République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique.
Université A/Mira de Bejaia.
Faculté des Sciences Exactes
Département d'Informatique

Ecole Doctorale d'Informatique

Mémoire de Magistère En Informatique

Option : Réseaux et Systèmes Distribués

Thème :

Monitoring des Web services au temps d'exécution

Présenté par :

ZIANE Meriem Epse GHERBI

Dirigé par :

Pr Mohand-Said.HACID

Mai 2009

Je dédie ce travail :

A mes très chers parents,
A mon très cher époux Badreddine,
A mes très chères sœurs Lynda et Sabrina,
A mes très chers frères A.Rahim et M^{ed} Lamine,
A mes très chères nièces Sarah et Maria,
A ma très chère amie Mouna,
A toute ma belle famille,
Ainsi qu'à toute ma famille.

Meriem.

Remerciements

Je tiens à remercier mon encadreur, Mohand Said HACID Professeur à l'université Claude Bernard de Lyon, pour avoir accepté de diriger ce travail. Mes remerciements les plus vifs vont aussi à Monsieur SABAHI Samir, pour tous ses conseils, ses encouragements, sa patience tout au long de cette période.

Je remercie Monsieur AbdelKamel Tari, Directeur de notre école doctorale, pour tous les conseils les plus constructifs qu'il nous a prodigués en permanence durant notre formation.

Je remercie tous les enseignants qui sont intervenus durant l'année théorique de ce Magister, afin de nous faire dispenser de leurs cours de grande qualité et de haut niveau.

Je tiens à remercier les différents membres du jury qui m'ont fait l'honneur de juger ce travail.

Je voudrai adresser ma gratitude à ma famille, surtout mes très chers parents, mon très cher époux, mes sœurs et mes frères pour leurs soutient et encouragements.

Résumé

Un des aspects critiques dans le développement des systèmes basés sur les services, est le besoin d'un support effective, flexible et facile à utiliser pour le cycle entier des processus métiers distribués, avec leur inévitable évolution qui nécessite une adaptation aux changements. De nos jours, le développement et la composition de services est une affaire statique. Toutes les interactions de services sont anticipées et connues à l'avance, et il y a une parfaite correspondance entre les signatures des entrées/sorties et les fonctionnalités.

Des compositions de services ad hoc et dynamiques sont en demande croissante dans le domaine des méthodologies du workflow. Afin de supporter la conception et le développement des SBA : applications basées sur les services, le monitoring et la gestion des services sont devenus des besoins fondamentaux. Le développement des services composites nécessite l'utilisation de mécanismes qui les fournissent dans des systèmes sains implantant les services. Les erreurs et les changements d'un composant d'une application peuvent conduire au mauvais fonctionnement de plusieurs applications d'une entreprise. Pour remédier à de telles situations, les entreprises ont besoin de surveiller constamment le bon fonctionnement de leurs applications lors de l'exécution, et prendre des mesures quand on est en présence de changement de situations, d'environnement ou la coordination des partenaires.

Les recherches qui visent actuellement le monitoring se focalisent sur les techniques dynamiques capables de capter des aspects pertinents du comportement des services composites définis (BPEL processes), ou encore sur la négociation incorporant des politiques de sécurité. D'autres travaux se sont focalisés sur la qualité de services en s'appuyant sur des métrique, et d'autres sur la notion de contrat avec des politiques (Service Level Agreement). Donc les travaux de recherche sur le monitoring sont épars. Cependant, les cadres développés restent toujours incomplets incluant les aspects discutés auparavant.

Mots clés : Services Web, Architecture Orienté Service (SOA), Applications Basés sur les Services (SBA), monitoring de services, processus métier, vie privé.

Table des matières

Liste des illustrations

Liste des figures

Extraits de code

Introduction générale

Le développement de l'Internet a favorisé l'échange de données entre partenaires et la multiplication de services en ligne. Des plateformes de services évoluées permettent aujourd'hui la conception, le déploiement et la mise en oeuvre de ces services dans des échelles de temps réduites. Au sein de ces plateformes, les services peuvent être perçus comme des composants, et des modèles de composition permettant de les combiner pour aboutir à des services plus élaborés.

Ces services présentent par conséquent une forte dynamicité, d'où la nécessité de créer des plateformes de monitoring qui soient adaptées. La richesse des spécifications fournies par les modèles de composition peut constituer un socle pour ce monitoring.

Le monitoring peut être défini comme un processus de rassemblement et le rapport d'informations appropriées sur l'exécution et l'évolution d'applications basées sur des services. Notre travail consiste à trouver de nouvelles méthodes permettant le monitoring des applications basées sur des services (SBA).

1. La problématique

Les services Web soutiennent un style architectural dynamique où la liaison entre les composants peut changer en temps d'exécution. De nouveaux services peuvent être développés et publiés dans l'annuaire et ensuite découvert par des clients. Des services précédemment disponibles peuvent disparaître ou devenir indisponibles. Cette situation a été caractérisée ailleurs par le terme openworld software [01], où les applications sont composées de parties qui peuvent changer d'une manière non prévisible et dynamiquement.

Il a été observé que openworld software présente l'exigence de validation continue [05]. Puisqu'une architecture logicielle se développe dynamiquement, la vérification doit s'étendre du temps de développement au temps d'exécution.

Les applications conventionnelles sont complètement vérifiées avant le déploiement, et les tests sont les moyens habituels de découvrir les défaillances. En revanche, les applications orientées services peuvent fortement changer au temps d'exécution : par exemple,

- Elles peuvent lier à différents services selon le contexte dans lequel elles sont exécutées
- Les fournisseurs du service peuvent modifier le contenu de leurs services.
- Les nouvelles versions des services choisis ainsi que les nouveaux services assurés par différents fournisseurs de différents contextes d'exécution pourraient entraver les niveaux d'exactitude et de qualité de ces applications.

Les activités de test ne peuvent pas prévoir tous ces changements, et elles ne peuvent pas être aussi puissantes qu'avec d'autres applications d'où la nécessité d'un système de monitoring pour surveiller le comportement des services Web d'une manière continue en temps d'exécution.

2. Objectifs du travail

Dans notre travail, on va baser sur le monitoring de la privacy dans les services Web, comment le fournisseur du service doit protéger et traiter les informations personnelles du consommateur.

Le but de notre travail est d'atteinte les points suivants :

- Etablir un état de l'art sur les services Web.
- Etudier les différentes approches de monitoring des services Web en citant quelques travaux de recherches sur ce domaine
- Faire une comparaison sur les différents travaux de recherches sur le monitoring.
- Développement des techniques du monitoring de la privacy (la vie privée) en fonction du contexte et tenant compte du rôle de l'utilisateur (ses données privés).
- Développement d'un langage de monitoring pour spécifier les propriétés de la privacy.

3. Structure du travail

Ce travail est structuré comme suit : Dans la première partie, nous établirons un état de l'art sur les services Web. Puis dans la deuxième partie, nous détaillerons les différents travaux de recherche en matière de monitoring des Services Web. Dans la troisième partie nous introduisant nos contributions. En conclusion, nous terminons par donner quelques limites de notre proposition et quelques perspectives.

Partie I :

État de l'art sur les services Web

Cette partie est consacrée aux différents concepts de services Web. Nous commençons cette partie en évoquant quelques concepts reliés aux services Web. Par la suite, nous étudions comment les services Web peuvent interagir grâce aux couches standard composant leur architecture, nous présenterons les différents standards sous-jacents. Enfin, nous terminons par une présentation de la composition de services Web.

I.1. Introduction

L'abstraction est une des plus importantes dimensions du développement de l'informatique. Celle-ci apporte, au fil des (r)évolutions, des solutions de plus en plus pragmatiques. L'abstraction concerne, pour une part, l'interaction entre applications. Cette relation porte plusieurs noms : interaction, interopérabilité, échange… Elle est aussi passée par plusieurs modèles client-serveur, fournisseur-consommateur toujours pour un même objectif : l'échange d'informations.

Les services Web répondent à une double demande : celle d'un échange sûr et celle d'un échange ouvert. Il s'agit de donner les moyens à deux applications d'échanger simplement. L'échange ouvert est permis, puisqu'un serveur ne doit pas nécessairement connaître à l'avance son client pour offrir son service vers l'extérieur [09].

Avant d'expliquer plus en détails le fonctionnement des *services Web,* il me parait important d'introduire l'architecture orientée service. Car comme leur nom l'indique, les *services Web* respectent l'architecture orientée services *SOA.*

I.2. SOA

I.2.1. Définition

Dans une architecture SOA (Service Oriented Architecture), la fonctionnalité d'une application est fractionnée en sections appelées « services » qui peuvent être distribués dans l'ensemble d'un réseau, jumelés et réutilisés pour créer d'autres applications de gestion. En se concentrant sur les services, les fonctionnalités des applications sont davantage fondé sur les processus opérationnels et permet ainsi aux applications de répondre plus fidèlement à ses attentes.

L'objectif de cette architecture et de renverser la tendance qui, trop fréquemment, démontre que les processus opérationnels à l'intérieur des entreprises se plient aux exigences des contraintes imposées par l'informatique [38].

La SOA peuvent se définir de la façon suivante :

- Le système d'information de l'entreprise est découpé en services. Le niveau de granularité n'est plus l'application mais le service. Les services délèguent les traitements aux applications, mais fournissent une interface indépendante de ces applications.

- Les services transverses utilisés par les applications sont mutualisés : ainsi, la notification, la sécurité ou le logging sont accessibles sous forme de services. Il n'est plus nécessaire, par exemple, de re-développer la gestion du logging pour chaque application.

- Les échanges entre les services sont gérés grâce à des fonctionnalités de Service Management. Il est alors possible de diagnostiquer et de prévenir les problèmes de montée en charge, d'indisponibilité d'une application, etc. Le Service Management permet aussi de remonter des informations basées sur des indicateurs métiers, à destination des décideurs.

- Les services sont utilisés dans les processus métiers de l'entreprise. On dépasse largement la notion de connecteur applicatif : les processus n'accèdent pas à des applications via des connecteurs, ils accèdent aux services du système d'information.

I.2.2. Composants d'une SOA

Une architecture de services est constituée de trois composants [03]. Le premier est le prestataire de services. Vient ensuite le demandeur du service (le client), autrement dit le composant qui accède au service. Enfin, une agence de services fournit des services de découverte et d'enregistrement.

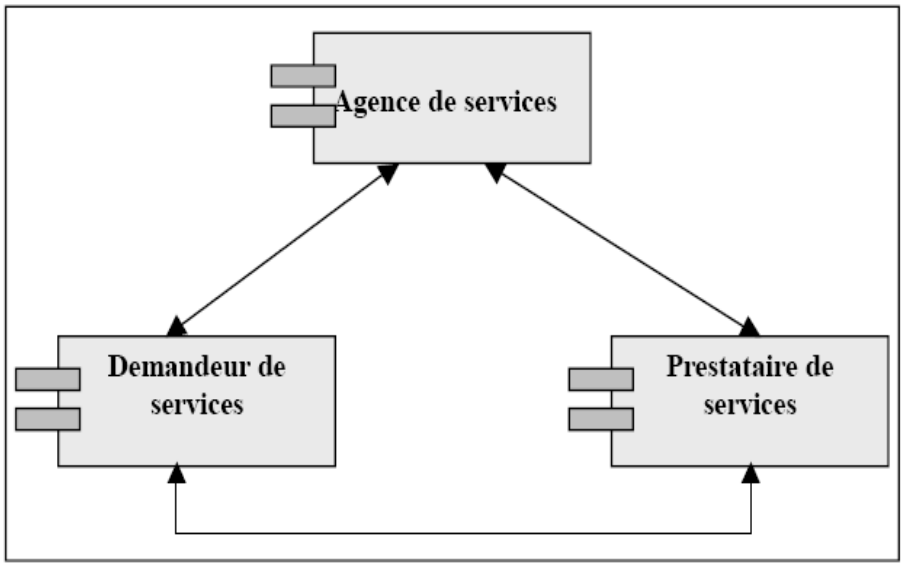


Figure I.1 : *Les éléments d'un service Web***.**

- ***Prestataire de services (serveur)***

  Un Prestataire de services est la source de la fonctionnalité des services. Il crée le service Web, puis publie un contrat d'interface ainsi que les informations d'accès au service, dans une agence de services.

- ***Agence de services***

  Une agence de services est un registre des services disponibles. Chaque prestataire publie son contrat d'interface à l'agence avec les informations à utiliser pour localiser le service. Le demandeur recherche les services appropriés dans l'agence et extrait le contrat d'interface correspondant.

- ***Demandeur de service (client)***

  Le client accède à l'agence de service pour effectuer une recherche afin de trouver les services désirés. Le client extrait le contrat d'interface correspondant de l'agence. Il utilise le contrat d'interface pour comprendre comment (méthodes disponibles) et où (adresse) accéder au service. Ensuite, il se lie au prestataire de service pour invoquer le service.

La présentation des composants SOA ci-dessus nous permet d'affirmer que les services Web implémentent l'architecture orientée service. Le prestataire de services est le service Web, le demandeur de service est le client du service et l'agence de services est l'annuaire UDDI.

I.2.3. Les objectifs de l'architecture orientée services SOA [4]

Les objectifs majeurs de *SOA* sont:

- La réutilisation et la composition, permettent le partage de modules entre applications et les échanges inter-applicatifs.
- La pérennité, qui implique notamment le support des technologies existantes et à venir.
- L'évolutivité, car toute application est vivante peut se voir ajouter de nouveaux modules et doit pouvoir répondre aux nouveaux besoins fonctionnels.
- L'ouverture et l'interopérabilité, pour partager des modules applicatifs entre plates formes et environnements.
- La distribution, pour pouvoir utiliser ces modules à distance et les centraliser au sein de l'entreprise par exemple.
- La performance, avec en priorité l'accent mis sur la montée en charge.

I.3. Service Web :

L'objectif de cette partie est d'introduire le concept de Services Web. Nous commencerons par étudier l'architecture générale de cette technologie avec son principe. Puis, nous présenterons les différents standards sous-jacents : SOAP[06], WSDL[07], UDDI[08]. Enfin, nous terminons en présentant la composition de services

I.3.1. Définition

Les services Web sont un ensemble de standards permettant aux applications d'interagir au travers du Web. Il place à la disposition des clients des méthodes pouvant être appelées à distance. L'interopérabilité est possible grâce à l'utilisation de la technologie SOAP (Simple Object Access Protocol) permettant de décrire des objets sous forme de texte structuré au format XML. SOAP sert de langage intermédiaire entre deux langages de programmation différents.

IBM donne dans un tutorial[1], la définition suivante des services Web :

"Les services Web sont la nouvelle vague des applications Web. Ce sont des applications modulaires, auto contenues et auto descriptives qui peuvent être publiées, localisées et invoquées depuis le Web. Les services Web effectuent des actions allant de simples requêtes à des processus métiers complexes. Une fois qu'un service Web est déployé, d'autres applications (y compris des services Web) peuvent le découvrir et l'invoquer."

Les services Web permettent l'appel d'une méthode d'un objet distant en utilisant un protocole Web pour le transport (HTTP : HyperText Transport Protocol) [18] en général) et XML : eXtensible Markup Language [19] pour définir un format d'échange universel. Les services Web fonctionnent sur le principe du client serveur.

L'appel de méthodes distantes n'est pas une nouveauté mais la grande force des services Web réside dans le fait qu'elle utilise les technologies standard de l'Internet ouverts et reconnus : HTTP et XML comme infrastructure pour la communication entre les composants logiciels (les services Web).

L'utilisation de ces standards permet d'écrire des services Web dans plusieurs langages et de les utiliser sur des systèmes d'exploitation différents. Le processus de standardisation touche actuellement trois couches du modèle de fonctionnement global : un protocole de communication permettant de structurer les messages échangés entre les composants logiciels, une spécification de description des interfaces des services Web et enfin une spécification de publication et de localisation de services Web.

Les services Web permettent d'intégrer des systèmes d'information hétérogènes en utilisant des protocoles et des formats de données standardisés, autorisant ainsi un faible couplage et une grande souplesse vis-à-vis des choix technologiques effectués. Par exemple ils permettent d'intégrer un système basé sur J2EE [20] avec un autre basé sur CORBA [21] en les faisant communiquer par un réseau Internet.

[1] http://webservices.xml.com/pub/a/2001/04/04/webservices/index.html

Ils permettent aussi l'interopérabilité entre les utilisateurs d'applications à travers le Web. Cette interopérabilité est possible du fait que les services Web repose sur une architecture SOA.

