Table des Matières

Tables des Figures

INTRODUCTION

Les web services sont l'aboutissement logique de la quête par les communautés académique et industrielle du middleware idéal. En effet, les web services s'appuient sur des standards techniques (HTTP, XML, SOAP, SMTP, FTP, WSDL, UDDI…) acceptés par tous les intervenants dans la chaine complexe d'un système d'information distribué. Ils ont réussi là où toutes les autres initiatives précédentes ont échoués à cause surtout de l'hétérogénéité des plateformes et des systèmes, mais aussi pour des raisons autres que techniques (financières, politiques, tentative d'accaparation de parts de marché …etc.). Le succès phénoménal de l'internet a permis de rendre la famille des protocoles TCP/IP incontournable, ce qui à son tour a rendu l'organisme de standardisation W3C très puissant au point d'imposer ses standards à tout le monde.

Cependant, avec le temps, d'autres problèmes sont apparus lors de l'exploitation de ces web services dans des conditions réelles, que ce soit lors de la phase de développement ou de l'exploitation.

Pour communiquer entre eux, les web services utilisent un paradigme d'échange de messages. Les messages eux-mêmes et leurs attributs sont bien documentés par les protocoles dédiés (WSDL, SOAP, UDDI), mais aucun protocole ou spécification ne décrit les séquences correctes de ces messages pour compléter une cession « correcte » (c'est-à-dire sans incidents) entre un web service et ses clients ou ses prestataires.

Pour résoudre ce problème, la communauté de recherche a proposé la notion de protocole métier ou « business protocole », c'est-à-dire un modèle formel qui permet de décrire toutes les séquences valides de messages (appelés aussi conversations)

échangés par un web service avec ses clients ou ses prestataires dans le cas où il se comporte lui-même comme un client.

Dans le monde académique, les business protocoles font l'objet d'un intense effort de recherche, d'où le développement de plusieurs variantes, chacune adressant un aspect ou un autre d'un web service. Bien sûr l'idéal serait de développer un business protocole qui englobe tous les aspects.

La privacité est l'un de ces aspects. Dans [NSEM00] les auteurs proposent un modèle formel hiérarchique qui utilise le paradigme « politiques/préférences » déjà présent dans P3P (The Platform for Privacy Preferences), mais malheureusement il ne prend pas en compte les contraintes temporelles. Ces derniers sont souvent une exigence de tout système informatique distribué réel. Tout business protocole qui ne les supporte pas serait incomplet et non expressif.

Problématique

Les web services sont de plus en plus utilisés comme solution pour les interactions application-à-application à l'intérieur et à l'extérieur de l'entreprise [ACKM04]. L'interopérabilité des applications était et reste toujours un problème difficile à traiter à cause de l'hétérogénéité et de l'autonomie des systèmes informatiques. Les web services fournissent les abstractions et les technologies adéquates pour exposer les applications de l'entreprise comme des services et rendre leur programmation accessible via des interfaces standards. En effet, les web services apportent à l'intégration des applications la standardisation du langage de description, des protocoles de communication et des protocoles de coordination (voir [ACKM04], [PaGe03], [PTDL07] et [PaHe07]). Concrètement, la standardisation du langage de définition de l'interface (WSDL) et du protocole de transport (SOAP) permet d'obtenir une interopérabilité basique au niveau de la couche messages, suffisante pour que les entreprises commencent à les utiliser réellement dans leurs systèmes opérationnels

[Vino04]. Ceci est un progrès énorme par rapport aux middlewares classiques (à base de systèmes RPC [1], ou MOM [2]), où le développement des adaptateurs pour l'interopérabilité des systèmes aux niveaux des points d'entrées des réseaux et des formats de données étaient nécessaire.

Bien qu'un long chemin ait été parcouru vers une interopérabilité basique, il reste beaucoup à faire pour faciliter le développement et l'interaction des systèmes. En particulier, un aspect important affecte les web services, c'est leur « couplage faible », c'est-à-dire qu'ils ne sont pas développés pour interagir avec un nombre restreint de clients spécifiques, mais ils sont destinés à servir un nombre plus grand (et même inconnus) de clients, développés par des équipes différentes ou même des entreprises différentes. D'où la nécessité pour les développeurs d'applications clientes de connaitre tous les aspects fonctionnels et non-fonctionnels d'un service pour pouvoir déterminer s'ils peuvent interagir correctement avec lui. La description du service doit donc spécifier les propriétés syntaxiques et sémantiques du service pour les mettre à disposition de ses clients potentiels dans le but de:

1. Assister les développeurs dans la création de clients qui peuvent correctement utiliser et interagir avec le service.
2. Rendre possible la sélection, en phase de développement ou d'exécution, du service qui convient le mieux au client,

D'où la nécessité d'une description de service plus riche que la description d'interface classique des middlewares (comme par exemple CORBA IDL). Pour être plus précis, nous avons besoin, en plus de la description de l'interface du service, du business protocole du service (la spécification des séquences de messages supportés par le service [BFFB04]). Les outils de développement qui existent actuellement, se

[1] Remote Procedure Call

[2] Message Oriented Middleware

focalisent majoritairement sur l'interopérabilité dans la couche des protocoles de base (ex: faire la correspondance entre JAVA et WSDL, ou rendre deux systèmes SOAP compatibles entre eux), mais il y a très peu d'outils pour l'analyse et la modélisation de protocoles.

Les business protocoles peuvent être spécifiés par BPEL [AAAB07] , ou par n'importe quel autre formalisme crée pour ce but (exemple [BFFB04] ou [Bera05]). La description du service par son business protocole n'est pas suffisante pour faciliter le développement et l'invocation du service. Nous aurons besoin de méthodes formelles et d'outils logiciels qui réalisent une analyse automatique dans le but de:

1. Identifier si une interaction entre un service et un client est possible (ou non).
2. Identifier les conversations qui sont supportées.
3. Identifier les points de divergences entre les protocoles du service et celui de son client, dans le but, si c'est possible, de proposer un adaptateur pour les rendre compatible.
4. Fournir un mécanisme de tolérance aux pannes, en identifiant la possibilité de remplacer un service par d'autres d'une façon complète ou partielle.

Le besoin de méthodes formelles et d'outils de génie logiciel est largement reconnu, et certains travaux ont été déjà réalisés dans ce but. Dans [BeCT04], [BeCT06] et [BFFB04], une approche est présentée. Elle consiste en un modèle formel de business protocole et un cadre de travail pour analyser la compatibilité et la remplaçabilité des BP. Un outil logiciel qui implémente ce concept est présenté dans [BeMo06].

Une catégorie très importante des business protocoles est celle qui prend en compte les contraintes temporelles (appelés *Business Protocole Temporisé)*. En effet le temps est une abstraction cruciale qui a été étudiée dans plusieurs travaux de recherche comme les systèmes de flux (workflow systems),(voir [BeWJ02], [MaMZ06] et [Maco05]) et même les web services (voir [BRSM03], [DCPV06] et [KaPP06]). On

peut citer un nombre infini d'exemples où l'aspect temporel est primordial pour le fonctionnement des systèmes distribués en général (comme par exemple l'interaction entre les utilisateurs des sites web du commerce électronique comme *ebay* ou *Amazon*), mais c'est aussi le cas des web services en particulier ([BeCT04] et [Khal07]). Ceci dit, dans la plupart des études, le temps est considéré, dans le cadre des vérifications temporelles traditionnelles, comme la présence d'une condition ou son absence, ou l'évitement des inter-blocages (deadlocks) (voir [BRSM03], [DCPV06] et [KaPP06])

Dans [PoBC10], les auteurs dotent le business protocole étudié dans [BeCT04], [BeCT06] et [BFFB04] de la capacité d'exprimer les contraintes temporelles. Pour cela ils substituent l'automate à états finis par une sous-classe de l'automate temporisé de (Alur et Dill) [RaDl94]. En effet, ce dernier, n'étant pas déterministe, pose des problèmes insurmontables lors de la modélisation des business protocoles. La sous-classe utilisée est ECTA (Event Clock Timed Automata) [AlFH99]. Elle possède la faculté d'être « déterminisable », c'est-à-dire qu'on peut calculer un automate déterministe équivalent pour un automate non-déterministe, ce qui n'est pas le cas des automates temporisés « généralistes ». Ensuite comme dans [BFFB04], quatre opérateurs spéciaux sont définis (composition compatible, intersection, différence, et projection) dans le but de réaliser une analyse de remplaçabilité et de compatibilité.

En plus des aspects cités plus haut, qualifiés souvent d'aspects fonctionnels, des aspects non-fonctionnels, réclament aussi l'attention lors de la phase de développement ou d'exploitation des web services. Ces aspects non-fonctionnels sont la sécurité, la qualité de service, la confiance (Trust), la privacité…etc. Ces aspects ne sont pas spécifiques uniquement aux web services, mais concernent tous les systèmes informatiques passés, présents ou à venir. A l'inverse des aspects fonctionnels, les aspects non-fonctionnels n'ont pas bénéficié d'un effort soutenu de recherche par la communauté de recherche sur les web services. Néanmoins on peut

citer les travaux [GBCH07a] , [HaPB07], [HeKi09] et [QaAb10]. La privacité a déjà été traité dans les systèmes distribués où des solutions ont été développées comme P3P [AKSX03] ou EPAL [AHKP03]. Il faut aussi citer un effort de la part de W3C envers la privacité mais dans le cadre d'un large Framework « la pile sécuritaire » qui contient parmi d'autres spécifications, les 2 spécifications WS-POLICY et une sous-spécification de cette dernière: WS-PRIVACY [Ibmc02].

Dans l'url : http://www.innoq.com/soa/ws-standards/poster/ on peut trouver un poster qui décrit « la pile sécuritaire » ainsi que les autres protocoles et standards des web services. Selon le poster précédent, la plupart de ces standards sont toujours à l'état de « draft » et ne sont donc pas finalisés. La spécification WS-POLICY décrit les politiques métiers qui doivent être imposées aux points terminaux et intermédiaires. Une politique métier décrit certains prérequis comme les jetons (tokens) de sécurité, les algorithmes de cryptage supportés et des règles de privacité. Concrètement, WS-POLICY est un infoset XML représentant un ou plusieurs déclarations de politique. WS-POLICY utilise d'autres spécifications comme WS-POLICYASSERTIONS pour définir des assertions de messages, ou WS-SECURITYASSERTIONS une sous-spécification de WS-SECURITY. Cependant la spécification WS-POLICY ne décrit pas les règles de privacité en détails. C'est le rôle de WS-PRIVACY de décrire un modèle de préférences de privacité des sujets et les pratiques de privacité des organisations. Cependant WS-PRIVACY n'a pas été finalisée [HuFC04].

CONTRIBUTION

Le travail présenté dans cette thèse a pour but d'améliorer la description de l'interface des web services, en renforçant cette dernière (exprimée dans le langage et protocole WSDL) par une spécification formelle qui décrit les aspects fonctionnels et non-fonctionnels d'un web service. La spécification formelle utilisée est le protocole

métier ou « Business Protocole: BP » introduit dans [BFFB04], [BeCT06], et [BeCT04]. Le BP nous permet d'imposer un ordre aux messages échangés entre les web services ainsi que la spécification des messages de début (un seul) et de fin (plusieurs cas possibles). Cette séquence de messages est appelée la conversation. D'autres travaux de recherches ont eu pour but d'étendre le BP afin de traiter efficacement les contraintes temporelles [PoBC10], [MBRH09] et [MBSC08]. En Effet, cette abstraction qui est le temps, s'est avérée essentielle dans l'expression des règles métiers (business rules) de toute modélisation d'application qu'elle soit distribuée ou non. En réalité « la distributivité » de l'application ne fait qu'amplifier le besoin de l'abstraction du temps pour les règles métiers et pour les contraintes techniques temporelles comme la latence des réseaux…etc.

