	Pieces of interface code acting in real time and available off-the-shelf for a number of file formats, DBMSs and packages	Idem
Où les trouver	Dans une solution EAI utilisant MOM.	Dans une solution EAI utilisant un serveur d'application
Ce qu'ils font	Transformation des données : ils accomplissent la traduction de bas niveau des données relatives aux appels et des formats de messages à partir de celle du middleware vers celle de l'application à laquelle les adaptateurs sont connectés.	API mapping: Les connecteurs fournissent une interface permettant de lier l'API du système à l'API utiliser par le système désirant se connecter à ce système
Relations entre adaptateurs et connecteurs	Comme un adaptateur accède à un système via un connecteur on peut dire qu'un adaptateur contient un connecteur et de ce fait effectue un alignement (*mapping*) d'API.	

Table 1 Comparaison adaptateurs - connecteurs [16 - 20]

Avantages et limitations

Dans la plupart des solutions *EAI*, il existe des adaptateurs et des connecteurs pour les applications les plus couramment utilisés en entreprise telles que celles de *SAP, Siebel* et *Oracle*. Parce que ces connecteurs sont pré-installés, ils permettent aux applications d'être connectées rapidement, nécessitant ainsi un effort de développement minime.

Mais tous les adaptateurs/connecteurs ne peuvent pas être pré-installés pour tous les systèmes disponibles : les entreprises doivent souvent développer leurs propres adaptateurs et connecteurs. Pour ce faire, elles auront peut-être besoin de concevoir l'application de façon à ce qu'elle puisse gérer elle-même les services comme la sécurité, les transactions, la gestion des connexions, etc. Cela prend beaucoup de temps, coûte cher et en conséquence le travail est souvent bâclé.

- Vers un développement plus facile des adaptateurs et des connecteurs …

  La J2EE Connector Architecture (présentée dans [45]) regroupe une ensemble de spécifications Java décrivant des interfaces simplifiant l'intégration avec les systèmes d'informations usuels (ERP, CRM,etc.) avec les applications basées sur J2EE uniquement..

Conclusions

Dans cette section nous avons vu qu'une sélection faite avec soin des composants principaux d'une solution *EAI* pouvait conduire à un couplage inférieur lors de l'intégration entre systèmes. Néanmoins, nous remarquons la difficulté et le temps qu'il faut pour mettre en place les systèmes de messagerie et développer les adaptateurs d'applications requis. De nos jours, les entreprises qui utilisent les solutions *EAI* le font conçues pour lier durablement, via des connexions discrètes, spécifiées au préalable, des applications souvent monolithiques. Mais pour la gestion des processus métiers, les entreprises ont besoin de technologies qui soient très flexibles et capables d'intégrer rapidement, des systèmes à travers toutes sortes de barrières technologiques, quelle que soit la durée d'existence des liens. Pour cette raison, dans le paragraphe suivant, nous étudierons les *Web Services,* une collection de standards basés sur *XML,* fournissant un moyen modulaire et dynamique de transmettre l'information entre les applications.

Web Services

Définition

« *Web Services* » est une expression peu précise. Pour cet article, nous proposons la définition suivante (dérivée de [21] et [22]) :

Les *Web Services (WS)* sont des composants logiciels qui peuvent être décrits, publiés, localisés et invoqués sur un réseau.
En d'autres termes, les *WS* procurent une méthode standardisée (utilisant des formats *XML*) pour publier et souscrire à des services logiciels via des protocoles Web tels que *HTTP*. On peut considérer la technologie *WS* comme étant l'évolution d'une technologie d'intégration telle que *CORBA* (Fig. 4).

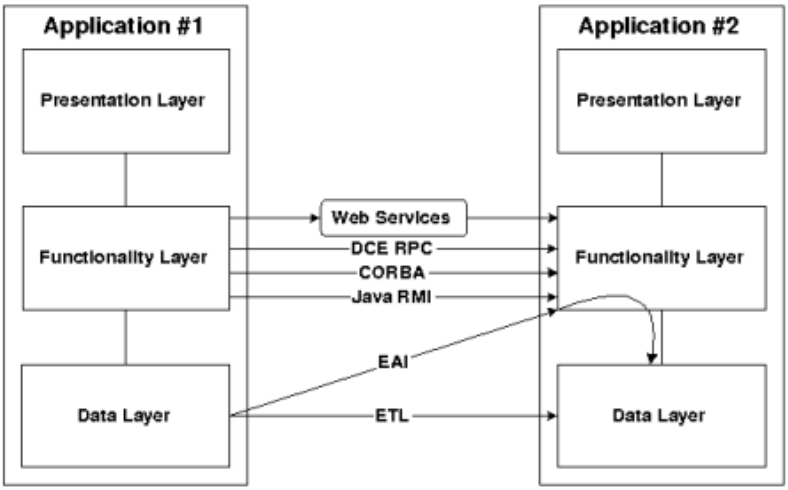

Figure 4 Principales méthodes d'intégration inter applicatives [23]