I.3.2. Principe des services Web

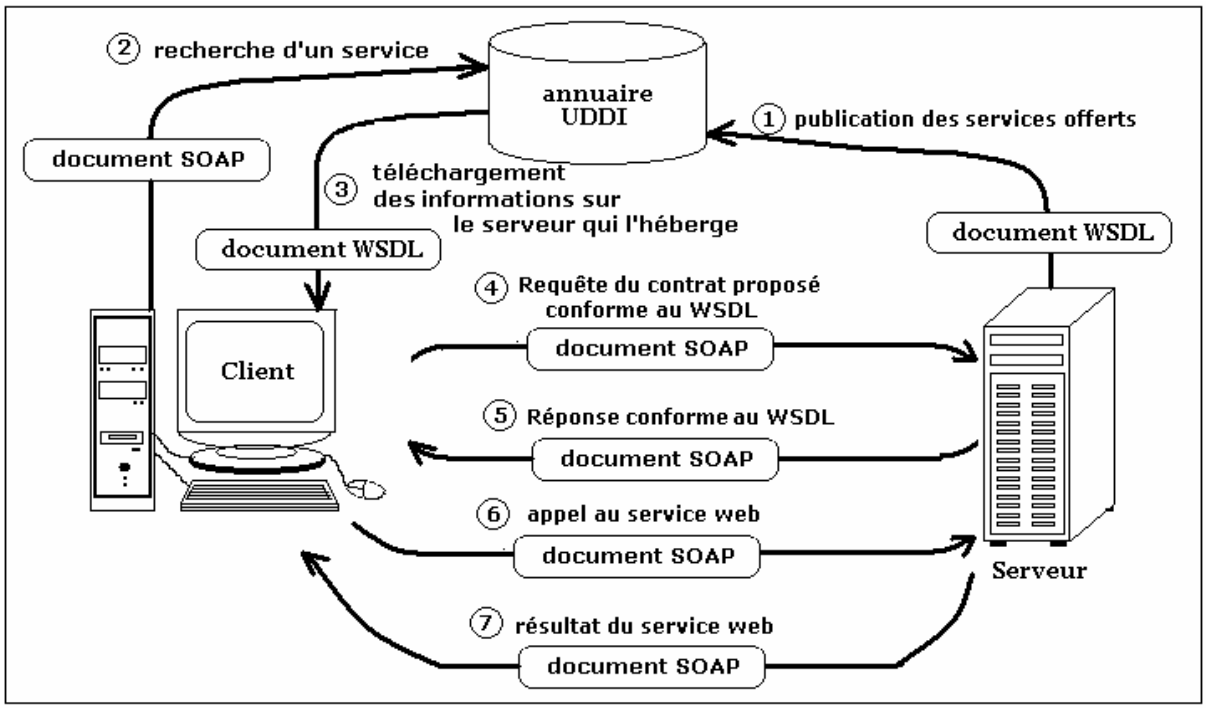


Figure. I.2: *Modèle d'interaction des services Web.*

I.3.3. Architecture des services Web

Différentes extensions de l'architecture de référence ont été proposées dans la littérature. Le groupe architecture du W3C travaille activement à l'élaboration d'une architecture étendue standard. Une architecture étendue est constituée de plusieurs couches se superposant les unes sur les autres, d'où le nom de pile des services Web.

La figure qui suit décrit un exemple d'une telle pile. La pile est constituée de plusieurs couches, chaque couche s'appuyant sur un standard particulier. On retrouve, au-dessus de la couche de transport, trois couches s'appuyant sur les standards émergents SOAP, WSDL et UDDI.

L'infrastructure de service Web de base définit les fondements techniques permettant de rendre les processus métier accessibles à l'intérieur d'une entreprise et au-delà même des frontières d'une entreprise. Dans ce contexte trois types de couches permettent de la compléter :

a. **L'infrastructure de base (*Discovery, Description, Exchange*)** : Ce sont les fondements techniques établis par l'architecture de référence. Nous distinguons alors les échanges de message (établis principalement par SOAP), la description de service (WSDL) et la recherche de services que les organisations souhaitent utiliser (via le registre UDDI).

b. **La couche Processus Métier (*Business Process*)** : Cette couche supérieure permet l'intégration de services Web. Elle établit la représentation d'un *processus métier* comme un ensemble de services Web. De plus, la description de l'utilisation des différents services Web composant ce processus est disponible par l'intermédiaire de cette couche.

c. **Les couches transversales (*Security, Transactions, Administration, QoS*)** :
Ce sont ces couches qui supportent les différents standards mis à jour fréquemment, intervenant dans l'ensemble du processus d'un service Web. Parmi ces derniers, nous pouvons mettre en relief la sécurité, l'administration, la qualité de service Web. Des travaux tentent d'intégrer le Web sémantique dans ces couches transversales en ajoutant une couche verticale représentant le Web sémantique et étant utilisable par les quatre couches horizontales représentant les standards.

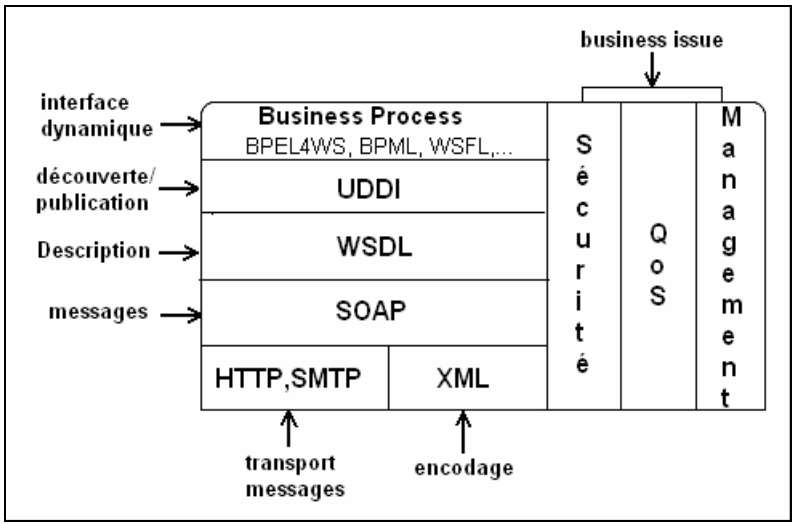


Figure. I.3: *Pile des services Web*

I.3.4. Les technologies standards

Les services Web utilisent trois technologies : SOAP pour le service d'invocation: il permet l'échange de messages dans un format particulier. WSDL pour le service de description: il permet de décrire les services Web. UDDI pour le service de publication: il permet de référencer les services Web.

I.3.4.1. Echange avec SOAP (Simple Object Access Protocol)

SOAP est un protocole de communication entre applications fondé sur XML, visant à satisfaire un double objectif : servir de protocole de communication sur les intranets, dans une optique d'intégration d'applications d'entreprise, et permettre la communication entre applications et services Web, en particulier dans un contexte d'échanges interentreprises. Conçu pour s'appuyer sur http, bien que formellement parlant on puisse y substituer un autre protocole de transport de messages (comme SMTP), SOAP ne requiert pas de modification majeure aux serveurs Web déjà installés sur la Toile, tout au plus des extensions. SOAP permet également, grâce au support de Schéma XML [22], de s'interfacer avec des applications préexistantes en capturant leurs structures de données quelle que soit la complexité de ces dernières [02].

SOAP est défini comme un protocole léger d'échange de données dans un réseau de pair à pair, c'est-à-dire décentralisé. S'appuyant sur XML, SOAP propose un mécanisme simple de représentation des différents aspects d'un message entre applications. N'imposant aucun modèle de programmation spécifique, SOAP peut donc être utilisé dans tous les styles de communication : synchrone ou asynchrone, point à point ou multipoint, intranet ou Internet.

La spécification SOAP se divise en trois parties :

- La structure de l'enveloppe SOAP définit une structure générale pour exprimer le contenu d'un message, le responsable de son traitement et le caractère facultatif ou obligatoire de ce message.
- Les règles de codage de SOAP définissent un mécanisme de sérialisation qui peut être utilisé pour échanger des instances de types de données définis par l'application.

- La représentation RPC de SOAP définit une convention qui peut être utilisée pour représenter les appels de procédures à distance et leurs réponses.

- ***Structure du Message SOAP***

  Un message SOAP comporte plusieurs parties :

  - une **enveloppe** qui définit la structure du message
  - un **en-tête** (optionnel) qui contient les informations d'en-tête (autorisations et transactions par exemple),
  - un **corps** contenant les informations sur l'appel et la réponse
  - une **gestion d'erreur** qui identifie la condition d'erreur
  - des **attachements ou pièces jointes** (optionnel)

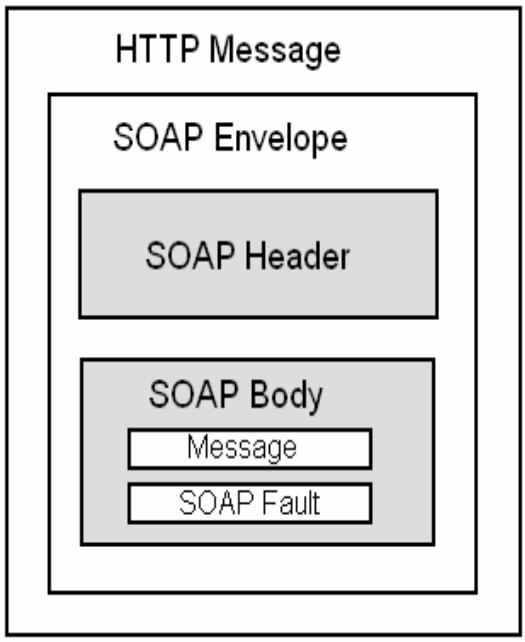


Figure. I.4 : *Structure d'un message SOAP*

I.3.4.2. Description de service avec WSDL (Web Services Description Language)

WSDL est un langage qui spécifie ce que doit contenir un message de requête et l'apparence du message de réponse dans une notation sans ambiguïté. La notation utilisée par un fichier WSDL pour décrire les formats de messages est basé sur la norme du schéma XML, ce qui signifie que WSDL est à la fois neutre par rapport au langage de programmation et à la plateforme.

Outre la description du contenu des messages, WSDL définit l'endroit où le service est disponible et le protocole de communications utilisé pour converser avec le service. Cela signifie que le fichier WSDL définit tout ce qui est nécessaire pour écrire un programme fonctionnant avec un service Web.

Le *WSDL* décrit quatre ensembles de données importantes:

- Information d'interface décrivant toutes les fonctions disponibles publiquement,
- Information de type de données pour toutes les requêtes de message et requêtes de réponse,
- Information de liaison sur le protocole de transport utilisé,
- Information d'adresse pour localiser le service spécifié.

Une fois qu'un *service Web* est développé, il faut publier sa description *WSDL* et faire un lien vers elle dans un annuaire *UDDI* de sorte que les utilisateurs potentiels puissent le trouver. Quand un utilisateur souhaite utiliser le service, il fait une demande de son fichier *WSDL* afin de connaître son emplacement, les appels de fonctions et comment y accéder. A partir de cela, il peut composer, par exemple, une requête *SOAP* (Simple Object Access Protocol) vers l'ordinateur du service pour consommer le service.

- ***Eléments de WSDL***

Un document WSDL (élément de définition) possède sept sous-éléments distincts, divisés en deux parties **(Figure.I.5)**. Une première partie réutilisable, appelée *abstraite*, détaille le service. Une seconde partie non réutilisable, appelée *concrète*, indique la location du service :

- Définitions

  Elément racine du document, il donne le nom du service, déclare les espaces de noms utilisés, et contient les éléments du service.

```
<definitions
        name="ServiceVol"
        targetNamespace="http:// localhost:8080/ServiceVol.wsdl"
        xmlns="http://schemas.xmlsoap.org/wsdl/"
        xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
        xmlns:ns="http:// localhost:8080/ServiceVol.wsdl"
        xmlns:xs="http://www.w3.org/XMLSchema">
```

Extrait de code 1 : *Exemple de definitions*

- Message

  Décrit un message unique, que ce soit un message de requête seul ou un message de réponse seul. L'élément défini les noms et types d'un ensemble de champs à transmettre. Peut-être comparé aux paramètres d'un appel de procédure.

```
<message name="GetPriceInput">
      <part name="body" element="xsd1: PriceRequest"/>
</message>
<message name="GetPriceOutput">
      <part name="body" element="xsd1: Price"/>
</message>
```

Extrait de code 2 : *Exemple de message*

- Types

  Décrit tous les types de données utilisés entre le client et le serveur. Contient les définitions de types utilisant un système de typage. Utilisation de XML Schema pour définir les types de données.

```
<types>
      <schema
targetNamespace="http://example.com/stockquote.xsd"
            xmlns="http://www.w3.org/2000/10/XMLSchema">
            <element name="PriceRequest">
                  <complexType>
                     <all>
                        <element name="destination" type="string"/>
                     </all>
                  </complexType>
            </element>
            <element name="Price">
...
            </element>
      </schema>
</types
```

Extrait de code 3 : *Exemple de type*

- Types de port (port Type)

  Combine plusieurs éléments message pour composer une opération. Chaque opération se réfère à un message en entrée et à des messages en sortie. Peut être comparé à une interface Java.

```
<portType name=" trajet PortType">
        <operation name="GetPrice">
                <input message="tns:GetPriceInput"/>
                <output message="tns:GetPriceOutput"/>
        </operation>
</portType>
```

Extrait de code 4 : *Exemple de port type*

Les éléments suivants représentent la section non réutilisable concrétisant les données abstraites de la première partie du document WSDL.

- Liaison (binding)

  Défini les spécifications concrètes de la manière dont le service sera implémenté: protocole de communication et format des données pour les opérations et messages définis par un type de port particulier.

```
    <binding name=" trajet SoapBinding"
type="tns: trajet PortType">
        <soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
        <operation name="GetPrice">
            <soap:operation
soapAction="http://example.com/GetPrice"/>
            <input>
                <soap:body use="literal"/>
            </input>
            <output>
                <soap:body use="literal"/>
            </output>
        </operation>
    </binding>
```

Extrait de code 5: *Exemple de liaison*

- Service

  Défini les adresses permettant d'invoquer le service donné, ce qui sert à regrouper un ensemble de ports reliés.

```
<service name=" trajet Service">
        <documentation>My first service</documentation>
        <port name=" trajet Port" binding="tns: trajet Binding">
            <soap:address location="http://example.com/ trajet "/>
        </port>
</service>
```

Extrait de code 6 : *Exemple de service*

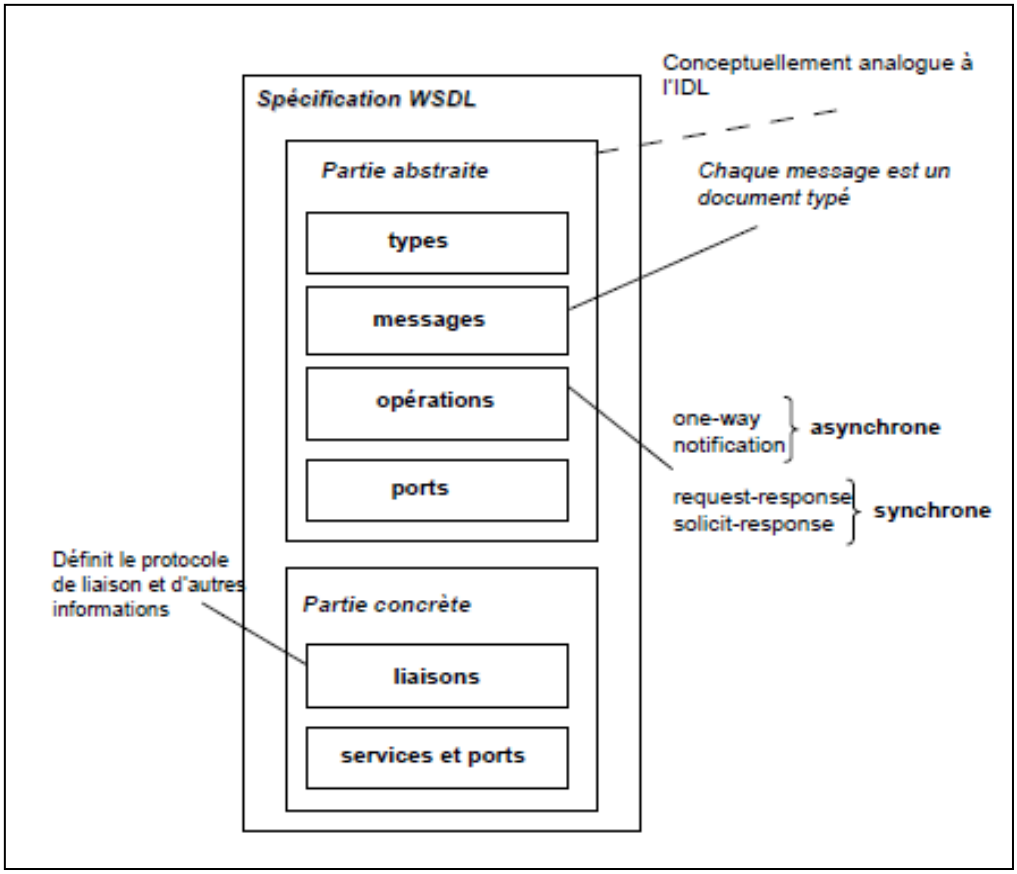


Figure. I.5 : *Structure d'une interface WSDL*

I.3.4.3. Recherche de service avec UDDI (Universal Description Discovery and Integration)

UDDI est une technologie d'annuaire basée sur XML et plus particulièrement destinée aux services Web, notamment dans le cadre d'architectures de type SOA (*Service Oriented Architecture*).

Un annuaire UDDI permet de localiser sur le réseau, le service Web recherché. Il repose sur le protocole de transport SOAP. Il permet de stocker à la fois des information techniques et formelles tel que l'adresse pour accéder au Services Web, mais également des informations beaucoup plus contextuelles, tel le nom de la personne qui s'occupe de leur gestion, la description sommaire de leur fonctionnalités ou encore le nom et la branche d'activité de l'entreprise dont ils dépendent.

L'annuaire UDDI est consultable de différentes manières :

- *Les pages blanches* comprennent la liste des entreprises ainsi que des informations associées à ces dernières. Nous y retrouvons donc des informations comme le nom de l'entreprise, ses coordonnées, la description de l'entreprise mais également l'ensemble des ses identifiants.

- *Les pages jaunes* recensent les services Web de chacune des entreprises sous le standard WSDL.

- *Les pages vertes* fournissent des informations techniques précises sur les services fournis. Ces informations concernent les descriptions de services et de information de liaison ou encore les processus métiers associés.

❖ ***Model d'informations UDDI:***

Comporte cinq structures de données principales [03]:

- ***Publisher Assertion*** (affirmation d'éditeur) : cette partie est facultative. Elle permet de décrire l'organisation dans son intégrité si elle est composée de plusieurs divisions. Toutes les parties de l'organisation et leurs liens sont décrits dans ce module.

- ***BusinessEntity*** (entité commerciale) : sont en quelque sorte les pages blanches d'un annuaire UDDI et elles décrivent les entreprises ayant publié des services dans l'annuaire. On y trouvera notamment le nom de l'entreprise, ses adresses (physiques et Web), des éléments de classification, une liste de contacts, etc. Chaque businessEntity est identifiée par une «businessKey ».