La contribution de cette thèse réside dans les points suivant:

- Etendre le business protocole pour qu'il modélise les contraintes temporelles, en utilisant comme outil formel sous-jacent l'automate d'états finis temporisé à horloges événementielles: ECTA (Event-Clock Timed Automata) de (Alur et al. [AlFH99]). ECTA est une sous-classe de l'automate temporisé de (Alur et Dill[RaDl94]), qui possède la caractéristique d'être «déterminisable» (une condition sine qua non pour modéliser un BP). Le BP obtenu nous permettra alors de faire des analyses de compatibilité et de remplaçabilité en prenant en compte les contraintes temporelles.

- Annoter le BP temporisé obtenu par un modèle hiérarchique de privacité, pour exprimer l'échange de données sensibles entre web services. Le modèle de privacité utilisé est basé sur le paradigme « politique/préférence», qu'on trouve auparavant dans P3P [AKSX03] ou EPAL [AHKP03]. Le modèle de privacité présenté possède la faculté d'exprimer les préférences des utilisateurs en matière de privacité même lorsqu'elles sont traitées par un service autre

que celui avec qui ils étaient en contact direct. En effet, le service prestataire doit imposer les préférences de ses clients, lorsqu'il doit les sous-traiter avec d'autres services prestataires pour lesquels il se comporte lui-même en client.

Notre BP appelé « Business Protocole Privé Temporisé » (BPPT) est donc capable de supporter l'aspect fonctionnel (spécification de l'ordre des messages – c'est-à-dire – la conversation avec prise en compte des contraintes temporelles), et un aspect non-fonctionnel (la prise en compte des préférences de privacité).

L'organisation du manuscrit

Cette thèse est organisée en 6 chapitres qui présentent dans le détail les différents aspects du travail de recherche qui a abouti à la notion de Business Protocol Privé Temporisé (BPPT).

Chapitre 1

Ce chapitre introduit les web services et la pile des protocoles de base. Les autres protocoles qui traitent les autres aspects non-fonctionnels (sécurité, privacité, trust, QoS,…etc.) sont aussi abordés. Ensuite la notion de composition de web services est introduite ainsi que les protocoles et les langages qui lui sont intrinsèques. Nous introduisons aussi les processus métier (Business Process), et nous clarifions leurs relations par rapport aux web services.

Chapitre 2

Dans ce chapitre, c'est la notion de privacité qui est introduite. Nous commençons par sa définition dans le cadre des technologies de l'information en général, puis dans le cadre des systèmes distribués. Nous la comparerons avec la notion de sécurité avec laquelle elle est souvent confondue. Deux implémentations principale (P3P et EPAL) sont abordés ainsi que quelques travaux de recherches qui traitent la privacité dans le cadre d'une architecture SOA (Service Oriented Architecture).

Chapitre 3

Dans ce chapitre nous introduisons un modèle privacité hiérarchique basé sur le paradigme « politique/préférence». Des opérateurs de comparaison seront introduits dans le but de pouvoir comparer des politiques entre elles ou avec des préférences. Le modèle de privacité sera ensuite utilisé pour annoter un business protocole dans le prochain chapitre.

Chapitre 4

Dans ce chapitre nous introduisons la notion de business protocole dont le modèle formel utilise les automates à états finis (final state machine-FSM).

Nous passons en revue les différents modèles du business protocoles. Dans un premier temps nous présentons le modèle original de [BFFB04], puis le modèle qui dote le business protocole de la capacité de préserver la privacité [GBCH07a]. Puis nous présentons les notions de remplaçabilité des politiques privés en précisant les différentes classes. Nous introduisons aussi la notion de simulation de protocoles, une notion proche de la remplaçabilité qu'on retrouve souvent dans la littérature spécialisée.

Chapitre 5

Dans ce chapitre nous introduisons les automates à état finis temporisés (ou TA pour Timed Automata) de (Dill et Alur) [RaDl94]. Les TA et surtout ECTA – (Event Clock Timed Automata) sont « le socle formel » sur lequel notre business protocole est bâti. Ensuite nous introduisons le BPPT (Business Protocole Privé Temporisé) généralement et formellement en précisant sa syntaxe et sa sémantique.

Le BPPT permet d'exprimer les contraintes temporelles de « la logique métier » d'un web service implémentant un système d'informations distribué e-business (aspect fonctionnel), et permet aussi d'exprimer les exigences de privacité (aspect non-fonctionnel).

Chapitre 6

Dans ce chapitre nous utilisons les capacités de calcul et de traitement des BPPTs pour vérifier que les exigences de privacité et les contraintes temporelles sont respectées (ou non).

Nous commençons par introduire les notions de « protocole courant » et de « protocole transversal », ainsi que la notion de « lien de transition » qui existe entre eux.

La vérification de l'aspect fonctionnel consiste dans la vérification des contraintes temporelles exprimées par les contraintes sur les horloges du BPPT.

La vérification de l'aspect non-fonctionnel consiste dans la vérification des contraintes de privacité.

Enfin, nous terminons par une conclusion et une brève discussion sur d'éventuels axes de recherches et perspectives.

Chapitre 1

Les Web Services et Processus Métier

1 SERVICES WEB ET PROCESSUS MÉTIER

1.1 Introduction

Les Services Web sont une implémentation concrète de l'architecture SOA (Service Oriented Architecture) [Davi00]. Cette dernière décrit un paradigme dans lequel les logiciels sont perçus comme un ensemble de services dont l'interface expose, d'une manière neutre par rapport aux plateformes logiciels et hardwares, les fonctions qui peuvent être appelées par toutes sortes de clients. Pour atteindre ce but, un langage universel qui permet une interopérabilité parfaite est nécessaire. Ce langage c'est XML [TJCE00]. De plus, nous aurons besoin de l'infrastructure des réseaux TCP/IP [Wiki11] (le type de réseaux les plus repandus) et plus particulièrement le protocole HTTP [Tber96] . Les services web peuvent aussi être perçus comme une évolution naturelle des Middleware des systèmes distribués [Hmah04], depuis les premiers mécanismes RPC (Remote Procedure Call) mono-langages [Sunm88] (principalement C et C++) aux EAI (Enterprise Application Integration) en passant par les Object Brokers [Grou11] et les MOM (Message-oriented middleware). L'adoption massive par le monde académique et industriel des web services fait qu'ils sont aujourd'hui incontournables dans tout système d'information moderne. Cette adoption est renforcée par une standardisation par le consortium WC3 (The World Wide Web Consortium -organisme de standardisation de toutes les technologies internet).
WC3 valide (on non), les propositions de ces membres (entité académique ou industrielle) et se propose aussi de définir des "profiles" qui correspondent à des degrés divers d'interopérabilité (exemple le profile WS-I[3] - The OASIS Web Services Interoperability - qui définit le strict minimum que chaque nœud doit avoir pour réussir une conversation web service).

[3] http://www.ws-i.org/

1.2 Les technologies des Web Services

Dans la littérature spécialisée, plusieurs taxinomies ont été proposées (par exemple dans [BeMH07]) pour classer les différents aspects que peuvent avoir les services Web. L'une d'elles (celle qui est pertinente à ce travail), consiste à classer leurs propriétés en 2 catégories: Propriétés fonctionnelles et Propriétés non- fonctionnelles.

1. **Les propriétés fonctionnelles (PFs)** décrivent ce que fait exactement un service (ses fonctionnalités) et comment ces fonctionnalités sont accomplies.
2. **Les propriétés non-fonctionnelles (PNFs)** décrivent des contraintes sur les propriétés fonctionnelles lors de l'exécution des services. Parmi ces propriétés non-fonctionnelles, nous avons la qualité de services, la privacité (privacy), la confiance, etc…

1.2.1 Les propriétés Fonctionnelles

Dans cette section nous abordons les propriétés fonctionnelles d'un service. Ces dernières font l'objet d'un grand effort d'étude et de standardisation par les communautés académiques et industrielles.
Le WC3[4] définit un service web comme suit:
« Un service web est un logiciel identifié par une URI (Uniform Resource Identifier,), dont l'interface est publique et les liaisons sont définies et décrites en utilisant XML. L'environnement du service offre le moyen à d'autres logiciels de découvrir celui-ci. Ce service interagit avec les autres services web en respectant cette définition, donc en utilisant des messages basés sur XML acheminés par des protocoles Internet ».
Cette définition ne présuppose ni l'utilisation du protocole SOAP (Simple Object Access Protocol,) [Mitr03] pour assurer le transport et la communication entre les entités, ni une description du service web par le langage de description WSDL (Web Services Description Language,)[EFGS01]. Mais l'architecture de référence d'un service web suppose que ce service possède un niveau d'abstraction dans lequel il est

[4] http://www.w3.org/

décrit par le langage de description WSDL et qu'il emploi le protocole de communication SOAP. Le protocole UDDI [Wwwo02] ne bénéficie pas aujourd'hui de l'appui des industriels (ni même du monde académique), et ne fait donc pas parti du "noyau de base" des protocoles web service. Malgré cela on le trouve souvent dans la littérature spécialisée associée à SOAP et WSDL, probablement pour des raisons historiques.

Description du modèle fonctionnel des web services

Les services web s'articulent autour du protocole d'échange de données SOAP (lui-même basé sur XML). Ce protocole se situe dans une couche supérieure des protocoles de niveaux applicatifs de l'Internet comme HTTP, FTP, SMTP, etc. Ces derniers encapsulent donc les messages XML issues du protocole SOAP dans leurs propres messages.

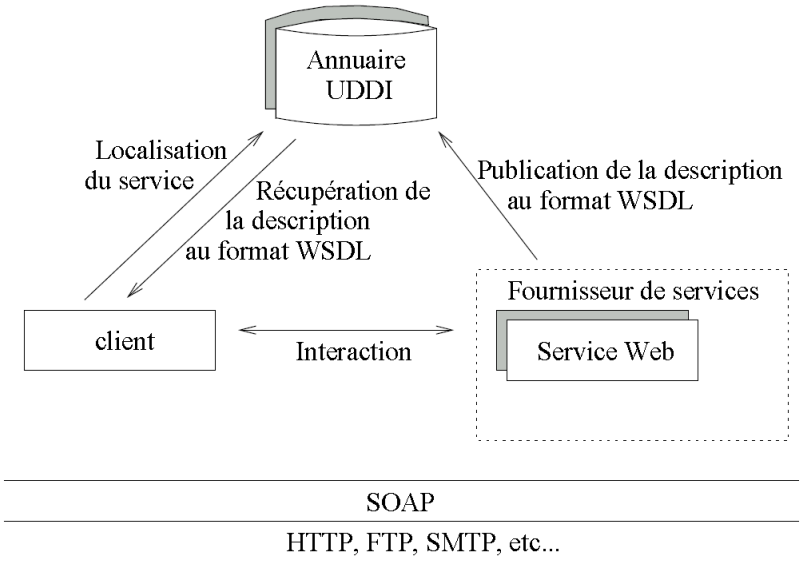


Figure 1 Modèle fonctionnel de l'architecture de publication et d'invocation d'un web service

Comme il est possible de le voir sur le modèle fonctionnel de la figure 1, le fournisseur de services est un serveur exécutant des applications ou composants assimilables à des web services. Leurs interfaces sont alors décrites en WSDL.