Les spécifications Web Services

Bien plus qu'une nouvelle technologie d'intégration, les *WS* représentent une pile de spécifications émergentes (Fig. 5) dont le but est de définir une architecture modulaire orientée services.

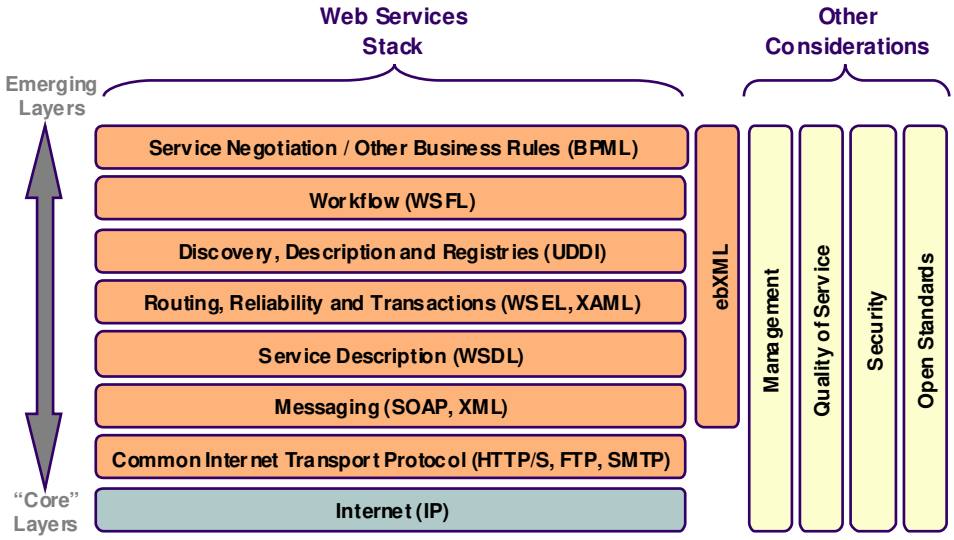

Figure 5 Les spécifications Web services [24]

Au bas de la pile se trouvent les spécifications définissant l'infrastructure permettant un accès dynamique aux *WS* ; par exemple: comment les définir *(WSDL),* comment les enregistrer et les localiser *(UDDI),* comment les invoquer et communiquer avec eux *(SOAP)*.

- **SOAP** est un protocole de type *RPC* utilisant *XML* pour définir ces messages et les envoie utilisant les protocoles standard Internet tels que *HTTP*. C'est une manière simple d'échanger des données dans un environnement distribué tel que le Web. Elle permet à deux points finaux de se connecter préalablement à la demande d'utilisation d'un service. Le message *SOAP* est codé en *XML* et transmis en utilisant *HTTP*. Il transporte les données échangées entre les deux objets éloignés et les données nécessaires à l'exécution du service sollicité : nom de la méthode, type de données, etc.
- Avec **WSDL** (WS Definition Language), les services (par exemple, les objets dans *Java, C++* ou *COM*), peuvent être décrits de façon à ce que leurs futurs utilisateurs sachent comment les activer et où les trouver. Par analogie avec CORBA WSDL correspond la définition de l'interface (IDL).
- **UDDI** (Universal Description, Discovery and Integration) vient compléter l'ensemble de ces technologies en fournissant un répertoire (public ou privé) où les prestataires peuvent enregistrer leurs services et les utilisateurs peuvent venir les chercher.

Plus haut dans la pile se trouvent les spécifications dont le but est de fournir une fonctionnalité ajoutant un contexte et des objectifs aux spécifications présentées plus haut. Par exemple, les spécifications pour un modèle de processus métiers seront utilisées dans un système de gestion des processus métiers pour faire en sorte que les processus (publiés comme *Web Services*) puissent être orchestrés de façon cohérente. Mais, contrairement aux spécifications essentielles qui sont presque considérées comme étant standards, il existe des approches divergentes (comme *Business Process Modelling Language / BPML* de BPMi.org en concurrence avec *Business Process Execution Language for Web Services (BPEL4WS)* de *IBM* et *Microsoft)*.