- ***BusinessService*** (offre de service) : décrivent de manière non technique les services offerts par les différentes entreprises. On y trouvera essentiellement le nom et la description textuelle des services ainsi qu'une référence à l'entreprise proposant le service et un ou plusieurs « bindingTemplates ».

- ***BindingTemplae*** (liaison *UDDI*) : donnent les coordonnées des services. Ils contiennent notamment une description des « points d'accès » aux services Web (URL), le moyen d'y accéder (les différents protocoles à utiliser) et les éventuels « tModels » associés.

- ***tModel*** (type de services) : sont les descriptions techniques des services. UDDI n'impose aucun format pour ces descriptions qui peuvent être publiées sous n'importe quelle forme et notamment sous forme de documents textuels (XHTML par exemple).

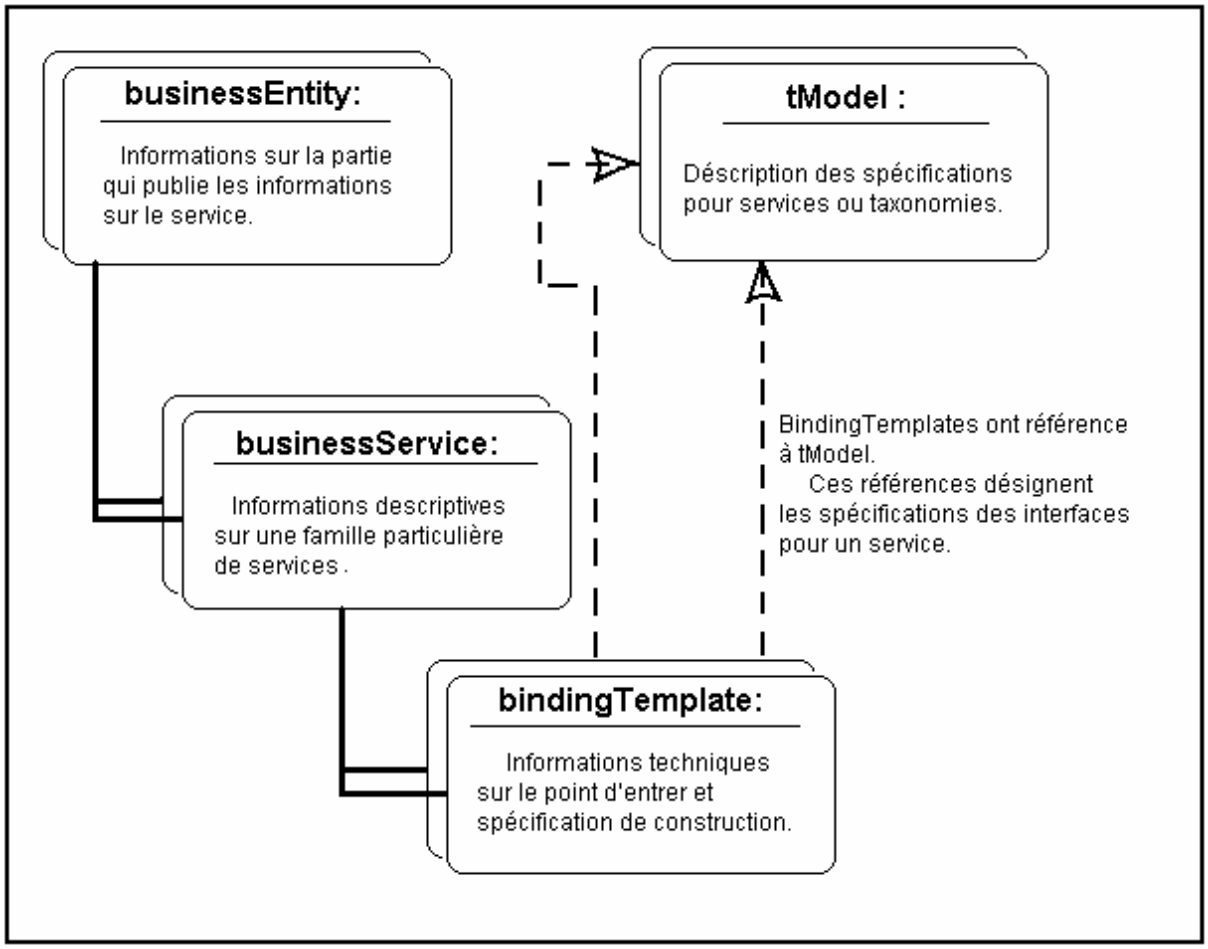


Figure. I.6 : *Structure de données d'un enregistrement de UDDI*

I.4. Processus métier

Un processus métier est une collection d'activités connexes et structurées qui produisent un service ou un produit qui satisfait les besoins d'un client. Ces processus sont critiques à n'importe quelle organisation car ils produisent du revenu et représentent souvent une proportion significative de coûts. [12]

Un processus métier est modélisé en plusieurs niveaux, et plus généralement en trois niveaux [11] :

- *Le niveau métier* : vue métier haut niveau du processus, définissant ses principales étapes et l'impact sur l'organisation de l'entreprise. Ce niveau est défini par les décideurs, et les équipes méthodes de l'entreprise.
- *Le niveau fonctionnel* : formalisation des interactions entre les participants fonctionnels du processus, où sont formalisées les règles métiers conditionnant son déroulement. Ce niveau est modélisé par les équipes fonctionnelles.
- *Le niveau technique* : lien entre les activités (participants modélisés dans le niveau fonctionnels, et les applications) services du système d'information, ainsi que les tâches utilisateurs (Workflow). Ce niveau est réalisé par les architectes et les équipes techniques de l'entreprise.

I.5. Le flux de travail

Un flux de travail est un flux d'informations au sein d'une organisation, comme par exemple la transmission automatique de documents entre des personnes [13].

Un flux de travail est un travail coopératif impliquant un nombre limité de personnes devant accomplir, en un temps limité, des tâches articulées autour d'une procédure définie et ayant un objectif global. Un flux de travail peut être un objet pouvant être décrit par un langage descriptif dans un fichier informatique, qu'une application adaptée (« workflow engine ») peut alors interpréter et exécuter.

Le moteur de flux de travail transfère des documents entre les participants d'un processus en leur assignant des tâches (valider le document, effectuer une modification, etc.). Cette approche pragmatique a l'avantage de l'efficacité : les concepts sont clairs, les outils relativement aisés à mettre en place.

I.6. Gestion de processus métier

I.6.1. Définition

L'enjeu de la gestion de processus métier BPM est de fournir à l'entreprise la possibilité de définir ses processus au niveau métier, et de faire intervenir les utilisateurs et les applications de l'entreprise en tant que partie prenante à ces processus. L'objectif est de permettre aux décideurs, analystes métiers, équipes fonctionnelles et équipes techniques de collaborer pour la définition et l'évolutivité des processus métiers via un seul outil agrégeant les différentes visions [11].

I.6.2. Cycle de vie de gestion de processus métier

Les activités qui constituent la gestion de processus métier peuvent être groupées dans cinq catégories : conception, modélisation, exécution, monitoring et optimisation [15].

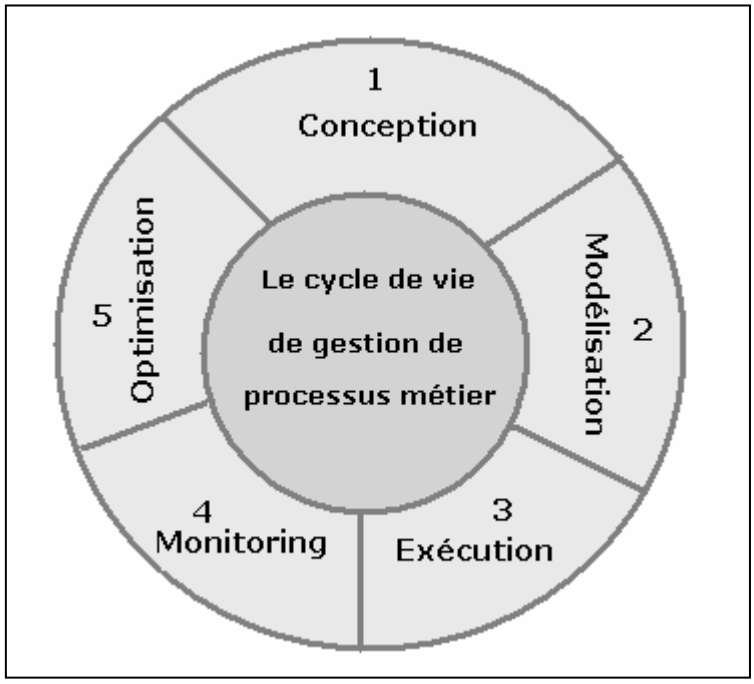


Figure.I.7 : *Cycle de vie de gestion de processus métier.*

La conception d'un processus englobe l'étude de l'entreprise en analysant ses objectifs et son organisation afin d'être en mesure de décomposer l'ensemble de son activité en processus métier. La bonne conception réduit le nombre de problèmes sur la durée de vie du processus. Le but de cette étape est d'assurer une conception théorique correcte et efficace.

La modélisation prend la conception théorique et elle essaye de représenter les processus graphiquement et de les simuler. Dans cette phase les processus métier sont modélisés de bout en bout. Les règles, rôles et séquences d'activités sont explicités dans des diagrammes structurés.

La seule façon d'automatiser des processus est de développer une application qui exécute les étapes exigés du processus; cependant, en pratique, ces applications exécutent rarement toutes les étapes du processus précisément ou complètement.

Le monitoring englobe la localisation de processus pour que l'on puisse facilement accéder aux informations sur leur état et les statistiques de la performance d'un ou plusieurs processus fournis. De plus, ces informations peuvent être utilisées pour améliorer les processus connectés entre le client et le fournisseur. Le degré de monitoring dépend sur quelle information, le service veut évaluer et analyser et comment il veut les contrôler, en temps réel ou en ad hoc.

Enfin, l'optimisation de processus inclut les informations de performance de processus recherché par les étapes de la modélisation et de monitoring. Il identifie les goulots d'étranglement potentiels ou réels et les parties éventuelles pour des réductions de coûts ou d'autres améliorations et appliquer ensuite ces améliorations dans la conception du processus en continuant ainsi le cycle de gestion de processus métier.

I.7. Composition de services

La composition de services permet de combiner des Services Web élémentaires afin d'obtenir des services plus élaborés. Typiquement, nous pouvons modéliser le Service Web de réservation d'une agence de voyages comme la composition d'un Service Web de compagnie aérienne, d'un Service Web d'une chaîne d'hôtels, d'un Service Web bancaire et d'un Service Web de livraison.[10]

La composition décrit un ensemble d'interactions ou processus métier, faisant intervenir différents Services Web. Elle est décrite indépendamment de son implémentation future i.e. elle indique uniquement les types de Services Web nécessaires (Service Web

bancaire) mais ne précise pas nominativement les Services Web qui seront utilisés (Service Web de la Banque d'Algérie). Par ailleurs, la composition peut être à plusieurs niveaux en permettant à des Services Web élaborés d'être à leur tour combinés pour construire de nouveaux services.

On distingue deux types principaux de composition de Web services :

- **Composition statique**

Il s'agit d'un type de composition dans lequel les services Web à composer sont précalculés avant qu'un client ne fasse une requête du service composite. Ce type de composition peut être appliqué dans des environnements « stables » où les services Web participants sont toujours disponibles et où le comportement du service composite est le même pour tous les clients.

- **Composition dynamique**

Dans ce type de composition, les services Web à composer sont déterminés lors de l'exécution de la requête d'un client. Ils peuvent être déterminés selon les besoins, les contraintes de chaque client, la disponibilité des services Web, etc. La composition dynamique apparaît la plus intéressante. D'une part, elle promet d'être capable de faire face à un environnement très dynamique dans lequel des services apparaissent et disparaissent rapidement. D'autre part, elle permet de mieux satisfaire les besoins de chaque client. La plupart des travaux actuels se concentrent sur la composition dynamique des services Web [09].

I.8. Conclusion

Les Services Web améliorent l'intégration et l'interopérabilité entre services à travers l'infrastructure Web en utilisant différents standards XML : SOAP pour l'échange de messages, WSDL pour la description de services et UDDI pour la publication et la découverte de services. Ils reposent sur une architecture orientée service (SOA) et correspondent à des composants logiciels qui peuvent être combinés, grâce à un langage de composition, pour former de nouveaux services plus élaborés.

Bien que beaucoup d'efforts aient été mis en œuvre pour que les Services Web deviennent une réalité, il reste beaucoup à faire. Aujourd'hui, les Services Web ont du succès, mais le développeur doit encore jongler avec un certain nombre d'aspects, par exemple le monitoring (la surveillance), l'adaptabilité, la sécurité, la gestion opérationnelle, les transactions, la fiabilité de la messagerie.

Partie II :

Travaux relatifs au monitoring des services Web

Le monitoring d'un service consiste à le surveiller et à agir sur ses paramètres opérationnels pour qu'il satisfasse les demandes des utilisateurs et les contraintes des fournisseurs. Les Services Web se distinguent des services classiques par leur forte dynamicité et nécessitent de créer de nouvelles infrastructures de monitoring mieux adaptées.

Dans ce chapitre, nous nous concentrons d'abord sur la nécessité de monitoring des services Web et ses différentes classes, puis nous présentons les différentes approches et les différents langages de monitoring.

II.1 Introduction

Le monitoring est une analyse continue des informations qui fournit des indications permettant de savoir si une application se déroule comme il avait été planifié initialement. Les entreprises ont besoin d'observer constamment le bon fonctionnement de leurs applications lors de l'exécution, et prendre des mesures quand on est en présence de changement de situations, d'environnement ou de coordination des partenaires.

Selon Papazoglou, le m*onitoring est la surveillance des activités d'un service qui sont placés dans un bus et de les contrôler explicitement avec des différentes métriques et statistiques. Sa fonction principale est l'aptitude d'être capable de détecter des problèmes et des anomalies dans des processus métier et d'être capable de les corriger au moment où ils sont exécuté.».*

Le rôle du monitoring en temps réel est d'observer l'exécution d'un service Web pour déterminer s'il est conforme à son comportement prévu. Le monitoring permet d'analyser et de détecter les erreurs d'un système, fournissant des activités de rétablissement pour empêcher l'échec catastrophique du système.

Il est employé pour le profilage, l'analyse d'exécution, l'optimisation des services aussi bien que la détection de défauts, le diagnostic, et le rétablissement. La détection des erreurs fournit l'évidence que le comportement d'un service Web est conforme ou non aux propriétés spécifiques, pendant son exécution.

II.2. Le besoin de monitoring les services

La nécessité de monitoring des applications orientée services au temps d'exécution a inspiré un grand nombre de projets de recherche, académique et industriels. Les différences entre ces recherches engendrent des propositions diverses, et sont évidentes après une analyse précise [37].

➢ ***Objectifs de monitoring :***

- Le rôle du monitoring est d'analyser la conformité du système en temps d'exécution c'est-à-dire de contrôler l'exécution du système et de déterminer s'il est approprié à son comportement attendu.
- Le monitoring peut être utilisé aussi pour réaliser le diagnostic et le rétablissement des fautes identifiées, c'est-à-dire, identifier la cause du problème et conduire les actions nécessaires pour rendre le système à l'exécution normale.
- Le monitoring des caractéristiques de qualité des services et des composants qui est souvent utilisé pour instancier et guider les activités de l'optimisation.
- Le monitoring peut aussi soutenir l'adaptation dynamique de l'application au changement approprié de son environnement.

II.3. Les phases de vérification

Il existe trois phases pour la vérification d'un système, la vérification pendant la conception qui surveille le comportement d'un système en utilisant des tests de vérification, le monitoring pendant l'exécution qui analyse et détecte des erreurs d'un système au moments de son exécution et le monitoring après l'exécution qui consiste à diagnostiquer un système après son exécution et de trouver des solutions pour corriger les erreurs.

II.3.1. Vérification pendant la conception (Design-Time Verification)

Il consiste à surveiller un système au cours de développement en utilisant des tests de vérification. Il consiste à s'assurer d'une certaine correction de la spécification formelle avant son utilisation pour une exécution. En effet, des comportements anormaux du service, dus une spécification incomplète ou des erreurs de conception, peuvent d'ores et déjà être établis en exécutant la spécification formelle.

II.3.2. Monitoring pendant l'exécution (Runtime Monitoring)

Le monitoring en temps d'exécution consiste à observer que le comportement du système est conforme à son comportement prévu. Le monitoring en temps d'exécution permet d'analyser et se remettre aux fautes découvertes. Bien que le monitoring pendant le

temps d'exécution ait été dans l'utilisation pendant plus de 30 ans, soit renouvelé grâce à l'intérêt dans son application pour la détection de faute et le rétablissement.

II.3.3. Monitoring après l'exécution (after runtime monitoring)

Cette classe de monitoring permet de récupérer les fichiers logs concernant l'exécution du système et de faire le diagnostic et des statistiques pour identifier les causes et proposer des solutions. Il est approprié pour les systèmes statiques qui ne sont pas à temps réel c'est-à-dire on surveille le système extérieurement de son exécution.

II.4. Les différentes approches de monitoring

Les approches de monitoring peuvent être identifiés par deux approches principales : des approches fonctionnelles par exemple, monitoring des processus métier et monitoring des services Web composés et des approches non fonctionnelles par exemple, monitoring des patrons de sécurité et monitoring de la qualité de services.

II.4.1. Approches fonctionnelles

Des propriétés fonctionnelles se réfèrent à la définition d'opérations de service, leurs signatures et des paramètres techniques (par exemple, des informations obligatoires), aussi bien qu'aux spécifications comportemental des applications à base de service, comme des modèles de composition de service.