Pour se faire connaitre, le fournisseur de services publie la description WSDL des services qu'il fournit sur un ou plusieurs annuaires de services UDDI (Universal Description, Discovery and Integration,). Ainsi, lorsque le demandeur de services (c'est-à-dire le client) veut savoir quels sont les serveurs fournissant un service correspondant à une certaine description, il fait appel à un annuaire UDDI dont il a la connaissance. Ce dernier lui renvoie alors le ou les fichiers WSDL des services web correspondant à la demande.

Le client fait alors son choix, s'adresse au fournisseur et invoque le service web, dont il n'avait pas nécessairement connaissance auparavant. L'annuaire peut être local à une application, à un réseau d'entreprises ou à l'échelle d'Internet.

Les langages et protocoles utilisés par les services web

La description fonctionnelle précédente montre l'utilisation de nombreux langages et protocoles durant le déploiement et l'invocation du service web. Ce sont ces principaux langages et protocoles que nous allons décrire maintenant :

Le Protocole SOAP

Les communications entre les différentes entités impliquées dans le dialogue avec le service web se font par l'intermédiaire du protocole SOAP (Simple Object Access Protocol). Ce protocole est normalisé par le W3C. La description qui va suivre est basée sur la version 1.2 de ce protocole.

Le protocole SOAP est une surcouche de la couche application du modèle OSI[5] des réseaux. Le protocole applicatif le plus utilisé pour transmettre les messages SOAP est HTTP, mais il est également possible d'utiliser les protocoles SMTP ou FTP ; la norme n'impose pas de choix. Le choix de l'utilisation de protocoles applicatifs, comme par

[5] http://www.acm.org/sigs/sigcomm/standards/iso_stds/OSI_MODEL/

exemple HTTP, est lié aux problèmes d'interconnexion connus des réseaux : le but des web services étant d'être réutilisables, après publication, à travers tout le réseau Internet, il faut alors leurs donner les moyens de passer les protections telles que les firewalls. Ces derniers autorisent généralement sans aucune restriction le trafic sur les ports liés aux protocoles tels que http (port 80), permettant ainsi le passage sans problème à travers les différents réseaux des messages générés par l'utilisation du protocole SOAP.

Dans un souci d'interopérabilité, et donc dans la continuité des différents langages et protocoles issus des web services, les messages échangés lors de l'utilisation du protocole SOAP sont basés sur le métalangage XML. Ces messages comportent plusieurs parties (voir la figure ci-dessous) que nous allons décrire et sont donc encapsulés dans des protocoles de niveau applicatif.

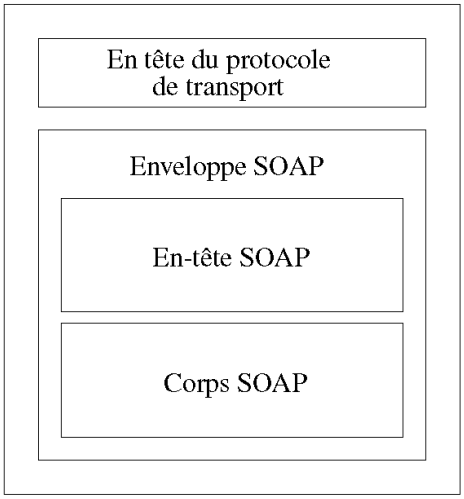


Figure 2 Schéma d'un message SOAP

L'en-tête du protocole de transport :
Cette partie dépend du protocole de transport utilisé. Par exemple, si le protocole HTTP est utilisé, cet entête contient :

– la version de HTTP utilisée,

– la date de génération de la page (qui est ici le message SOAP lui-même),

– le type d'encodage du contenu (ici, l'encodage est généralement le type text/xml), etc.

Les messages SOAP (l'enveloppe) :

La partie principale d'un message SOAP est l'enveloppe (symbolisée par la balise <Envelope>). Cette enveloppe (SOAP Envelope) est-elle-même, subdivisée en deux sous-parties : la partie en-tête et la partie corps du message. Elle permet également de spécifier des environnements de noms XML utilisés dans la suite du message.

L'entête du message SOAP :

L'entête SOAP (SOAP Header) est optionnel et extensible.

Les balises XML permettant de symboliser cette partie sont <env :Header> et </env :Header>.

Ces balises peuvent être complétées par des attributs permettant de définir le domaine de noms du service web.

En fait, l'en-tête permet principalement d'ajouter des informations sur le comportement que doivent avoir les différents nœuds intermédiaires, lors du traitement du message. Un nœud étant un intermédiaire SOAP, incluant le récepteur et l'émetteur SOAP, désignable depuis un message SOAP. Son rôle est de traiter l'en-tête (et d'effectuer les actions qui y sont décrites) et ensuite de transférer le résultat (le message SOAP modifié) à un autre intermédiaire (qui peut être le récepteur final).

Par extension, la description du comportement des différents nœuds permet également de réaliser une "composition de services", car le message peut être routé entre différents nœuds, chacun étant capable de réaliser une action précise et décrite dans le bloc d'en-tête.

Les principaux attributs des éléments formant le bloc sont :

- **env:role** : permet d'indiquer à quel nœud la fonction décrite est destinée. Et donc par extension, cela permet le routage d'un message.
- **env:mustUnderstand** : c'est une valeur booléenne, elle permet de préciser que le traitement devient obligatoire pour un nœud intermédiaire. Par exemple, pour un calcul très long, il peut être utile d'envoyer un e-mail à chaque étape.
- **env:relay** : cet attribut permet de relayer un message à un autre nœud si le premier nœud n'est pas capable de le traiter.

Le corps du message SOAP :

Les données spécifiques à l'application se trouvent dans le corps du message SOAP (SOAP Body). Tout ce qui est présent dans cette section est défini lors de la conception de l'application. Enfin, les balises symbolisant cette partie sont <env :Body> et </env :Body>. Les données doivent donc être sérialisées selon l'encodage choisi. L'utilisation du XML 1.0 à l'intérieur des blocs XML <element> permet d'envoyer absolument tous les types de documents comme par exemple des fichiers zip ou rar, des documents XML, des images au format binaire, etc... En plus des données, cette partie peut transporter un type spécial : les messages d'erreurs (SOAP Fault). Comme précédemment on peut ajouter un attribut dans la balise <env :Fault> permettant d'indiquer un type d'encodage des données, mais également d'autres attributs permettant la gestion de l'erreur, comme **code**, **reason**, **node**, **role** et **detail**.

Le Langage WSDL

Le langage de description WSDL (Web Services Description Language), a été créé dans le but de fournir une description unifiée des web services. Il se présente comme un standard dans ce domaine, normalisé par le W3C. Il est issu d'une fusion des langages de descriptions NASSL (Network Acessibility Service Specification Language

- IBM) et SCL (Service Conversation Langage - Microsoft). Son objectif principal est de séparer la description abstraite du service de son implémentation même.

Le langage de description WSDL se comporte donc comme un langage permettant de décrire l'interface visible (ou publiée) du service web. Il décrit, à l'aide du langage de balises XML, les différents éléments du service (voir Figure 3 L'architecture d'un fichier WSDL) :

- les messages (intervenant lors des échanges) et leurs types associés (pour gérer l'interopérabilité entre les différents intervenants de l'échange)
- les opérations (composées de un ou plusieurs messages)
- les liaisons et les ports de communication (permettant de lier les opérations à un protocole de transport sous-jacent)
- la description du service lui-même.

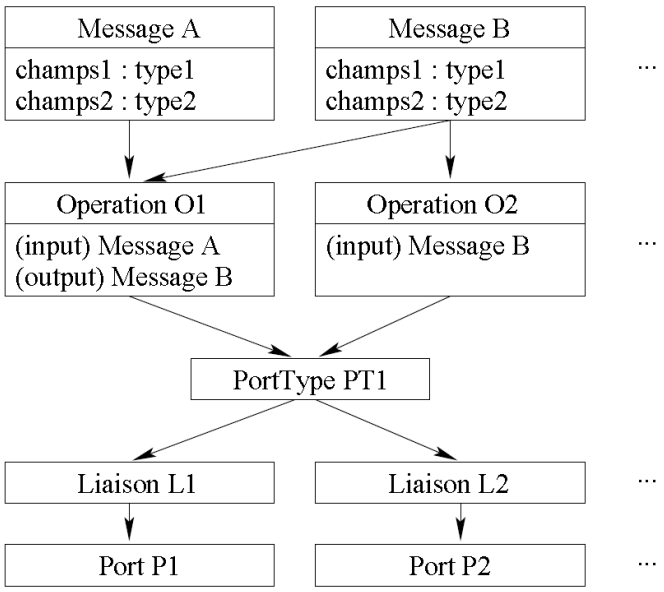


Figure 3 L'architecture d'un fichier WSDL

Les messages :

Ils sont composés de plusieurs parties. Ces parties correspondent, par exemple, dans le paradigme objet aux différents champs d'une structure. Ils sont décrits en XML et un type leur est associé. Les types de base XML peuvent être utilisés (entier, chaîne

de caractères, etc.), mais également des types complexes définis dans un fichier WSDL (fichier identique à celui de la description du service ou externe).

Les opérations :

Cette partie associe les messages aux opérations (une opération peut être vue comme une méthode). Les différents messages étant les différents paramètres de cette méthode. Un mot clé permet de distinguer le mode :

- input : mode entrée ou paramètre de donnée
- output : mode sortie ou paramètre de résultat

Plusieurs opérations sont associées pour former un PortType, utilisé par la suite.

Les liaisons :

Elles permettent d'associer les opérations (ou un ensemble d'opérations identifiés par un PortType) au protocole de transport de niveau inférieur. Ainsi, dans le cas d'un service web utilisant le protocole de communication SOAP :

- pour chaque opération, on définit le type d'échange utilisé (mode document ou mode XML-RPC).
- pour chaque message (composant les opérations), on décrit l'espace de nom associé au type de message à transporter.

Le service :

Enfin, le service lui-même est décrit dans la partie service de cette description WSDL. Le mot clé port permet d'associer des adresses Internet (URL) à chaque PortType : ces adresses seront utilisées par le client pour l'invocation du service. Il est également possible d'ajouter des informations permettant de trouver une documentation sur le service. C'est également cette partie qui peut être étendue par un autre langage que WSDL.

Le Protocole UDDI

UDDI (Universal Description, Discovery and Integration) est plus qu'un simple protocole : il fournit un protocole, une API[6] et une plateforme permettant aux utilisateurs de web services, depuis n'importe quel système, de localiser dynamiquement à travers Internet les web services qu'ils désirent utiliser. Ceci passe par le biais d'annuaires qui peuvent être maintenus dans plusieurs optiques :

- par une seule et même personne pour son usage interne ;
- par un groupe d'utilisateurs regroupant des services web répondant à certains critères ;
- par un organisme comme base de données mondiale de web services.

UDDI, dans une optique d'interopérabilité, est basé sur les standards tels que HTTP, XML, XML Schema, SOAP. Ainsi, la version 3 de ce protocole, normalisée par le consortium Oasis2, considère UDDI comme un méta-service de localisation de web services.

