



Université de Fribourg, Suisse
Département d'informatique
Bachelor en informatique de gestion

TRAVAIL DE BACHELOR

Sujet:

*Conception et développement d'un Service Web
Pour l'échange d'information
dans
le domaine humanitaire*

Présenté par:

Bouchahma Karim
Chemin des Epinettes 53
1723 Marly
Suisse

Encadré par :

Dr. Stefan Hüsemann

À MES PARENTS

À MES SŒURS

À MES PROCHES

À TOUS MES AMIS

KARIM

RÉSUMÉ

Dans le cadre de ce travail de Bachelor, l'auteur essaie de relier des sujets traités au cours de Système d'informations du professeur S. Hüsemann avec un projet personnel, à savoir celui d'implémenter un Service Web. Ce travail s'articule en deux parties. Une première partie théorique traite le Service Web. Ensuite, le document évolue vers l'analyse et la compréhension des Services Web. Cette partie décrit les caractéristiques, la technologie ainsi que l'architecture d'un Service Web.

Dans la deuxième phase de ce travail, l'auteur expose et explicite le "Development Information Exchange System" DIES. En utilisant des documents XML le Service Web « DIES_Service » facilite l'échange de données entre systèmes d'information autonomes hétérogènes. Cette partie permet de réaliser un Service Web tout en se concentrant sur sa conception et son implémentation. Le projet se termine en évaluant le programme en utilisant la méthode SWOT afin de démontrer les forces et faiblesses d'un tel système.

ABSTRACT

During this bachelor thesis, the author tries to combine subjects learned at the class of Information systems thought by Professor S. Hüsemann with an own project of implementing a web service. This paper consists of two parts. The first theoretical part deals with the Web Service. The document then continues with the analysis and comprehension of web services. This part describes the characteristics, technologies as well as the architecture of such a web service.

In the second phase of this paper, the author exposes and explains what a "Development Information Exchange System", shortly DIES, is. Using XML files, the web service « DIES_Service » helps the exchange of information among different heterogeneous information systems. This section of the paper consists of creating a web service while concentrating on its conception and implementation. The project will then be finished by evaluating the program using a SWOT method in order to illustrate its strengths and weaknesses.

Keywords:

DIES, XML files, Service Web, First Contract Development, WSDL, UDDI, SOAP, information sharing,

Remerciements

Au terme de ce projet, je tiens à exprimer mes reconnaissances à Mr. Hüsemann Stefan pour son soutien, son assistance, sa disponibilité et pour ses précieux conseils.

Je tiens à remercier profondément ma famille et mes proches pour leurs conseils et l'aide qu'ils m'ont accordé.

Je tiens également à remercier tous mes amis pour leurs encouragements et leurs soutiens qu'ils m'ont fourni avant l'aboutissement de mon projet de fin d'études.

Je tiens à remercier tous mes professeurs à l'université pour la formation qu'il m'on apporté tout au long de mes études.

Je ne manquerai pas de remercier vivement tous ceux qui m'ont encouragé à aller jusqu'au bout.

Sommaire

I	Introduction	1
1	Problématique du sujet	2
2	Objectif du travail.....	3
3	Questions traitées et manière de procéder.....	3
II	Partie Théorique : Services Web	5
1	Introduction	6
2	Définition des Services Web	6
3	Principe du fonctionnement des Services Web	7
4	Caractéristiques des Services Web.....	8
4.1	Interopérabilité	8
4.2	Simplicité d'utilisation	8
4.3	Couplage souple des applications	8
5	Technologies liées aux services Web.....	9
5.1	XML : eXtensible Markup Language	9
5.2	Triptyque SOAP/WSDL/UDDI	11
5.3	WS-BPEL.....	21
6	Architecture	22
7	RPC vs Doc Style.....	23
8	Conclusion.....	26
III	Partie Pratique : Services Web	27
1	Introduction	28
2	DIES	28
2.1	Architecture du Development Information Exchange System (DIES).....	29
2.2	Objectif du DIES	30
2.3	Bénéfices du Service Web pour DIES	31
3	Implémentation.....	32
3.2	Contract First Development :	37
3.3	WSDL et WebMethod :.....	38
3.4	Service Consumer : utilisation du Service Web « DIES_Service»	44
4	Evaluation.....	49
4.1	Exigences à l'égard de DIES_Service.....	49
4.2	Forces et Faiblesses	50
4.3	Opportunités et Menaces	50

IV	Conclusions	52
V	Références	55
VI	Annexes	60
1	Annexe A : Code source de SearchDoc	61
2	Annexe B : Code source de AffichDoc	66
3	Annexe C : Code Source de Transform	68
4	Annexe D : WSDL	69

Liste de figures

Figure II.1 Principe général du fonctionnement des Services Web	7
Figure II.2 Les relations entre les éléments du triptyque SOAP/WSDL/UDDI	11
Figure II.3 Forme générale d'une requête SOAP	12
Figure II.4 Architecture du système d'information	22
Figure III.1 Architecture du DIES (Hüsemann 2005).....	29
Figure III.2 Diagramme de cas d'utilisation (propre élaboration).....	33
Figure III.3 Description du cas d'utilisation "Search For document" (propre élaboration)	34
Figure III.4 Description du cas d'utilisation "Call Document" (propre élaboration)	35
Figure III.5 Contract First Development [Site 21].....	37
Figure III.7 Requête: DIES_Service SOAPSearchDoc.....	44
Figure III.8 Résultat de la Requête: DIES_Service SOAPSearchDoc.....	46
Figure III.9 Résultat de la Requête: DIES_Service SOAPAffichDoc	47

Liste de Codes

Code II-1 Exemple de requête SOAP [Nicolescu 2004].....	14
Code II-2 Exemple de requête de réponse SOAP [Nicolescu 2004].....	14
Code II-3 Exemple d'une description d'un Service Web	17
Code II-4 Requête SOAP avec Style RPC	24
Code II-5 Message de réponse avec Style Document	24
Code II-6 Message de réponse avec Style RPC	24
Code III-1 Bout de code de la Méthode "SearchDoc"	39
Code III-2 Bout de code de la Méthode "AffichDoc"	41
Code III-3 Bout de code de la Méthode "Transform"	42
Code VI-1 Méthode "SearchDoc"	65
Code VI-2 Méthode "AffichDoc"	67
Code VI-3 Méthode "Transform"	68
Code VI-4 WSDL.....	71

Tableaux

Tableau I-1 Liste d'abréviation.....	1
Tableau II-1 Les technologies liées aux services Web	9

Abréviations

Abréviations	signification
HTTP	Hypertext Transfer Protocol
RPC	Remote Procedure Call
SOAP	Simple Object Access Protocol
UDDI	Universal Description Discovery and Integration
W3C	World.Wide.Web Consortium
WS-BPEL	Web Service-Business Process Execution Language
WSDL	Web Service Description Language
XML	eXtensible Markup Language
XSL	eXtensible Stylesheet Language
SOA	Services Oriented Architecture

Tableau I-1 Liste d'abréviation

I *Introduction*

1 *Problématique du sujet*

Une multitude d'organisations humanitaires internationales tentent d'améliorer les conditions de vie des personnes démunies dans le monde. La coordination de projets de développement humanitaire entre organisations est importante afin d'augmenter l'efficacité de l'aide. Pour coordonner efficacement ces projets, il faut avoir accès à des données qui se trouvent dans des systèmes d'informations de donateurs institutionnels (par exemple la Banque Mondiale ou le Département du Développement et de la Coopération Suisse) et d'organisations humanitaires (comme la Croix Rouge ou Caritas). Ces systèmes sont autonomes et peuvent être très hétérogènes.

Bien que les avantages de partage d'information puissent être assez évidents, l'information n'est pas souvent échangée librement ; et cela, même dans un environnement à but non lucratif, où les objectifs financiers ne sont pas au sommet de la hiérarchie des priorités.

Plusieurs barrières au partage d'information entre organisations ou individus peuvent être trouvées. D'une part il existe des problèmes d'organisation, qui sont liés à la coordination/transparence de projets, la distance géographique, les décisions ; souvent influencées par la politique, la confidentialité de certaines informations ou encore le manque de motivation pour le partage d'information. De plus des problèmes culturels en rapport avec les gens et leurs coutumes peuvent surgir. D'autre part nous trouvons aussi des problèmes techniques liés aux manquements de la télécommunication et de l'infrastructure de la technologie d'information pour les pays en voie de développement, les difficultés d'accès à l'information et enfin l'hétérogénéité des systèmes d'information. [Hüsemann 2002]

2 *Objectif du travail*

L'objectif principal de ce travail est l'élaboration d'un Web Service assurant la communication entre les organisations humanitaires via la plate-forme DIES (Development Information Exchange System) en se basant sur les problèmes de partage et d'échange d'informations entre les institutions, ainsi que les défis techniques quant aux échanges de données entre systèmes.

3 *Questions traitées et manière de procéder*

A notre époque, les développeurs des systèmes d'information optent de plus en plus pour l'intégration des fonctionnalités des systèmes logiciels connus sous le nom des services web [Site 1]. Ceux-ci sont destinés à supporter l'interaction ordinateur à ordinateur sur le réseau.

Au cours de mon travail de Bachelor nous allons traiter différents aspects concernant les Services Web en répondant aux questions suivantes :

1. Qu'est ce qu'un Web Service?
2. Quelles sont les raisons de créer des services Web ?
3. Quelles sont les Standards liées aux Web Services ?
4. Comment fonctionne un Web Service ?
5. Comment implémenter un Web Service ?
6. Quels bénéfices du Web Service pour DIES ?

Nous commençons par une définition d'un Service Web afin de mieux comprendre son phénomène qui est de grande actualité. Par la suite, nous traiterons les raisons de créer des Services Web et analyseront les avantages que ce-dernier nous fournit. Tout au long du travail, nous allons illustrer les différents standards utilisés et leur impact sur le fonctionnement d'un Service Web. Puisque ce système logiciel est destiné à supporter l'interaction d'ordinateur à ordinateur nous allons donc analyser son fonctionnement ainsi que la procédure à son implémentation en choisissant la technologie approprié.

I Introduction

Ce travail abordera, dans un premier temps, l'aspect théorique des Services Web qui concerne le principe de fonctionnement, l'architecture, les caractéristiques ainsi que les différentes technologies utilisées.

Dans la seconde partie, l'aspect pratique sera exposé, affairant à la préparation du cadre applicatif, la conception, l'implémentation logicielle, aux bénéfices du Service Web pour DIES et enfin à l'évaluation.

II *Partie Théorique : Services Web*

1 Introduction

Les services web constituent un phénomène assez récent, il a eu un grand intérêt de la part des développeurs des systèmes d'information. De cela, il est considéré comme étant la solution idéale intégrant des fonctionnalités déjà implémentées afin de construire et développer des systèmes d'information. D'une part, et malgré le succès de cette technologie, les services Web ont confrontés le problème de l'interopérabilité, et pour s'en débarrasser, une jungle de standards a été élaborée, ce qui nous invite à aborder les technologies reliées telles que eXtensible Markup Language (XML) [Site 2], Simple Object Access Protocol (SOAP) [Site 7], Web Services Description Language (WSDL) [Site 8] et Universal Description Discovery and Integration (UDDI) [Site 13].

2 Définition des Services Web

Durant ce chapitre nous allons définir ce qu'est un Service Web. Le groupe de W3C qui travaille sur les services web, a utilisé dans un document appelé « Web Services Architecture » la définition suivante :

« Un Service Web est un système logiciel destiné à supporter l'interaction ordinateur à ordinateur sur le réseau. Il a une interface décrite en un format traitable par l'ordinateur WSDL. Autres systèmes réagissent réciproquement avec le Service Web d'une façon prescrite par sa description en utilisant des messages SOAP, typiquement transmis avec le protocole HTTP et une sérialisation XML, en conjonction avec d'autres standards relatifs au web » [Site 1].

3 **Principe du fonctionnement des Services Web**

Ce chapitre consiste à démontrer le fonctionnement ainsi que les différentes étapes de la mise en place d'un Service Web.

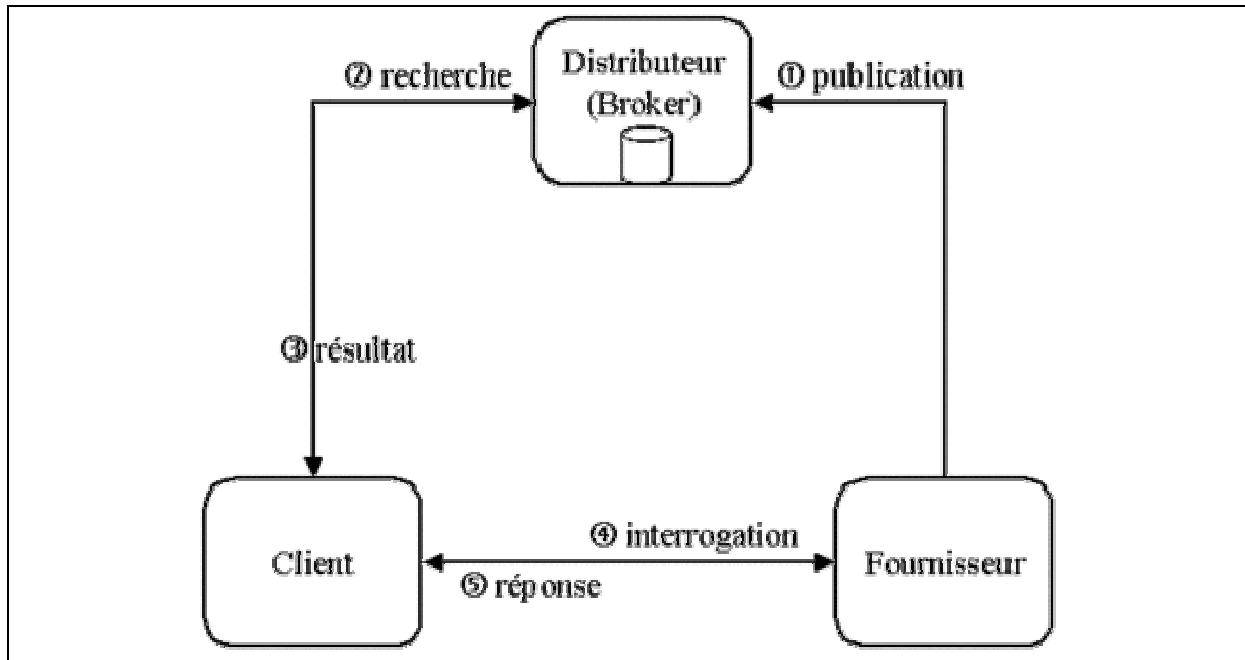


Figure II.1 Principe général du fonctionnement des Services Web

Le schéma de la Figure II.1 ci-dessus, concrétise les grandes phases de la mise en œuvre d'un Service Web. Au préambule, le fournisseur réalise un Service Web ainsi que son interface au format WSDL [Site 12]. (les numéros se réfèrent au Figure ci-dessus)

(1) **Publication d'un Service Web** : le fournisseur commence par enregistrer son Service Web auprès d'un distributeur (sur un annuaire UDDI). Cette opération se fait en envoyant directement un message SOAP via un protocole de transport.