II.4.1.1. Monitoring de processus métiers

Monitoring de processus métiers est important dans la gestion de systèmes. Tous les grands fournisseurs présentent leur solution de monitoring comme un outil de monitoring de processus métiers.

Les moniteurs de processus métiers ne nous indiquent pas seulement si on peut commander des marchandises, mais également si elles sont livrées aux clients. De tels moniteurs maintiennent des transactions très longues et qui sont satisfait de rendre compte le fonctionnement approprié des télécopieurs et du progrès des chariots de livraison. Monitoring de processus métiers répond aux questions concernant si le service s'exécute bien.

II.4.1.2. Monitoring des Services Web composés

Les services composés sont, dans le cas général, distribués. L'architecture Web Services Management Network ou WSMN [33] a été proposée par les laboratoires HP pour faciliter leur monitoring. WSMN est un réseau d'overlay, au-dessus de SOAP, constituait d'un ensemble d'intermédiaires qui communiquent entre eux (Figure II.1).

Chaque intermédiaire joue le rôle de proxy entre deux Services Web de la composition. Il permet le monitoring mais aussi la corrélation de messages (détection des dépendances entre Services Web) et la gestion de fautes.

Les échanges entre intermédiaires sont réalisés à l'aide de trois protocoles :

- **Life-Cycle Protocol** : il permet la construction du réseau à partir d'un ensemble d'intermédiaires indépendants,
- **Measurement Protocol** : les mesures effectuées sur les messages doivent parfois être réalisées sur plusieurs sites différents. Ce protocole permet l'échange de mesures entre intermédiaires.
- **Assurance Protocol** : ce protocole permet d'émettre des messages d'alarme pour avertir les intermédiaires en cas de détection de faute.

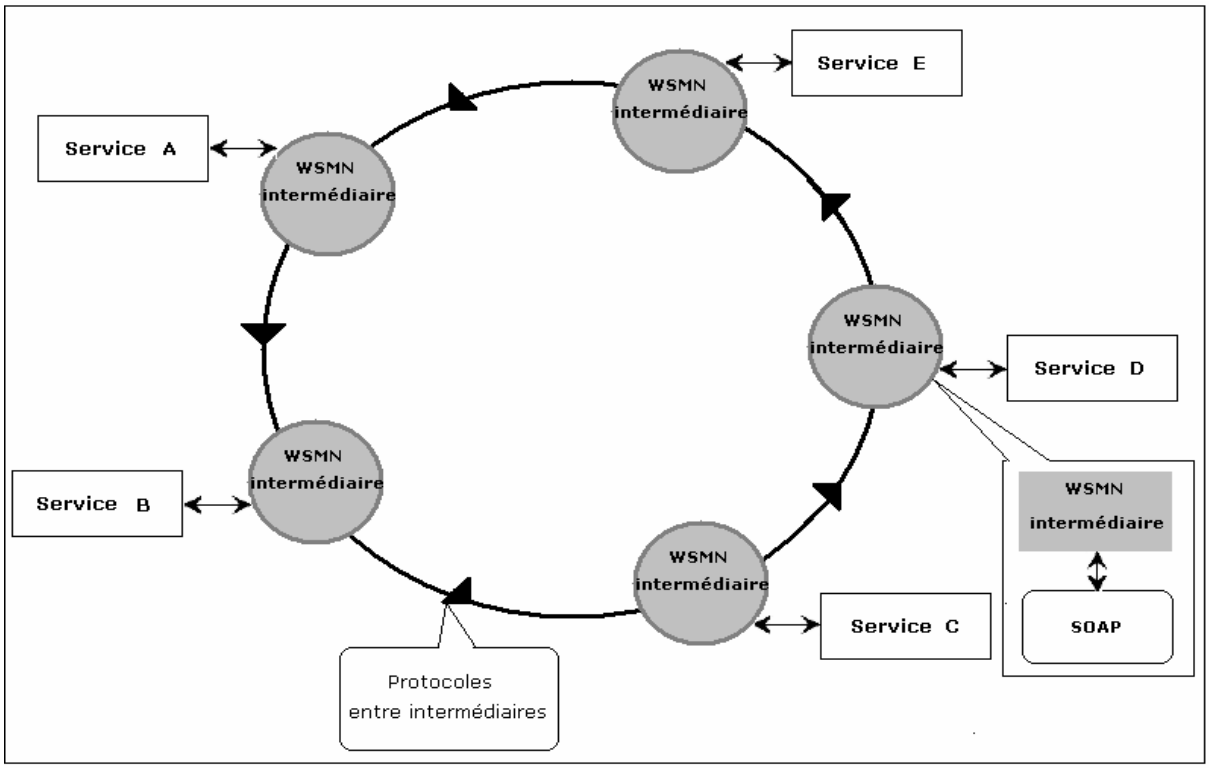


Figure. II.1 : *Architecture du réseau WSMN*

Les intermédiaires ont besoin d'un modèle qui représente le Service Web composé, afin de pouvoir attacher à ce modèle les mesures effectuées :

- Si le business process (BPEL4WS) est connu, un modèle du service est automatiquement généré dans une base de données.
- Sinon, les intermédiaires disposent d'un module de corrélation, qui en ajoutant des éléments aux entêtes des messages SOAP, permettent d'analyser les dépendances et de générer un modèle du Service Web.

Le réseau WSMN constitue une approche importante dans le monitoring et la gestion des fautes des Services Web composés.

II.4.2. Approches non fonctionnelles

Le monitoring de propriétés non-fonctionnelles, vise à mesurer certaines caractéristiques du système. Les propriétés non-fonctionnelles appropriées qui sont soumises de cette analyse quantitative sont la disponibilité, l'accessibilité, la performance et la fiabilité [08].

II.4.2.1. Monitoring les patrons de sécurité

Le problème de monitoring des propriétés de sécurité de systèmes basés sur des services est abordé par plusieurs auteurs. Tandis que les techniques d'analyses statiques sont largement utilisées pour vérifier les propriétés de sécurité pendant la conception et la vérification en temps d'exécution de ces propriétés [17].

II.4.2.2. Monitoring de la qualité des services

La gestion de la qualité joue un rôle essentiel dans le monitoring. Elle consiste à assurer un niveau de service donné (performance, disponibilité) pour un Service Web.

- **Contrat de services**

La gestion de la qualité passe souvent par l'établissement d'un contrat de services (Service Level Agreement[39]) entre un client et un fournisseur. Ce contrat spécifie les objectifs à respecter en termes de qualité de service. Au cours de l'utilisation du service, des mesures sont réalisées par l'infrastructure de supervision afin de vérifier que le contrat est bien respecté [14]. Le cas échéant, des pénalités et des mesures correctives doivent être prises.

Les différents éléments d'un contrat de services sont décrit par l'ouvrage Foundations of Service Level Management [16] :

- Parties : les entités concernées par le contrat i.e. les signataires (typiquement le fournisseur et le client) et éventuellement des entités tierces pour effectuer des mesures,
- Term : la durée de validité du contrat,
- Service level objectives : les objectifs à respecter (en matière de disponibilité, performance, sécurité),
- Service level indicators : les métriques (temps de réponse par exemple) que l'on utilise pour vérifier que les objectifs sont atteints,
- Penalties : les sanctions appliquées en cas de non respect du contrat,
- Scope, limitations : le cadre exact du contrat (portée, limitation, exclusion).

Le contrat de services permet de confronter les besoins du client avec les limites du fournisseur. Il doit être à la fois flexible pour s'adapter au plus grand nombre d'instances et suffisamment respectueux pour qu'il n'y ait pas de malentendus sur la qualité du service fourni.

- **Langages de contrat de services pour les Services Web**

Deux langages WSLA [34] et Web Service SLA [35] ont été proposés afin de formaliser les contrats d'un service dans le cas des Services Web. Ils utilisent des syntaxes différentes. Néanmoins, nous retrouvons, dans chacun des cas, les sections caractéristiques d'un contrat de services.

Ces langages définissent des métriques de haut niveau SLA Parameters ou MeasuredItem (temps de réponse, débit), qui sont exprimées sous la forme de métriques de bas niveau Metrics ou Item (compteurs, jauges). Á partir de ces langages sont construits différentes architectures de monitoring, qui permettent d'assurer le respect des contrats de service.

II.5. Quelques travaux sur le monitoring des services Web

Dans cette section, nous décrivons plusieurs approches de monitoring de services Web. Nous nous concentrons sur des méthodes basées sur des règles, des méthodes de planification ainsi les approches algébriques et les approches d'automates.

II.5.1. Requirements monitoring based on event calculus

G. Spanoudakis, K. Mahbub [23,24] présentent une approche dans laquelle les exigences à contrôler dans un workflow BPEL sont définies en utilisant le calcul d'événement (event calculus), une logique de premier ordre qui introduit des prédicats pour exprimer les caractéristiques temporelles. Un composant d'intercepteur d'événement est nécessaire pour capturer des phénomènes, comme des invocations d'opérations, des messages de retour, etc. En liant l'intercepteur d'événement à un moteur d'exécution centralisé, avec cette approche on a pas besoin d'orchestrer les services individuels dans la collaboration.

On considère deux sortes d'exigences : des propriétés comportementales, automatiquement obtenues de la spécification de collaboration BPEL et des hypothèses comportementales qui sont manuellement spécifiées. Quand les événements sont collectés au temps d'exécution, ils sont stockés dans une base de données d'événements. Les propriétés indiquées sont alors vérifiées sur les données collectées, en utilisant des variantes de techniques vérifiant l'intégrité dans des bases de données temporelles.

Cette approche permet de capturer les situations incorrectes. Bien que l'approche se focalise sur le monitoring de propriétés fonctionnelles, les propriétés non fonctionnelles peuvent aussi être exprimées et vérifiées, comme propriétés relatives aux temps de réponse.

II.5.2. Planning and monitoring execution with business assertions

A. Lazovik, M. Aiello, and M. P. Papazoglou [25] proposent une approche significativement différente. Ils présentent une architecture de planification dans laquelle les requêtes de service sont présentées dans un langage de haut niveau appelée XSRL (Xml Service Request Language). Ils adoptent une approche orchestrée du propriétaire à la collaboration, puisqu'ils prétendent que des standards actuels, comme BPEL, n'ont pas la flexibilité nécessaire pour satisfaire les exigences d'utilisateur qui dépendent des informations de contexte en temps d'exécution.

L'architecture de planification est basée sur une insertion continue des étapes de planification et des étapes d'exécution. Parce que BPEL manque de la sémantique formelle, les auteurs ont décidé de concevoir des systèmes de transition d'état à partir des spécifications BPEL et les enrichir d'opérateurs de domaine.

Cette structure est basée sur le monitoring réactif. C'est-à-dire les développeurs peuvent définir trois sortes de propriétés : (1) les buts doivent être corrects avant de passer vers l'état suivant (2) les buts doivent être corrects pour l'exécution de processus entière et (3) les buts doivent être corrects pour l'exécution de processus et l'ordre d'évolution. Le langage XSRL tient aussi compte de la définition de contraintes comme les combinaisons booléennes d'inégalités linéaires et les propositions booléennes.

La plate-forme de livraison boucle continuellement entre l'exécution et la planification. Particulièrement la dernière activité est réalisée en prenant en compte le contexte et les propriétés indiquées pour le système de transition d'état. Cela permet de découvrir, chaque fois qu'il est entrepris, si une propriété a été violée par l'étape exécutée précédemment, ou si l'exécution passe correctement.

II.5.3. Monitoring conversational web services

D. Bianculli and C. Ghezzi [26], proposent une approche pour la vérification en temps d'exécution c-à-d si le comportement réel d'un service conforme au comportement attendu de ce service. Des services conversationnels, dont le comportement dépend de l'état local résultant de l'interaction de service de client, augmentent de nouveaux défis aux langages de spécification et au parcours que le comportement réel peut être vérifié contre la spécification.

En effet, la spécification comportementale devrait formellement décrire ce que le service contrôlé doit faire de la part de son client et inclut non seulement les aspects syntactiques, mais aussi la logique métier du service.

Il explore un style de spécification qui est basé sur la représentation algébrique de la conversation des services Web. Dans cette approche les opérations de service sont classifiées dans des constructeurs (qui crée des nouveaux objets de données), des observateurs (récupère des informations sur les données) et des transformateurs.

La spécification algébrique permet de spécifier les pré et post conditions sur les opérations de service, mais aussi les conditions et les axiomes qui donnent la sémantique aux opérations et font la spécification à base d'état. L'évaluation de la spécification durant le temps d'exécution peut être faite par l'exécution symbolique (la simulation).

L'architecture de monitoring adopte une programmation orientée aspect. Les aspects sont dynamiquement attachés au moteur BPEL pour contrôler la composition de service. AspectJ est utilisé pour ajouter les facilités de monitoring à ActiveBPEL. Leur solution étend l'implémentation d'un standard du moteur ActiveBPEL avec ces trois composants supplémentaires principaux :

- **L'intercepteur** pour intercepter l'exécution d'un processus dans le moteur (l'utilisation de la programmation orienté aspect) pour passer au moniteur des informations appropriées (c'est-à-dire, d'interactions externes).

- **Le registre de spécification** contient les spécifications contre lequel les services sont vérifiés pour la conformité.

- **Le moniteur** est le contrôleur de la conformité actuel. Le composant moniteur est un wrapper pour l'évaluation des spécifications algébriques. Cela garde une description exploitable de la machine pour les états des instances de processus et met à jour cette description quand les nouvelles informations de l'intercepteur sont reçues.

II.5.4. Assumption-based monitoring of service compositions

F. Barbon, P. Traverso, M. Pistore, and M. Trainotti, [27, 28] présentent une approche de monitoring des composition des services implémentée dans BPEL. Le but général de cette approche est la vérification en temps d'exécution les hypothèses conformément à lesquelles les services composantes supposées à participer à la composition et les conditions que la composition doit les satisfaire. De plus, l'approche proposée vise à vérifier si la propriété se tient ou pas, ainsi au rassemblement des statistiques et des informations sur le temps de telles propriétés, aussi bien que les informations de l'historiques agrégées sur tous les instances du processus métier donné.

Pour aborder le problème, les auteurs proposent une structure de spécification et un monitoring en temps d'exécution de l'environnement qui étend le moteur BPEL avec les capacités de contrôle et de monitoring. L'architecture créée sépare clairement la logique d'affaires d'un service Web de contrôle de ses fonctionnalités.

La structure de spécification repose sur un langage de monitoring expressif RTML (Run-Time Monitoring Language), qui tient compte de l'expression des propriétés comportementales des instances de composition de service (par exemple, le paiement ne devrait pas commencer avant que la reconnaissance ne soit reçue), en même temps avec le chronométrage (par exemple, durée de la procédure de paiement) et les informations statistiques (par exemple, nombre de nouvelles tentatives sur reconnaissance négative) et de classes de composition (par exemple, nombre total de violations ou nombre moyen de nouvelles tentatives).

II.5.5. Monitoring privacy-agreement compliance

S. Benbernou, H. Meziane, and M.-S. Hacid [29] abordent le problème de monitoring au temps d'exécution de la conformité de la régularité de privacy définissant les droits de privacy d'utilisateurs et leur traitement possible par le fournisseur de services.

La solution proposée présente le modèle d'agreement de privacy, où les exigences sur le management et le traitement des données de privacy sont spécifiées, associé avec l'approche pour le monitoring de la conformité au temps d'exécution.

Les propriétés de la privacy sont données sous forme de data-rights (des opérations autorisées) et data-obligations (des actions exigées).L'ensemble d'exigence, des unités de la privacy et des scénarios de mauvaise utilisation est défini en se basant sur ces propriétés.

Le formalisme adopté pour la représentation des unités de privacy et la mauvaise utilisation est basé sur la logique temporelle linéaire (LTL). Pour monitoring purpose (but), les unités de privacy sont transformées dans la représentation de machine d'état qui correspond à l'évolution de gestion de données de privacy et définit la bonne et la mauvaise utilisation de ces données.

La structure de monitoring incorpore trois composants principaux, à savoir la spécification d'exigences, l'observateur d'unité de privacy et le moniteur. La spécification d'exigences est transformée dans les machines d'état d'unité de privacy correspondantes, qui au temps d'exécution se développent en parallèle avec l'exécution de service. Le moniteur rassemble les informations sur les données de privacy en utilisant les journaux de service et met à jour le statut de l'observateur d'unité de privacy. Le dernier annonce les violations des exigences de privacy quand un état d'échec spécifique de la machine d'état correspondante est atteint.

La structure proposée compte sur un modèle clair et simple d'agreements de privacy, tandis que le modèle d'exigences sous-jacent compte sur un formalisme compréhensif pour la représentation d'utilisation correcte d'informations privées. L'approche de monitoring au temps d'exécution exploite des techniques automatisées pour l'extraction et l'exécution de programmes de moniteur.

II.5.6. Query-based business process monitoring

C. Beeri, A. Eyal, T. Milo, and A. Pilberg [30] proposent une approche de monitoring de processus métier indiqués dans BPEL. Le but de cette approche est semblable aux d'autres approches proposées, mais l'environnement spécifique de l'approche est différent.

En particulier, les auteurs essayent d'adresser aux problèmes suivant la conception et l'implémentation du monitoring: la spécification de moniteur devrait viser le même niveau d'abstraction que les processus original. L'activité de monitoring devrait prendre en compte les caractéristiques spécifiques des modèles de processus; les moniteurs devraient être déployés et exécutés dans le même environnement que le processus original, sans mettre des exigences supplémentaires sur cet environnement.

Pour ces buts, un système de monitoring des processus métier (BP-Mon) est présenté. Pour spécifier les propriétés de monitoring, ils ont proposé un langage de requêtes graphique intuitif de haut niveau qui permet aux utilisateurs de définir visuellement les tâches de monitoring. Les requêtes de BP-Mon consistent en deux composants principaux : les modèles d'exécution qui devraient être correspondus contre les traces d'exécution réelles et la spécification de rapport engendrée par ces modèles.