Les différents rôles d'UDDI :

UDDI fournit trois services de bases :

- **Publish :** ce service gère comment le fournisseur de web service s'enregistre lui-même ainsi que ses services (en utilisant UDDI).
- **Find :** ce service gère comment un client peut localiser le web service désiré (cela peut passer par des invocations de web services pour une utilisation automatique par un programme ou par une consultation d'annuaire en utilisant des mots clés).
- **Bind :** ce service gère également comment un client peut se connecter et utiliser le web service une fois celui-ci localisé.

Les entreprises qui fournissent ce service et hébergent un annuaire global UDDI sont appelées des opérateurs UDDI. Ils sont responsables de la synchronisation de

[6] Application Programming Interface

l'information des annuaires : cette synchronisation d'annuaires s'appelle la réplication.

1.3 Composition des web services et processus métier

1.3.1 Définition d'une composition (web service complexe)

Un web service est dit composé ou composite lorsque son exécution implique des interactions avec d'autres web services afin de faire appel à leurs fonctionnalités. La composition de web services spécifie les services qui ont besoin d'être invoqués, dans quel ordre et comment gérer les conditions d'exception.

La composition des web services peut se faire de deux manières: orchestration et chorégraphie.

Orchestration : [Sara07]
Un processus principal (web service) prend le contrôle du déroulement de la composition et coordonne donc les différentes opérations des différents web services. A aucun moment les autres services servant à la composition n'ont connaissance de cette composition: Ils remplissent leur rôle de service sans se soucier si un client humain ou applicatif interagit avec eux.
L'orchestration est une méthode très utilisée, et pour plusieurs raisons :

1. il est relativement simple d'écrire un processus centralisé gérant l'invocation de sous-services ;
2. dans le cas d'une réutilisation de web services basiques, seule la partie centrale est à développer.

Chorégraphie : [DaFH04]
La chorégraphie ne repose pas sur un web service principal. Chacun des services intervenant dans la composition sait exactement ce qu'il doit faire, quand il doit le faire et avec qui il doit le faire. Donc ils ont tous une connaissance plus ou moins globale du processus métier dans lequel ils se retrouvent. Contrairement à la méthode basée sur l'orchestration, la chorégraphie demande nettement plus de développement et de test : il faut développer chaque service dans le but de la composition qu'ils formeront. Cela dit, en utilisant des méthodes de développement et des stratégies bien pensées, l'intérêt de la méthode apparait: la non-centralisation du traitement.

L'approche la plus simple dans le passage à une architecture orientée service (SOA) est l'orchestration. En effet, le but principal de la composition est la réutilisation de services (basiques ou non) sans modifier ceux-ci (car ils peuvent très bien être hébergés par une autre compagnie et nous n'avons alors aucun moyen de contrôle sur eux). Le processus principal doit alors pouvoir s'adapter aux différentes erreurs possibles (non disponibilité d'un service, annulation, etc.).

L'architecture orientée service permet également d'utiliser l'approche chorégraphie : la preuve en est de l'existence de nombreux standards comme WSCI (Web Services Choreography Interface) [AAFJ02], ou encore WS-CDL (Web Services Choreography Description Language) [KBRF04].

1.3.2 Les différents langages de web services complexes

De nombreux langages ont été développés pour répondre à ces besoins. Parmi ceux-ci, on peut citer :

- WSFL [Wesl02] : le langage Web Services Flow Language développé par IBM dans le but de rendre possible la description de la composition d'un ensemble de web services en se basant sur la composition des flux de manière hiérarchique.
- XLANG [That01] : le langage XLANG a été développé par Microsoft en 2001, pour les besoins de sa plateforme de gestion de processus BizTalk. Ce langage permet de représenter les éléments clés, d'un point de vue algorithmique, des processus métier et se classe ainsi dans les langages de description comportementale.
- BPEL [TFHJ03] : le langage BPEL est développé en 2003, par un regroupement de constructeurs informatiques, parmi ceux-ci, on retrouve IBM, BEA et Microsoft. Ce langage est une fusion des langages, dit de première génération : XLANG et WSFL. Il reprend donc les avantages de ces deux langages en tirant parti de l'apprentissage lié à leur mise en place. Il se focalise sur la représentation du processus métier en se rapprochant des structures algorithmiques. BPEL est un langage de plus en plus utilisé par les industriels, devenant ainsi un standard de fait, en plus d'être en phase de devenir un standard OASIS. BPEL est aussi connu dans la littérature spécialisée par BPEL4WS.

1.4 Les propriétés Non-Fonctionnelles

On désigne par PNFs les concepts comme la qualité de service (QoS), la sécurité, la confiance (trust), et la privacité (privacy). Cette dernière est le sujet principal de ce travail.

Contrairement aux PFs, les PNFs ne sont pas étudiés avec le même intérêt (qualitativement ou quantitativement) par la communauté des chercheurs web services. Ceci est dû principalement aux causes suivantes :

- Les PNFs sont souvent abstraites et sont présentés d'une manière informelle ;
- Les PNFs sont souvent considérées après la description des PFs ;
- Une grande difficulté dans la modélisation de PNFs ;

1.4.1 Qualité de Service des Web Services (QoS)

Cette caractéristique est souvent utilisée dans les systèmes d'exploitation et dans la gestion de réseaux informatique pour exprimer des exigences que doivent remplir les systèmes pour fournir une prestation de service de qualité. (Exemple : une bande passante minimum 2 Mb est nécessaire pour faire de la TV sur IP en simple définition (SD), et il faut 6 Mb de bande passante pour faire de la TV IP en haute définition (HD)). La qualité d'un service est une notion subjective. Selon le type d'un service envisagé, la qualité pourra résider dans le débit (téléchargement ou diffusion vidéo), le délai (pour les applications interactives ou la téléphonie), la disponibilité (accès à un service partagé) ou encore le taux de pertes de paquets (pertes sans influence pour de la voix ou de la vidéo, mais critiques pour le téléchargement).

Pour les services web, QoS a une signification plus étendue pour englober toutes propriétés non-fonctionnelles qui peut affecter une utilisation satisfaisante. Parmi les paramètres à surveiller on peut citer : la disponibilité, l'accessibilité, l'intégrité, la performance, la fiabilité, la Normalisation et la Sécurité.

1.4.2 La Sécurité dans les services Web

Bien que la technologie des web services soit relativement récente, les principes de base, qui gouvernent sa sécurité ne le sont pas. Les caractéristiques fondamentales qui gouvernent une architecture classique de sécurité, sont aussi valables pour le paradigme SOA. Ces principes sont :

- **Identification :** Le récepteur d'un message à besoin d'identifier l'envoyeur – (la question posée est : "qui êtes –vous ? ").
- **Authentification :** Le récepteur d'un message à besoin de vérifier que l'identité de l'envoyeur est valide (la question posée est : "comment être sûr que vous êtes ce que vous prétendez être ? ").

- **Intégrité :** Le message doit être inchangé depuis la transmission jusqu'à sa réception (la question posée est : "Est-ce que ce message à changer depuis que vous l'avez envoyé ? ").
- **Autorisation :** Le récepteur du message doit déterminer le degré du pouvoir accordé à l'envoyeur. (La question posée est : " Quelles sont les actions qu'on vous autorise à accomplir")
- **Confidentialité :** Le contenu du message, ne doit pas être vu, pendant la transmission, excepté par des entités autorisées. (La question posée est : "Est-ce que des entités non autorisées ont pu accéder au contenu du message ? ")

Pour comprendre comment SOA traite ces obligations de sécurité, il faut se pencher sur les spécifications de sécurité, qui sont tous le temps en évolutions que ce soit pour XML ou pour les web services. Le tableau ci-dessous montre les principaux standards de sécurité en relation avec les principes cités plus haut.

Tableau 1

Identification Authentification Autorisation	- WS-Security Framework - Extensible Access Control Markup Language (XACML) - Extensible Rights Markup Language (XrML) - XML Key Management (XKMS) - Security Assertion Markup Language (SAML) - .NET Passport
Confidentialité	- WS-Security Framework - XML-Encryption - Transport Layer Security (TLS)
Intégrité	- WS-Security Framework - XML-Digital Signatures

XML Key Management (XKMS) [05]

Le but de cette spécification est d'obtenir et de gérer des clés publiques (PK). XKMS est compatible avec les infrastructures clés publiques existantes (PKI), mais il peut aussi fonctionner d'une manière autonome par rapport à eux. XKMS est composé de deux spécifications complémentaires "XML Key Registration Service" et "XML Key Information Service". Ensembles, elles permettent d'intégrer plusieurs technologies de sécurités comme la signature digitale, les certificats, et la vérification de l'état de révocation. Par exemple, XKMS utilise une signature digitale pour protéger l'intégrité du contenu d'un document XML.

Extensible Access Control Markup Language (XACML) [PaLL11] **et Extensible Rights Markup Language (XrML)** [HHKM02]

La spécification XACML consiste en deux langages qui sont liés l'un à l'autre: un qui traite le contrôle d'accès et l'autre qui traite les échanges de type Requête-Repense. A travers ces deux langages on peut créer des politiques de sécurité de type "petits grains" (fine-grained). La spécification XACML ne doit pas être confondue avec la spécification WS-POLICY qui fait partie du WS-Security Framework et qui est aussi utilisée pour définir des politiques de sécurité. La spécification XrML est utilisée surtout dans les environnements où les documents à transférer utilisent des formats digitaux variés.

Security Assertion Markup Language (SAML) [CKPM04] **ET .NET Passport (Microsoft)**[7]

Le but de ces deux spécifications est de permettre de mettre en place des systèmes à signature unique. En effet dans certains systèmes composés de plusieurs applications, chacune avec ces propres contrôles d'accès et authentifications, le système devient vite ingérable. Ces deux spécifications permettent à l'utilisateur de se présenter au système global avec un "login-password" unique, partagé avec toutes

[7] Microsoft a changé cet appellation en 2010 pour ***Windows Live ID***

les applications du système. Les 2 spécifications concurrentes (.NET Passport est un système propriétaire de Microsoft, alors que SAML est ouvert) fournissent les mécanismes d'authentification et d'autorisation. Les deux formats de messages (requête et repense) sont définis pour faciliter la transmission des références sécuritaires des utilisateurs dans le cadre d'une activité web-service. .NET Passport utilise un modèle centralisé des références sécuritaires alors que SAML utilise un modèle décentralisé. L'interopérabilité entre SAML et .NET Passport existe et elle s'améliore avec les versions successives.

XML-Encryption et XML-Digital Signatures [ImDS02]

Le but de ces deux spécifications est de protéger le contenu d'un document XML. La spécification "XML-Encryption" fournit un modèle standard pour crypter les données textuelles et binaires, mais aussi un moyen pour communiquer au récepteur les informations nécessaires pour décrypter le contenu du message reçu. La spécification "XML-Digital Signatures" établie un format standardisé pour représenter les signatures digitales. Ces dernières permettent de donner de la crédibilité à un message en assurant le récepteur que le message à bien été envoyé par un service partenaire agrée. Elles permettent aussi de s'assurer de la non-altération du message en cours de route. Comme XML-Encryption, XML-Digital Signatures supporte les données textuelles et binaires.