(2) **Recherche d'un Service Web** : le client envoie un message encapsulé dans une enveloppe SOAP via un protocole de transport dont le but est la recherche d'un Service Web auprès du distributeur (annuaire UDDI).

(3) **Résultat de la recherche** : le client recevra la réponse sous forme d'un message WSDL encapsulé dans une enveloppe SOAP.

II Partie Théorique : Services Web

(4) *Interrogation du service au près du fournisseur de service* : Selon le message reçu, le client accèdera directement au Service Web chez le fournisseur. La demande de service s'effectue à l'aide d'un message SOAP via un protocole de transport.

(5) *Réponse et exécution du Service Web* : le client reçoit une réponse (via un protocole de transport) du Service Web sous la forme d'un message SOAP. Le client exploitera directement la réponse.

4 Caractéristiques des Services Web

4.1 Interopérabilité

L'interopérabilité présente la caractéristique principale des Services Web. En effet, quels que soient la plate-forme utilisée (Windows, Unix ou autres) et le langage de développement employé, les Services Web se basent sur des standards XML pour simplifier la construction des systèmes distribués et la coopération entre ces derniers.

4.2 Simplicité d'utilisation

L'utilisation des standards tels que XML et HTTP a mis en valeur la simplicité de la manipulation des Services Web et a encouragé les acteurs forts du marché tels que Microsoft et IBM pour intégrer de nouveaux produits.

4.3 Couplage souple des applications

Les Services Web constituent un support d'échange des documents structurés qui traversent les contrôles d'accès dans un environnement hétérogène. La collaboration entre différentes applications afin d'échanger des documents se fait d'une manière directe entre objets.

5 Technologies liées aux services Web

Ce point nous permet de traiter les différents standards liés aux Services Web. Pour les utiliser, il faut savoir où les trouver, comment y accéder et les utiliser correctement. Pour ce faire, une jungle de standards a été proposée illustrés dans le Tableau II-1.

Acronyme	Désignation	Fonction dans le contexte des services Web
XML	eXtensible Markup Language	Edition des documents
XML Schema	Schéma XML	Définition des types de données lors de la description des services web
WSDL	Web Services Description Language	Description des services web
UDDI	Universal Description, Discovery, and Integration	Publication des services web
SOAP	Simple Object Acces Protocol	Communication entre les services web
OWL-S	Web Ontology Language for Services	Description sémantique des services web
RDF	Ressource Description Framework	Représentation des métas donnés pour les services web
DL	Description Logics	Description formelle des services web

Tableau II-1 Les technologies liées aux services Web

5.1 XML : eXtensible Markup Language

XML « eXtensible Markup Language » est un format de texte simple, très flexible tiré du SGML « Standard Generalized Markup Language » (l'ISO 8879). À l'origine, conçu pour la publication électronique à grande échelle, XML joue aussi un rôle de plus en plus important dans l'échange d'une large variété de données sur le web et ailleurs.

W3C recommande depuis 1998 XML en tant que standard de description de données. XML est un méta langage permettant d'identifier la structure d'un document.

Un document XML est composé d'une structure et d'un contenu. La structure d'un document XML est souvent représentée graphiquement comme un arbre. La racine du document constitue le sujet du document, et les feuilles sont les éléments de ce sujet. De ce fait, XML est alors flexible et extensible, et est devenu rapidement le standard d'échange de données sur le web [Site 2] ».

II Partie Théorique : Services Web

XML a été conçu pour des documents arbitrairement complexes, tout en s'appuyant sur cinq grands principes simples et clairs:

- ✚ lisibilité à la fois par les machines et par les utilisateurs ;
- ✚ définition sans ambiguïté du contenu d'un document ;
- ✚ définition sans ambiguïté de la structure d'un document ;
- ✚ séparation entre documents et relations entre documents ;
- ✚ séparation entre structure du document et présentation du document.

Suite à l'apparition du standard XML, plusieurs autres langages et technologies qui se basent sur ce dernier ont vu le jour, comme XSL [Site 4] (eXtensible Stylesheet Language) qui constituent des feuilles de style XML dont le fonctionnement est semblable à celui des CSS (Cascading Style Sheets) pour le Web [Chauvet 2002], la norme XSLT (XSL transformation) définit le vocabulaire des ces feuilles de style destinées à transformer un document XML en vue de le présenter à son utilisateur ; XPath [Site 5] pour l'accès aux éléments d'un document XML et finalement, OWL [Site 6] pour la description sémantique des données XML.

5.2 Triptyque SOAP/WSDL/UDDI

Le langage WSDL, le protocole SOAP et l'annuaire UDDI représentent les trois éléments clés pour le développement des Services Web. De ce fait, la figure ci-dessous montre les interactions directes et indirectes des Services Web avec ces trois standards.

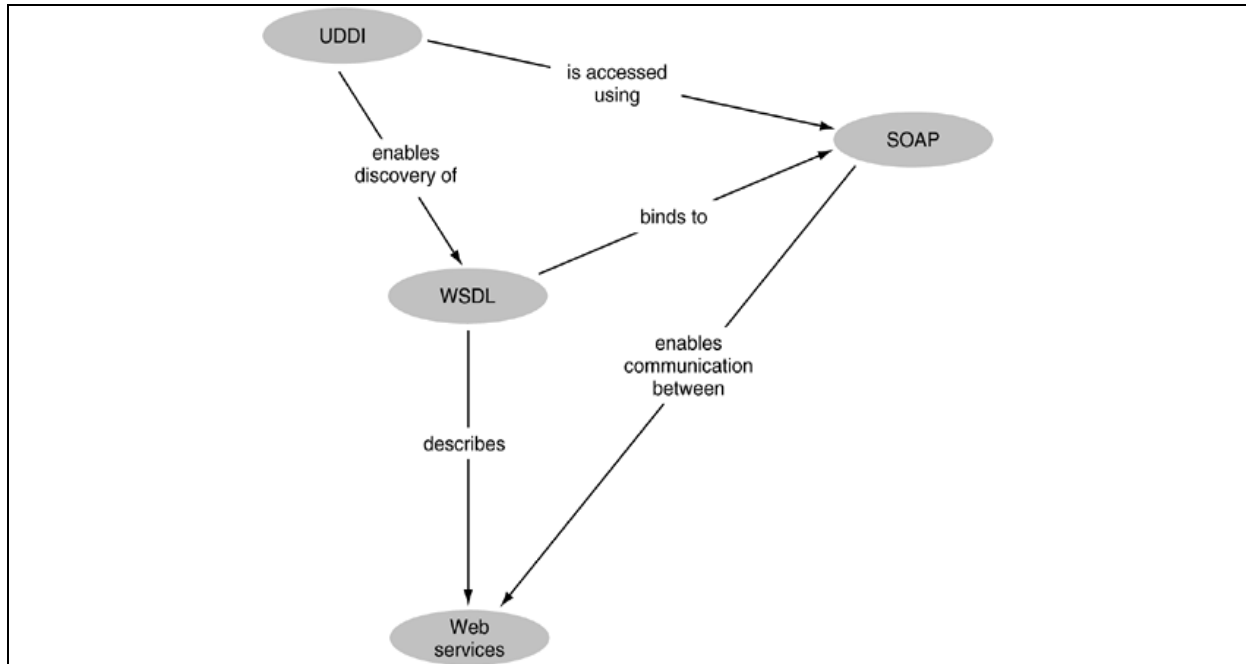


Figure II.2 Les relations entre les éléments du triptyque SOAP/WSDL/UDDI

Les relations observées au niveau de la Figure II.2 montrent d'une part que la description des Services Web est assurée via le langage WSDL [Site 8]. D'autre part, la communication entre les Services Web est garantie par le protocole SOAP [Site 7] et les descriptions des services sont liées via SOAP. En effet, l'annuaire UDDI [Site 9] permet la découverte des Services Web via leurs descriptions et l'accès au catalogue UDDI est assuré par le protocole SOAP.

5.2.1 SOAP : Simple Object Access Protocol

Le World Wide Web Consortium a défini le SOAP comme étant : «Un protocole léger destiné à l'échange d'informations structurées dans un environnement décentralisé et distribué. Il utilise des technologies XML pour définir une structure d'échange de messages fournissant une construction de messages pouvant être échangés sur divers protocoles sous-jacents. La structure a été conçue pour être indépendante de tout modèle de programmation et autres sémantiques spécifiques d'implémentation.» [Site 7].

SOAP a été proposé au consortium W3C en 2000 par UserLand, Ariba, Commerce One, Compaq, Developer, HP, IBM, IONA, Lotus, Microsoft et SAP. Ce protocole de transmission de messages a été employé pour les services web afin d'offrir aux utilisateurs du web des fonctionnalités pratiques et afin de rendre l'échange de l'information entre des systèmes hétérogènes une opération très pratique. La communication entre les Services Web se fait par l'échange des paquets de données via le protocole SOAP.

Un message SOAP n'est en effet pas qu'un fichier XML classique, avec seulement un espace de nom prenant en charge les diverses extensions. Le document décrivant un message de ce type est en réalité une déclaration très structurée, appelée "enveloppe", et contenant le nécessaire à l'exécution de l'opération qu'elle contient.

Principalement, une enveloppe SOAP peut être décomposée en deux composants : l'entête (header) et le corps (body) du message. Il peut y avoir plusieurs entêtes, ou un entête vide. Le squelette complet prend la forme suivante :

```
<?xml version="1.0" ?>  
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">  
  <env:Header />  
  <env:Body ... >  
    [...]  
  </env:Body>  
</env:Envelope>
```

Figure II.3 Forme générale d'une requête SOAP

Par ailleurs, les messages SOAP reposent fortement sur l'usage des espaces de nom : cela permet au message de transporter tout type de contenu XML, et d'éviter les conflits de noms d'éléments.

L'enveloppe

C'est l'élément supérieur du document : il englobe entête et corps. Il est obligatoire - sans enveloppe, le message ne peut pas être transporté -, et doit répondre qualifié, c'est-à-dire répondre à l'espace de nom définissant SOAP, comme présenté dans l'exemple ci-dessus.

L'entête

Placé au sein de l'enveloppe avant même le corps, l'entête peut-être utilisé pour compléter les informations nécessaires à une requête. Le plus souvent, y sont précisés des extensions SOAP (prédéfinies ou propre à l'application), des identifiants de cibles SOAP intermédiaires, ou plus généralement des métadonnées relatives au message.

Les informations de l'entête peuvent être traitées, modifiées ou effacées par les applications intermédiaires, afin que le destinataire final puisse au mieux analyser son contenu. Par ailleurs, pour assurer le bon traitement de ces informations, tous les éléments de Header doivent être qualifiés par un espace de nom.

L'entête reconnaît plusieurs attributs spécifiques :

- ✚ actor, qui indique le destinataire de l'information indiqué par l'entête. Cela permet de viser une application intermédiaire spécifique, via une URL.
- ✚ mustUnderstand, qui prend une valeur booléenne et indique si le traitement de l'élément est obligatoire ou non.
- ✚ encodingStyle, qui spécifie les règles d'encodage s'appliquant à l'élément.

Le corps

Cette section contient les données transportées par le message SOAP (le payload) qui, comme pour les éléments précédents, doit voir tous ces sous-éléments correctement qualifiés par des espaces de nom. Il doit contenir, en envoi, le nom de la méthode appelée, ainsi que les paramètres appliqués à cette méthode. En réponse, il contiendra soit un appel méthode, soit une réponse à sens unique, ou finalement un message d'erreur détaillée.

II Partie Théorique : Services Web

Comme on peut le voir dans l'exemple ci-dessous, la première ligne du message SOAP contient une déclaration XML qui n'est pas obligatoire. L'enveloppe SOAP est ensuite déclarée via la balise <SOAP-ENV:Envelope>. Cette enveloppe est composée d'un corps (<SOAP-ENV:Body>). Dans le corps de ce message, il y'a la méthode « GetNombre » et son paramètre qui est égale à 10 dans notre exemple.

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<SOAP-ENV:Envelope SOAPENV:=" "
  encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:SOAP-ENC=http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:xsi=http://www.w3.org/1999/XMLSchema-instance
  xmlns:xsd="http://www.w3.org/1999/XMLSchema" >
  <SOAP-ENV:Body>
    <ns1:GetNombre
      xmlns:ns1="urn:MySoapServices" >
      <param1 xsi:type="xsd:int">10</param1>
    </ns1:GetNombre>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Code II-1 Exemple de requête SOAP [Nicolescu 2004]

Le message de réponse ci-dessous a la même structure que celle du message envoyé. Ceci veut dire qu'il contient une déclaration XML, ainsi qu'une enveloppe SOAP composée d'un corps.

```
<?xml version="1.0" encoding="UTF-8" ?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi=http://www.w3.org/1999/XMLSchema-instance
  xmlns:xsd="http://www.w3.org/1999/XMLSchema" >
  <SOAP-ENV:Body>
    <ns1:GetNombreResponse
      xmlns:ns1="urn:MySoapServices"
      SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" >
      <return xsi:type="xsd:int">10</return>
    </ns1:GetNombreResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Code II-2 Exemple de requête de réponse SOAP [Nicolescu 2004]

II Partie Théorique : Services Web

Dans le corps du message de réponse, nous pouvons toujours voir notre méthode « GetNombre » : le mot « Response » a été rajouté à la fin du nom de la méthode pour bien préciser qu'il s'agit du retour d'une requête sur la méthode. La valeur renvoyée par la méthode « GetNombre » est égale à 10.

Dans notre exemple, l'enveloppe SOAP n'était pas composée d'un en-tête : en effet, l'en-tête d'un message SOAP est optionnelle et est utilisée pour transmettre des données d'authentification ou de gestion de sessions.

Certes, SOAP permet l'invocation à distance des méthodes, il inscrit l'espace de nom qui le définit dans la première partie qui englobe l'entête et le corps. Pour cela la partie « Enveloppe » est essentielle. Puis, dans une deuxième partie, « l'entête », qui est un élément optionnel, reconnaît plusieurs attributs (actor, mustUnderstand, encodingStyle), il constitue la première sous partie de l'enveloppe et contient des informations complémentaires pour le message (sessions, transactions...). De suite, le corps ou « Body » est réservé pour le contenu du message. Elle doit contenir :

- ✚ En envoi, le nom de la méthode appelée (GetNombre ; dans notre exemple), ainsi que les paramètres appliqués à cette méthode.
- ✚ En réponse, elle contiendra soit un appel méthode, soit une réponse à sens unique, ou finalement un message d'erreur détaillé.

La spécification SOAP se divise en quatre parties:

- ✚ L'enveloppe SOAP, qui définit le contexte d'un message, son destinataire, son contenu et différentes options ;
- ✚ Les règles de codage SOAP, définissant la représentation des données d'une application dans le corps d'un message SOAP (en particulier leur structure) ;
- ✚ Un protocole de RPC définissant la succession des requêtes et des réponses ;
- ✚ La définition de l'utilisation de HTTP comme couche de transport des messages SOAP.

Les règles de codage utilisent abondamment XML Schema pour décrire la structure des données constitutives des messages SOAP.

5.2.2 WSDL : *Web Services Description Language*

WSDL a été défini par le consortium comme étant : « Un langage XML pour la description des Services sur le réseau à travers une collection de communications entre les terminaux par l'échange des messages, les définitions WSDL des Services fournissent de la documentation pour les systèmes distribués et servir comme une recette pour automatiser les détails engagés lors de la communication entre les applications » [Site 8].

Il a été soumis par Ariba, IBM et Microsoft pour décrire les activités XML du consortium W3C (*World Wide Web Consortium*) en mars 2001, ce standard a atteint récemment sa version 2.0, il définit, d'une manière abstraite et indépendante du langage, l'ensemble des opérations et des messages qui peuvent être transmis vers et depuis un Service Web donné.

Nous tenons au niveau de cette section à présenter un exemple de description d'un Service Web "TemperatureService.wsdl" permettant de retourner la température.

II Partie Théorique : Services Web