Les modèles représentent les événements composites comme des spécifications de flux de contrôle partiel, où les éléments spécifient les activités qui devraient apparaître dans une exécution normal. De plus, la requête contient la définition de la fenêtre de temps (la période et l'intervalle), dans lequel la requête devrait être évaluée et la condition de limiter l'ensemble d'exécutions correspondues. Quand la requête est accordée, le rapport est émis. Le rapport représente un modèle d'XML paramétrique, qui est instancier quand le modèle est correspondu. Deux modes qui sont fournis : **un local**, où un rapport individuel est publié pour chaque instance de processus et **un global** qui considère toutes les instances.

Pour faire l'analyse au temps d'exécution, ils ont proposé un algorithme d'appariement de modèle spécifique. L'algorithme essaye de simuler le modèle pour correspondre aux événements le plutôt possible et faisant marche arrière en cas d'échec. Un rapport est publié quand l'accord pour le modèle est identifié.

Le système est implémenté comme suit. Une requête de BP-Mon est compilée dans une spécification de processus BPEL, dont les instances exécutent la tâche de monitoring, qui est traduite dans un code exécutable pour être exécuté sur le même moteur BPEL que le processus métier contrôlé.

Un composant supplémentaire, le distributeur, est utilisé pour écouter les événements sur les activités de processus et offre eux à l'instance de processus de requête. Une caractéristique importante de l'approche est qu'il ne vise pas à un but particulier de monitoring. En effet, les rapports produisent juste les valeurs exigées et peuvent donc être utilisés pour des buts divers relativement aux processus de BPEL.

II.6. Langages

Plusieurs langages de monitoring ont été développés depuis que les recherches sur monitoring ont commencés. Nous allons citer deux langages WS-CoL et BP-QL :

II.6.1. WS-CoL (Web Service Constraint Language) [15]

Le Langage de Contrainte de Service Web, nommé WS-CoL, est un langage d'assertion des politiques indépendante du domaine pour spécifier les exigences d'utilisateur (des contraintes) de l'exécution des services Web. WS-CoL est un langage d'affirmation standard augmentée avec des fonctions de propositions spéciales pour récupérer des données "externes".

Il distingue entre la collection de données et l'analyse de données à différencier la phase dans laquelle les informations sont rassemblées (de sources externes) de la phase dont les expressions exposées sont évaluées contre les valeurs identifiées. Les données peuvent être collectées par le processus directement (Par exemple, la variable interne), mais ils peuvent aussi venir de sources externes (par exemple, des messages SOAP échangés).

Des variables internes sont eues accès au moyen de l'instruction suivante :

```
$<name of variable>\<part of variable>
```

Dans la spécification WS-BPEL, une variable est une instance d'un schéma XML. Puisqu'une variable peut être composée de plusieurs parties, cette instruction nous permet d'avoir accès aux parties différentes.

Si les données viennent de sources externes, appelées des collectionneurs de données, nous utilisons l'instruction suivante :

```
\return[Int|String|Boolean](WSDL, OpName, <parameters>)
```

II.6.2. BP-QL (Business Process Query Language)

Business Process Query Langage (BP-QL) est un nouveau langage de requête pour interroger les spécifications de processus métier. Il permet aux utilisateurs à demander des processus métiers visuellement, dans une façon très analogue, comment de tels processus sont typiquement spécifiés et peuvent être employés dans un cadre distribué, où on peut fournir des composants de processus.

II.7. Conclusion :

Dans ce chapitre, nous avons présenté le concept de monitoring, ainsi que les trois classes de monitoring (monitoring pendant la conception, pendant l'exécution et après l'exécution).

Nous avons étudié les différentes approches existantes pour monitoring : fonctionnelles (processus métier, services Web composés…) et non fonctionnelles (sécurité, qualité de services…).

Nous avons citées quelques travaux de recherches reliées au monitoring des services Web. Enfin nous avons présenté deux langages de monitoring WS-CoL et BP-QL.

Partie III :

Contribution

Dans cette partie, nous présentons une structure pour le monitoring des services Web, en se basant sur monitoring des processus métier « business protocol» au lieu des processus métiers «business process » qui vont être modélisés par des automates à états finis déterministes et qui admettent une représentation abrégée.

Dans ce cas, nous proposons un langage de monitoring qui va être exprimé en «Bpath» un langage développé par l'équipe Liris, c'est une extension du langage Xpath qui consiste à spécifier et surveiller les propriétés de « privacy » dans les services Web.

III. Présentation

Dans ce chapitre, on va présenter une approche pour spécifier le monitoring de la privacy dans les services Web basé sur un protocole métier. Au lieu de surveiller les processus métiers « business process» pour protéger les données privées du client, on prend l'exécutable de ce processus et on essaye d'obtenir son protocole métier « business protocol» correspondant.

D'ailleurs, il existe maintenant des méthodes pour générer automatiquement le protocole métier à partir du son processus métier. Le protocole métier est plus pratique pour le monitoring puisqu'il est modélisé sous forme d'un automate d'état finis, on aura que les interactions entre les services.

III.1. Notion de protocole métier

III.1.1. Pourquoi le Protocole métier ?

Un protocole métier est un formalisme très intéressant pour représenter des services Web en termes d'interactions avec d'autres services. Il a un niveau d'abstraction qui facilite l'écriture de règles de monitoring dont il facilite cette tache.

III.1.2. Définition de protocole métier

Le protocole métier est un formalisme pour la spécification de service Web qui définit des messages et leur ordre d'invocation. Avec le protocole métier, le client peut savoir comment interagir correctement avec un service. Nous avons choisi un modèle de protocole métier utilisant la machine d'état fini déterministe, puisque ce schéma semble être plus adéquat à l'exécution de service Web et facilite leur monitoring.[32]

Tel que :

- **États :** représentent les phases différentes de la conversation de service.
- **Transitions :** représentent les messages échangés avec d'autres services.

III.1.3. Un exemple de Protocole métier :

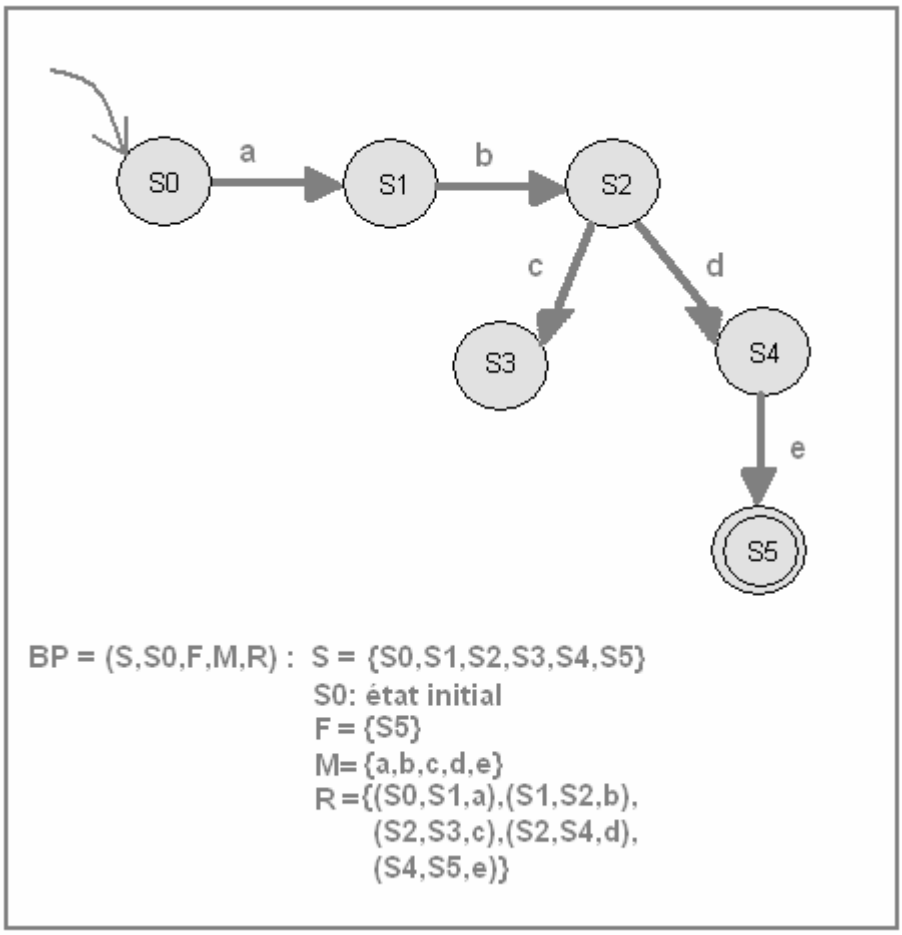


Figure III.1: *Un Protocole métier.*

III.2. Notion de Privacy (la protection de la vie privée)

III.2.1. Définition

L'Internet a révolutionné la façon dont nous communiquons, en permettant un partage sans limites ni frontières de l'information. Cependant, avec l'appétit des nouvelles entreprises venues profiter de cette aubaine, il n'est pas surprenant de constater que la vie privée et la sécurité ont été les grands oubliés de cette révolution. La considérable augmentation des actes de violation de la vie privée et de la sécurité depuis la naissance d'Internet illustre parfaitement cette tendance alarmante.

A chaque fois que vous visitez un service Web sans être protégé, vous ne vous exposez pas seulement à un risque d'intrusion : vous fournissez quantité d'informations sur vous-même, qui peuvent inclure les mots-clés que vous recherchez, votre localisation géographique, votre adresse, votre numéro de téléphone, votre emploi, votre numéro de carte bancaire, etc. Beaucoup de ces services compilent les informations, et conservent de véritables dossiers sur leurs clients. Et même si, la plupart du temps, vous faites confiance à la personne qui est derrière le site que vous visitez, vos informations sont tout de même menacées en cas d'intrusion par des pirates.

Le droit à la vie privée ne se limite pas à la liberté de ne pas être dérangé et interrompu dans ses activités. Le pouvoir d'exercer un contrôle sur nos renseignements personnels est aussi un aspect essentiel de ce droit fondamental. À l'âge du numérique, où la collecte de données personnelles représente une valeur inestimable, la protection de la vie privée prend une importance de plus en plus grande.

- Les critères Communs de la protection (ISO 15408), il y avait quatre propriétés pour la privacy sur l'Internet :
  - ***Anonymat*** : impossibilité pour d'autres utilisateurs de déterminer le véritable nom de l'utilisateur associé à un sujet, une opération, un objet.
  - ***Pseudonymat***: sauf que l'utilisateur peut être tenu responsable de ses actes.
  - ***Non-"chaînabilité"***: impossibilité pour d'autres utilisateurs d'établir un lien entre différentes opérations faites par un même utilisateur.
  - ***Non-observabilité***: impossibilité pour d'autres utilisateurs de déterminer si une opération est en cours.

III.2.2. Quelques définitions concernant les règles de privacy :

Nous présentons le modèle de règles de la privacy comme une extension des catégories de règles définies dans la plate-forme de préférences de privacy P3P [36].

Pour établir une interaction entre un client et un fournisseur du service, Le client spécifie des règles appelées des préférences de la privacy décrivant la façon comment le fournisseur peut utilisé ses données privées, et le fournisseur doit spécifie des règles appelées les politiques de la privacy décrivant comment il va utiliser et traiter les données privées du client.

Pour établir une conversation entre un client et un fournisseur (le client fournit ses données privées), les préférences du client doivent être compatibles avec les politiques du fournisseur.

• Une préférence **pref** est compatible avec une politique **plcy** si cette politique **plcy** est plus restrictive que la préférence **pref**. Puisqu'une préférence est définie comme une politique, nous serons capables de vérifier la cohérence d'une préférence avec une politique [32].

Puisque le client peut aussi demander un autre service Web, il devrait spécifier des préférences. Ainsi, chaque service Web possède des préférences et des politiques.

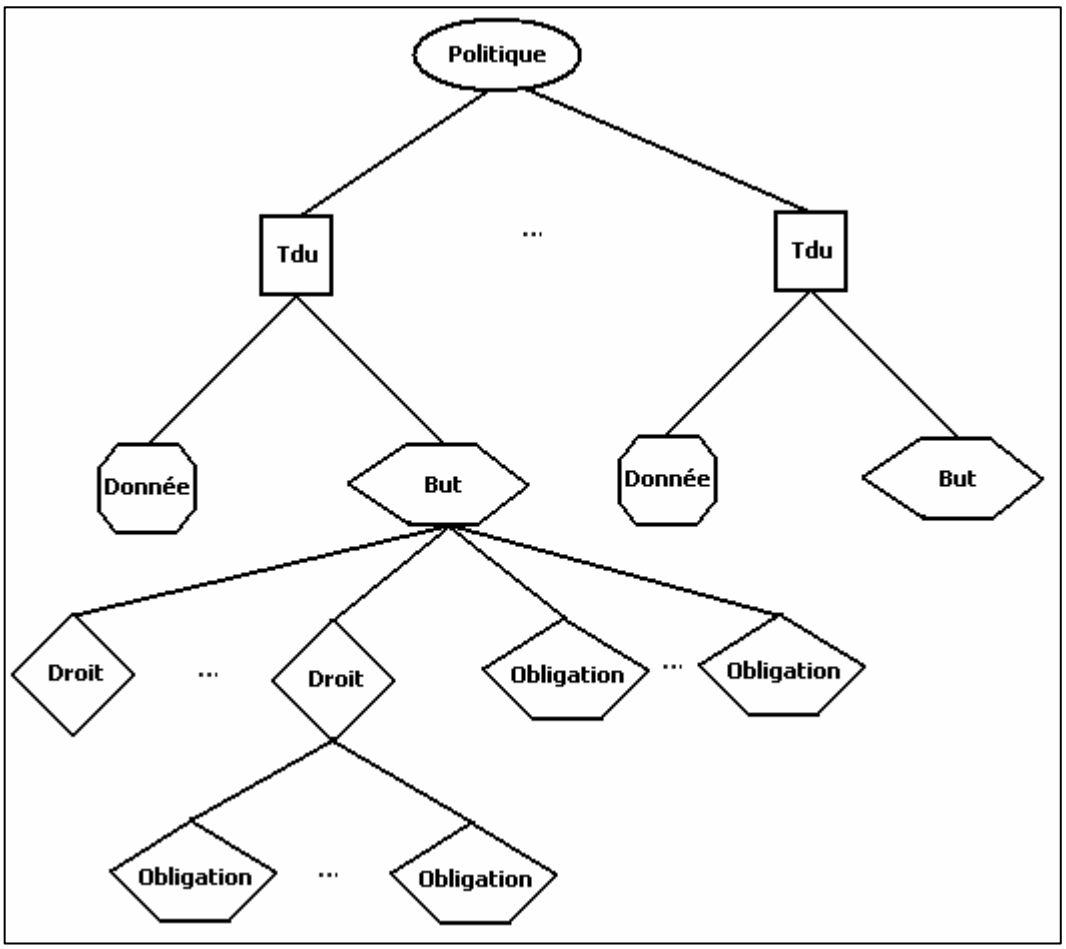


Figure III.2 : *Politique de privacy*

Les politiques de la privacy sont des règles indiquées par le fournisseur décrivant comment il traitera les données privées de son client. Comme c'est montré dans la figure suivante une politique de privacy est un ensemble fini des **Termes d'Utilisation de Données**, dénoté TDU. Un TDU consiste en données privées.

But *:* un but est une action représentant le besoin d'un client et exécuté par une entité de donnée. Les entités sont ceux qui peuvent utiliser les données pour accomplir le but du client dans des délais de donnés.

En outre, le fournisseur peut exiger l'obtention d'un choix pour exécuter ou pas d'autres actions appelées des **Droits**. Puisque l'accomplissement de buts et des droits implique l'utilisation des données privées du client, le fournisseur doit garantir leur sécurité. Pour cela, il doit spécifier des actions appelées des **obligations** pour sécuriser les données.

Droit : un droit est une action de fournisseur. Pour chaque droit, nous spécifions les entités autorisées pour l'exécuter, le délai pendant lequel les entités possèdent le droit et le délai pendant lequel le droit doit être réalisé une fois activé. Aussi un droit peut inciter un ensemble d'obligations.

Obligation : une obligation est l'action où le fournisseur doit réaliser après le rassemblement de données privées pour assurer leur sécurité. Une obligation est spécifiée comme un droit mais il n'implique pas d'action.

Remarques :

- **Comparaison de Termes d'Utilisation de Données** $\lesssim_{TDU}$**.** Un terme d'utilisation de données tdu2 = (donneé2, but2) est plus restrictive qu'un terme d'utilisation de données tdu1 = (donnée1, but1) noté: tdu1 $\lesssim_{TDU}$ tdu2 si et seulement si **donnée1** est inclut dans **donnée2** et **but1** est plus petit que **but2**.

Et comme une politique est définie par un ensemble de terme d'utilisation de données TDU, la comparaison précédente sert à comparer entre deux politiques.

- **Comparaison de politiques** $\lesssim_{privée}$. Une politique plcy2 est plus restrictive qu'une politique plcy1, noté **plcy2** $\lesssim_{privée}$ **plcy1**, si seulement si $\forall$ tdu1= (d1, b1) $\in$ plcy1, $\exists$ tdu2= (d2, b2) $\in$ plcy2 tel que **tdu1** $\lesssim_{TDU}$ **tdu2**.

- Donc on dit qu'une préférence **pref** est consistante avec une politique **plcy** si cette politique **plcy** est plus restrictive que **pref** $(pref \lesssim_{private} plcy)$. Ainsi une préférence est définie comme une politique avec un ensemble finis de termes d'utilisation de données **TDU**. On peut alors utilisé la deuxième remarque pour comparer entre une politique et une préférence.

III.3. Protocole métier de privacy pour les services Web

Nous présentons ici comment intégrer notre modèle de règles de privacy dans des protocoles métiers. Un protocole métier vise à spécifier le comportement externe (les ordres de messages) de service Web. Les protocoles métiers sont spécifiés par des machines d'état finis, où les états correspondent aux états différents du service et les transitions correspondent aux messages échangés (input et output messages).