Avantages

Soutenus dans son développement par la quasi-totalité de l'industrie du logiciel, indépendants de tout langage de programmation ou systèmes, les *Web Services* permettent de promouvoir l'utilisation d'une intégration centrée sur les services, où les fonctions automatiques sont mises à la disposition de tous les utilisateurs éventuels, quelles que soient la technologie et les spécifications interfaces qu'ils utilisent. Les *Web Services* ont en outre les avantages suivants:

- HTTP
  Les WS utilisent HTTP comme protocole de transport. Ils profitent donc de ses avantages:
  - Une architecture opérationnelle
  - Une infrastructure simple qui a fait ses preuves
  - Sûr and scalable
  - Standard
  - Offre des performances "acceptables" (*best effort*)
  - Compatible avec les *parefeux*
- Avantages des WS sur les architectures de type RPC
  Fortement inspiré des précédents modèles informatiques d'architecture modulaire, comme *CORBA ou RPC*, les WS ont d'importants avantages par rapport à ces derniers:
  - Les *WS* supportent non seulement le système synchrone d'envoi de messages de *RPC,* mais également le système asynchrone, rendant ainsi possible une architecture asynchrone couplée sans rigidité, adaptée aux transactions commerciales durables, qui peut encore fonctionner sans réponse immédiate du réseau.
  - Un avantage majeur des *WS* sur *CORBA* est que les *WS* font l'économie de l'intégration complexe d'un *ORB (Objet Request Broker)*.
- Avantages des WS sur les solutions de messageries EAI classiques
  Les *WS* évitent le principal problème des messageries EAI, à savoir leur dépendance vis-à-vis de solutions propriétaires. Ils le font en se basant sur des standards comme *HTTP, XML* et *WDSL*. Les solutions *MOM,* au contraire, sont par exemple souvent liées à la plateforme de l'éditeur de logiciels. Même *JMS,* qui fournit une façon standard d'accéder à la fonctionnalité *MOM,* ne résout pas le problème de l'incompatibilité des formats de messages.

Limitations

Le développement des *Web Services* n'a commencé qu'il y a deux ans. C'est ce qui explique qu'un grand nombre de questions importantes n'aient pas encore été résolu ce qui laisse même douter qu'elles le soient un jour. C'est pourquoi nous examinerons ci-dessous ces questions afin de déterminer leur réelle importance et présenterons quelles sont les réponses possibles.

- Sécurité

  Si certains standards de facto sont déjà disponibles pour une sécurité au niveau transport (par exemple *SSL*) et au niveau des messages (par exemple *SMIME*), les entreprises qui utiliseront les *Web Services* extérieurs comme des « boîtes noires » dans leurs processus auront besoin d'établir des rapports de confiance avec leurs prestataires. A ce niveau il y a encore du travail à faire, notamment au niveau corps de message (*payload*) et au niveau processus. Fort heureusement, de nombreuses propositions ont été faites pour assurer la sécurité des *WS*, mais ces efforts ont eu pour résultat un grand nombre de spécifications concurrentes *(XACML, XKMS. SAML, WS-Security, HTTP-R...).* Ajouté à cette confusion il y a le fait que ces initiatives sont loin d'être parfaites : par exemple, *XACML* et *SAML* ne suivent pas les directives classiques en matière de sécurité et *XML Encryption* et *XKMS* répondent aux mêmes problèmes mais de deux manières différentes. Ce manque de sécurité constitue l'obstacle principal à l'adoption générale des *Web Services*. En conséquence, ils devraient tout d'abord être déployés à l'intérieur du domaine protégé par un pare-feu pour éviter tout problème de sécurité.

- Performance & montée en charge

  Les *Web Services,* tout comme les autres technologies basées Internet, la fiabilité et la disponibilité des connexions, les performances des serveurs et le trafic sont des préoccupations importantes. En particulier nous craignons que leur protocole de transport *SOAP* qui utilise un format synchrone de requêtes ne soit pas adapté aux applications industrielles qui mobilisent un nombre important de ressources sur des périodes plus ou moins longues. De plus, dans les cas où différentes applications utilisent le même *Web Service,* il y a un risque que toute défaillance de ce *Web Service* puisse à son tour avoir pour résultat un mauvais fonctionnement de ces applications. Nous conseillons de s'assurer que les prestataires de *Web Services* puissent être capable de gérer la montée en charge de leur infrastructure au fur et à mesure que la popularité de leurs *Web Services* s'accroît.