```
<?xml version="1.0"?>
<definitions name="TemperatureService"

targetNamespace="http://www.xmethods.net/sd/TemperatureService.wsdl"

xmlns:tns="http://www.xmethods.net/sd/TemperatureService.wsdl"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
        xmlns="http://schemas.xmlsoap.org/wsdl/">
  <message name="getTempRequest">
    <part name="zipcode" type="xsd:string"/>
  </message>
  <message name="getTempResponse">
    <part name="return" type="xsd:float"/>
  </message>
  <portType name="TemperaturePortType">
    <operation name="getTemp">
      <input message="tns:getTempRequest"/>
      <output message="tns:getTempResponse"/>
    </operation>
  </portType>
  <binding name="TemperatureBinding" type="tns:TemperaturePortType">
    <soap:binding style="rpc"
transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="getTemp">
      <soap:operation soapAction=""/>
      <input>
        <soap:body use="encoded" namespace="urn:xmethods-Temperature"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
      </input>
      <output>
        <soap:body use="encoded" namespace="urn:xmethods-Temperature"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
      </output>
    </operation>
  </binding>
  <service name="TemperatureService">
    <documentation>Returns current temperature in a given U.S. zipcode
</documentation>
    <port name="TemperaturePort" binding="tns:TemperatureBinding">
      <soap:address
location="http://services.xmethods.net:80/soap/servlet/rpcrouter"/>
    </port>
  </service>
</definitions>
```

Code II-3 Exemple d'une description d'un Service Web

A travers le code source du fichier WSDL [Site 15] présenté par le Code II-3, on distingue que la structure est composée de six éléments essentiels et un septième optionnel, à savoir :

- ✚ **Définitions** : élément racine du document, il inscrit le nom du service, les espaces de noms utilisés, et contient les éléments du service.
- ✚ **Types** : décrit tous les types de données utilisées entre le client et le serveur. WSDL n'est pas exclusivement lié à un système spécifique de typage, mais utilise par défaut la spécification de *XML Schema* [Site 3].
- ✚ **Message** : décrit les données échangées, autrement dit, les paramètres des opérations.
- ✚ **PortType** : définit les messages pour composer une opération. Chaque opération se réfère à un message en entrée et à des messages en sortie.
- ✚ **Binding** : décrit les spécifications concrètes de la manière dont le service sera implémenté : protocole de communication et format des données.
- ✚ **Service** : définit les adresses permettant d'invoquer le service donné, ce qui sert à regrouper un ensemble de ports reliés.
- ✚ **Documentation** : contient la documentation lisible. (Optionnel).

Le W3C a publié le 26 mars 2004 une copie brouillon de la spécification de la nouvelle version WSDL 2.0 [Site 18]. De ce fait, plusieurs changements et améliorations ont été apportés à la version précédente (WSDL 1.1.) dont on peut citer :

- ✚ l'ajout de la sémantique pour le langage de description ; pour ce faire, ils ont proposé que l'attribut `targetNamespace` sera un élément nécessaire dans la section « Definition » du document WSDL.
- ✚ Au niveau de la section « Message », les types seront désormais spécifiés à partir du schéma XML.
- ✚ Ne plus supporter la surcharge de l'opérateur.

II Partie Théorique : Services Web

- ✚ La section « PortTypes » a été renommée « Interfaces ».
- ✚ le support d'autres langages de schéma
- ✚ « Ports » (dans la section service) a été renommée « Endpoints ».

Ayant cité les améliorations les plus importantes de la nouvelle version WSDL 2.0 nous comprenons à présent mieux ce standard.

5.2.3 *UDDI: Universal Description, Discovery and Integration*

Le consortium international OASIS [Site 13] a défini le protocole UDDI comme étant : « le membre clé du groupe interdépendant des standards qui comporte l'empileur des Services Web. Il définit une méthode standard pour publier et découvrir au niveau du réseau les composants logiciels d'une architecture orientée service (SOA).

Le modèle d'un catalogue UDDI est un élément central de l'approche orientée service pour la conception des logiciels. Via la distribution à base des règles et la gestion des Services Web de l'entreprise, le registre de l'UDDI délivre une valeur significative. » [Site 9].

La norme UDDI propose une formalisation complète des informations nécessaires au bon fonctionnement de l'architecture de Services Web et, en particulier, du protocole de découverte dynamique essentiel à la vision du logiciel comme service. Il existe également d'autres dictionnaires ou annuaires dans les standards métiers comme ebXML ou RosettaNet, élaborés, pour la plupart, avant UDDI. Tous néanmoins proposent ce protocole de recherches et de découvertes dynamiques des services. UDDI a bénéficié de ces premiers travaux de spécification et propose un référentiel partagé entre les informations métier et les informations techniques relatives à l'implémentation de ces échanges métier.

Dans la première vision de la spécification, différents scénarios d'usage d'UDDI sont proposés. Dans l'un, adapté à l'intégration d'application d'entreprises via un intranet, les Services Web représentant les interfaces des logiciels de l'entreprise, par exemple, sont publiés dans un annuaire UDDI « privé » soit manuellement, soit automatiquement au moyen d'un outil de développement utilisant l'API UDDI. Dans l'autre, un utilisateur cherche et trouve les coordonnées d'un fournisseur de Service Web via un portail, un moteur de recherche ou une place de marché indépendante, automatiquement alimenté par échange de messages SOAP avec un ou plusieurs annuaires UDDI « externes »

II Partie Théorique : Services Web

Logiquement centralisé mais physiquement réparti, l'Annuaire universel recueille les inscriptions des fournisseurs des Services Web et permet à tout un chacun d'effectuer des recherches pour y trouver les services particuliers dont il a besoin. Le site www.uddi.org construit par Ariba, IBM et Microsoft est le point focal de l'initiative UDDI. Il rassemble les publications officielles des groupes de travail et, en particulier, les spécifications de l'Annuaire universel des processus d'inscription et de recherche, et celles qui ont trait au rôle et aux responsabilités formelles des opérateurs UDDI. Les deux premiers opérateurs de l'Annuaire UDDI universel ont été IBM [Site 25] et Microsoft [Site 26], suivis de Hewlett-Packard [Site 27] et de SAP.

Exemple de recherche dans l'Annuaire universel

Quel que soit le site opérateur UDDI choisi, la recherche suit le même procédé. Les recherches peuvent s'effectuer selon différents critères : nom de l'entreprise, adresse, URL, secteur d'activité, etc.

XMethods est une organisation dédiée à la promotion des Services Web et des technologies sous-jacentes. Son site [Site 24] propose de nombreux exemples de Services Web à des fins de tests, d'illustration et de formation.

Le résultat de la recherche, c'est-à-dire la fiche descriptive de XMethods, est similaire aux informations qui peuvent être obtenues sur le site de tout autre opérateur UDDI, puisque ces sites d'opérateurs offrent une interface d'accès au même annuaire global, seule la présentation des informations pouvant varier d'un site à l'autre. La fiche descriptive de l'entreprise contient, outre des informations sur l'entreprise, ses dirigeants et ses secteurs d'activités, la liste des Services Web qu'elle propose. A chaque Service Web est associé une clé UDDI (Service Key), identifiant unique permettant à toute application d'accéder directement et automatiquement au service. Toutes les informations concernant l'entreprise enregistrées dans l'Annuaire universel sont rassemblées dans un document au format XML, dont l'adresse est donnée dans le champ Discovery URL.

5.3 WS-BPEL

Web Service - Business Process Execution Language (WS-BPEL), en général appelé BPEL, est un standard ouvert pour la modélisation et l'exécution de processus indépendamment de la plateforme. Ces processus sont formés dans WS-BPEL en reliant des Services Web.

D'un point de vue technique, ceci constitue un pas important dans l'intégration des SI hétérogènes.

Le problème des Services Web réside dans le fait qu'ils ne peuvent pas remplir toutes les exigences nécessaires à l'exécution d'un processus, car :

- Un processus à un état.
- Un processus peut durer plusieurs jours.
- Divers parties peuvent participer.
- Séquences d'échanges entre parties (communication synchrone ou asynchrone).

Pour faire face à ce problème, le standard WS-BPEL a été adopté comme une solution par les informaticiens.

WS-BPEL est basé sur une syntaxe XML et des spécifications pour Services Web [Site 16]. Il utilise WSDL et XML Schema comme modèles de données.

Les bénéfices offerts par le WS-BPEL peuvent se décliner sur deux niveaux ; celui de la gestion et celui de la technique, qui comportent chacun un certain nombre d'avantages.

Premièrement, au niveau de la gestion, le WS-BPEL permet une définition des processus B2B, il apporte des informations rapides sur l'état d'une instance des processus et enfin, il offre des indicateurs permettant l'optimisation de ces derniers.

Au niveau technique, le WS-BPEL constitue un standard ouvert. Il est, de plus, indépendant de la plate-forme. Il permet une implémentation de la vision SOA et forme un modèle exécutable.

6 Architecture

Entre le Client et la plate forme DIES, On a décidé de placer une couche de web service afin d'avoir une architecture SOA.

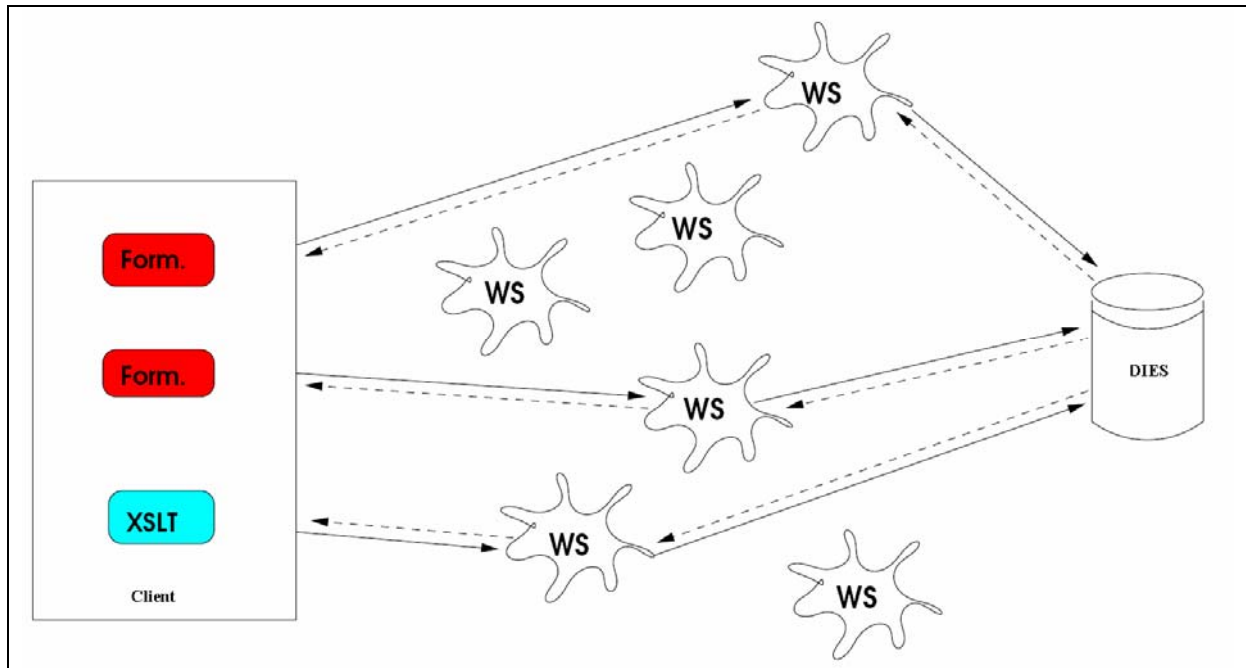


Figure II.4 Architecture du système d'information

L'architecture SOA est une architecture orientée services (notée SOA pour Services Oriented Architecture) est une architecture logicielle s'appuyant sur un ensemble de services. L'objectif d'une telle architecture est de décomposer une fonctionnalité en un ensemble de fonctions basiques, appelées services, fournies par des composants et de décrire finement le schéma d'interaction entre ces services. L'idée sous-jacente est de cesser de construire la vie de l'entreprise autour d'une application tout-en-un pour faire en sorte de construire une architecture logicielle globale d'composées en services correspondant aux processus métiers de l'entreprise.

Les principaux avantages lors de l'utilisation d'une architecture orientée services sont:

- ✚ Une modularité permettant de remplacer facilement un composant (service) par un autre.
- ✚ Une réutilisabilité possible des composants (par opposition à un système tout-en-un fait sur mesure pour une organisation).
- ✚ De meilleures possibilités d'évolution (il suffit de faire évoluer un service ou d'ajouter un nouveau service)
- ✚ Une maintenance plus facile.

7 *RPC vs Doc Style*

Remote Procedure Call (RPC), est un protocole permettant de lancer une procédure d'un programme sur un ordinateur distant. Cette méthode permet au programmeur de réaliser assez simplement des programmes Client-serveur.

La distinction qui est faite entre le style « document » et le style « RPC » pour les messages WSDL est assez obscure. En effet, les deux styles de représentation des données dans des messages SOAP coexistent dans le document WSDL. Dans le style « document », les types des messages peuvent être définis en termes de schémas XML arbitrairement complexes et, dans ce cas, le premier élément de corps de message SOAP (balise <Body>) pointe vers le namespace correspondant. Dans le style RPC, les messages peuvent être définis en plusieurs parties, et le premier élément de corps de message SOAP est une encapsulation des définitions WSDL.

Il y a deux façons de structurer un message du SOAP. Dans les premières versions de SOAP (avant qu'il ait été accessible au public), le SOAP a été conçu pour supporter seulement le style RPC. A partir du moment où le SOAP 1.0 a été publié, il a été étendu pour supporter les RPCs et les “unstructured messages” (document). Lorsqu'on utilise le style “Document”, on peut structurer le contenu du Corps du SOAP sans restrictions. En revanche, lorsqu'on utilise le style “RPC”, le contenu du Corps du SOAP doit se tenir à une structure qu'indique le nom de la méthode et contient un ensemble de paramètres. Voir ci après :

II Partie Théorique : Services Web

```
<env:Body>
<m:&methodName xmlns:m="someURI">
<m:&param1>...</m:&param1>
<m:&param2>...</m:&param2>
...
</m:&methodName>
</env:Body>
```

Code II-4 Requête SOAP avec Style RPC

Le message de réponse a une structure semblable qui contient la valeur du retour et tous paramètres de sortie. Par exemple, on peut passer un “purchaseorder” comme un document ou comme un paramètre dans une méthode appelée “placeOrder”. C'est votre choix:

Document style:

```
<env:Body>
  <m:purchaseOrder xmlns:m="someURI">
    ...
  </m:purchaseOrder>
</env:Body>
```

Code II-5 Message de réponse avec Style Document

RPC style:

```
<env:Body>
  <m:placeOrder xmlns:m="someURI">
    <m:purchaseOrder>
      ...
    </m:purchaseOrder>
  </m:placeOrder>
</env:Body>
```

Code II-6 Message de réponse avec Style RPC

La plus grande différence se trouve dans la façon d’encoder le message. Dans la plupart des circonstances, on utilise le codage littéral avec le style Document et le codage du SOAP avec le style RPC. Le codage du SOAP utilise un ensemble de règles basé sur le XML Schéma datatypes pour chiffrer les données. En revanche, le message ne suit pas un schéma particulier.

D’autre part, Microsoft MS SOAP Toolkit et.NET ne supportent que le style RPC, mais utilisent le style Document par défaut [Site 29].

Avantages de RPC

Quelques avantages du RPC sont la simplicité typique de la Request/response typiquement simple. De plus, l'usage en est simplifié.

Avantages de Document-Style

Le style Document favorise une description des données plus riche, plus complète. De plus, il offre une plus grande flexibilité. Le style Document convient mieux pour workflows et traitement asynchrone. Il assure une interopérabilité avec les représentations XML existantes[Site 30].

8 Conclusion

Un Service Web est un composant logiciel représentant une fonction applicative (ou un service applicatif). Il est accessible depuis une autre application (un client, un serveur ou un autre service Web) à travers les réseaux Internet en utilisant les protocoles de transports disponibles. Ce service applicatif peut être implémenté comme une application autonome ou comme un ensemble d'applications. Il s'agit d'une technologie permettant à des applications de dialoguer à distance via Internet, et ceci indépendamment des plate-formes et des langages sur lesquelles elles reposent.

Pour ce faire, les Services Web s'appuient sur un ensemble de protocoles standardisant les modes d'invocation mutuels de composants applicatifs. Ces protocoles sont répartis selon quatre axes :

1. Couche de transport (HTTP, FTP ou SMTP) : cette couche s'occupe de transporter les messages entre les applications.
2. Messages XML (SOAP) : il s'agit de formaliser les messages à l'aide d'un vocabulaire XML commun.
3. Description des services (WSDL) : il s'agit de la description de l'interface publique des services Web.
4. Recherche de service (UDDI) : il s'agit de la centralisation des services Web (de leurs descriptions) dans un référentiel commun.

Un Service Web peut donc être compris comme une application exécutable disponible "en self-service" pour être utilisée par des clients (entreprises, applications, etc.). Le Service Web est ainsi "exposé" sur le Web, avec la description de son fonctionnement et des paramètres dont il a besoin pour fonctionner. L'un des énormes avantages est par exemple, de pouvoir utiliser un Service Web développé en Java sous Unix, dans une application tournant en Visual Basic dans un environnement .NET de Microsoft, puisque l'ensemble du dialogue se fera via des standards XML.

III *Partie Pratique :
Services Web*

1 Introduction

Au niveau de ce chapitre, on va implémenter du web based application, cette partie pratique va être développée en utilisant .Net et XML (eXtensible Markup Language ou langage de balisage extensible) qui est un standard du consortium World Wide Web et qui sert de base pour créer des langages balisés spécialisés.

2 DIES

Le “ Development Information Exchange System DIES” est un mediator-wrapper-architecture qui utilise des documents XML pour associer des systèmes d'information hétérogènes. Le système peut aider à fermer la « Control Loop » en fournissant de l'information qualitative à propos des projets humanitaires. Cela rend la gestion du projet plus efficace.

Une étude de marché sur les systèmes d'informations accessibles par le Web a montré que jusqu'à présent, il est difficile de trouver des informations détaillées sur des projets humanitaires. Ces informations sont nécessaires pour coordonner des activités et pour permettre un apprentissage au niveau de l'organisation (organisational learning). Un des problèmes se trouve au niveau de l'échange d'informations entre les stakeholders. Il n'y a pas de standard pour l'échange électronique de rapports et d'évaluations de projets humanitaires.

Un concept et un prototype ont été conçus afin de permettre l'échange d'informations structurées entre des systèmes d'informations autonomes et hétérogènes. La plate-forme est basée sur Internet et porte le nom Development Information Exchange System (DIES). L'échange dans le DIES se fait par des documents XML (Extensible Stylesheet Language) (W3C/XML 1998). La structure des documents est définie avec des XML Schemas. Les deux schémas International Development Markup Language (IDML) et idmlReporting sont utilisés. IDML est un standard émergent pour l'échange d'informations générales sur des projets humanitaires (IDML 2000). IdmlReporting est une extension d'IDML proposée par l'auteur permettant de structurer des rapports et des évaluations de projets [Hüsemann 2002].

2.1 Architecture du Development Information Exchange System (DIES)

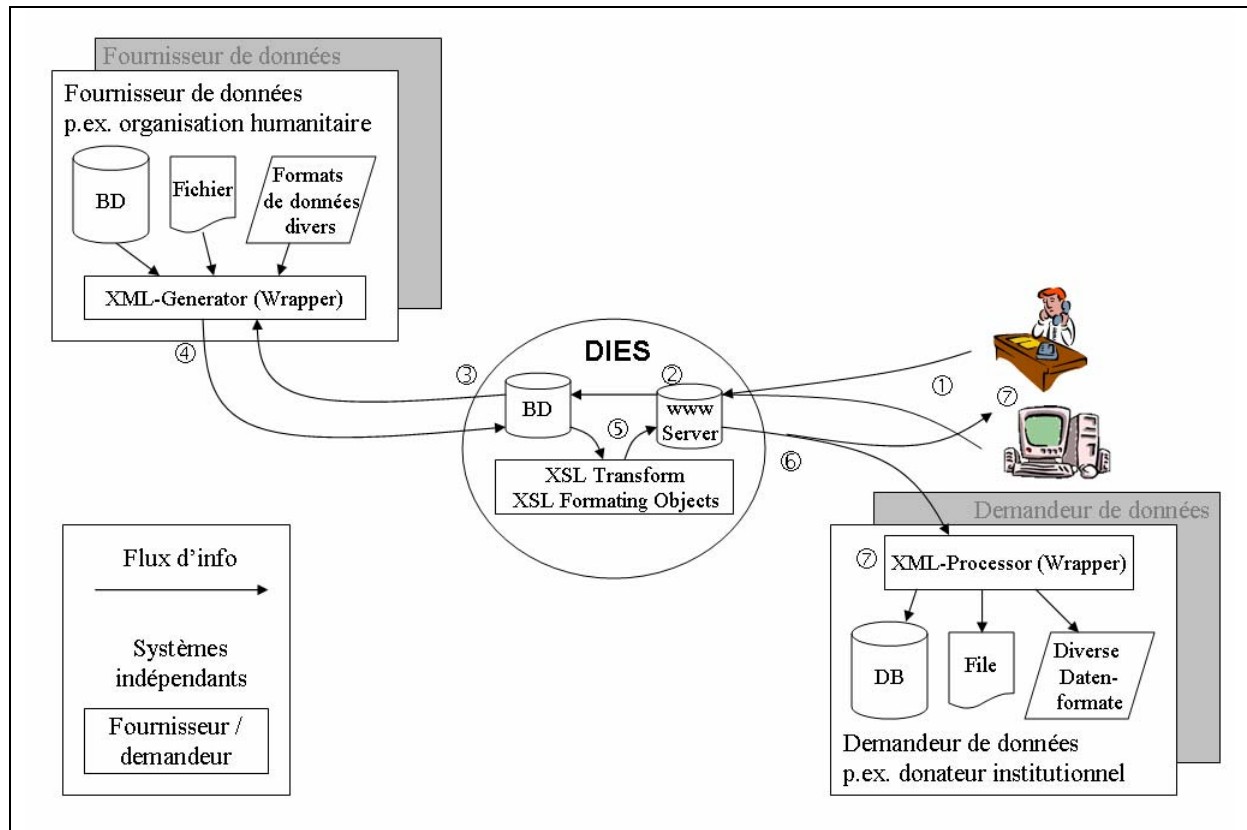


Figure III.1 Architecture du DIES (Hüsemann 2005)

Le DIES fonctionne de la manière suivante (les numéros se réfèrent au Figure III.1):

- (1) Une personne ou un système d'information se connecte au DIES par le Web.
- (2) Le DIES identifie l'utilisateur et ses droits d'accès à l'aide d'une base de données (BD). L'utilisateur choisit parmi les documents et les feuilles de style (Extensible Stylesheet Language, XSL) à sa disposition.
- (3) Le DIES prend contact par Internet avec les systèmes des fournisseurs d'informations sur lesquels se trouvent les documents et feuilles de style sélectionnés.
- (4) Les systèmes connectés renvoient les informations au DIES dans le format XML.

III Partie Pratique : Services Web

- (5) Selon les droits d'accès et le format choisi le document XML est transformé et formaté (XSL Transform et XSL Formatting Objects).
- (6) Le résultat est envoyé à l'utilisateur. Il est par exemple possible de générer des pages HTML, des documents PDF ou de nouveau du XML.
- (7) Selon le format, le document peut ensuite être importé de manière automatisée dans le système du demandeur d'informations en utilisant un processeur XML.

Quand un fournisseur d'informations veut mettre à disposition de nouveaux documents à certains membres du DIES il doit enregistrer l'URL (Unique Resource Locator) sur la plate-forme. Le document ne doit pas forcément exister physiquement à cette URL. Il peut aussi être généré au moment de la requête du DIES par un wrapper (enveloppe) c'est-à-dire un générateur XML qui a été programmé pour un système de gestion de projets par exemple.

2.2 Objectif du DIES

Dans le cadre de l'amélioration de l'efficacité des organisations humanitaires, le DIES a pour objectifs une meilleure coordination entre projets de ces dernières. De plus, il a pour but d'améliorer les rapports et assister le processus d'apprentissage organisationnel ; ce dernier point, consistant à apprendre à partir d'autres projets.

Cette plate-forme d'échange, à aussi pour objectifs de donner un aperçu des projets de diverses organisations ainsi que permettre l'échange d'informations détaillées sur les projets (rapports de projets).

Pour faciliter l'échange d'information par le DIES, on a recours à des concepts et prototype. Ceux-ci constituent des moyens permettant d'atteindre les objectifs et d'échanger des informations détaillées sur les projets (rapports de projets).

2.3 Bénéfices du Service Web pour DIES

Le Service Web DIES_Service résout la communication machine à machine. En effet, Il utilise des standards et protocoles ouverts dont les protocoles et les formats de données sont au format texte dans la mesure du possible, facilitant ainsi la compréhension du fonctionnement global des échanges.

Le Service Web offre une interface interopérable qui favorise le fonctionnement sur diverses plateformes. Enfin, le Service Web met à disposition du DIES des fonctionnalités telles que SearchDoc et AffichDoc.

3 ***Implémentation***

Au niveau méthodique pour l'implémentation du système, on a d'abord fait une analyse, puis la conception (dans cette phase on a créé un Use Case qui décrit les fonctionnalités du Service Web). Ensuite nous avons choisi dot Net en tant qu'environnement logiciel pour l'implémentation de notre système en se basant sur quelques points avantageux. Afin d'expliquer mon choix de procédure d'implémentation, on a défini le « contract first web service » en listant les avantages ainsi que les inconvénients. Nous terminons ce chapitre par la partie d'évaluation qui consiste à appliquer la méthode SWOT sur notre Service Web dont le but de connaître les forces, les faiblesses ainsi que les occasions et les menaces.