Chaque transition est étiquetée par un nom de message suivi par la polarité de message indiquant si le message est l'arrivée (de signe +) ou sortant (de signe -).

- ***Les états du protocole métier pour la privacy***

  Dans les protocoles métiers, un état peut être une source d'un ensemble de transitions permettant une production d'un message privé. De là, les préférences correspondantes sont associées à un état.

- ***Les transitions du protocole métier pour la privacy***

  Avec les protocoles métier, la politique de la privacy doit être associée aux messages d'entrés privés. Donc, nous proposons d'annoter chaque transition permettant un message d'entré privé par la politique correspondante.

- **<u>Un protocole métier pour la privacy:</u>**

La définition suivante étend la définition traditionnelle du protocole métier en incorporant les aspects de la privacy:

Un protocole métier privé Q est un tuple **Q = (*S, S0, F, M, PREF,* ð*, PLCY, T*)** Qui consiste en composants suivants :

- ✓ **S** est un ensemble fini d'états, où **S0 Є S** est l'état initial
- ✓ **F Є S** est un ensemble d'états finaux. Si **F = Ø**, donc on dit que Q est un protocole vide.
- ✓ Le **M** est un ensemble fini de messages. Pour chaque message **m Є M**, nous définissons deux variantes de la fonction la Polarité (Q, m):
  - La polarité d'un message d'entrée s'écrit : (+, ***Donnée privé du client***)

Donnée privée du client { **1** Si le message m importe les données privées du client ; **0** Sinon

  - La polarité d'un message de sortie s'écrit : (**-**, ***Donnée privé du client, Donnée privée du service***)

Donnée privée du service { **O** Si le message m exporte les données privées du service ; **N** Sinon

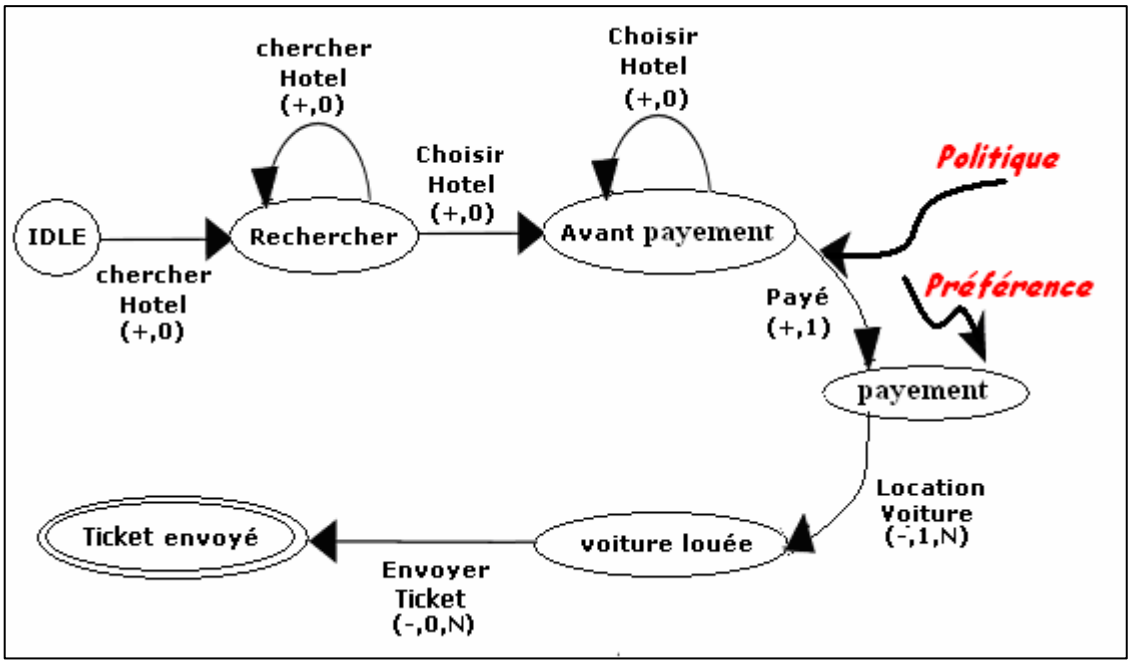


***Figure III.3**: Un exemple d'un protocole métier privé Q.*

La figure précédente permet de réserver un hôtel et de louer une voiture, le message **Chercher Hotel (+,0)** est un message envoyé par le client donc on le signe par + puisque c'est une entrée et **0** puisque c'est un message qui n'est pas privé. Pour le message **Payé (+,1)** c'est un message privé qui importe les données privées du client donc on met **1.** Le dernier message **Envoyer Ticket (-, 0, N)** est un message envoyé par le fournisseur du service au client donc on le signe par **-** et ce message n'a pas importé les données privées du client ni exporté les données privées des services.

III.4. Monitoring de la privacy dans les services Web

On propose un langage de monitoring qui consiste à surveiller les propriétés de privacy dans les services Web. D'abord le client va présenter ses données privées qui vont être protégé et respecté (surtout si les services à invoqué sont des services bancaires, d'achat ou de santé…), le fournisseur du service doit traiter ces données privées et il ne doit pas les perdre.

On génère un processus métier pour cette privacy, puis on essaye de créer son protocole métier correspondant et à partir de ce dernier on génère la trace d'exécution du protocole qui va nous aider à surveiller et à vérifier les propriétés de privacy. Les politiques et les préférences de la privacy vont être extraites facilement à partir de l'automate d'état finis puisque les politiques vont être exprimées dans les messages privés (transitions du protocole) et les préférences sont spécifiées dans les états du protocole.

Le schéma suivant va nous montrer la structure générale de monitoring de la privacy pour les services Web:

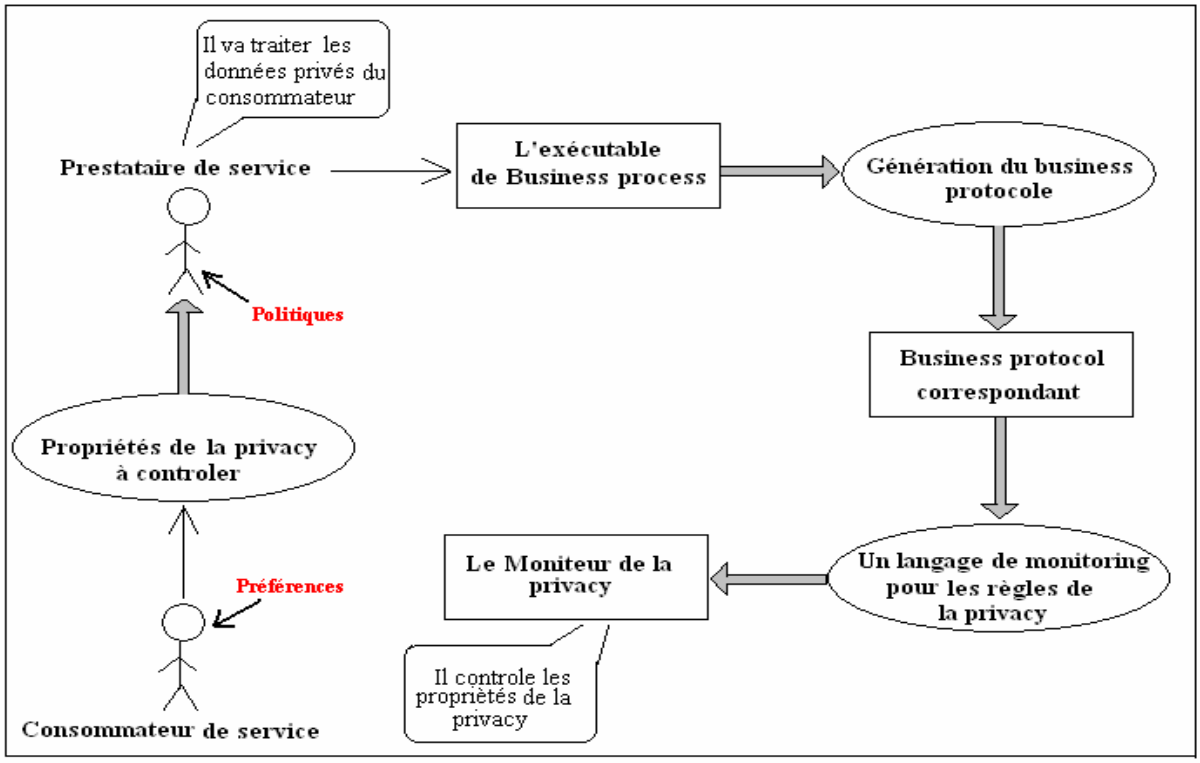


Figure III.4 : *Schéma général du monitoring*

III.4.1. Quelques règles de la privacy :

Le fournisseur de services doit protéger les informations privées du client d'une manière d'assurer que ces informations ne vont pas être violés ou perdus. Avant d'invoquer un service, le client doit bien lire et doit révérer le règlement sur le respect de la vie privée. On donne quelques règles de la privacy dans les services Web par exemple :

- Lorsqu'un fournisseur du service a reçu un message privé d'un client donc il ne doit pas le renvoyer à un autre utilisateur.
- Les informations personnelles du client doivent être protégés (nom, prénom, Email…).
- Si un client envoi son code confidentiel de sa carte bancaire, le fournisseur doit effacer directement ce code avant de continuer l'invocation d'autres messages.

III.4.2. Les composants de la structure de protocole métier pour la privacy

Les différents éléments de la structure de protocole métier pour la privacy sont structurés dans la figure suivante:

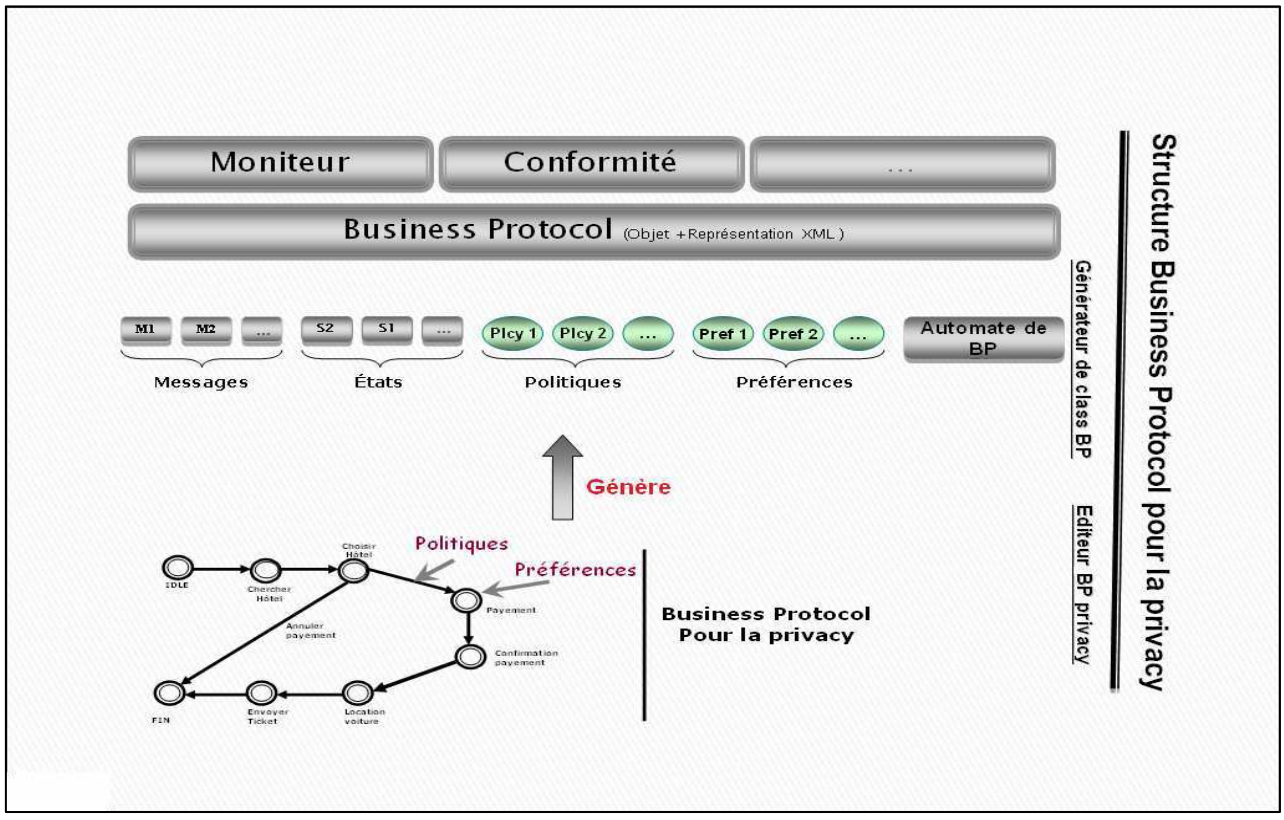


Figure III.5 : *Les composants de la structure de protocole métier pour la privacy.*

Dans ce schéma nous avons décrit les différentes composantes de la structure de protocole métier pour la privacy. Le premier élément est le procotole métier qui est un automate d'état finis déterministe, il contient des états et des transitions (messages).

Ces messages peuvent être privés ou pas donc on aura besoin des entités qui vont protéger les données privées du client, pour que le fournisseur et le client peuvent se communiqué en sécurité, le fournisseur doit signalé les conditions d'utilisation et du règlement sur le respect de la vie privée et le client doit les acceptés.

Le fournisseur va créer ses règles décrivant comment il traitera les données privées de son client appelées politiques et le client va spécifie ses propres règles appelées des préférences décrivant comment le fournisseur peut utiliser ses données privées.

Il existe deux fonctions principales pour ce protocole: un éditeur et un générateur de classes :

- **L'éditeur** du business protocol crée, publie ou introduire des PBEL.
- **Le générateur de classes** pour le business protocol engendre les différentes classes pour les états, les messages, les politiques et les préférences.

Ce protocole va être représenté par des objets donc on utilise une présentation Xml pour structuré les éléments du protocole. Quand le protocole métier soit bien spécifié, on pourra alors faire du monitoring et la conformité de la privacy.

III.4.3. Le système de monitoring :

Le service Web à surveiller et le client du service vont partager lors de leur conversation des messages SOAP, un moniteur va être un intermédiaire entre les deux, il doit satisfaire des propriétés comportementales du système et avoir des connaissances sur les statistiques pour mesurer les performances de ce système.

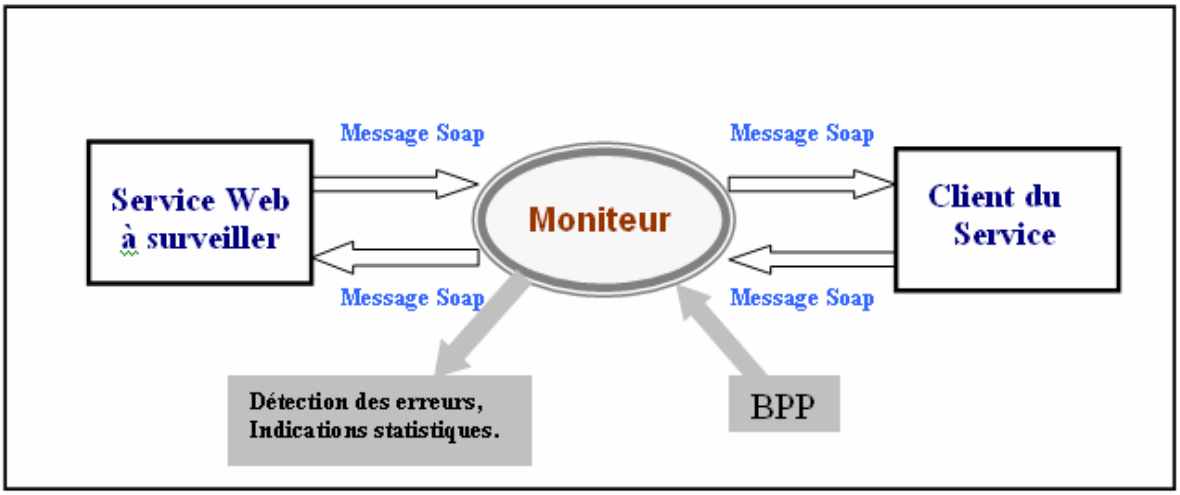


Figure III.6 : *Monitoring de la conversation.*

La figure suivante va expliquer le déroulement du système de monitoring (le moniteur) :