Secure Sockets Layer[8] (SSL) [DiRe08]

SSL est une technologie qui n'est pas propre aux Web service, mais elle est utilisée principalement par les browsers et les sites web pour établir un canal sécurisé afin de transmettre des données sensibles entre les 2 entités. Le problème avec SSL, c'est qu'il correspond à un protocole de sécurité de niveau « transport » selon le modèle OSI (protection point à point). Le message est protégé par cryptage pendant le

[8] Nouvelle appellation TLC (Transport Layer Security)

transport d'un point à l'autres mais pas pendant le traitement par ces points. Ceci ne signifie pas que SSL est inutile dans le cadre d'une architecture SOA, mais qu'elle n'est pas suffisante et doit être complétée par d'autres spécifications pour assurer une sécurité de bout en bout.

1.5 Les Protocoles de sécurité des Web Services

1.5.1 WS-Security (WSS) [NKMH06]

WS-Security est une spécification de sécurité pour web services définie par le consortium OASIS-Open. En avril 2004, la version 1.0 de la spécification a été publiée, suivie par une version 1.1 en fév. 2007. Le Framwork WS-Security se présente sous la forme d'un important document qui établit les standards conceptuels et fondamentaux de la sécurité, en plus de plusieurs autres spécifications qui ensembles, forment un cadre (framwork) de sécurité taillé pour les Web Service. WS-Security (appelé aussi langage de sécurité Web Service) est utilisé pour combler les failles qui existent entre les différents modèles de sécurité existants, et aussi pour assurer une sécurité de bout en bout des messages SOAP et non pas uniquement de « point à point », comme c'est le cas des modèle de niveau « transport ». En effet dans un système orienté services, le message passe souvent par différents intermédiaires avant d'arriver à sa destination finale. Les modèles de sécurité de niveau « transport » de la pile OSI (ex : SSL), ne protègent le message que de « point à point ». La protection du message n'est pas assurée à l'intérieur des nœuds intermédiaires (voir figure suivante).

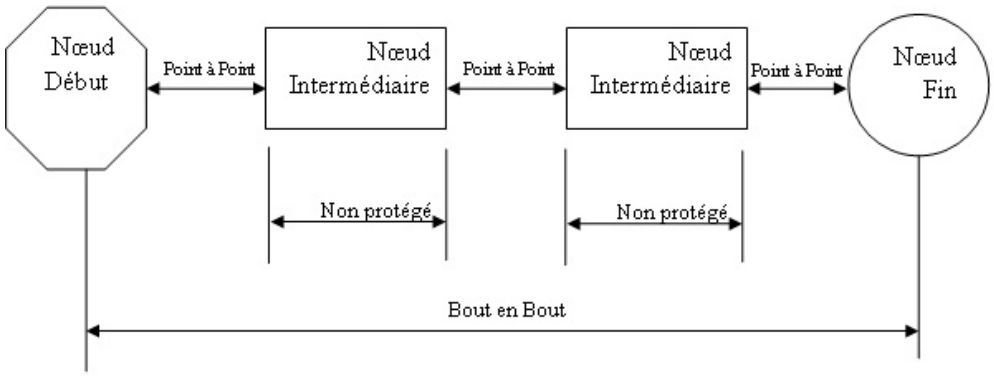


Figure 4 - Type de communications

Pour assurer une sécurité de bout en bout, WS-Security implémente des mesures de sécurité grâce aux blocs "header" de SOAP, et qui sont transmis avec le message. Le Framework WS-Security est encore épaulé par d'autres spécifications (appelés spécifications associées), qui ensembles, forment la pile sécuritaire.

Tableau 2

WS-SecureConversation	WS-Federation	WS-Authorization
WS-Policy	WS-Trust	WS-Privacy
WS-Security Framework		

La couche en dessus du framework est souvent appelée « la couche politique » (policy layer) puisque elle fournit les briques de base pour construire la confiance (trust) et le couple « politique – préférences » de la privacité. La couche suivante est appelée « la couche de fédération » (Federation Layer). Elle se base sur ces politiques pour unifier des domaines de confiance disparates.

WS-Policy [VOHH07]

WS-Policy permet aux web services (consommateurs et fournisseurs) d'exprimer (à l'aide de XML) leurs politiques et leurs exigences (sur la sécurité, la privacité, la qualité de services, les règles de métiers,…).

WS-Trust [NGGB09a]

Cette Spécification établie un modèle standard de confiance, dans le but d'unifier les autres modèles de confiance, en permettant de vérifier la validité des jetons de sécurité (Security Tokens) échangées. WS-Trust fournie un processus de communication pour demander l'implication des autorités de confiance tierces dans la vérification.

WS-Privacy

Cette spécification travail en collaboration avec WS-Policy et WS-Trust. Elle permet à un fournisseur de service d'exprimer sa politique en matière de privacité et de réclamer que le consommateur adhère explicitement à ces politiques.

WS-SecureConversation [NGGB09b]

Le but de cette spécification est de créer une session qui englobe plusieurs messages SOAP pour ne pas évaluer l'autorisation et l'authentification de chaque message individuel, afin d'éviter les problèmes de performances.

WS-Federation [SSCI03]

Parce que plusieurs parties (consommateurs et fournisseurs) peuvent participer dans une session web service, en utilisant des technologies sécuritaires différentes (ex : un service utilise des certificats KERBEROS, alors qu'un autre utilise X.509), on a besoin de traduire des données sécuritaires entre les parties impliqués. WS-Federation est une spécification qui permet de décrire comment la négociation doit se faire entre les parties impliquées. Cette spécification opère sur la couche au-dessus de WS-Policy et WS-Trust.

WS-Authorization

Cette spécification se charge d'exprimer les règles d'accès et leur gestion. Concrètement elle permet de définir comment des revendications peuvent être représentés en jetons, et comment ces jetons doivent être interprétés par les nœuds. Cette spécification possède beaucoup en commun avec le standard XACML (Extensible Access Control Markup Language) cité plus haut.

1.6 Les principes fondamentaux des processus métiers

Les processus métiers sont fondés sur deux éléments corrélés : les processus et les tâches. Un processus métier est organisé en séquences logiques et chronologiques de tâches transformant des éléments d'entrés en produits finis ou semi-finis à la sortie. Globalement, les règles et contraintes métiers gouvernent la mise en œuvre et l'exécution des processus métiers. Les ressources humaines affectées à l'exécution des tâches du processus métier laissent apparaître une répartition formée de rôles divers. La figure ci-dessous illustre le regroupement de ces principes en système.

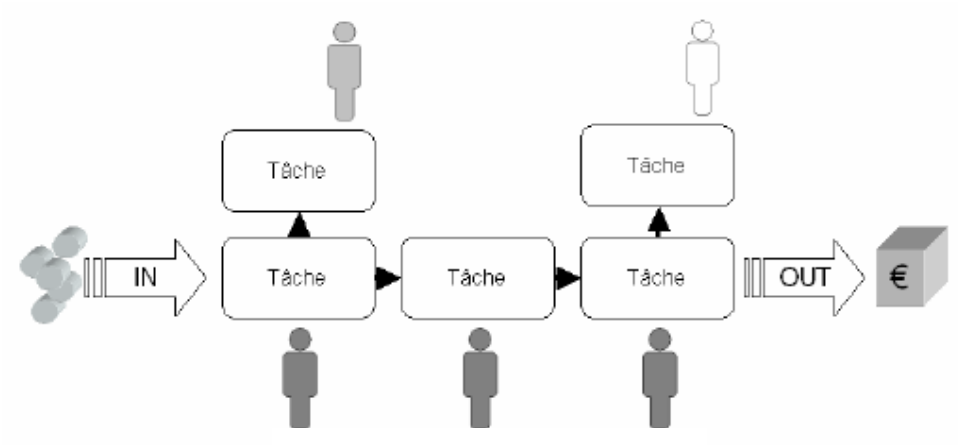


Figure 5 - Processus Métier

L'efficacité recherchée est obtenue en ordonnançant de manière optimale les tâches et la collaboration entre les intervenants.

1.6.1 Les processus métiers

Un processus métier se compose d'une séquence logique et chronologique d'une ou plusieurs tâches produisant conjointement un résultat mesurable à valeur ajoutée. Une tâche est la partie élémentaire d'un processus et correspond plus généralement à un traitement ayant pour finalité la transformation de matières premières en produits semi-finis ou finis.

1.6.2 Les tâches des processus métiers

Un processus métier est une exécution ordonnée de tâches. Chaque tâche produit un résultat en transformant ses entrées en sorties. Elle se subdivise en plusieurs phases

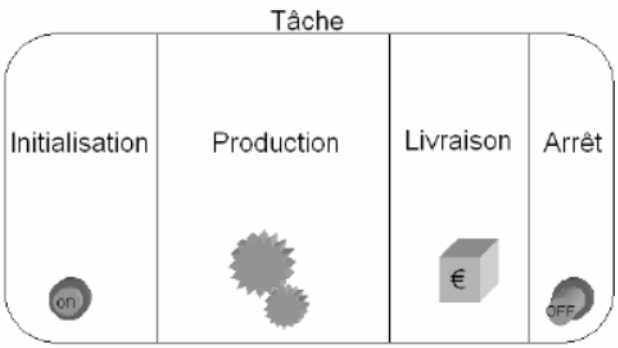


Figure 6 - Les différentes phases d'une tâche

comme l'illustre la Figure 6:

- **L'initialisation** ou préparation de la tâche. Avant de produire, une tâche nécessite un démarrage ou une préparation qui consomme des ressources pendant un certain délai. Les actions menées sont uniquement destinées à initialiser le dispositif de production comme la mise sous tension de l'ordinateur ou encore la préparation de la matière première à traiter.
- **La production** reprend les actions de transformation des intrants en produits à valeur ajoutée.
- **La livraison** des biens et services produits à la tâche suivante.
- **La finalisation** de la tâche suite à l'arrêt du dispositif nécessitant éventuellement une consommation supplémentaire de ressources et de moyens dans un délai déterminé.

Chaque phase est dépendante de la tâche et ne peut y être dissociée. L'enchaînement des tâches décrit à chaque fois une relation client-fournisseur entre les tâches. Ce qui est produit de l'une est consommé de l'autre en produisant éventuellement des déchets.

Soit un opérateur exécute manuellement une tâche, soit celle-ci est confiée à un dispositif autonome d'automatisation. Il existe cependant une solution intermédiaire faisant intervenir l'opérateur sur le dispositif afin de compléter ou contrôler le résultat obtenu. Le traitement d'une tâche est éventuellement régulé ou soumis à des conditions désignées comme **«règles métiers internes».** Ces règles influencent directement la production du bien ou du service en sortie de la tâche.

Remarquons que les concepts introduits plus haut ne supposent aucune sorte d'automatisations (informatiques ou autres). En effet un processus métier peut (et c'est le cas généralement) être implémenté manuellement dans sa globalité ou automatisé partiellement. L'informatisation des processus peut être appliquée « avant », « pendant » et « après ».

« L'avant » consiste dans la modélisation du processus pendant sa conception ou, dans le cas où il existe déjà, son « reverse-engineering ».
« Pendant » consiste dans l'implémentation des tâches (toutes les tâches ou uniquement certaines d'entre elles), par des logiciels.
« Après » consiste dans l'analyse des différents rapports (automatiques ou manuels) générés pendant l'exécution du processus métiers, afin de l'optimiser.