3.1.1 ***Diagramme de cas d'utilisation***

Un cas d'utilisation (Use Case) décrit une fonctionnalité du système dans un langage non technique. Un modèle de cas d'utilisation (Use Case Model) permet de représenter de manière graphique les différents cas d'utilisation d'un système ainsi que les acteurs pouvant initier ces cas d'utilisation. Chaque cas d'utilisation a une description qui décrit de manière précise et concise ce cas d'utilisation [Hüsemann 2005].

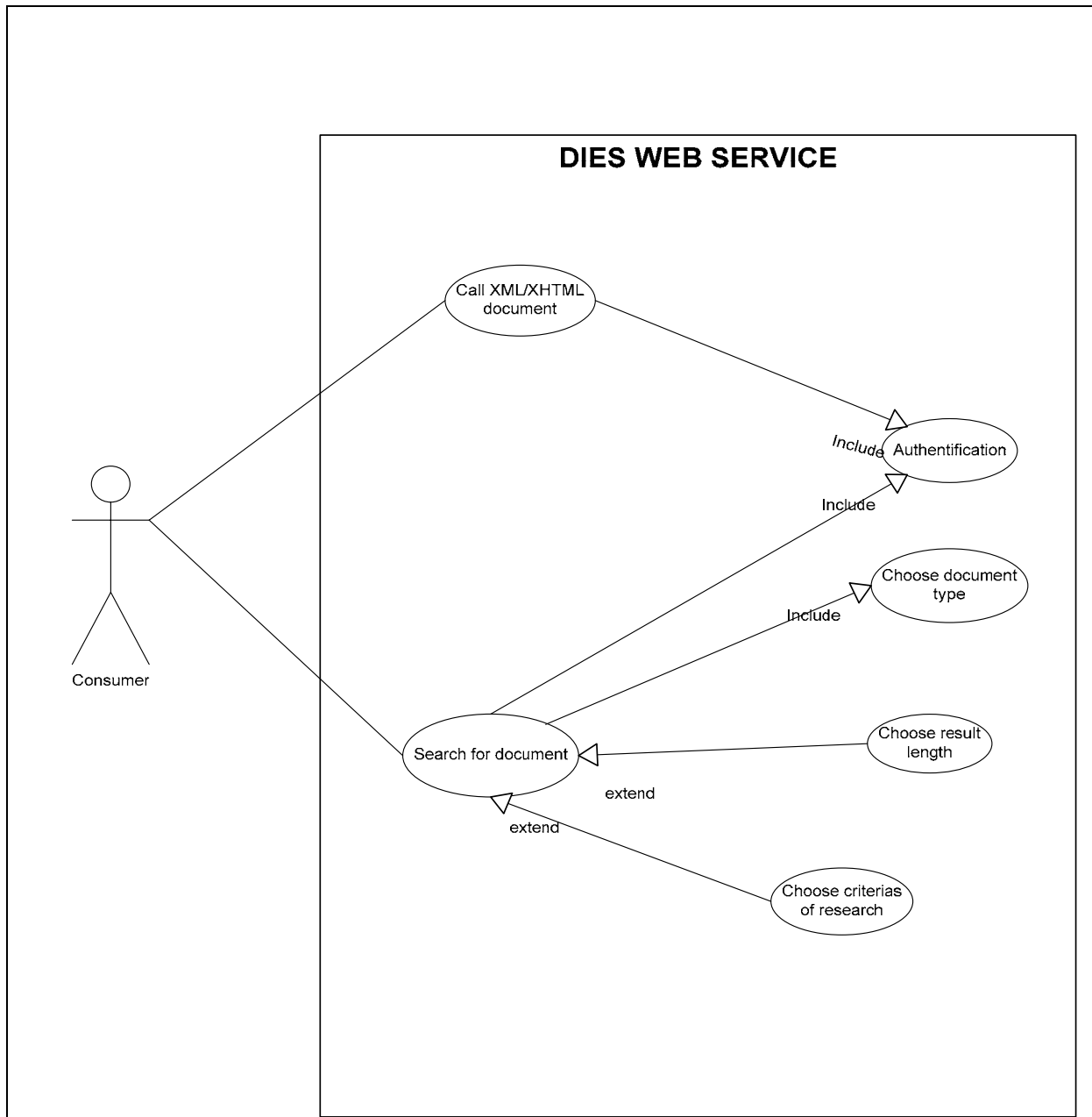


Figure III.2 Diagramme de cas d'utilisation (propre élaboration)

Dans le but d'éclaircir chaque cas d'utilisation inséré au niveau du diagramme, on tient à donner une description de chacun.

Le use case « Search for document » fait appel aux use cases « Authentication » et « Choose document type ». Tandis que les use cases « Choose result length » et « Choose criterias of research » peuvent être défini comme une extension de « Search for document » afin d'ajouter des fonctionnalités en plus. En revanche, le use case « Call XML/HTML document » s'appelle **AffichDoc** plus tard, inclus seulement le use case « Authentication ».

[UML livre]

Recherche de Documents ou de Stylesheets:

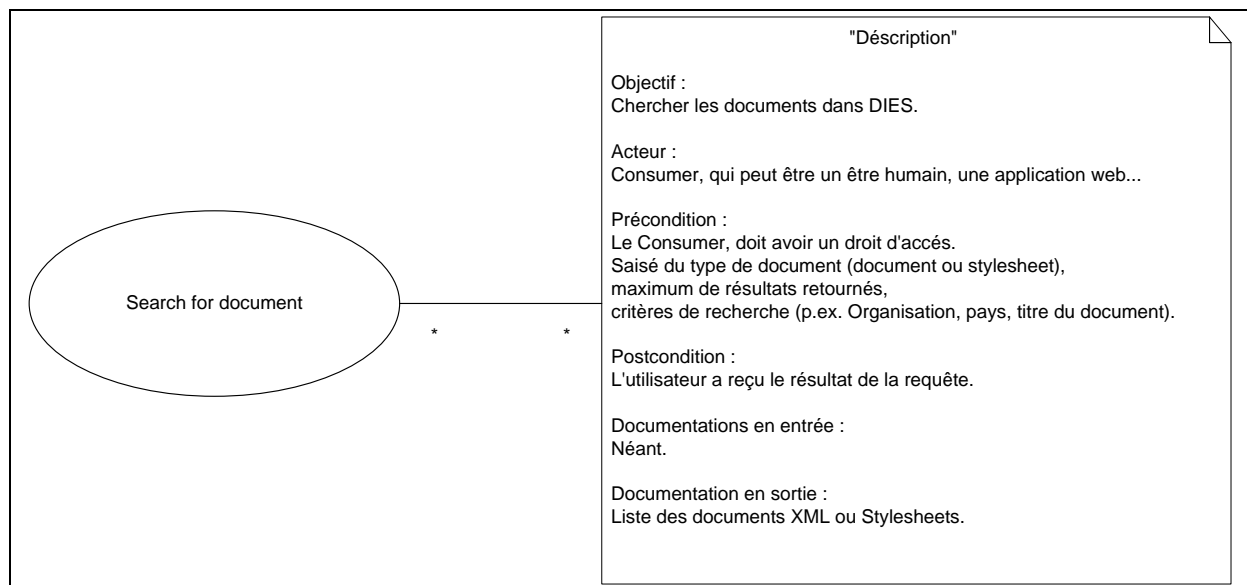


Figure III.3 Description du cas d'utilisation "Search For document" (propre élaboration)

Ce cas d'utilisation prend comme Input : username, type de document (document ou stylesheet), maximum de résultats retournés, critères de recherche (p.ex. Organisation, pays, titre du document).

Output : Total results matching Search, Liste des documents auxquels l'utilisateur a accès - par document trouvé : document ID, type de doc, Titre du document et tous les autres méta-données d'un document dans la base de données DIES (mais pas le contenu même du document).

Appeler Document avec Stylesheet:

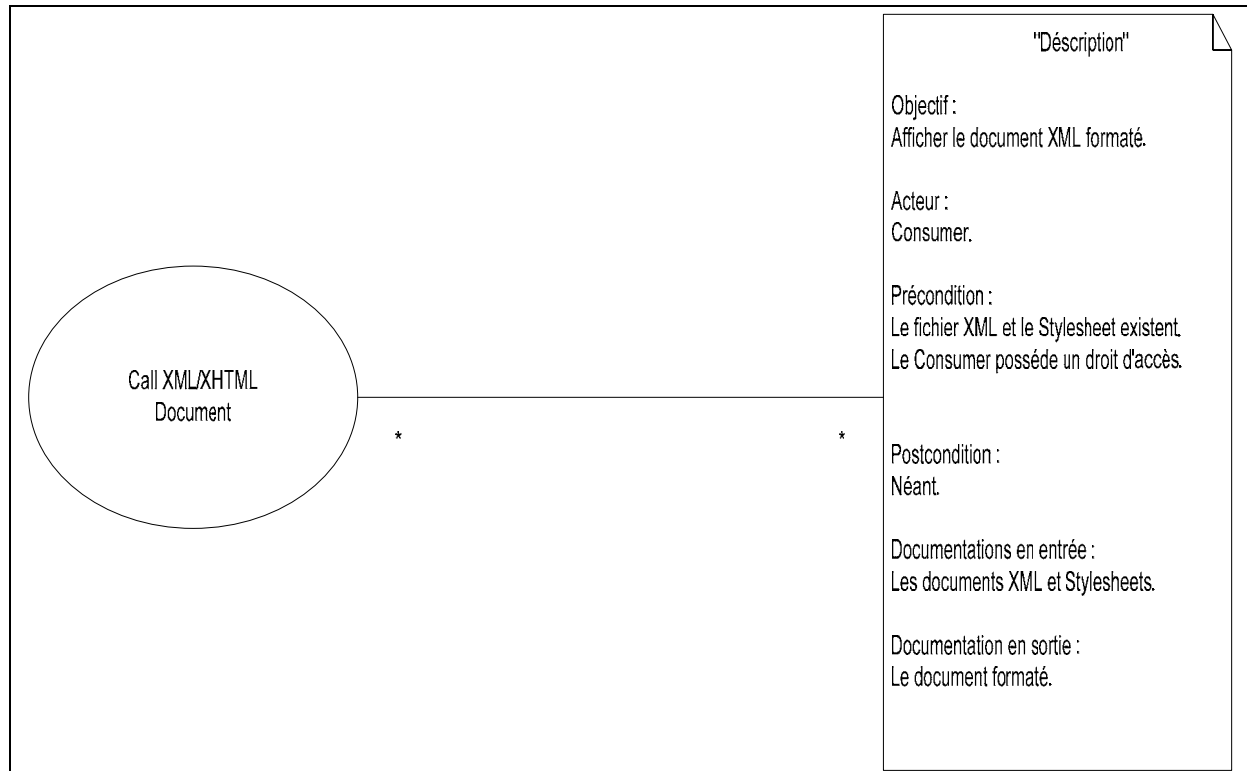


Figure III.4 Description du cas d'utilisation "Call Document" (propre élaboration)

Le use case « Call XML/HTML document » a pour but d'afficher le document XML formaté suivant le stylesheet pré-choisit. Cette méthode utilise comme input: Document ID, Stylesheet ID et retourne un output: Document formaté/transformé (limitation au format XML/XHTML).

3.1.2 *Environnement logiciel*

Pour l'implémentation du Service Web, on a opté pour la technologie dot NET. En effet, utiliser Visual Studio .NET facilite le travail du développeur et rend plus productif le projet. Lorsque vous commencez un nouveau Service Web, Visual Studio .NET vous crée tous les fichiers nécessaires au bon fonctionnement de votre Service Web et fournit quelques exemples de code.

La stratégie .NET de Microsoft est lancée le 22 juin 2000 lors d'une conférence donnée par Bill Gates et Steve Ballmer à l'occasion du Forum 2000, ".NET" ("dot Net" en anglais) dessine la stratégie 'tout Internet' de Microsoft. Objectif affiché : faire évoluer les solutions Windows vers un modèle ASP (applications hébergées) et proposer une plate-forme logicielle sur laquelle les entreprises pourront s'appuyer pour échanger et mettre à disposition des données et des services applicatifs. En quelque sorte, après s'être imposé dans le domaine des systèmes d'exploitation pour les PC, Microsoft prépare avec .NET le système d'exploitation du Web: une architecture logicielle au sein de laquelle des services applicatifs pourront collaborer via Internet. En ce sens, .NET représente l'adaptation Microsoftienne de ce que d'autres éditeurs nomment les Services web.

Les composants applicatifs et leur enveloppe de Web Services sont traités au sein d'un environnement d'exécution appelé "Common Language Runtime" (CLR). Installé sur la machine de l'utilisateur final, CLR s'apparente à une machine virtuelle Java capable d'interpréter les standards qui décrivent des Services Web.

3.2 Contract First Development :

L'approche typique en utilisant le Contract First Web Service consiste en cinq étapes. Dans un premier temps on commence par la définition de messages XML en utilisant le XSD. Ensuite on définit les opérations par l'utilisation de WSDL. Ceci génère le ASMX, qui est un code produit par le générateur et qui est conforme au contrat. La quatrième étape consiste à l'implémentation des opérations. Finalement, le processus est mis en marche et l'itération continue.

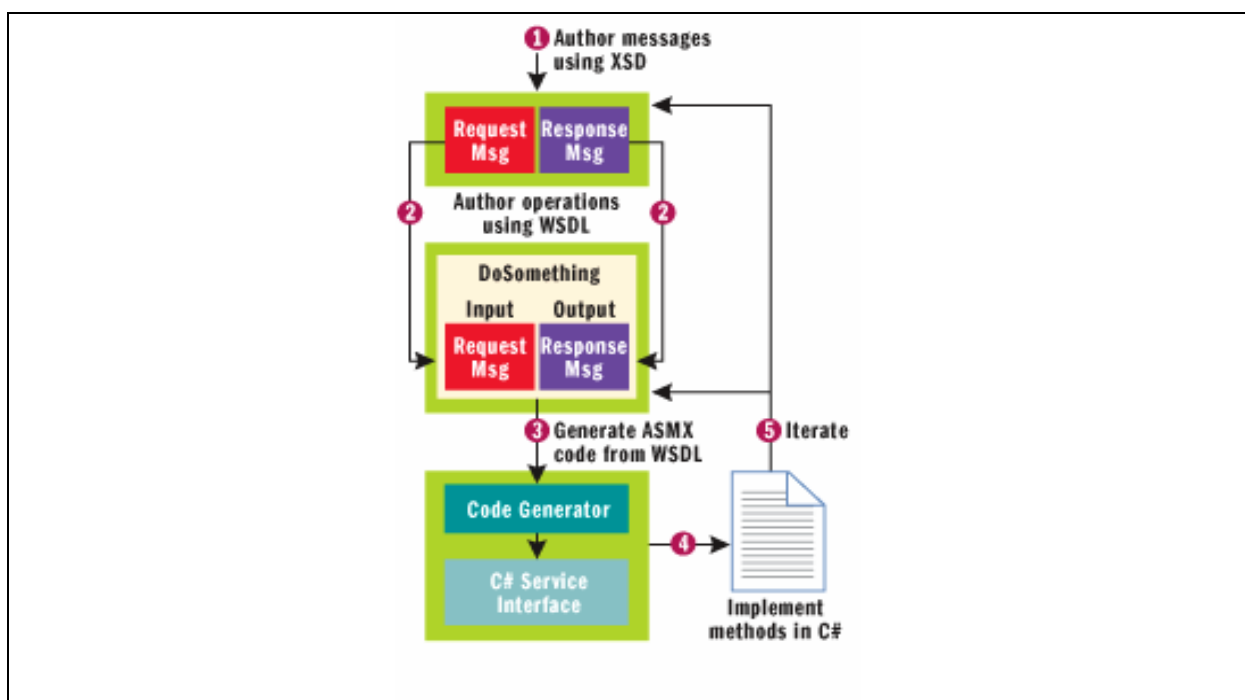


Figure III.5 Contract First Development [Site 21]

Les avantages en commençant l'implémentation par le WSDL et le schéma consiste en ayant un schéma plus descriptif, une plus grande flexibilité lors d'un changement du code et facilite l'utilisation des standards car ils sont écrits pour la description WSDL et les messages SOAP [Site 22]. En plus, cette approche permet que l'interface demeure inchangeable lors d'une modification d'une ligne de code. Toutefois, il existe un lien serré entre le WSDL et la méthode ce qui est un inconvénient.

En revanche l'élaboration de notre projet a été faite différemment en optant à l'approche « décontract first web service ». Cette approche consiste en premier par

III Partie Pratique : Services Web

l'implémentation du Service Web sur dot Net. Par la suite, Microsoft Visual Studio génère automatiquement le WSDL.

3.3 WSDL et WebMethod :

3.3.1 *WSDL* :

Chaque Service Web possède un fichier de spécification : c'est ce qu'on appelle la WSDL (Voir chapitre II.5.2.2 WSDL). C'est à partir de ce fichier que le développeur pourra savoir comment interroger le service web, quels sont ses différentes WebMethods, quels types ces WebMethods retournent elles.

La WSDL décrit en fait tout le fonctionnement d'un Service Web et il est donc indispensable pour pouvoir utiliser ce dernier. Ce fichier est structuré en XML. Le Microsoft Visual Studio génère automatiquement la WSDL d'un Service Web ce qui facilite grandement la tâche du développeur.

Enfin, pour que l'application puisse utiliser un Service Web, il faut qu'elle dispose d'une classe Proxy appelé Proxy Web qui sera créée à partir du Contrat du Service Web, c'est-à-dire le fichier de description WSDL. Le Proxy va aider le client à savoir où trouver le service web. Le Proxy contiendra également les détails des communications avec le Service Web. L'utilitaire « wsdl.exe » fourni avec le Microsoft Visual Studio permettra de générer notre classe Proxy à partir d'un fichier de description WSDL.

Une telle description des Services Web est absolument indispensable à la réalisation de l'objectif d'interopérabilité et d'accès généralisé des applications web (voir aussi Annexe D : WSDL).

3.3.2 *WebMethod* :

Le Common Language Runtime nous permet d'ajouter des déclarations descriptives de type mot clé, appelées attributs, pour annoter les éléments de programmation tels que les types, champs, méthodes et propriétés. Les attributs sont enregistrés avec les métadonnées d'un fichier Microsoft .NET Framework et peuvent être utilisés pour décrire votre code au Runtime ou affecter le comportement de l'application au moment de l'exécution. Bien que le .NET Framework fournisse de nombreux attributs utiles, vous pouvez également concevoir et

III Partie Pratique : Services Web

déployer vos propres attributs personnalisés. La terminologie **WebMethod** de Microsoft est en fait un attribut.

Le fait d'attacher l'attribut **WebMethod** à une méthode **publique** indique que vous voulez que la méthode soit exposée comme faisant partie du service Web XML. Vous pouvez aussi utiliser les propriétés de cet attribut pour configurer le comportement de la méthode de Service Web XML. L'attribut WebMethod fournit les propriétés suivantes : BufferResponse, CacheDuration, Description, EnableSession, MessageName et TransactionOption [Site 19].

Lorsque nous devons créer un Service Web se nommant par exemple «MyWebService», nous devons créer une classe «MyWebService». C'est pour cela, on a nommé notre classe «DIES_Service». En effet tout est objet en .NET et tout code doit se trouver dans une classe.

Comme on le savait, une classe dispose de valeurs ainsi que de méthodes. Ceci ne change pas avec les Services Web, mais pour qu'une méthode soit utilisable via notre Service Web, il faut donner l'attribut [WebMethod] à cette même méthode.

Notre Service Web contient deux méthodes :

- SearchDoc
- AffichDoc

Examinons à présent le code suivant de SearchDoc:

```
[WebMethod]
public DataSet SearchDoc(string username, string typeDoc,
int max_result, string organisation, string titre_doc)
{
    [...]

    return null;
}
```

Code III-1 Bout de code de la Méthode "SearchDoc"

« SearchDoc » est une WebMethode car elle possède l'attribut « WebMethod », donc elle pourra être atteinte via le Service Web. En effet, Toutes WebMethods doivent être déclarées en public et ne peuvent pas être « private » ou « protected », sinon elle ne pourra pas être atteinte depuis l'extérieur.

III Partie Pratique : Services Web

La WebMethod **SearchDoc** va retourner des documents de type « string » correspondant aux paramètres suivants:

- ✚ username : les documents auxquels l'utilisateur correspondant a le droit d'accès.
- ✚ typeDoc : peut prendre deux valeurs XML pour les documents XML et STYLESHEET pour les stylesheet.
- ✚ max_result : nombre maximal de documents retournés.
- ✚ organisation : organisation à laquelle appartient le document.
- ✚ titre_doc : titre de document comme argument.

Tous ces paramètres sont du type « string ».

« **SearchDoc** » commence tout d'abord par la vérification du code utilisateur (voir aussi Annexe A : Méthode "SearchDoc"). Dans cette partie, on va instancier la connexion à la base de données en adaptant une requête SQL afin de chercher l'identifiant correspondant au code utilisateur donné en paramètre. Ensuite on va instancier le « DataAdapter » qui va exécuter la requête SQL ainsi que la « DataTable » qui va contenir le résultat. À la fin, le programme vérifie si l'utilisateur existe, sinon il lance une exception qui donne comme message d'erreur : Utilisateur inexistant.

La deuxième partie « **XML** », débute par une requête XML qui sélectionne les documents XML en vérifiant les droits de l'utilisateur et en filtrant le nom de l'organisation et le nom du document. Comme au paravent, on va instancier le « DataAdapter » pour l'exécution de la requête ainsi que la « DataTable » qui va contenir le résultat. Le processus termine par retourner le « DataSet » qui contient la « DataTable ».

Dans la dernière partie « **Stylesheet** » une requête SQL sélectionne des stylesheets en vérifiant les droits de l'utilisateur et en filtrant le nom de l'organisation et le nom du document. La suite est identique au processus de la partie « **XML** ».

III Partie Pratique : Services Web

```
[WebMethod(Description = "", EnableSession = true)]

    public string affichDoc(string docName, string styleName)
    {
        OdbcConnection cn;
        OdbcDataAdapter da;
        DataTable dt;

        string sql, connectionString, documentURL, styleURL;
        [...]
    }
```

Code III-2 Bout de code de la Méthode "AffichDoc"

Le code ci-dessus (voir aussi Annexe B : Méthode "AffichDoc") de la deuxième WebMethode de notre Service Web, commence dans la clause « try » par instancier la connexion à la base de données puis l'exécution d'une requête SQL qui a pour but de rechercher l'identifiant correspondant au code d'utilisateur donnée en paramètre. Après l'étape d'instancier le « DataAdapter » et la « DataTable » qui va contenir le résultat de la requête SQL, le code récupère l'URL du fichier XML s'il existe, sinon il lance un message d'erreur comme quoi « Le document XML recherché n'existe pas ». Le déroulement de la partie correspondante au stylesheet est identique que le processus qu'on vient de traiter. À la fin de la clause « try » on fait appelle à la méthode " transform " ci-dessous qui prend comme paramètre le documentURL ainsi que le styleURL et retourne le document XML formaté avec le style correspondant.

« **AffichDoc** » finit par la clause « catch » qui retourne au cas où les différentes étapes de la clause « try » ne sont pas vérifier, un message d'erreur : « document or stylesheet not found ».

```
public string Transform(string xml, string stylesheet)
{
    XmlDocument XDoc;
    System.Xml.Xsl.XslCompiledTransform XTrans;
    XmlTextWriter wr;
    System.IO.StreamReader rd;
    HttpRequest req = HttpContext.Current.Request;
    string t = "", tmpPath = Server.MapPath(req.ApplicationPath) +
"\\tmp.htm";

    try
    {
        //instancier l'objet XmlDocument qui servira à lire le fichier
XML
        XDoc = new XmlDocument();
        [...]

    }
    //retourner le contenu HTML
    return t;
}
```

Code III-3 Bout de code de la Méthode "Transform"

Il s'agit dans cette méthode (voir aussi Annexe C: Méthode "Transform") de donner le résultat de la transformation du fichier XML (dont le chemin est donné par le paramètre XML) par le style XSL (dont le chemin est donné par le paramètre stylesheet). Pour ce ci nous aurons besoin d'écrire le résultat dans un fichier temporaire (tmp.htm) que nous allons enregistrer dans le répertoire courant. L'instruction: tmpPath = Server.MapPath(req.ApplicationPath) + "\\tmp.htm"; nous donnera le chemin complet de ce fichier temporaire.

L'objet XDoc (de la classe XmlDocument) sert à lire le fichier XML.

L'objet XTrans (de la classe XslCompiledTransform) sert à lire le fichier XSL, c'est-à-dire le stylesheet. Après avoir instancié les deux objets XDoc et XTrans, on les charge.

L'objet wr (de la classe XmlTextWriter) sert à créer et à écrire dans le fichier temporaire. Après avoir instancié cet objet, en lui donnant le chemin (variable tmpPath) et le type d'encodage (UTF8 dans notre cas) en paramètre, on applique la transformation par la méthode Transform de l'objet XTrans en lui donnant en paramètres l'objet XDoc (le document XML) et l'objet wr (le XmlTextWriter qui contiendra le résultat de la transformation). Une fois la transformation effectuée, on ferme wr.

III Partie Pratique : Services Web

Maintenant nous avons le résultat de la transformation que voulons, mais elle se trouve dans le fichier tmp.htm, donc on doit le lire et le retourner.

L'objet rd (de la classe StreamReader) sert à lire le fichier temporaire, dont le chemin est tmpPath.

La méthode ReadToEnd de l'objet rd va nous retourner le contenu du fichier. Ensuite, on ferme rd, et on supprime le fichier temporaire.

Enfin, on retourne t, la variable de type string dans laquelle nous avons mis le contenu du fichier temporaire.

3.4 Service Consumer : utilisation du Service Web « DIES_Service »

Le Service Web DIES_Service permet aux utilisateurs autorisés de chercher des rapports de projets et choisir un format de sortie. Les documents sont d'ailleurs dans un format XML. La transformation et le formatage des documents sont faits par stylesheets XSL.

Si un utilisateur désire obtenir des informations sur un projet spécifique, comme par exemple, celui d'une organisation partenaire, il peut choisir dans le **Document type** entre XML et Stylesheets. Il obtiendra une liste de documents, selon ses droits d'utilisateur. Enfin, tous ces documents pourront être formatés avec un des stylesheets disponible dans le champ **Stylesheets**.

Le Service Consumer est l'utilisateur. Afin de le tester, on a utilisé des outils qui jouent un rôle de stimulateur du Service Consumer, tels que XMLSpy et SOAPUI (grâce au WSDL) afin de créer les requêtes SOAP à savoir SOAPSearchDoc et SOAPAffichDoc. La figure ci-dessous est un ScreenShot qui représente la requête SOAP de la première WebMethod SearchDoc dans l'outil SOAPUI [Site 23]. On remarque que le Body de la requête contient les différents champs de la WebMethod, ces informations vont être envoyées au serveur dans l'enveloppe SOAP <Soapenv:envelope>.

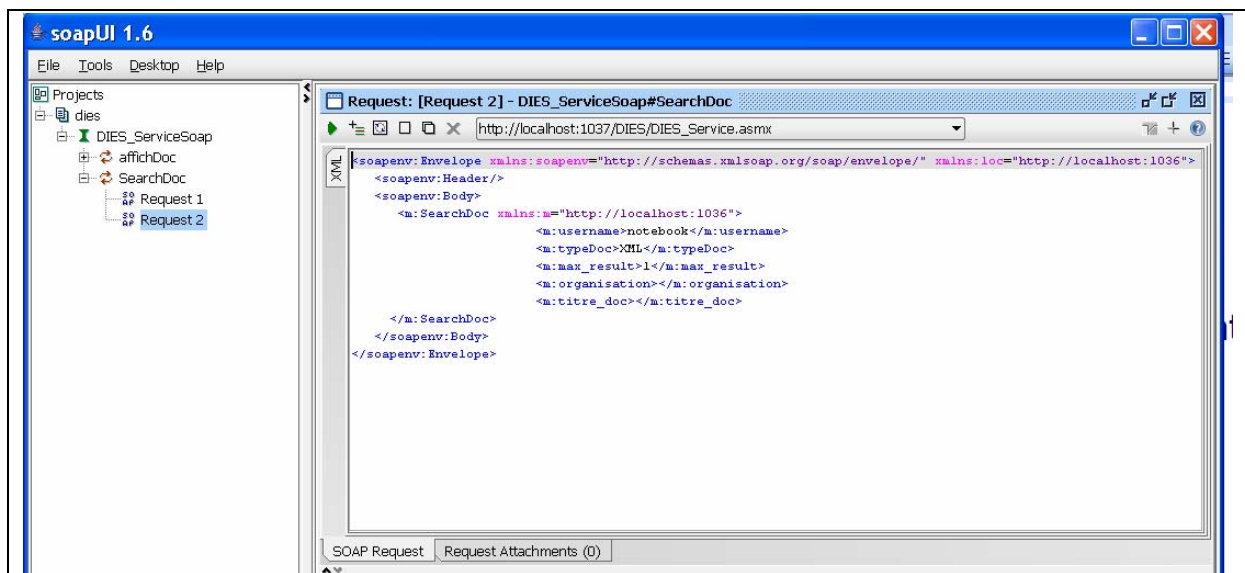


Figure III.6 Requête: DIES_Service SOAPSearchDoc

Ci-après, le message SOAP présente les résultats de la requête contenant les réponses dans l'enveloppe SOAP.

III Partie Pratique : Services Web

```

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <soap:Body>
    <SearchDocResponse xmlns="http://localhost:1036">
      <SearchDocResult>
        <xs:schema id="XML_documents" xmlns=""
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">
          <xs:element name="XML_documents"
msdata:IsDataSet="true"
msdata:UseCurrentLocale="true">
            <xs:complexType>
              <xs:choice minOccurs="0"
maxOccurs="unbounded">
                <xs:element name="Table1">
                  <xs:complexType>
                    <xs:sequence>
                      <xs:element name="key_document" type="xs:int" minOccurs="0"/>
                      <xs:element name="document_name" type="xs:string" minOccurs="0"/>
                      <xs:element name="document_URL" type="xs:string" minOccurs="0"/>
                      <xs:element name="document_public_restricted" type="xs:string"
minOccurs="0"/>
                      <xs:element name="document_belongs_to_Data_provider_system"
type="xs:int" minOccurs="0"/>
                      <xs:element name="document_follows_schema" type="xs:int"
minOccurs="0"/>
                      <xs:element name="fKey_document_registered_by" type="xs:int"
minOccurs="0"/>
                      <xs:element name="document_blocked" type="xs:boolean"
minOccurs="0"/>
                      <xs:element name="document_date_registered" type="xs:dateTime"
minOccurs="0"/>
                      <xs:element name="document_comment" type="xs:string" minOccurs="0"/>
                      <xs:element name="organisation_name" type="xs:string" minOccurs="0"/>
                    </xs:sequence>
                  </xs:complexType>
                </xs:element>
              </xs:choice>
            </xs:schema>
          <diffgr:diffgram xmlns:msdata="urn:schemas-microsoft-com:xml-msdata"
xmlns:diffgr="urn:schemas-microsoft-com:xml-diffgram-v1">
            <XML_documents xmlns="">
              <Table1 diffgr:id="Table1"
msdata:rowOrder="0">
                <key_document>16</key_document>
                <document_name>
                  Global Environment Facility (ohne ns, SH)
                </document_name>
                <document_URL>
                  <a href="http://localhost/server/xml/testreport_ohne_ns.xml">http://localhost/server/xml/testreport_ohne_ns.xml</a>
                </document_URL>
              </Table1>
            </XML_documents>
          </diffgr:diffgram>
        </xs:element>
      </SearchDocResult>
    </SearchDocResponse>
  </soap:Body>
</soap:Envelope>

```

III Partie Pratique : Services Web

```
<document_public_restricted>restricted</document_public_restricted>

<document_belongs_to_Data_provider_system>3</document_belongs_to_Data_provi
der_system>

<document_follows_schema>3</document_follows_schema>

<fKey_document_registered_by>8</fKey_document_registered_by>
<document_blocked>false</document_blocked>
<document_date_registered>2002-10-19T14:20:50+02:00</document_date_registered>
<document_comment>Proposed Project Concept and request for a PDF Block B Grant.&lt;br&gt;Report im
idmlReporting Format ohne Angabe des Namespace (referenz auf Laptop SH)</document_comment>
<organisation_name>University of Fribourg</organisation_name>
      <Table1>
        </XML_documents>
      </diffgr:diffgram>
    </SearchDocResult>
  </SearchDocResponse>
</soap:Body>
</soap:Envelope>
```

Figure III.7 Résultat de la Requête: DIES_Service SOAPSearchDoc

III Partie Pratique : Services Web

Le message SOAP ci-dessous présente le résultat de la requête SOAPAffichDoc. On remarque qu'il y a XML dans le HTML.

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
  <soap:Body>
    <affichDocResponse xmlns="http://localhost:1036">
      <affichDocResult><![CDATA[<html>
        <head><title>Project Information</title></head>
        <body>
          <div STYLE="font-family:Arial Black;"><center><TABLE width="100%"><tr><td
            align="center"><h1>Project Information</h1></td></tr><tr><td><br /></td></tr></TABLE><p>This is a list
            of all projects with summary information</p><table width="100%" cellpadding="2" cellspacing="2"
            border="0"><tr bgcolor="blue"><td colspan="2" width="100" STYLE="font-family:Verdana; font-
            weight:bold; color:white">Project:</td><tr bgcolor="silver" STYLE="font-size:10pt; font-
            family:Verdana; font-weight:bold; font-style:normal;"><td valign="top" style="color:blue;"><b>Activity
            ID</b></td><td valign="top" align="right">1234</td></tr><tr bgcolor="gray" STYLE="font-size:11pt; font-
            family:Verdana; font-weight:bold; font-style:normal;"><td valign="top" style="color:black;"><b>Project
            Title</b></td><td valign="top" align="right">
              Education of homeless children in Africa
            </td></tr>
            [...]
          </td></tr><tr bgcolor="gray" STYLE="font-size:10pt; font-family:Verdana; font-weight:bold; font-
            style:normal;"><td valign="top" style="color:red;"><b>Location</b></td><td valign="top"
            align="right">TZA; ZMB; </td></tr><tr bgcolor="blue"><td colspan="2" width="100" STYLE="font-
            family:Verdana; font-weight:bold; color:white">Project:</td></tr><tr bgcolor="silver" STYLE="font-
            size:10pt; font-family:Verdana; font-weight:bold; font-style:normal;"><td valign="top"
            style="color:blue;"><b>Activity ID</b></td><td valign="top" align="right">70</td></tr><tr bgcolor="gray"
            STYLE="font-size:11pt; font-family:Verdana; font-weight:bold; font-style:normal;"><td valign="top"
            style="color:black;"><b>Project Title</b></td><td valign="top" align="right">
              STA Statistics Department - Southern African Development Community - SADC
            </td></tr><tr bgcolor="gray" STYLE="font-size:10pt; font-family:Verdana; font-weight:bold; font-
            style:normal;"><td valign="top" style="color:red;"><b>Location</b></td><td valign="top"
            align="right">QMA; </td></tr></table></center></div></body></html>]]></affichDocResult>
        </affichDocResponse>
      </soap:Body>
    </soap:Envelope>
```

Figure III.8 Résultat de la Requête: DIES_Service SOAPAffichDoc

Dans ce chapitre, nous avons pu voir l'interface graphique de la Requête: DIES_Service SOAPSearchDoc. Un message SOAP est envoyé au serveur avec des informations concernant l'utilisateur, les données dont il a eu besoin et le format de l'output. Le DIES_Service rapporte l'information de la source et la transforme. Le document est, ensuite, retourné à l'utilisateur dans une enveloppe du SOAP.