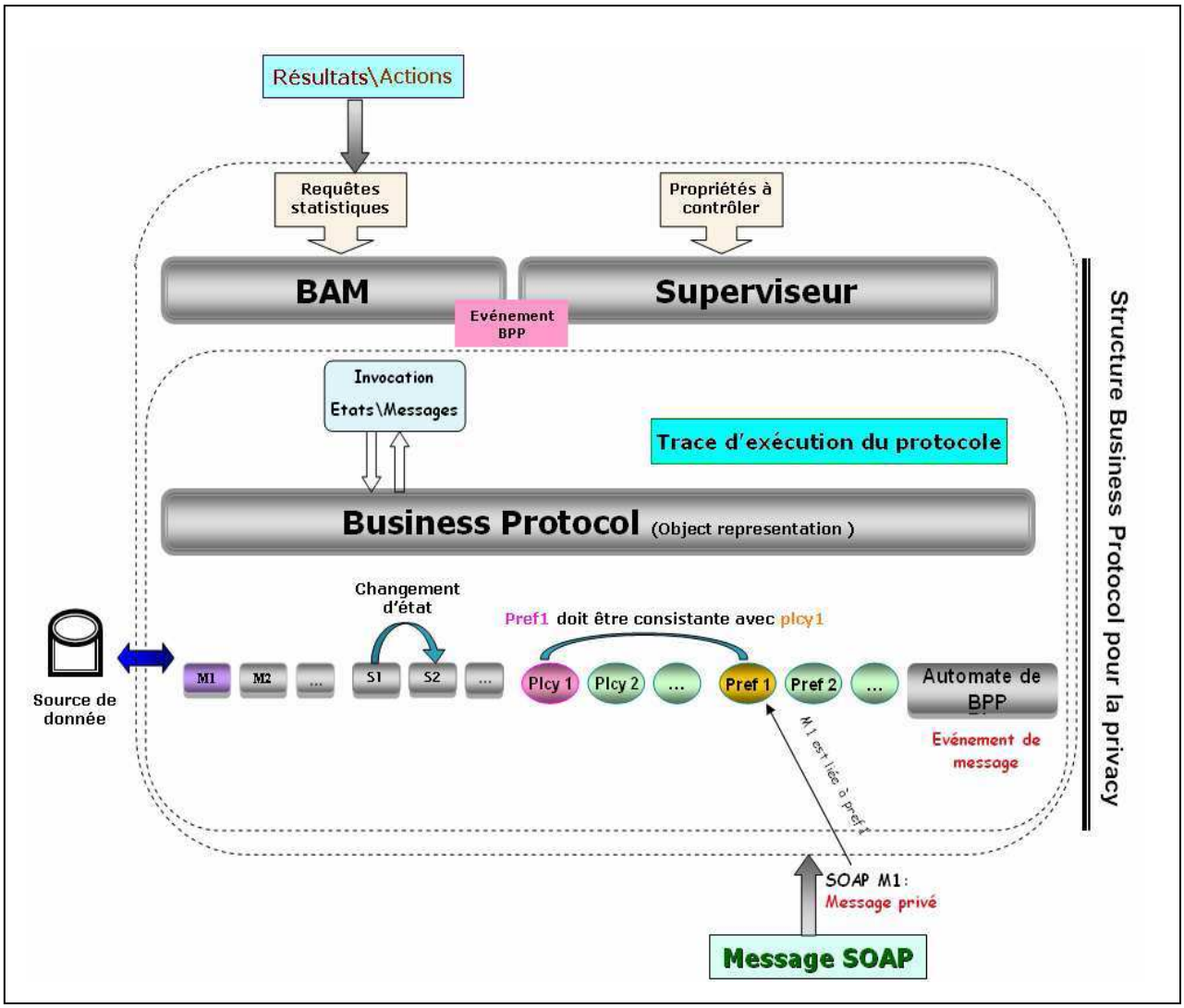


Figure III.7 : *Moniteur.*

Le moniteur a pour rôle le contrôle des propriétés de la privacy en utilisant le protocole métier. Si un message SOAP arrivé est un message privé donc il est attaché à une préférence qui doit être consistante avec la politique du fournisseur pour compléter l'exécution du reste du protocole. La vérification de cette politique sera faite par la trace d'exécution du protocole.

Si on prend comme exemple d'une règle de privacy, l'envoi du code secret de la carte bancaire du client au fournisseur du service. Ce dernier va protéger ce code en le rejetant avant de passer à un autre état.

- ***La préférence du client*** *:* le client exige la protection de son code bancaire.
- ***La politique du fournisseur*** *:* pour que le fournisseur protège le code bancaire du client, il ne va pas garder ce code après la transaction bancaire.

Le client doit accepter la politique du fournisseur qui doit lui même accepter la préférence du client. **Pref** du client doit être consistante avec **Plcy** du fournisseur de service et si on compare la consistance entre Pref est Plcy en utilisant les remarques précédentes, on trouve que **Tdu de Préf < Tdu de Plcy** comme **Donnée Pref** (protéger ma carte bancaire) est inclut dans **Donnée Plcy** (protéger la carte bancaire du client en le rejetant avant de passer à l'état suivant).

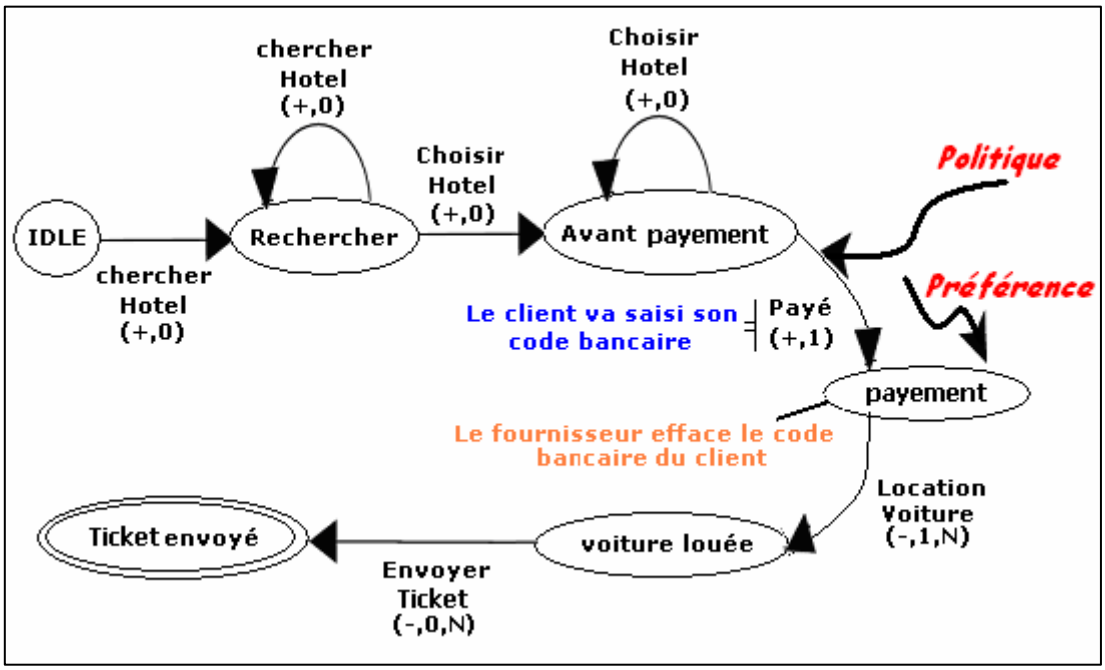


Figure III.8: *Protocole métier de la privacy pour la réservation d'hôtel.*

III.4.4. Trace d'exécution du protocole :

La trace d'exécution est un arbre qui spécifie les différents états et messages du protocole métier.

Chaque application de service Web peut avoir plusieurs instances, si un client arrive et demande un service en lui crée une instance et s'il n'en reste plus en lui demande de refaire sa demande plus tard.

Pour un seul protocole métier, on peut générer plusieurs chemins à suivre pour exécuter ce protocole. Par exemple, pour le schéma suivant, chaque instance a cinq chemins possibles pour le protocole métier correspondant. Les états sont estampillés par un temps.

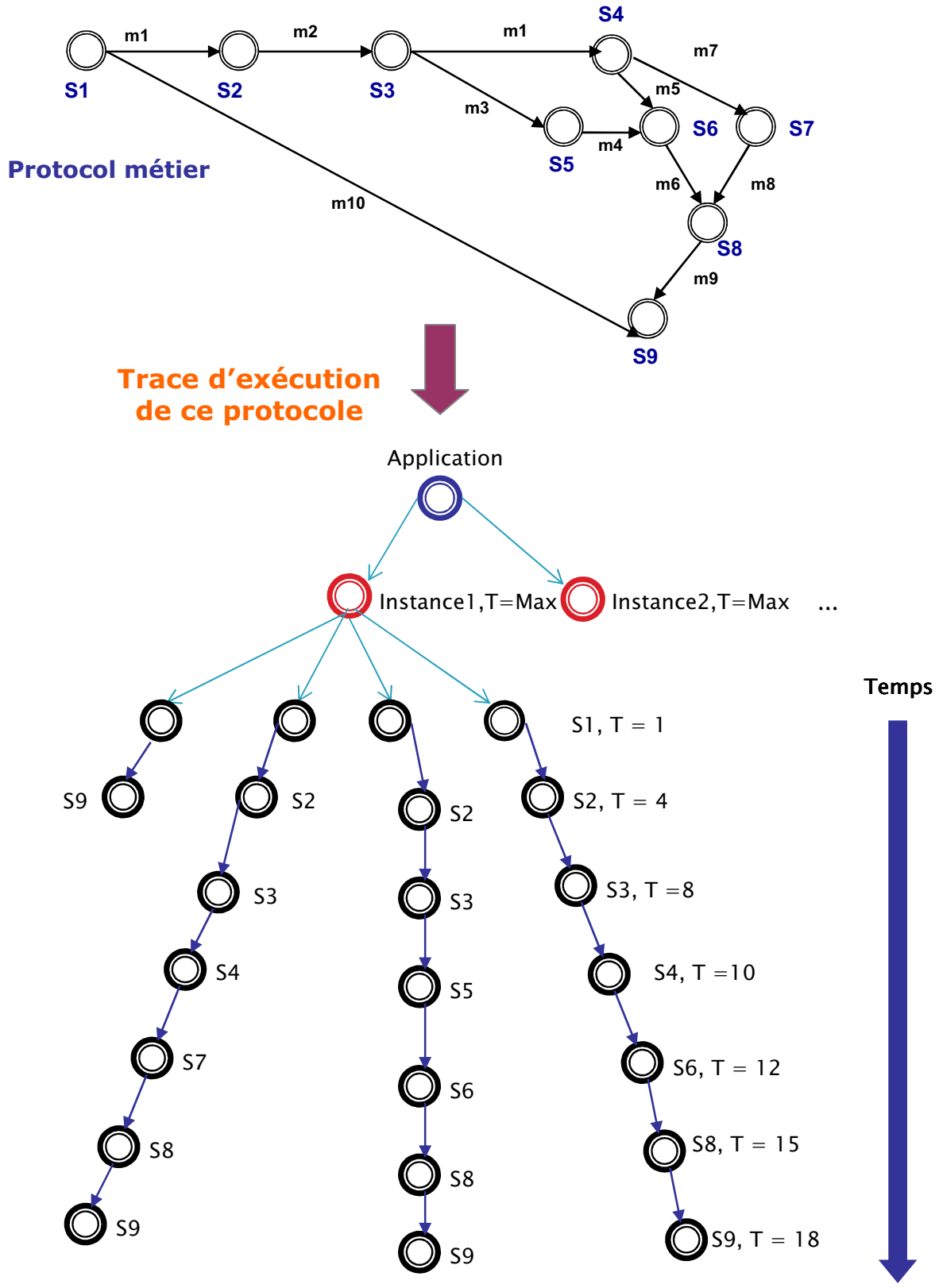


Figure III.9: *Trace d'exécution de protocole de la privacy.*

III.4.5. Monitoring de la privacy en utilisant le protocole métier :

Le moniteur ou le manager va utiliser les données stockées (messages échangés entre le client et le fournisseur) par le protocole métier de privacy d'un service Web ainsi que la trace d'exécution de ce protocole pour faire le monitoring d'une propriété de la privacy.

Chaque message arrive, on enregistre le temps de son arrivée. Le moniteur va savoir à partir des données stockées quel chemin de la trace d'exécution utilisée par le client et il compare le temps des états de la trace d'exécution et le temps des messages stockés, et la valeur de message M à l'état Si est la dernière valeur de message Mi avec Mi.T < Si.T.

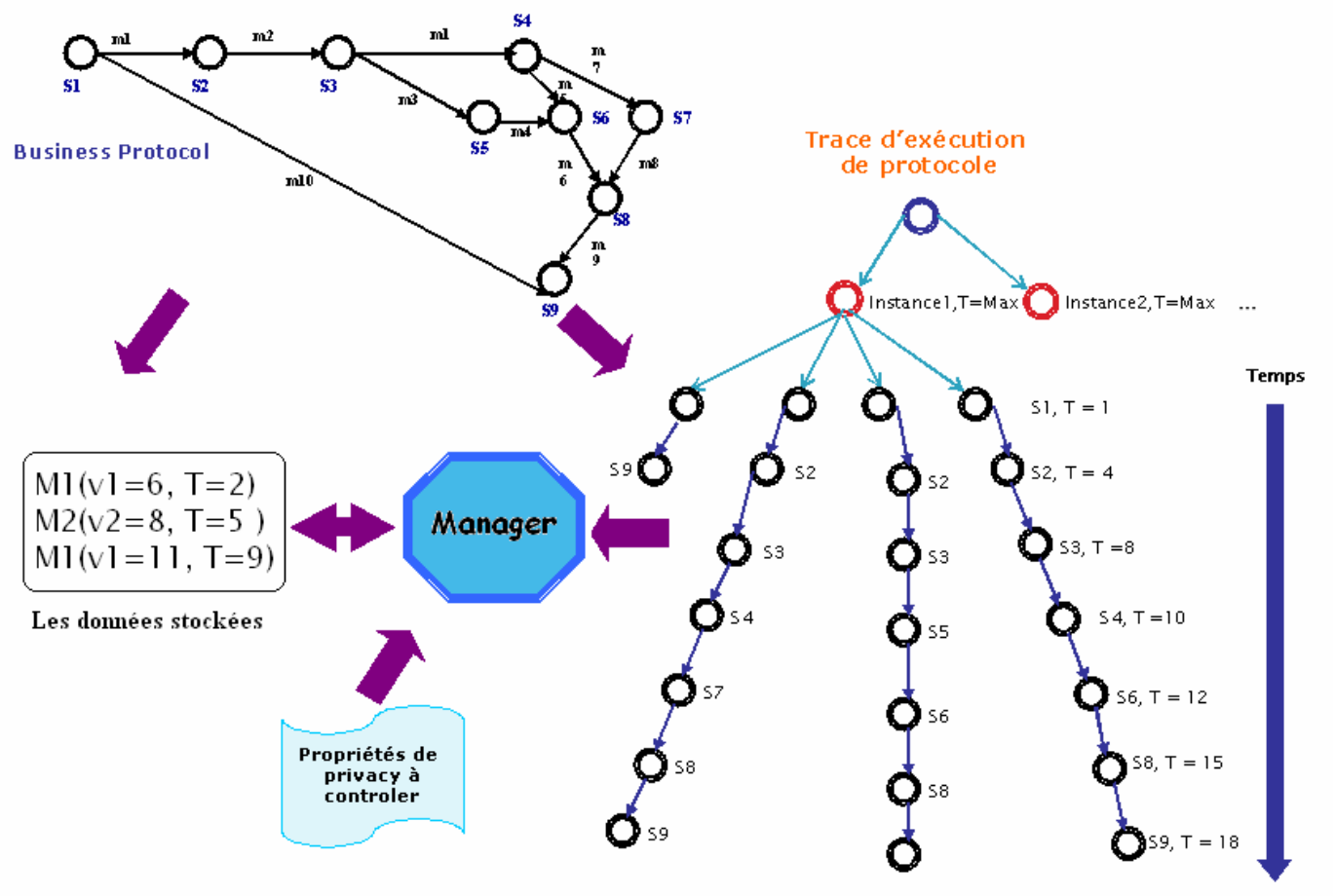


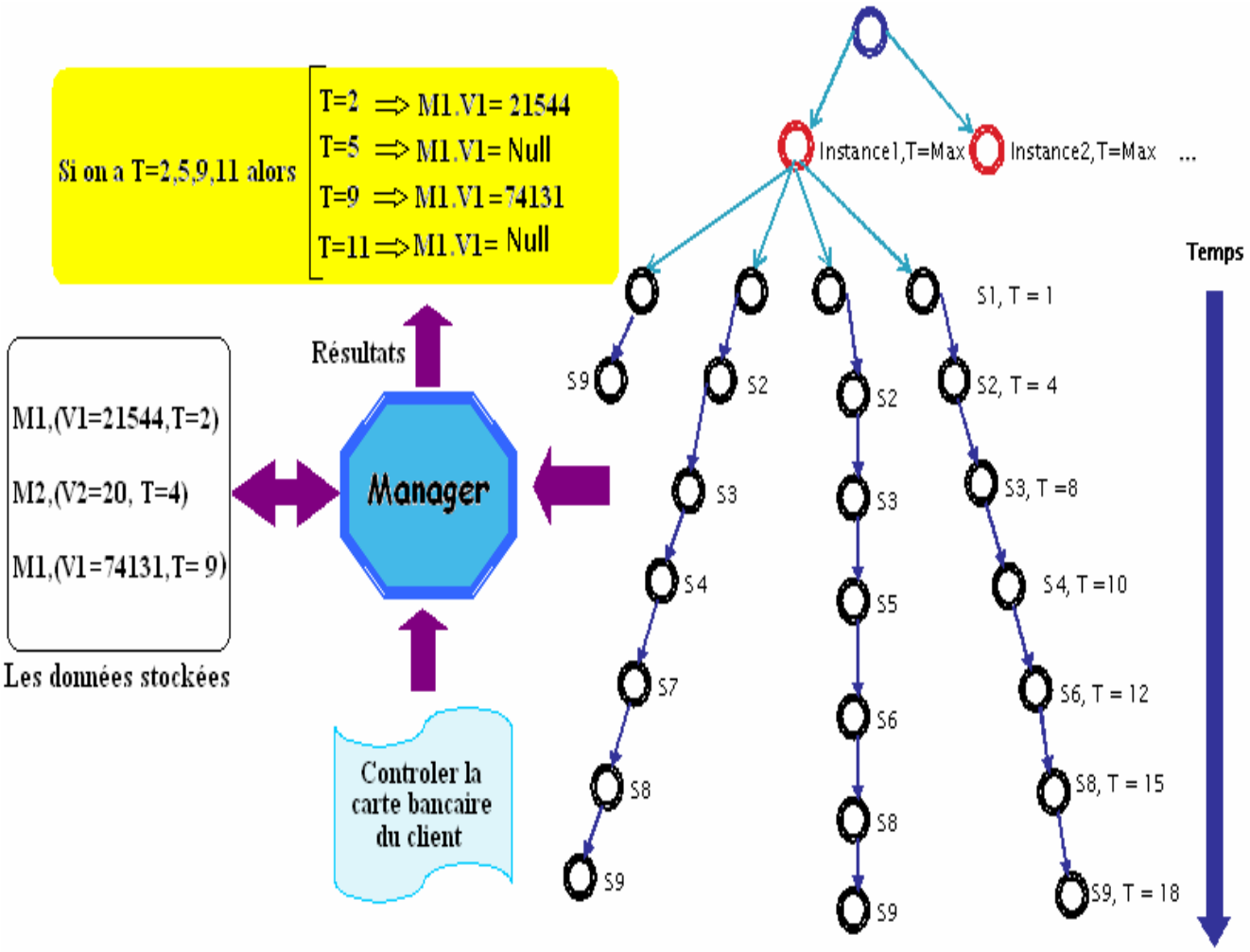
Figure III.10: *Monitoring de la privacy en utilisant un protocole métier*

Exemple d'illustration :

Si on veut par exemple surveiller la propriété de privacy suivante : l'envoi du code secret de la carte bancaire du client au fournisseur du service qui va le protéger en le rejetant avant de changer l'état on aura le résultat présenté dans le schéma suivant :

Si M1 est le message d'envoi de code bancaire du client et si on respecte le chemin dans la trace d'exécution, ce message va être envoyé deux fois dans la même conversation mais avec des valeurs différentes.