Les systèmes informatiques qui permettent de gérer les 3 aspects (« avant », « pendant », « après ») sont appelés les **BPMS** (Business Process Management suite). Un BPMS est constitué au minimum d'un éditeur de modélisation (ou modeleur) et d'un moteur d'exécution automatisé des processus métier. Le modeleur permet de construire le processus tâche par tâche et les relations (logique métier) qui existent entre elles en utilisant une notation standard BPMN (Business Process Management Notation). La figure suivante représente un processus métier représenté en BPMN dans le BPMS Intalio.

.

Processus Métier (Business Process) et Web Service

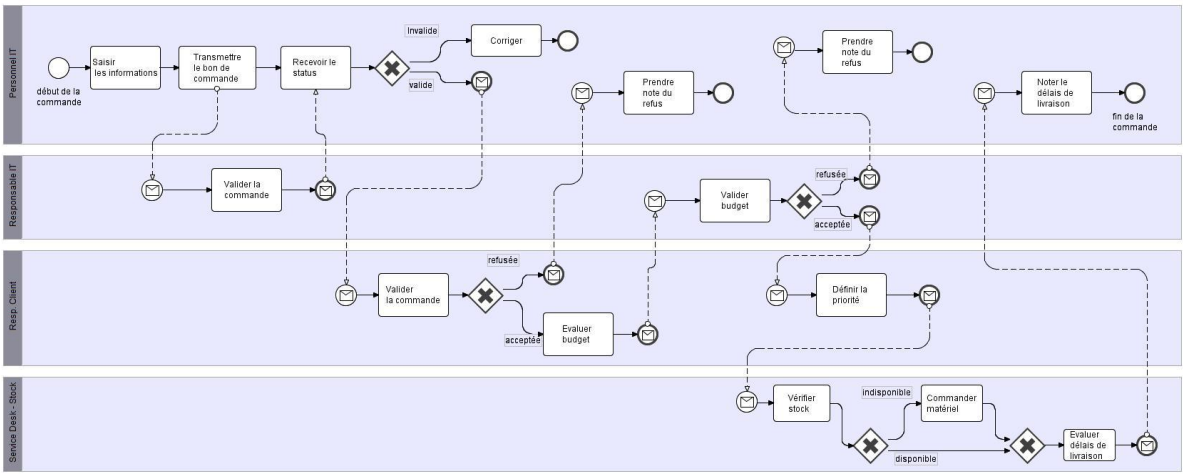


Figure 7 - Notation BPMN d'un processus métier

Il faut insister sur le fait que les 2 concepts **« Processus Métier »** et **« Web services »** ne sont pas corrélés. En effet on peut implémenter l'un sans recourir à l'autre. Un processus métier n'est pas forcément une composition de web services mais un ensemble de tâches humaines et/ou automatiques à effectuer. La confusion de leurs association, est due au fait, qu'il existe des moteurs de processus métiers qui implémentent ces derniers comme une composition de web services (langage BPEL / Soft Apache ODE ou OW2 Orchestra).

La pluparts des vendeurs de systèmes BPMS proposent d'implémenter les tâches des processus métiers par des web services à cause du faible couplage de ces derniers et de leur indépendance envers les plateformes matérielles et logicielles et donc d'obtenir des processus métiers qui fonctionnent à l'intérieur d'une même entreprise ou à travers plusieurs entreprises, mais ils utilisent d'autres méthodes d'implémentation de ces tâches (tableur, traitement de textes, email, lien directe vers RDBMS, Scripts, code JAVA, code C++ …etc.).

Les BPMS permettent aussi de réaliser des compositions de web services par orchestration (exemple: Bonita), la logique de la composition étant assurée par le moteur d'exécution des processus métier du BPMS.

L'association des BPM et des Web services est bénéfique aux 2 concepts. Les BPM bénéficient des avantages des web services suivants:

- L'indépendance vis-à-vis des plateformes matérielles et logicielles.
- Le faible couplage entre les clients et les serveurs.
- La standardisation des protocoles.
- La gestion implicite de la distributivité des systèmes.

Les Web services bénéficient des avantages des BPM suivants:

- Des fondements formels solides des BPM (l'algèbre des processus – connu aussi par : pi-calculus)
- Des Moteurs d'exécution des BPMS pour réaliser des compositions (le plus souvent des orchestrations).

1.7 Conclusion

Dans ce chapitre nous avons passé en revue les spécifications de base des web services (propriétés fonctionnelles et non-fonctionnelles). Nous avons aussi vu comment ces spécifications s'imbriquent les unes aux autres pour essayer de résoudre les problèmes rencontrés. Nous avons aussi abordé la composition des web service – un sujet très actuel – et nous avons introduit la notion de processus métier et sa relation avec les web services.

Un aspect important concernant l'interopérabilité des web services (et des architectures SOA en général) est que les services ont un couplage faible, c'est-à-dire que les web services ne sont pas développés pour interagir avec des clients spécifiés

mais ils sont aussi conçus pour satisfaire les besoins de différents clients, probablement développés par des équipes différentes ou même par des sociétés différentes. En conséquence, les développeurs d'application clientes doivent être informés de tous les aspects fonctionnels et non-fonctionnels d'un service, pour savoir s'ils peuvent interagir correctement avec ce dernier. Pour cette raison, la description des services doit être plus riche que la description d'interface conventionnelle des middlewares. Pour être précis, il est maintenant établi que la description d'un service doit non seulement inclure l'interface du service mais aussi inclure le protocole métier (Business Protocol) supporté par ce service, c'est-à-dire la spécification des séquences de messages possibles échangés (conversations) ainsi que d'autres abstractions utiles (les transactions, les politiques des vendeurs etc.).

Bien que les web services fournissent des abstractions pour simplifier l'intégration dans les niveaux bas de la pile d'interaction (la syntaxe des données et les protocoles de communication), où plusieurs questions ont été identifiées (et pour certaines même résolues), ils n'ont pas contribué (jusqu'à maintenant) à simplifier l'intégration à des niveaux d'abstraction élevés (types de message/donnée et les protocoles d'interaction métier).

Chapitre 2

La Privacité

2 LA PRIVACITE

2.1 Introduction

La vie privée (appelée aussi privacité) est un sujet très vaste qui englobe les aspects juridiques, éthiques, politiques et d'autres aspects encore. Dans ce chapitre, nous nous contenterons seulement de l'aborder d'un point de vue informatique, même si cet aspect n'est pas complètement indépendant des autres. En effet dès le début de l'informatique, la crainte de l'utilisation abusive des données personnelles des citoyens, collectées par des moyens informatiques à des fins qui serviraient à les contrôler ou à les intimider, a conduit à la création d'organismes de contrôle qui ont pour but de s'assurer que les pouvoirs politiques ou autres n'abusent pas des moyens informatiques énormes mis à leur disposition. Exemple La CNIL[9] (Commission Nationale d'Informatique et Liberté) en France ou la CPVP[10] (Commission de la Protection de la Vie Privée) en Belgique.

Le but de ces organismes et de s'assurer que le citoyen:

- Peut maitriser qui détient des informations sur lui.
- Peut maitriser quel est le type d'informations personnelles détenues.
- Peut maitriser l'usage qui est fait de cette information.

Le développement des systèmes distribués et des réseaux larges comme Internet n'a fait qu'accentuer le problème de la privacité. Cette dernière est maintenant sous la menace potentielle internationale et non pas uniquement nationale. En particulier, Internet a mis en avant la nécessité de protéger les 3 aspects suivants de la privacité:

[9] http://www.cnil.fr/

[10] http://privacy.fgov.be/fr/

- **Privacité personale contre l'offense morale:** Ne pas être exposés à des informations qui choquent notre sens moral.
- **Privacité Territoriales contre l'envahissement:** Ne pas envahir notre propriété (contenu local de notre ordinateur).
- **Privacité Informationnelle contre les rumeurs:** Décider quelles sont les données qui peuvent être liés à nous en tant qu'individu.

Il existe plusieurs définitions de la privacité, celle qui nous convient le plus ici c'est celle définie par Alan Westin en 1967 [West69], et qui a mis l'accent sur l'autodétermination:

"La privacité, c'est la revendication des individus, des groupes ou des institutions du droit de déterminer par eux-mêmes comment, quand, et à quel degré, des informations sur eux sont communiquées à d'autre."

A partir de cette définition on peut définir des technologies qui ont pour but de préserver la privacité. Le commuté international CCG (Common Criteria group) a défini 4 types de technologies qui améliorent la privacité (privacy-enhancing technologies: PETs) [DeCa06]:

1. **Anonymisation:** C'est l'impossibilité des autres systèmes ou utilisateurs de déterminer l'identité d'un utilisateur lié à une opération ou sujet.
2. **Pseudonymisation:** C'est l'impossibilité des autres systèmes ou utilisateurs de déterminer l'identité d'un utilisateur lié à une opération ou sujet, Mais avec la possibilité que cet utilisateur soit toujours responsable de ces actions.
3. **Non-Asociabilité (Unlinkability):** C'est l'impossibilité des autres systèmes ou utilisateurs de déterminer si un même utilisateur a effectué une opération spécifique dans le système.
4. **Non-Observabilité:** C'est l'impossibilité des autres systèmes ou utilisateurs de déterminer si une opération spécifique est en cours d'exécution.

2.2 Privacité Et Sécurité

La tentation est toujours grande de confondre privacité et sécurité. Dans la majorité des cas, c'est ce qui arrive souvent. Pourtant la privacité et la sécurité sont complètement différentes, même si elles sont intimement liées, elles doivent travailler en concert pour garantir un système fiable et sécurisé.

La privacité est concernée par:

- La protection des données d'identification de l'utilisateur.
- Comment les informations d'identification de l'utilisateur sont collectées et stockées.
- Les standards et les politiques.
- La pratique honnête, c'est-à-dire:
    - Qui peut utiliser l'information
    - Quel est le but pour lequel l'information a été collectée.
    - Où l'information est stockée et pour quelle durée.
    - Qui a accès à cette information.
    - Quelles sont les mesures mise en œuvre pour la sécuriser.
    - Comment elle est transmise.
    - Quelles sont les droits d'accès consentis aux autres utilisateurs pour elle.
    - Qui est tenu responsable en cas de litige.
    - Est-ce que on demande le consentement des utilisateurs avant d'utiliser leurs informations.

Par contre la sécurité est concernée par:

- Réduire le risque de l'exposition des systèmes.
- Comprendre les questions métiers est leurs risques.

- Minimiser les facteurs humains et leurs risques.
- Protéger les données et les systèmes.
- La Confidentialité: Uniquement les utilisateurs autorisés peuvent lire et utiliser les données.
- L'intégrité: La certitude que les données n'ont pas été altérées par des entités non autorisées ou par accident.
- La disponibilité: Les données et les systèmes peuvent être utilisés pour le but pour lequel ils ont été conçus.

Ensemble la sécurité et la privacité augmentent:

- La confiance de l'utilisateur.
- La fiabilité des données.
- La compétitivité des organisations.
- La viabilité des systèmes en ligne.

Ensemble la sécurité et la privacité réduisent :

- La probabilité du vol (physique ou électronique).
- L'utilisation abusive des données.
- Les fuites.

En Conclusion: Privacité + Sécurité = Liberté du choix.

2.3 La Plateforme P3p (Platform for Privacy preferences) [LGMM05]

La plateforme P3P est une combinaison de protocoles et d'architectures conçues pour informer les utilisateurs Web des usages en matière de collecte de données des sites Web qu'ils visitent. Le but est de donner aux utilisateurs un meilleur contrôle sur leurs informations personnelles. P3P a été développée par le WC3 et elle été adoptée officiellement le 16 Avril 2002.