- Interopérabilité (SOAP)

  Un certain nombre d'implémentations *SOAP* risquent de ne pas fonctionner les unes avec les autres parce que ces implémentations ont été conçues pour utiliser des logiciels de développement *SOAP* différents. Il est en effet possible que ces logiciels implémentent *SOAP* de manière différente, créant ainsi des messages *SOAP* légèrement différents (voir l'exemple ci-dessous).

  - Exemple: Implémentation de *SOAP* utilisant *Java* par opposition à celle de *Microsoft .NET*
    Quand il s'agit de chiffres, les implémentations *Microsoft .NET* et *Java de SOAP* diffèrent. Avec *.NET* vous pouvez utiliser des chiffres jusqu'à la 29ème décimale, tandis qu'avec *Java* vous ne pouvez aller que jusqu'à la 18ème décimale.

  Bien que cette question soit loin d'être spécifique aux *WS,* elle devient importante pour une collaboration *B2B* où l'interopérabilité est cruciale. Malheureusement, la seule solution semble être celle qui consiste à essayer différentes implémentations *SOAP* afin d'être préparé aux manoeuvres requises pour assurer l'interopérabilité dans le futur.

- Gestion des transactions

  La gestion des transactions est cruciale lors qu'on intègre des processus métiers puis que ces derniers comprennent une multitude d'activités pouvant s'échelonner sur de longues périodes de temps. Mais, à l'heure actuelle, ces mécanismes (comme, par ex., les transactions compensatoires) ne sont pas disponibles pour les *Web Services*. En attendant que les spécifications proposées *(Transaction Authority Markup Language/XAML, Business Transaction Protocol)* soient finalisées, nous recommandons (cf. [45]) d'éviter les solutions propriétaires de gestion de transactions. Elles peuvent fonctionner à l'intérieur d'une chaîne de valeur bien définie, où l'on peut passer des contrats propriétaires ciblant une industrie particulière, mais risquent d'être contre-productives dans un réseau de collaboration où les partenaires appartiennent souvent à une autre industrie.

Conclusions

Les *Web Services* permettent de développer de manière standard des connexions aux applications sans se soucier de leurs spécificités. C'est exactement ce dont nous avons besoin pour la *BPI.* Mais nous avons vu que des questions importantes (sécurité, gestion des transactions) restent encore sans réponses. Et, malgré toute l'attention consacrée par l'industrie à ces carences, nous considérons qu'il faudra un certain temps (de 2 à 3 ans) avant que les *Web Services* ne deviennent le moyen le plus efficace pour intégrer les applications et suffisamment stable pour servir la *BPI.*

Conclusions finales

Lorsque les entreprises décident d'automatiser la gestion de leurs processus, elles doivent tout d'abord s'assurer qu'elles peuvent gérer l'échange de l'information entre entreprises à travers une sémantique processus métiers. Pour partager cette information, elles doivent "brancher" ces processus dans leurs systèmes et ceux de leurs partenaires tout en permettant la participation de protagonistes humains.

Étant donné que les solutions *EAI* traditionnelles ne se préoccupent que des processus des systèmes informatiques, négligeant la participation humaine, les entreprises ont besoin de rechercher une solution qui fournisse une intégration plus proche du *workflow* et de la solution *EAI.* Cette solution elle-même exige une plateforme capable de gérer la nouvelle complexité. Un serveur d'applications pourrait être cette plateforme, puisqu'il est robuste et propose des services utiles pour une collaboration *B2B* tels que la gestion des transactions et la traduction du format de données. Mais ce nouvel *EAI* risque comme son aîné d'être trop dépendant d'être trop lié aux produits et trop complexe à implémenter pour la gestion des processus métiers, en comparaison de la toute nouvelle technologie d'intégration, *Web Services.* Contrairement à l'*EAI,* l'approche modulaire proposée par les *Web Services* promet de fournir un moyen standardisé de gérer l'intégration permettant ainsi aux entreprises de se lancer dans une collaboration *B2B* sans avoir à se préoccuper des spécificités des systèmes de leurs partenaires. Mais les *Web Services* sont une technologie encore immature notamment dans le domaine de la sécurité et de la gestion des transactions.