Il est important de remarquer que les documents XML et les feuilles de style XSL ne sont pas dupliqués dans le DIES. Seule la référence aux documents est sauvegardée dans la base de données sous la forme d'une URL. Ainsi, le contenu et le formatage des documents sont séparés et téléchargés au moment de la demande de l'utilisateur ; ce qui assure

III Partie Pratique : Services Web

l'indépendance de la plate-forme de l'output. L'accès aux systèmes sous-jacents peut être limité au DIES afin d'éviter que les informations ne soient vues par des tiers non-autorisés [Hüsemann 2002].

4 Evaluation

L'analyse SWOT (strengths weaknesses opportunities threats) est une méthode qui permet d'analyser l'environnement externe et interne au projet. Dans l'environnement externe, on distingue les opportunités et les menaces pour le projet. Dans l'environnement interne, on distingue les forces et les faiblesses que l'entreprise transmet au projet [Site 20].

Environnement externe

Les *opportunités* constituent le domaine d'action dans lequel le projet peut espérer jouir d'un avantage différentiel. Ce dernier point, se manifeste dans une entreprise porteuse d'un projet lorsqu'elle a la possibilité d'exploiter les facteurs clés de succès plus facilement.

Les *menaces* correspondent à un problème posé par une tendance défavorable ou une perturbation de l'environnement externe au projet.

Environnement interne

Les *forces* et *faiblesses* de l'environnement interne au projet sont à évaluer sur deux critères : la performance et l'importance.

Afin d'évaluer l'utilité du DIES_Service, nous allons, tout d'abord, nous concentrer sur les exigences à l'égard de ce dernier. Après cela, nous essayerons d'appliquer les critères d'évaluation proposés par l'analyse SWOT sur notre Service Web.

4.1 Exigences à l'égard de DIES_Service

Le DIES_Service accomplit les exigences fonctionnelles telles que chercher documents et feuilles de style et la possibilité d'appeler un document formaté avec une feuille de style.

En ce qui concerne l'évaluation, il permet de séparer le contenu et du formatage des documents à échanger et donner une reproduction minimale des données dans le système.

De plus, grâce au DIES_Service, le DIES est accessible de n'importe où via le WWW. Ce qui explique que l'information confidentielle peut être rendue accessible aux personnes

III Partie Pratique : Services Web

autorisés, sans pour cela susciter de coûts d'administrions de plate-forme élevés ; étant donné que la plupart des tâches sont dirigées par les utilisateurs en question.

4.2 Forces et Faiblesses

Visual Studio .NET facilite le travail du développeur et rend plus productif le projet. En effet, lorsqu'on commence un nouveau service web, Visual Studio .NET crée tous les fichiers nécessaires au bon fonctionnement de notre service web et fournit quelques exemples de code.

L'environnement choisi pour ce projet a une particularité, c'est qu'il ne permet pas au programmeur d'affiner le WSDL. Cette particularité peut être vue comme un défaut, dans la mesure où elle ne permet pas au programmeur toute la liberté dont il a parfois besoin.

En fin, cette particularité peut être intéressante dans le cas qu'elle facilite la programmation, mais qu'elle peut être vue comme un défaut aussi (pas de liberté d'action).

4.3 Opportunités et Menaces

Le Service Web a comme opportunité, la possibilité de relier de manière standardisée les SI hétérogènes des organisations humanitaires et d'échanger les documents de projet. Ce qui favoriserait la collaboration avec les nouveaux partenaires. D'autant plus, qu'une bonne coordination de projets est très importante voire même essentielle pour les ONG afin d'éviter certaines inefficacités. Le service Web DIES_Service demeure facile à installer sur le serveur grâce aux programmes DIES_ServiceSetup mis en place.

En revanche, ce service court un danger qui réside essentiellement dans le risque de la divulgation d'informations confidentielles. En effet, ce projet contient quelques lacunes au niveau de la sécurité vu l'inexistence du champ « mot de passe » qui limite les utilisateurs agréés. De plus, il n'y a pas de cryptage de la communication.

Néanmoins, afin de pouvoir remédier à ce problème deux solutions pourraient être envisageables : à savoir la sécurisation de l'infrastructure et celle des connexions.

En effet, concernant la sécurisation de l'infrastructure, Microsoft offre dores et déjà diverses technologies qui, intégrées dans un plan de sécurité global, permettent à une entreprise d'assurer la sécurité de son infrastructure informatique. Par ailleurs, plusieurs

III Partie Pratique : Services Web

techniques peuvent permettre la sécurisation des connexions comme la mise en place d'un pare-feu, du protocole SSL (Secure Sockets Layer) et des réseaux privés virtuels (VPN, Virtual Private Network).

Pour l'implémentation de DIES_Service, je n'ai pas opté pour l'approche technique Contract First Development qui consiste à commencer tout d'abord par l'implémentation de WSDL. En effet, les avantages en commençant l'implémentation par le WSDL et le schéma consiste en ayant un schéma plus descriptif, une plus grande flexibilité lors d'un changement du code et facilite l'utilisation des standards car ils sont écrits pour la description WSDL et les messages SOAP [Site 22]. En plus, cette approche permet que l'interface demeure inchangeable lors d'une modification d'une ligne de code. Toutefois, il existe un lien serré entre le WSDL et la méthode ce qui est un inconvénient. De plus, la gestion des erreurs n'est pas encore au point.

Enfin, la structure du format des réponses ne correspond pas à un schéma prédéfini mais à une structure produite par le .Net de manière automatique.

IV *Conclusions*

IV Conclusions

Ce travail a donné une vue d'ensemble des problèmes de partage d'information qui se produisent dans les projets du développement humanitaire. Le concept DIES en tant qu'architecture technique pourrait permettre aux partenaires du projet d'échanger les rapports entre organisations.

La coordination de projets devrait être décentralisée afin d'éviter les inefficacités. En effet, le DIES ne coordonne pas, mais il met à disposition les informations aux partenaires. Par conséquent, il encourage la décentralisation et la consolidation de la prise de décision. Le DIES améliore l'impact des projets du développement et aide les organisations humanitaires à optimiser l'allocation de leurs ressources limitées.

Résultats du travail

Durant ce travail, j'ai pu voir de près le fonctionnement du Service Web, son fonctionnement ainsi que les standards qui y sont liés. Ces derniers se composent d'une collection de standards que l'on regroupe sous le terme de *Web Services Protocol Stack*. En voici donc quelques-uns :

- ✚ XML : Toutes les données à échanger sont formatées en XML.
- ✚ WSDL : L'interface publique au service Web est décrite par ce protocole en cours de normalisation. C'est une description XML qui décrit la façon de communiquer en utilisant le service Web.
- ✚ UDDI : Le service Web est connu sur le réseau au moyen de ce protocole. Il permet à des applications de rechercher le service web dont elles ont besoin.
- ✚ BPEL4WS : (Business Process Execution Language for Web Services) fournit un langage pour la spécification des processus business et le protocole d'interaction business. Ceci dit, il permet d'étendre le modèle d'interaction des services web afin d'assurer des transactions Business. BEPL définit donc un modèle d'intégration inter-opérable dans les mondes inter-entreprises et intra-entreprises.

Mon expérience personnelle

Pour l'implémentation du Service Web **DIES_Service**, on a opté pour le .NET. Ceci, pour deux raisons principales. Premièrement la plate-forme DIES est implémentée et fonctionne suivant les technologies Microsoft. Deuxièmement, cette recherche m'a permis de me familiariser avec un nouvel outil de programmation qui est le « Visual Studio » et son langage de programmation .Net qui se révèle un avantage dans mon avenir professionnel.

Perspectives futures

Au long de notre travail, nous avons vu qu'un Service Web est un phénomène assez récent. De plus, il suscite un grand intérêt de la part des développeurs des systèmes d'information. Les Services Web sont considérés comme étant une solution idéale, intégrant des fonctionnalités déjà implémentées, afin de construire et développer des systèmes d'information.

Une des raisons qui encourage à créer des Service Web est son interopérabilité entre divers logiciels fonctionnant sur diverses plates-formes. De plus, les Services Web utilisent des standards et des protocoles ouverts et peuvent fonctionner au travers de nombreux Firewalls sans pour cela nécessiter des changements sur les règles de filtrage.

Ce projet pourrait être utilisé par des organisations humanitaires, pour autant d'assurer l'objectif principal qui est de séparer le contenu ainsi que le format des documents à échanger, l'interopérabilité.

V *Références*

Pages Web

- [Site 1] W3C World Wide Web Consortium; “Web Services Architecture”; W3C Working Group Note 11; February 2004; <http://www.w3.org/TR/ws-arch/> Dernier accès le 02.12.2006
- [Site 2] W3C World Wide Web Consortium; “Extensible Markup Language” (XML) <http://www.w3.org/XML/> Dernier accès le 02.12.2006
- [Site 3] W3C World Wide Web Consortium; “XML Schema” <http://www.w3.org/XML/Schema> Dernier accès le 02-12.2006
- [Site 4] W3C World Wide Web Consortium; “The Extensible StyleSheet Language (XSL)” <http://www.w3.org/Style/XSL/> Dernier accès le 02.12.2006
- [Site 5] W3C World Wide Web Consortium; “XML Path Language (XPath)” <http://www.w3.org/TR/xpath> Dernier accès le 02.12.2006
- [Site 6] W3C World Wide Web Consortium; “Web Ontology Language (OWL)” <http://www.w3.org/2004/OWL/> Dernier accès le 02.12.2006
- [Site 7] W3C World Wide Web Consortium; “SOAP Version 1.2 Partie 1: Structure pour l'échange de messages, Recommandation W3C 24 June 2003” <http://www.w3.org/2002/07/soap-translation/soap12-part1.html> Dernier accès le 02.12.2006
- [Site 8] W3C World Wide Web Consortium; “Web Services Description Language (WSDL)” <http://www.w3.org/TR/2005/WD-wsdl20-adjuncts-20050803/> Dernier accès le 02.12.2006
- [Site 9] UDDI Executive Overview: Enabling Service-Oriented Architecture, disponible à <http://www.uddi.org/whitepapers.html> Dernier accès le 02.12.2006
- [Site 10] « DAML Services » disponible à <http://www.daml.org/services/owl-s/> Dernier accès le 02.12.2006
- [Site 11] “The ebXML Messaging Service” <http://webservices.xml.com/pub/a/ws/2003/03/18/ebxml.html> Dernier accès le 02.12.2006
- [Site 12] “WSDL Usage Experience with XML Schema” <http://www.w3.org/2005/05/25-schema/WSDL.html> Dernier accès le 02.12.2006
- [Site 13] Site du consortium OASIS: <http://www.oasis-open.org/home/index.php> Dernier accès le 02.12.2006
- [Site 14] Resource Description Framework disponible à www.w3.org/RDF/ Dernier accès le 02.12.2006

V Références

- [Site 15] Exemple du fichier de description d'un service web (WSDL)
<http://search.cpan.org/src/STODGHIL/SOAP-Clean-0.02/t/example-Temperature.xml>
Dernier accès le 02.12.2006
- [Site 16] BPEL4WS: Disponible à:
<http://msdn.microsoft.com/library/default.asp?url=/library/enus/dnbiz2k2/html/bpel1-1.asp> Dernier accès le 02.12.2006
- [Site 17] RPC Vs Doc Style: Disponible à:
<http://www.stylusstudio.com/xmldev/200406/post50190.html>] Dernier accès le 02.12.2006
- [Site 18] Description de services web: WSD: <http://dyomedea.com/papers/2004-wsc/3-soap.html> Dernier accès le 17.01.2007 Dernier accès le 18.01.2007
- [Site 19] L'attribut WebMethod:
[http://msdn2.microsoft.com/fr-fr/library/byxd99hx\(VS.80\).aspx](http://msdn2.microsoft.com/fr-fr/library/byxd99hx(VS.80).aspx)
Dernier accès le 19.01.2007
- [Site 20] SWO: http://erwan.neau.free.fr/Toolbox/Analyse_SWOT.htm
Dernier accès le 21.01.2007
- [Site 21] Contract-First Development
<http://msdn.microsoft.com/msdnmag/issues/05/06/ServiceStation/>
Dernier accès le 08.03.2007
- [Site 22] Contract-First Web services (CFWS)
http://webservices.sys-con.com/read/143909_1.htm
Dernier accès le 08.03.2007
- [Site 23] Outils de simulation du Service Web: SOAPUI
<http://soapui.org> Dernier accès le 30.03.2007
- [Site 24] XMethods disponible à: <http://xmethods.net>
Dernier accès le 30.03.2007
- [Site 25] Opérateur de l'Annuaire UDDI universel: IBM, disponible à:
www.ibm.com/services/uddi/ Dernier accès le 30.03.2007

V Références

- [Site 26] Opérateur de l'Annuaire UDDI universel: Microsoft, disponible à:
uddi.microsoft.com Dernier accès le 30.03.2007
- [Site 27] Opérateur de l'Annuaire UDDI universel: Hewlett-Packard, disponible à:
uddi.hp.com Dernier accès le 30.03.2007
- [Site 28] Opérateur de l'Annuaire UDDI universel: SAP, disponible à:
udditest.sap.com Dernier accès le 30.03.2007
- [Site 29] RPC Vs Document Style, disponible à:
http://searchwebservicestechtarget.com/ateQuestionNResponse/0,289625,sid26_cid494324_tax289201,00.html Dernier accès le 31.03.2007
- [Site 30] RPC Vs Document, disponible à:
<http://www-128.ibm.com/developerworks/library/ws-castor/>
Dernier accès le 31.03.2007

Livres

- [Chauvet 2002] Chauvet, Jean-Marie: Services Web avec SOAP, WSDL, UDDI, ebXML... Editions Eyrolles, Paris 2002.
- [Birman 1996] Birman, Kenneth P: Technologies, Web Services, and Applications. Springer New York, Heidelberg, Berlin 1996.
- [Daniel 2003] Daniel, Jérôme: Services Web - Concepts, techniques et outils. Vuibert, Paris, 2003.
- [Singh 2004] Singh I, Brydon S, Murray G, Ramachandran V, Violleau T, Stearns B: Designing Web services with the J2EE 1.4 Platform JAX-RPC, SOAP, and XML Technologies. Addison Wesley 2004.
- [Hüsemann 2005] Hüsemann Stefan, Cours du système d'information I.
- [Hüsemann 2002] Hüsemann Stefan, Dissertation: WEB-BASIERTE INFORMATIONSAUSTAUSCHPLATTFORM IM BEREICH INTERNATIONALER HUMANITÄRER PROJEKTE: PROBLEME UND LÖSUNGSVORSCHLÄGE.
- [UML 2002] Morley C, Hugues J, Leblanc B: UML POUR L'ANALYSE D'UN SYSTEME D'INFORMATION. Dunod, Paris, 2002.