Pour établir leur propre politique de privacité, les utilisateurs doivent remplir un questionnaire standardisé à choix multiples dont les réponses sont compilées dans un document XML (les préférences de sa vie privé). Les navigateurs peuvent ainsi facilement décrypter cette déclaration sur la vie privée. Cela permet de bloquer automatiquement les actions pour lesquelles les internautes ont préalablement établi leur désaccord. Cette pratique a pour but d'établir une relation de confiance entre les usagers et Internet. P3P est déjà intégré aux navigateurs les plus récents (Internet Explorer, FireFox et Opera).

Néanmoins, le P3P présente certaines faiblesses. Les sites agréés par cette plateforme ne doivent répondre à aucune obligation officielle de respecter leur engagement. Cela ouvre la porte à tous les intervenants ayant de mauvaises intentions... Éventuellement, des sanctions seront prévues pour limiter les abus ainsi qu'une certification pour endosser les engagements de confidentialité établis. De plus, les usagers doivent fréquemment accepter les cookies lors de leur navigation pour avoir accès aux fonctionnalités de base d'un site. Cette réalité démontre le grand pouvoir des techniques de marketing dans cet univers en constante évolution. L'efficacité du P3P est donc régulièrement menacée par de nouvelles contraintes.

2.3.1 Objectifs du protocole P3P

Les deux objectifs principaux de la plateforme P3P sont les suivants:

- Permettre aux sites Web de présenter leurs usages en matière de collecte de données d'une manière standardisée, lisible par l'ordinateur et facile à localiser.
- Permettre aux utilisateurs du Web de savoir quelles données seront collectées par quels sites, comment ces données seront utilisées et quelles données et utilisations ils peuvent autoriser ou refuser.

2.3.2 La spécification P3P

La spécification P3P définit:

- Un fichier Référence écrit en XML qui fait correspondre à un ensemble de pages web leur politique de privacité.
- Un ou plusieurs fichiers politiques écrit en XML qui définissent la politique de privacité.
- Un fichier de préférence codé en langage APPEL (lui-même basé sur XML) pour exprimer les préférences de privacité de l'utilisateur.

Le Fichier Référence

Le protocole P3P est une simple extension du protocole http. La figure ci-dessous montre le déroulement type d'une cession p3p entre un client p3p ('user agent' qui est concrètement le browser) et le service p3p (concrètement le site web).

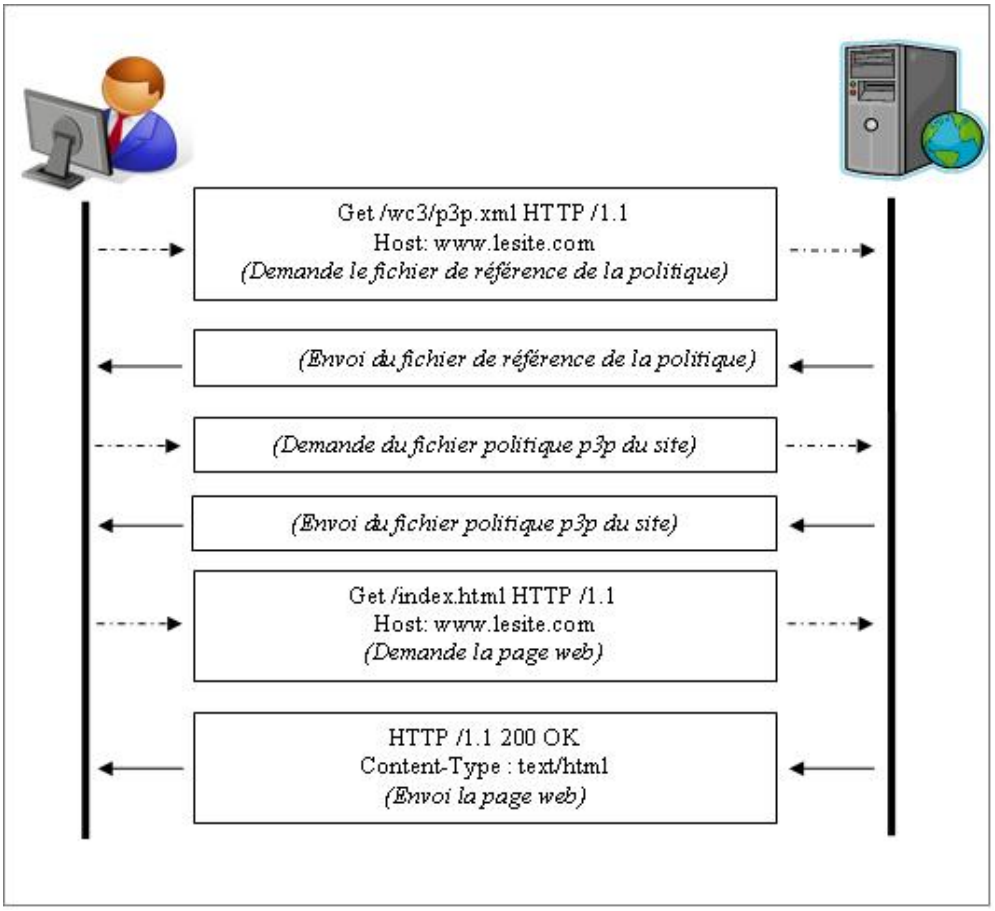


Figure 8 - P3P Principe de fonctionnement

Le fichier de référence de la politique indique la localisation du fichier de politique, qui s'applique sur chaque partie du site. On peut avoir un fichier de politique globale pour tout le site, ou plusieurs fichiers politique, chacun couvrant une partie donnée du site. Pour le trouver, le « user agent » le cherche dans une "well-known location", c'est-à-dire dans /w3c/p3p.xml à partir de la racine du site. Remarquons aussi que la localisation de ce fichier référence peut ce faire aussi par un autre mécanisme, celui d'inclure son adresse dans un lien inséré dans un header html.

Les Politiques de Privacité

Les politiques P3P sont des documents XML qui définissent les pratiques de privacité d'un site web. Les informations de ce fichier sont plus faciles à déchiffrer par la machine que pour un être humain. La figure ci-dessous montre un exemple d'un fichier politique.

```
<POLICIES xmlns="http://www.w3.org/2002/01/P3Pv1">
<POLICY discuri="http://p3pbook.com/privacy.html" name="policy">
  <ENTITY>
    <DATA-GROUP>
      <DATA
       ref="#business.contact-info.online.email">privacy@p3pbook.com
      </DATA>
      <DATA
       ref="#business.contact-info.online.uri">http://p3pbook.com/
      </DATA>
      <DATA ref="#business.name">Web Privacy With P3P</DATA>
    </DATA-GROUP>
  </ENTITY>
  <ACCESS><nonident/></ACCESS>
  <STATEMENT>
    <CONSEQUENCE>Our Web server collects access logs containing
      this information.</CONSEQUENCE>
    <PURPOSE><admin/><current/><develop/></PURPOSE>
    <RECIPIENT><ours/></RECIPIENT>
    <RETENTION><indefinitely/></RETENTION>
    <DATA-GROUP>
      <DATA ref="#dynamic.clickstream"/>
      <DATA ref="#dynamic.http"/>
    </DATA-GROUP>
  </STATEMENT>
</POLICY>
</POLICIES>
```

Figure 9 - P3P policy

A partir du fichier exemple, on peut présenter quelques éléments et attributs utilisés dans un document XML d'une politique P3P.

POLICY

L'élément POLICY décrit une politique. En particulier l'attribut "discuri" de l'élément POLICY indique la localisation d'un fichier qui présente la politique dans un format lisible par un humain.

TEST

L'élément TEST indique qu'une politique est pour des besoins de test uniquement.

ENTITY

L'élément ENTITY identifie les entités légales qui imposent la politique de privacité du site. Il indique aussi leurs références (électroniques ou physiques).

ACCESS

L'élément ACCESS fournit des informations sur la capacité des individus à consulter les données, qu'un site a récolté sur eux. Ils existent 6 niveaux qui vont de "pas d'accès" à "un accès total".

DISPUTE

L'élément DISPUTE décrit les procédures de résolution des conflits qui peuvent se produire à propos d'une politique de privacité d'un site. Exemple: contacter le représentant de l'utilisateur, une organisation indépendante ou déposer une plainte légale.

REMEDIES

L'élément REMEDIES désigne les traitements à appliquer dans le cas où une brèche à une politique de privacité a été constatée.

STATEMENT

L'élément STATEMENT est utilisé pour indiquer un ensemble de données pour lesquelles une même pratique de privacité s'applique. Cet élément contient d'autres éléments qui représentent le type de données et les pratiques de privacité. En général une politique contient un ou plusieurs élément STATEMENT chacun pour un groupe de données. Exemple un STATEMENT pour les données enregistrés dans le log, et un autre pour les données saisies par l'utilisateur dans un formulaire html.

NONIDENTIFIABLE

La présence de l'élément NONIDENTIFIABLE dans une politique indique que le site ne collecte pas de données d'identification ou que les données collectées sont rendues anonymes.

CONSEQUENCE

L'élément CONSEQUENCE permet de fournir une description succincte, humainement lisible, de la politique d'un site.

PURPOSE

L'élément PURPOSE décrit comment les données collectées vont être utilisées. Les sites ont le choix entre 12 valeurs possibles pour cet élément. Il existe aussi un attribut optionnel "required", pour indiquer qu'une utilisation particulière d'une donnée et de type " à prendre ou à laisser".

RECIPIENT

L'élément RECIPIENT décrit l'étendue du partage des informations avec d'autres entités autre que celle qui possède le site. On peut choisir un ou plusieurs valeurs parmi 6 possibles. Cela inclus:

1. Une autre société uniquement parce qu'elle aide à traiter la requête du client.
2. Les sociétés de transports et de coulissage.
3. Des sociétés qui ont les mêmes pratiques de privacité que la société du site.
4. Une société avec une pratique de privacité différente mais qui est comptable envers la société du site.
5. N'importe quelle société, quelle que soit sa politique.
6. Les personnes qui accèdent à l'information à partir de cercles publics comme les forums et les chat-rooms.

RETENTION

L'élément RETENTION décrit combien de temps le site peut garder des données privées. Au lieu de préciser le temps en termes de jour, mois, etc. Les concepteurs de P3P ont choisi de spécifier 5 valeurs possibles:

1. Retenir l'information uniquement pendant la cession en ligne courante.
2. Retenir l'information uniquement pendant la période nécessaire au traitement de la requête de l'utilisateur.
3. Retenir l'information uniquement pendant la période nécessaire pour un traitement légal.
4. Retenir l'information en conformité d'une politique bien précisée dans la portion humainement lisible du site.
5. Retenir l'information indéfiniment.

DATA

L'élément DATA décrit le type de données qu'un site peut collecter. On peut spécifier des informations bien définies comme "Nom" ou "Prénom" ou des types généraux de données. La spécification P3P v1.0 définie 17 catégories.

Les Préférences de L'utilisateur

Pour spécifier les préférences de l'utilisateur, les concepteurs de P3P ont choisi d'utiliser une syntaxe différente (de la spécification de la politique). C'est "A P3P Preference Exchange Language" (APPEL). APPEL [CrLM02] est un langage à base de règles encodé en XML. Les « user agents » comparent les fichiers APPEL contre des fichiers politiques pour déterminer si un site respecte les préférences d'un utilisateur. Ceci dit, les « user agents » ne sont pas obligés d'utiliser APPEL pour la comparaison, vu que ce dernier n'est pas un standard approuvé par le WC3, et il est toujours dans une phase d'expérimentation.