Pour respecter la propriété de privacy citée précédemment, la valeur de M1 va prendre la valeur de la carte bancaire si le temps T ∈ [M1.T, Si.T] (avec Si est l'état prochain du protocol) et elle prend la valeur Null si le temps T ≠ [M1.T, Si.T]. (Puisque la valeur de ce message va être rejetée avec le changement d'état). Et si le moniteur trouve à la place de Null un numéro donc il va signaler une erreur et il cherche directement à la corriger.



III.4.6. Langage de monitoring :

Nous allons spécifier les propriétés de la privacy par un nouveau langage BPath (développé par le groupe Liris de Lyon). Ce langage est une extension de XPath et exécuté à partir de la trace d'exécution du protocole métier, puisque le langage XPath n'arrive pas à accéder à tous les types de données (base de données, service, processus…).

XPath est un langage d'expressions permettant de décrire des ensembles de nœuds d'un document XML.

❖ ***Spécification des propriétés de la privacy par le langage BPath :***

- Exemple de spécification d'une propriété de privacy par ce langage (on a choisi de surveiller la propriété citée précédemment dans l'exemple d'illustration) :

```
IF Time IN [CurrentState/@M1.T, NextState.T] THEN
        CurrentState/@M1.V1=NumCarteBancaire
ELSE CurrentState/@M1.V1=Null
```

- Vérification de propriétés temporales de la privacy en utilisant le BPath :

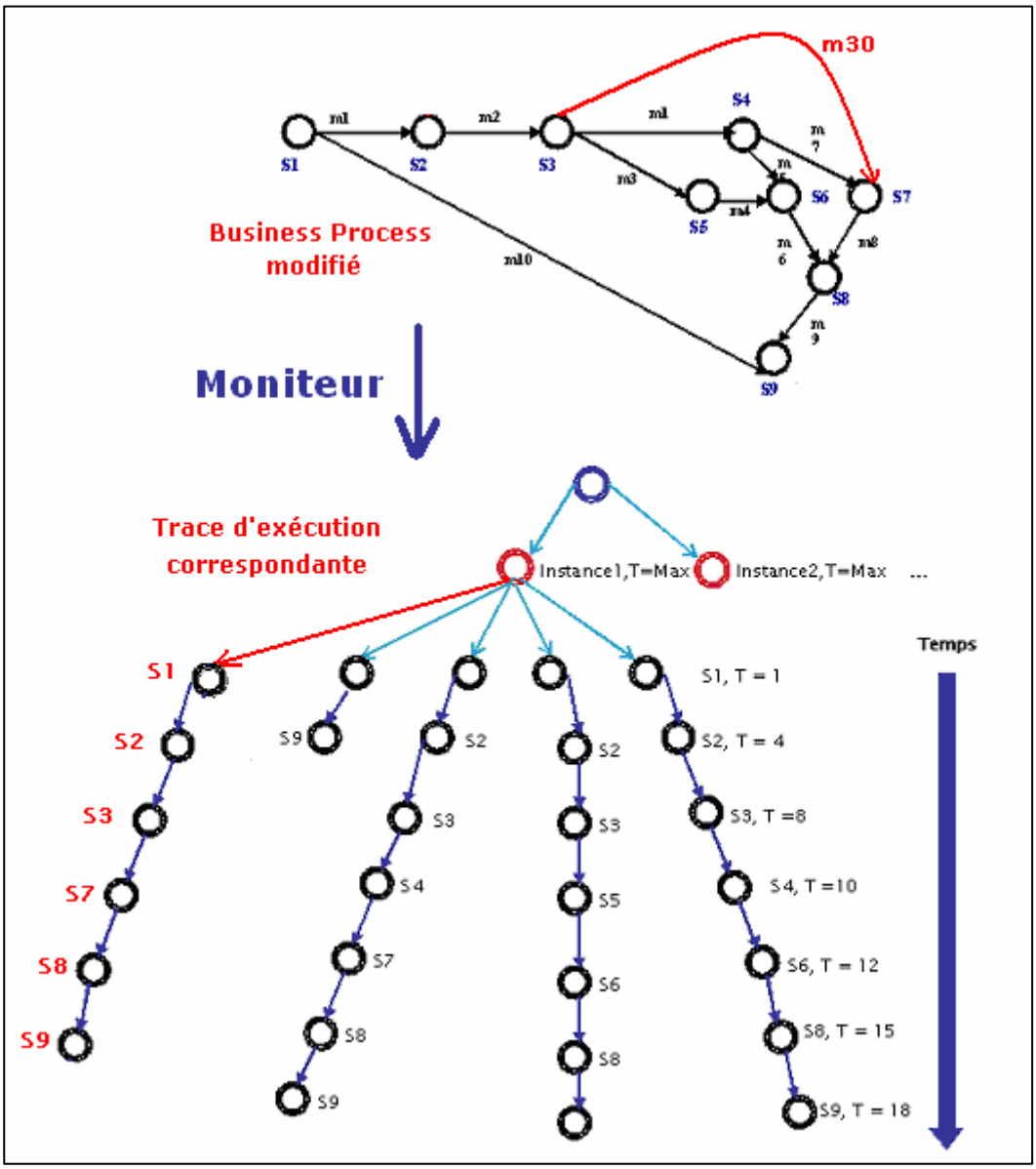
Le numéro de la carte bancaire d'un client ne sera disponible qu'au moment de son envoi jusqu'à l'état prochain du protocole, si par exemple on est dans un état dont la valeur du message envoyé est le numéro de la carte bancaire, on aura :

```
Quelqu'un dans le futur :
CurrentState//*[@M1.v1=Null]?
Quelqu'un dans le passé :
CurrentState/successor::* [@M1.v1= Null]?
Juste avant :
CurrentState/../@M1.v1= Null?
Maintenant :
CurrentState/@M1.v1= NumCarteBancaire?
Juste après :
CurrentState/*[@M1.v1= Null]
```

III.4.7. Les propriétés du langage de monitoring :

- ✓ **Dynamique:** possibilité de vérifier et de surveiller les propriétés qui sont spécifiées en temps d'exécution. Puisque on a utilisé un automate d'états pour spécifier le protocole métier donc s'il aura un changement dans ce protocole, le moniteur va changer directement la trace d'exécution de ce protocole :

  -Si par exemple, on veut ajouter un message **m30** dans le protocole métier précédent donc le moniteur va ajouter dans la trace d'exécution un autre chemin :



- ✓ **Interroger les états du protocole métier :** si on prend le schéma de la figure III.9 et on veut avoir les valeurs de M1 dans les différents états du protocole, on aura:

Pour le temps **T=9, 10, 12 ➔ Instance1/@ M1.v1= 6, 11, 11**

S3/@ M1.v1= 6, 6, 6

S4/@ M1.v1= Null, 11, 11

S6/@ M1.v1=Null, Null, 11

- ✓ **Flexibilité à découvrir les situations indésirables du système :** il existe deux manières pour régler les situations indésirables soit :

▪ L'exécution du protocole métier va être interrompue. Pour contrôler les propriétés de la privacy, le moniteur doit bloquer les messages qui possèdent des erreurs (c'est-à-dire qui ne vérifies pas les règles de privacy) au moment de leurs réceptions et d'essayer de corriger ces erreurs.

▪

```
Bloqué l'exécution
M1.au réception () {
// Faire quelques choses pour
vérifier les propriétés de privacy…}
```

▪ L'exécution du protocole métier va continuer normale et les messages vont être traités au moment de leurs envoies au système à contrôler.

```
Ne pas bloqué l'exécution
M1.à l'envoi () {
// Faire quelques choses pour
vérifier les propriétés de privacy…}
```

✓ **Spécification de politiques et de préférences de la privacy :**

- Les différentes politiques de fournisseurs vont être exprimés dans les messages privés du protocole métier.

```
M1.En Message () {
// Politique de privacy
}
```

- Et les préférences de client vont être spécifiés dans les états du protocole métier

```
S1.En État () {
// Préférence de privacy
}
```

Conclusion générale et perspective

Dans ce travail, nous constatons l'importance des travaux accentuant sur le problème de monitoring, et dans cette perspective, nous avons abordé sur les différents points qui se rapprochent avec le monitoring, plusieurs travaux dans ce domaine ont représenté, puis les plus importantes approches de monitoring ont eux aussi leurs parts. Le coeur de travail réside dans les différentes solutions, outils et contributions présentés pour résoudre le problème d'adaptation de monitoring au temps d'exécution (runtime monitoring).

Le premier effort dans ce travail est la spécification d'une nouvelle approche de monitoring au temps d'exécution. Cette approche s'appuie sur un formalisme qui s'appelle protocole métier, ce dernier nous a facilité la tache de monitoring puisqu'il se pose sur un automate d'états finis donc on aura que les états et les messages et afin d'apporter une plus grande souplesse, nous avons proposé quelques modifications de ce formalisme et puisque on a basé sur la privacy donc ce protocole va avoir comme modification, l'ajout des politiques et des préférences de la privacy. À partir de ce protocole le moniteur va réaliser une trace d'exécution qui est un ensemble de chemins existants pour invoquer le service avec des états estampillés par un temps. Enfin on a présenté des spécifications de ce système de monitoring.

Comme il n'existe pas de formalisme standard sur le problème de monitoring des services Web alors les perspectives ne peuvent être que nombreuses. Tout d'abord, améliorer notre approche afin qu'il soit capable de surveiller toutes les propriétés de la privacy avec les changements du système et qu'il soit plus adaptatif, Enfin, implémenter un prototype pour valider notre approche.

Bibliographie

[01] Luciano Baresi, Elisabetta Di Nitto, and Carlo Ghezzi: « Toward Open-World Software: Issue and Challenges ». Computer, IEEE, Volume 39, Number 10: 36-43, October 2006.

[02] Jean-Marie Chauvet, « *Services Web avec SOAP, WSDL, UDDI, ebXML* », Eyrolles Editions, 2002.

[03] Hubert Kadima, Valérie Monfort, « *Les Web Services : Techniques, Démarches et outils* », DUNOD, March 2003.

[04] DotNetGuru.fr, Approcher SOAP en douceur, 17.02.2004, 21.05.2004 http://dotnetguru.org/articles/dossiers/soadouceur/soaendouceur.htm.

[05] Domenico Bianculli, Radu Jurca, Walter Binder, Carlo Ghezzi, Boi Faltings : «Automated Dynamic Maintenance of Composite Services Based on Service Reputation». ICSOC 2007: 449-455.

[06] Box D., Ehnebuske D., Kakivaya G., and Layman A: « Simple object access protocol (soap) specification ». Technical report, World Wide Web Consortium, W3C Note, May 2000.

[07] Christensen E., Curbera F., Meredith G., and Weerawarana S: « Web services description language (wsdl) specification ». Technical report, World Wide Web Consortium, W3C Note, March 2001. http://www.w3.org/TR/wsdl/.

[08] Bellwood T., Clément L., Ehnebuske D., and Hately A. Universal description, discovery and integration (uddi) specification. Technical report, OASIS Committee, July 2002. http://www.UDDI.org/specification.html.

[09] N. Halfoune: « Vers une composition dynamique de services Web », mémoire de magistère de l'école doctorale de Bejaia, 2004.

[10] R. Badonnel: **«** Composition de services et supervision : application aux Web Services », mémoire DEA de l'Université Henri Poincaré – Nancy I, 2003

[11] Tanguy Crusson, « Business Process Management », 2003.

[12] CRP Henri Tudor : « Modélisation des processus métiers : Etat de l'art et conseils pratiques », Projet SPINOV 2006.

[13] M. Tissot, « Environnements d'édition de workflow », projet Opéra- INRIA 2000.

[14] A. Sahai and al : « Automated sla monitoring for web services». DSOM 2002, IEEE Press, October 2002.

[15] L. Baresi, S. Guinea, and P. Plebani: « WS-Policy for Service Monitoring ».TES 2005, LNCS 3811, pp. 72–83.

[16] R. Sturm, W. Morris, and J. Mary: « Foundations of Service Level Management ». SAMS Publishing, January 2000.

[17] G. Spanoudakis, C. Kloukinas, and K. Androutsopoulos, « Towards security monitoring patterns, » in Proceedings of the 2007 ACM Symposium on Applied Computing (SAC), 2007, pp. 1518–1525.

[18] W3C, *HTTP protocol*, HTTP Protocol Web Page: http://www.w3.org/Protocols/

[19] W3C. *XML : eXtensible Markup Language*. XML Web Page: http://www.w3.org/XML

[20] SUN. J2EE overview. URL: http://www.sun.org/j2ee/overview.html.

[21] CORBA. URL: http://www.corba.org/.

[22] W3C. *Xmlschema***.** Xmlschema Web Page: http://www.w3.org/TR/Xmlschema-0

[23] G. Spanoudakis and K. Mahbub, « Requirements monitoring for servicebased systems: Towards a framework based on event calculus, » in 19th IEEE International Conference on Automated Software Engineering (ASE 2004), 20-25 September 2004, Linz, Austria, 2004, pp. 379–384.

[24] K. Mahbub and G. Spanoudakis: « Run-time monitoring of requirements for systems composed of web-services », Initial implementation and evaluation experience. In: ICWS. 2005 IEEE International Conference on Web Services, IEEE Computer Society Press, Orlando, Florida, USA (2005).

[25] A. Lazovik, M. Aiello, and M. P. Papazoglou, « Associating assertions with business processes and monitoring their execution, » in Service-Oriented Computing - ICSOC 2004, Second International Conference, 2004, pp. 94–104.

[26] D. Bianculli and C. Ghezzi, "Monitoring conversational web services," in IW-SOSWE'07, 2007.

[27] M. Pistore and P. Traverso, « Assumption-based composition and monitoring of web services, » in Test and Analysis of Web Services, L. Baresi and E. D. Nitto, Eds. Springer, 2007, pp. 307–335.

[28] F. Barbon, P. Traverso, M. Pistore, and M. Trainotti, « Run-time monitoring of instances and classes of web service compositions, » in IEEE International Conference on Web Services (ICWS 2006), 2006, pp. 63–71.

[29] S. Benbernou, H. Meziane, and M.-S. Hacid: « Run-Time Monitoring for Privacy-Agreement Compliance». In Service-Oriented Computing – ICSOC 2007, Fifth International Conference, 2007, pp. 353–364.

[30] C. Beeri, A. Eyal, T. Milo, and A. Pilberg: « Monitoring business processes with queries», in Proceedings of the 33rd International Conference on Very Large Data Bases, 2007, pp. 603–614.

[31] A. Azough and E. Coquery and M-S. Hacid: « Management of Changes in Web Service Protocols ».

[32] Guermouche, N., Benbernou, S., Coquery, C.E, Hacid, M.: « Privacy-aware web service protocol replaceability ». In: ICWS'07. IEEE International Conference on Web Services, IEEE Computer Society Press, Salt Lake City, USA (2007).

[33] Machiraju V., Sahai A., and van Moorsel A: « Web services management network ». IM 2003, IEEE Press, March 2003.

[34] A. Keller and H. Ludwig: « The wsla framework: Specifying and monitoring service level agreements for web services ». Technical report, IBM Research, March 2003.

[35] A. Sahai, A. Durante, and V. Machiraju: « Towards automated sla management for web services ». Technical report, Hewlett-Packard Labs. July 2002.

[36] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. « Implementing p3p using database technology». International Conference on Data Engineering, 2003.

[37] C. Ghezzi and S. Guinea: « Run-Time Monitoring in Service-Oriented Architectures », Test and Analysis of Web Services 2007, pp. 237-264

[38] Présentation inspirée de: ZDNet.fr, SOA, 28.10.2003,21.05.2004. http://www.zdnet.fr/builder/architecture/services_web/0,39021055,39127269-*00.htm*.

[39] SLA: "Service Level Agreement" disponible à : http://www.service-level-agreement.net

Index

BPM : Business Process Management

BP-QL : Business Process Query Language

CORBA : Common Object Requesting Broker Architecture

BPP : Business Process pour la Privacy

HTTP : HyperText Transport Protocol

J2EE : Java 2 platform Enterprise Edition

LTL : Linear Temporal Logic

P3P : Platform of Privacy Preferences

RPC : Remote Procedure Call

RTML : Run-Time Monitoring Language

SLA : Service Level Agreement

SOA : Service Oriented Architecture

SOAP : Simple Object Access Protocol

TDU : Term Data Unit

UDDI : Universal Description, Discovery and Integration

W3C : World Wide Web Consortium

WSDL : Web Services Description Language

WS-CoL : Web Service Constraint Language

WSMN : Web Services Management Network

XML : eXtensible Markup Language

XSRL : Xml Service Request Language