V Références

[Nicolescu 2004] Nicolescu Matthieu, WebServices: SUPINFO DOT NET TRAINING COURSE. Novembre 2004.

[Meier 2006] Meier Andreas, Introduction pratique aux bases de données relationnelles. 2^{ème} Edition 2006.

VI *Annexes*

1 Annexe A : Code source de SearchDoc

```
[WebMethod]
public DataSet SearchDoc(string username, string typeDoc,
int max_result, string organisation, string titre_doc)
{
    OdbcConnection cn;
    OdbcDataAdapter da;
    DataTable dt;
    DataSet ds;
    string sql, connectionString;
    int UserID;

    try
    {
        ///////////////vérification du code utilisateur/////////////////

        connectionString =

        System.Configuration.ConfigurationManager.AppSettings["connect
ionString"];

        cn = new OdbcConnection(connectionString);

        sql = "select Person.ID from Person where Person.Code='" +
            username + "'";

        da = new OdbcDataAdapter(sql, cn);

        dt = new DataTable("Person");

        da.Fill(dt);

        if (dt.Rows.Count > 0)
            UserID = int.Parse(dt.Rows[0]["ID"].ToString());
        else throw new Exception("Utilisateur inexistant");

        ////////////////////////////////////////////////////////////////////XML//////////////////////////////////////////////////////////////////
    }
}
```

```

if (typeDoc.ToUpper().Trim() == "XML")
{
    sql = " set rowcount " + max_result.ToString() +
        " select XML_documents.*,
            isnull(organisation.organisation_name, '')
            as organisation_name " +
        " from XML_documents inner join
            XML_document_can_be_used_by_person on
            (XML_documents.key_document=
            XML_document_can_be_used_by_
            person.fkey_XML_document)" +
        " left outer join data_provider_system on
            (data_provider_system.key_system=
            XML_documents.document_belongs_
            to_Data_provider_system)" +
        " left outer join organisation_is_data_provider on
            (organisation_is_data_provider.fKey_data_provider=
            data_provider_system.key_system)" +
        " left outer join organisation on
            (organisation_is_data_provider.fKey_organisation=
            organisation.Key_organisation)" +
        " where XML_document_can_be_used_by_person.fkey_person=
        " + UserID.ToString() +
        " and isnull(organisation.organisation_name, '')
            like '%" + organisation + "%'" +
        " and XML_documents.document_name like '%" +
            titre_doc + "%'" +
        " set rowcount 0";

    da = new OdbcDataAdapter(sql, cn);
    dt = new DataTable();
    da.Fill(dt);
    ds = new DataSet("XML_documents");
    ds.Tables.Add(dt);

    return ds;
}

```

//////////////////////////////////STYLESHEET////////////////////////////////////

```

if (typeDoc.ToUpper().Trim() == "STYLESHEET")
{
    sql = " set rowcount " + max_result.ToString() +
        " select stylesheet.*,
            isnull(organisation.organisation_name, '')
            as organisation_name" +
        " from stylesheet inner join
            stylesheet_can_be_used_by_person on
            (stylesheet.key_stylesheet= stylesheet_can_be_used_by_
            person.fkey_stylesheet)" +
        " left outer join stylesheet_belongs_to_organisation on
            (stylesheet_belongs_to_organisation.fKey_stylesheet=
            stylesheet.key_stylesheet)" +
        " left outer join organisation on
            (stylesheet_belongs_to_organisation.fKey_organisation=
            organisation.Key_organisation)" +
        " where stylesheet_can_be_used_by_person.fkey_person=
        " + UserID.ToString() +
        " and isnull(organisation.organisation_name, '')
            like '%" + organisation + "%'" +
        " and stylesheet.stylesheet_name
            like '%" + titre_doc + "%'" +
        " set rowcount 0";
    da = new OdbcDataAdapter(sql, cn);
    dt = new DataTable("Stylesheet");
    da.Fill(dt);
    ds = new DataSet("XML_documents");
    ds.Tables.Add(dt);
    return ds;
}
}
catch (Exception ex)
{
    throw new Exception(ex.Message);
}
return null;
}
        if (typeDoc.ToUpper().Trim() == "XML")
        {
            sql = " set rowcount " + max_result.ToString() +

```

```

" select XML_documents.*,
    isnull(organisation.organisation_name, '')
    as organisation_name " +
" from XML_documents inner join
    XML_document_can_be_used_by_person on
(XML_documents.key_document=
    XML_document_can_be_used_by_
    person.fkey_XML_document)" +
" left outer join data_provider_system on
    (data_provider_system.key_system=
    XML_documents.document_belongs_
    to_Data_provider_system)" +
" left outer join organisation_is_data_provider on
    (organisation_is_data_provider.fKey_data_provider=
    data_provider_system.key_system)" +
" left outer join organisation on
    (organisation_is_data_provider.fKey_organisation=
    organisation.Key_organisation)" +
" where XML_document_can_be_used_by_person.fkey_person=
" + UserID.ToString() +
" and isnull(organisation.organisation_name, '')
    like '%" + organisation + "%'" +
" and XML_documents.document_name like '%" +
    titre_doc + "%'" +
" set rowcount 0";

```

```

da = new OdbcDataAdapter(sql, cn);
dt = new DataTable();
da.Fill(dt);
ds = new DataSet("XML_documents");
ds.Tables.Add(dt);

    return ds;
}

```

```

//////////////////////////////////////STYLESHEET//////////////////////////////////////

```

```

if (typeDoc.ToUpper().Trim() == "STYLESHEET")
{
    sql = " set rowcount " + max_result.ToString() +

```

```

        " select stylesheet.*,
          isnull(organisation.organisation_name, '')
          as organisation_name" +
        " from stylesheet inner join
          stylesheet_can_be_used_by_person on
          (stylesheet.key_stylesheet= stylesheet_can_be_used_by_
          person.fkey_stylesheet)" +
        " left outer join stylesheet_belongs_to_organisation on
          (stylesheet_belongs_to_organisation.fKey_stylesheet=
          stylesheet.key_stylesheet)" +
        " left outer join organisation on
          (stylesheet_belongs_to_organisation.fKey_organisation=
          organisation.Key_organisation)" +
        " where stylesheet_can_be_used_by_person.fkey_person=
        " + UserID.ToString() +
        " and isnull(organisation.organisation_name, '')
          like '%" + organisation + "%'" +
        " and stylesheet.stylesheet_name
          like '%" + titre_doc + "%'" +
        " set rowcount 0";
    da = new OdbcDataAdapter(sql, cn);
    dt = new DataTable("Stylesheet");
    da.Fill(dt);
    ds = new DataSet("XML_documents");
    ds.Tables.Add(dt);
    return ds;
  }
}
catch (Exception ex)
{
    throw new Exception(ex.Message);
}
return null;
}

```

Code VI-1 Méthode "SearchDoc"

2 Annexe B : Code source de AffichDoc

```
[WebMethod(Description = "", EnableSession = true)]

public string affichDoc(string docName, string styleName)
{
    OdbcConnection cn;
    OdbcDataAdapter da;
    DataTable dt;

    string sql, connectionString, documentURL, styleURL;

    try
    {

        connectionString = System.Configuration.ConfigurationManager.
            AppSettings["connectionString"];

        cn = new OdbcConnection(connectionString);

        sql = " select document_url from DIES_
            User.XML_documents where

            document_name ='" + docName.Trim() + "'";

        da = new OdbcDataAdapter(sql, cn);

        dt = new DataTable("XML_documents");

        da.Fill(dt);

        if (dt.Rows.Count > 0)
            documentURL = dt.Rows[0]["document_url"].ToString();
        else throw new Exception(
            "Le document XML recherché n'existe pas"
            );
    }
}
```

```
        sql = " select stylesheet_URL from DIES_User.stylesheet where  
                stylesheet_name ='" + styleName.Trim() + "'";  
  
        da = new OdbcDataAdapter(sql, cn);  
  
        dt = new DataTable("Stylesheet");  
  
        da.Fill(dt);  
  
        if (dt.Rows.Count > 0) styleURL =  
            dt.Rows[0]["stylesheet_URL"].ToString();  
        else throw new Exception(  
                "Le style XSL recherché n'existe pas"  
            );  
  
        return Transform(documentURL, styleURL);  
    }  
    catch (Exception ex)  
    {  
  
        return "Document or Stylesheet not accessible at the moment...";  
    }  
}
```

Code VI-2 Méthode "AffichDoc"

3 Annexe C : Code Source de Transform

```

public string Transform(string xml, string stylesheet)
{
    XmlDocument XDoc;
    System.Xml.Xsl.XslCompiledTransform XTrans;
    XmlTextWriter wr;
    System.IO.StreamReader rd;
    HttpRequest req = HttpContext.Current.Request;
    string t = "", tmpPath = Server.MapPath(req.ApplicationPath) +
        "\\tmp.htm";

    try
    {
        //instancier l'objet XmlDocument qui servira à lire le fichier
        XML
        XDoc = new XmlDocument();
        //instancier l'objet XslCompiledTransform qui servira à lire
        le fichier XSL
        XTrans = new System.Xml.Xsl.XslCompiledTransform();

        //charger le fichier XML
        XDoc.Load(xml);
        //charger le fichier XSL
        XTrans.Load(stylesheet);

        //créer le fichier HTML temporaire
        wr = new XmlTextWriter(tmpPath, System.Text.Encoding.UTF8);
        //transformer le fichier XML
        XTrans.Transform(XDoc, null, wr);
        //fermer le stream
        wr.Close();

        //instancier l'objet StreamReader en donnant le path du
        fichier HTML temporaire
        rd = new System.IO.StreamReader(tmpPath);
        //lire le contenu du fichier HTML
        t = rd.ReadToEnd();
        //fermer le stream
        rd.Close();
        //suppression du fichier temporaire
        try
        {
            System.IO.File.Delete(tmpPath);
        }
        catch (Exception) { }
    }
    catch (Exception ex)
    {
        throw new Exception(ex.Message);
    }
    //retourner le contenu HTML
    return t;
}

```

Code VI-3 Méthode "Transform"

4 **Annexe D : WSDL**

```

<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/" xmlns:tns="http://localhost:1036"
xmlns:s="http://www.w3.org/2001/XMLSchema" xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/" targetNamespace="http://localhost:1036"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  <wsdl:types>
    <s:schema elementFormDefault="qualified" targetNamespace="http://localhost:1036">
      <s:element name="SearchDoc">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="username" type="s:string"/>
            <s:element minOccurs="0" maxOccurs="1" name="typeDoc" type="s:string"/>
            <s:element minOccurs="1" maxOccurs="1" name="max_result" type="s:int"/>
            <s:element minOccurs="0" maxOccurs="1" name="organisation" type="s:string"/>
            <s:element minOccurs="0" maxOccurs="1" name="titre_doc" type="s:string"/>
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="SearchDocResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="SearchDocResult">
              <s:complexType>
                <s:sequence>
                  <s:element ref="s:schema"/>
                  <s:any/>
                </s:sequence>
              </s:complexType>
            </s:element>
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="affichDoc">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="docName" type="s:string"/>
            <s:element minOccurs="0" maxOccurs="1" name="styleName" type="s:string"/>
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="affichDocResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="affichDocResult" type="s:string"/>
          </s:sequence>
        </s:complexType>
      </s:element>
    </s:schema>
  </wsdl:types>
  <wsdl:message name="SearchDocSoapIn">
    <wsdl:part name="parameters" element="tns:SearchDoc"/>
  </wsdl:message>

```

```

<wsdl:message name="SearchDocSoapOut">
  <wsdl:part name="parameters" element="tns:SearchDocResponse"/>
</wsdl:message>
<wsdl:message name="affichDocSoapIn">
  <wsdl:part name="parameters" element="tns:affichDoc"/>
</wsdl:message>
<wsdl:message name="affichDocSoapOut">
  <wsdl:part name="parameters" element="tns:affichDocResponse"/>
</wsdl:message>
<wsdl:portType name="DIES_ServiceSoap">
  <wsdl:operation name="SearchDoc">
    <wsdl:input message="tns:SearchDocSoapIn"/>
    <wsdl:output message="tns:SearchDocSoapOut"/>
  </wsdl:operation>
  <wsdl:operation name="affichDoc">
    <wsdl:input message="tns:affichDocSoapIn"/>
    <wsdl:output message="tns:affichDocSoapOut"/>
  </wsdl:operation>
</wsdl:portType>
<wsdl:binding name="DIES_ServiceSoap" type="tns:DIES_ServiceSoap">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http"/>
  <wsdl:operation name="SearchDoc">
    <soap:operation soapAction="http://localhost:1036/SearchDoc" style="document"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="affichDoc">
    <soap:operation soapAction="http://localhost:1036/affichDoc" style="document"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
</wsdl:binding>
<wsdl:binding name="DIES_ServiceSoap12" type="tns:DIES_ServiceSoap">
  <soap12:binding transport="http://schemas.xmlsoap.org/soap/http"/>
  <wsdl:operation name="SearchDoc">
    <soap12:operation soapAction="http://localhost:1036/SearchDoc" style="document"/>
    <wsdl:input>
      <soap12:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap12:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="affichDoc">
    <soap12:operation soapAction="http://localhost:1036/affichDoc" style="document"/>
    <wsdl:input>
      <soap12:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap12:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>

```

VI Annexes

```
</wsdl:binding>
<wsdl:service name="DIES_Service">
  <wsdl:port name="DIES_ServiceSoap" binding="tns:DIES_ServiceSoap">
    <soap:address location="http://localhost:1037/DIES/DIES_Service.asmx"/>
  </wsdl:port>
  <wsdl:port name="DIES_ServiceSoap12" binding="tns:DIES_ServiceSoap12">
    <soap12:address location="http://localhost:1037/DIES/DIES_Service.asmx"/>
  </wsdl:port>
</wsdl:service>
</wsdl:definitions>
```

Code VI-4 WSDL