Pour conclure, nous pensons que les technologies utilisées pour l'intégration des processus métiers vont dépendre des exigences des applications. Par exemple de gros volumes demanderaient une solution *EAI* sur mesure afin d'optimiser le temps d'exécution. Mais s'il faut une certaine flexibilité (et c'est souvent le cas avec la *BPI*), alors l'approche des *Web Services* nous semble plus bénéfique à long terme car elle offre un développement plus rapide avec un couplage moindre.▲

Alexandre Giess, Wilson Lau, Siham Sinaceur
France Telecom R&D San Francisco
siham.sinaceur@rd.francetelecom.com

Bibliographie

[1] Emergence of Business Process Management. *CSC Research Services*, 2002

[2] Siham Sinaceur. eProcess project presentation. *France Telecom R&D*, 2001

[3] Michael Aubin. Simplifying the Integration Market: BPI Is Here to Stay. *Metaserver*, 2002

[4] Andre Yee. Demystifying Business Process Integration. *Saga Software*

[5] The Role of the Enterprise Repository in Enterprise Application Integration. *Rochade*, 2000

[6] Eric Sanchez, Dan Beery, Joshua Shehab. EAI Systems Bring IT Together. *Network Computing*, 2002

[7] David McGoveran. Architected Scalability. *EAI Joumal*

[8] Sue Hildreth. The Marriage of EAI and BPM: A conversation with Gartner's David McCoy. *ebizQ.net*

[9] John Mann. Workflow and Enterprise Application Integration

[10] Rashid Kahn. A Perfect Marriage: Workflow and EAI. 2002

[11] IBM. Enterprise Integration with IBM Connectors and Adapters. 2002

[12] Jane Stanhope. J2EE Connector Architecture Promises to Simplify Connection to Back-End Systems. *Giga Group*, 2000

[13] JCA: vers la standardisation des connexions inter-applications (Parties 1 et 2). *Journal du Net*, 2002

[14] Theo Stolker. Why you need to look at the J2EE Connector architecture in 2001. *WRQ*, 2001

[15] Middleware. *Carnegie Mellon Software Engineering Institute*

[16] Everything You Need to Know About Middleware. *Talarian*

[17] Message-Oriented Middleware. *Carnegie Mellon Software Engineering Institute*

[18] Barry Nance. MOM Implementation Issues

[19] Martin Erzberger, Marcel Altherr. Every dad needs a MOM. *Softwired*, 1999

[20] Linking software with message oriented middleware. *Infochain*

[21] Paul Timberlake. The Pitfalls of Using Multicast Publish/Subscribe for EAI. *Promenix*, 2002

[22] Johanna Ambrosio. Interoperability: You can count on MOM for message delivery. *Techtarget.com*, 2001

[23] Kurt Lingel. Security Requirements for Message-Oriented Middleware. *eAI Journal*, 2001

[24] Gopalan Suresh Raj. Java Message Service (JMS).

[25] Michael Hudson, Craig Miller. IT and the NOW Economy.2001

[26] R.Schulte. Java Message service: A standard that works. *Gartner*, 2001

[27] John Rymer. Is Your Application Server Ready for 2002? *IONA,* 2002

[28] Pat Coleman. App Servers and EAI Vendors Duke It Out.

[29] Beth Gold-Bernstein. Defining EAI Technologies and Solutions.

[30] OASIS Business Transactions Technical Committee

[31] Sam Wong. Web Services: The Next Evolution of Application Integration. *Patkai Network*, 2001

[32] Erick Schonfeld. What the *&%@!! Are Web Services? (And Why You Should Care.). *Business 2.0*, 2002

[33] Carla King. Getting Started on Developing Web Services. *Sun Microsystems*, 2001

[34] Itamar Ankorion. Revolutionizing Process Automation with Web Services. *Attunity*, 2002

[36] John Fontana. Top WS worry: Security. *Network World*, 2002

[37] Edd Dumbill. XML on the Move. 2001

[38] Steven Nicks. WS and Telco: A Milestone on the Road to IP Convergent Intermediation, Vol. 1: An introduction to WS. *France Telecom R&D*, 2001

[39] www.webservices.org

[40] Web Services: Protocol and Vendor Report. *France Telecom R&D*, 2001

[41] John Hagel, John Seely Brown, Dennis Layton-Rodin. Go Slowly with Web Services. *CIO Magazine*, 2002

[42] Understanding how SOAP works *Preston Gralla, searchwebservices.com*, 2002

[43] Peter Bals. Web Services and the Move to Loosely Coupled Computing. *Glue Technology*, 2001

[44] Patrick Gannon. OASIS & ebXML: The Building Blocks for eBusiness Web Services. 2002

[45] Alexandre Giess. Integration of business process in back-end systems. *France Telecom R&D*, 2002