Le code source suivant est le contenu des préférences d'un client écrites en APPEL.

```
<appel:RULESET xmlns:appel="http://www.w3.org/2002/03/APPELv1"
      xmlns:p3p="http://www.w3.org/2002/01/P3Pv1">

      <appel:RULE behavior="limited" description="Site does not allow you to
            remove yourself from marketing/mailing list">
            <p3p:POLICY>
                  <p3p:STATEMENT>
                        <p3p:PURPOSE appel:connective="or">
                              <p3p:contact required="always"/>
                              <p3p:telemarketing required="always"/>
                        </p3p:PURPOSE>
                  </p3p:STATEMENT>
            </p3p:POLICY>
      </appel:RULE>

      <appel:RULE behavior="limited" description="This site collects data for an
      unknown purpose">
      <p3p:POLICY>
            <p3p:STATEMENT>
                  <p3p:PURPOSE>
                  <p3p:other-purpose required="always"/>
                  </p3p:PURPOSE>
            </p3p:STATEMENT>
      </p3p:POLICY>
      </appel:RULE>

      <appel:RULE behavior="limited" description="Unless you opt-out, this site
      collects data for an unknown purpose">
            <p3p:POLICY>
                  <p3p:STATEMENT>
                        <p3p:PURPOSE>
                              <p3p:other-purpose required="opt-out"/>
                        </p3p:PURPOSE>
                  </p3p:STATEMENT>
            </p3p:POLICY>
      </appel:RULE>

      <appel:RULE behavior="request">
            <appel:OTHERWISE/>
      </appel:RULE>


</appel:RULESET>
```

Figure 10 - Fichier APPEL

Ici les préférences consistent en un ensemble de 3 règles avec une règle "otherwise" pour les cas où aucune d'elles ne peut pas être appliquée. Les règles sont appliquées selon l'ordre de leur position dans le fichier APPEL. Lorsqu'une règle est déclenchée, c'est l'action décrite dans l'attribut "behavior" qui est exécutée.

2.4 Enterprise Privacy Authorization Language (EPAL 1.1) [AHKP04]

EPAL est un projet IBM développé principalement comme un outil B2B (business to business) pour contrôler le flux de données qui s'opère entre organisations lors de leurs interactions.

Il permet de s'assurer que l'information est protégée et qu'elle est utilisée en accord avec la politique de privacité de l'entreprise. IBM a définie EPAL comme un langage formel dont le but est d'imposer une politique de privacité sur toutes les applications et systèmes de l'entreprise, ainsi qu'avec ses partenaires.
Les politiques EPAL, à la différence de celles de P3P, sont réellement imposables. En effet, dans le cas d'EPAL, un moteur d'imposition (enforcement engine) s'assure qu'aucune information utilisée, stockée, ou échangée dans l'entreprise ou avec ses partenaires n'est en désaccord avec la politique de privacité de l'entreprise.

Le langage EPAL permet à une organisation d'exprimer un ensemble de règles concernant la manipulation et la consultation des informations qu'elle utilise (une politique de privacité). Autorisations et interdictions doivent contrôler l'utilisation des ressources informatiques dans une organisation. Le modèle demande de dresser des listes hiérarchiques de catégories de données, de catégories d'utilisateurs, et des finalités de collecte, ainsi que des ensembles d'actions, d'obligations et de conditions. Les **actions** décrivent comment les données sont utilisées, les **obligations** décrivent quelles actions doivent être effectuées, les **conditions** évaluent des éléments du contexte.

Une politique EPAL catégorise les données détenues par l'entreprise et établit des règles pour chaque catégorie. Dans la politique, l'ordre des règles est important car un principe d'ordre de précédence (descending precedence) s'applique, rendant inopérantes les règles suivantes dans une liste. Comme une règle XACML [PaLL11],

une règle EPAL comporte un **sujet**, une **action** et une **ressource** avec une indication de permission ou d'interdiction, et elle comporte en plus une mention de finalité. Une règle peut aussi contenir des conditions et des obligations.

La première étape pour une organisation est d'élaborer un vocabulaire comprenant tous les termes nécessaires à l'interprétation des règles. Ceux-ci sont listés dans l'ordre des éléments ci-dessous:

- **<vocabulary-information>**: Liste de tous les termes d'un vocabulaire EPAL permettant à une organisation de mieux l'exploiter comme un îlot de stabilité pouvant agir comme pivot dans l'interprétation des règles et la construction des tables de correspondance entre organisations. Le vocabulaire comprend les définitions de données nécessaires pour évaluer les conditions. Cet élément contient trois attributs: nom du vocabulaire, identification de source, et version.
- **<user-category>:** définit les catégories d'utilisateurs par rapport à l'accès aux données, qu'il s'agisse d'individus, de rôles, de groupes, d'applications informatiques ou du sujet des données personnelles lui-même. De type hiérarchique, la catégorie d'utilisateurs comporte un identifiant, un attribut facultatif parent, des éléments description-brève, description-enrichie, et propriété. La structure d'ensemble des catégories d'utilisateurs est une liste de hiérarchies, l'attribut parent permettant la mise en ordre en indiquant l'identifiant de la catégorie d'utilisateurs supérieure dans laquelle elle se trouve regroupée.
- **<data-category>:** définit les catégories de données par rapport à la privacité. Par exemple coordonnées domiciliaires, dossier médical, et statistiques. Les niveaux distincts de privacité de ces regroupements et de leurs éléments de données peuvent ainsi être pris en compte.
- **<purpose>:** définit les finalités pour lesquelles l'information a été collectée. Définition d'une hiérarchie de finalités. Principal apport de l'EPAL qui fait de la finalité une partie de toute requête.

- **<action>:** définit les actions associées à la privacité et qui sont autorisées. Les actions ne sont pas classées en hiérarchie. Des tables de correspondance avec les applications sont nécessaires pour déployer les actions EPAL.
- **<container>:** définit une structure abstraite pour des données faisant partie des conditions qui devront être évaluées ensembles avec des données en provenance du contexte externe au moment où une requête a été traitée. Plusieurs regroupements pratiques peuvent s'avérer utiles.
- **<obligation>:** définit les obligations qui peuvent être associées obligatoirement à une certaine action. Par exemple, la destruction de données personnelles après un certain délai peut être obligatoire dans divers contextes; ou que certains accès doivent être enregistrés dans un log.

Une politique EPAL est formulée comme un ensemble de règles relatives à la privacité selon un ordre précis de sous-éléments:

- **<policy-information>:** sert à décrire la politique au moyen d'un identifiant, d'une indication de détenteur (issuer), et d'une version.
- **<epal-vocabulary-ref>:** désigne le vocabulaire EPAL utilisé, sa localisation, son identifiant, son numéro de révision, et facultativement une description et une propriété. Le moteur EPAL y puisera son vocabulaire.
- **<condition>:** désigne plusieurs (ou aucune) évaluations des conteneurs du vocabulaire afin d'y repérer les conditions préalables à ajouter aux règles servant à la vérification des instances. Les conditions de contexte sont vérifiées par le biais de requêtes XACML dans son environnement.
- **<rule>:** les règles de confidentialité qui expriment vraiment les dispositions effectives de la politique dans ce qui est autorisé ou interdit. Une règle énonce que certaines catégories d'utilisateurs sont autorisées ou non à effectuer une action sur des données de différentes catégories et en fonction de certaines finalités et si les conditions établies sont satisfaites.

Dix types de données EPAL sont définis pour répondre aux besoins courants de représentation d'éléments dont on veut protéger la privacité. (Les noms formels EPAL sont dans le lexique ci-dessous):

- **identifiedObjectType:** Type d'objet identifié. Par exemple, document.
- **referringObjectType**: Type d'objet référant. Par exemple, url.
- **describedObjectType:** Type d'objet décrit. Par exemple, document de transaction.
- **hierarchicalType:** Type hiérarchique.
- **contactInfoType:** Type information d'un contacte. Par exemple, nom, organisation, email, adresse, pays…
- **attributeDefinitionType:** Type définition d'attribut. Les attributs ajoutés sont ceux de XACML, par exemple pour la date.
- **containerAttributeDefinitionType:** Type conteneur de définition d'attribut. Le conteneur est le moyen d'indiquer l'obligation de journaliser (pour rendre vérifiables) les actions relatives à cet attribut.
- **epalSimpleType:** Type simple EPAL. Types de données primaires qui sont ceux de XACML.
- **infoType:** Type information. Information qui identifie et décrit une politique ou un vocabulaire EPAL. Au minimum, le détenteur (issuer), la localisation de la ressource et sa version.
- **importStatementType:** Type énoncé d'importation. Pour importer un fichier EPAL avec son identifiant, sa localisation et sa version.

L'annexe 5 de la norme explique le recours au langage XACML pour l'expression des conditions en EPAL. Les désignations d'attributs d'environnement en XACML sont mises en correspondance avec les valeurs de l'attribut conteneur en EPAL.

L'annexe 6 situe EPAL par rapport à d'autres standards proches:

-**Platform for Privacy Preferences (P3P)** est la publication par un site de ce qu'il fait des données personnelles qu'il recueille, de l'utilisation qu'il en fait, et à quelles fins. EPAL diffère de P3P par les faits quelle est plus détaillée et quelle est imposée. P3P peut servir de résumé simplifié dérivé d'une politique EPAL.
-**Customer Profile Exchange Specification (CPExchange)** définit un format de données pour l'échange concernant les clients. Des types de données simples et complexes sont définis pour plusieurs sortes de données personnelles (par exemple, champs d'une adresse. Il est possible d'y ajouter un attribut de privacité, mais l'expressivité est limitée.

-**eXtensible Access Control Markup Language (XACML)** a une portée plus large pour le contrôle d'accès, mais il n'a pas la notion spécifique de finalité, propre au domaine de la privacité.

2.5 Autres travaux de recherche

Le privacité n'est pas un sujet qui bénéficie d'un grand engouement de la part de la communauté de recherche des web service / SOA. En effet en comparaison avec les autres aspects (surtout fonctionnels), la privacité (et les autres aspects non-fonctionnels) sont largement moins étudiés.

Dans [GBCH07b] les auteurs proposent un formalisme inspiré de P3P pour exprimer des politiques et des préférences dans les web services. Ils commencent par compléter la notion de business protocoles temporisés proposé dans [Pong06], c'est-à-dire la spécification de la séquence de messages échangés, en la dotant de la notion de privacité. Ensuite ils proposent des algorithmes pour déterminer si un web service peut remplacer un autre en respectant les contraintes de privacité (politiques et préférences).

Dans [Weix06], les auteurs considèrent la privacité des web services dans le cas de la composition. Ils proposent un framework (qui s'inspire aussi de P3P) et qui repose sur le mode de fonctionnement suivant: Le client demande un modèle du service fournisseur. Ce dernier résume la manière dont le fournisseur utilise les données du client (la politique). Le client vérifie par des techniques automatiques ses exigences (ses préférences) et s'il y a violation il demande au fournisseur de s'adapter en lui envoyant en retour des obligations à respecter. Le fournisseur doit s'adapter dynamiquement aux nouvelles exigences.

Dans [Anne04], les auteurs proposent un langage appelé WSPL: Web Services Policy Language, pour exprimer des politiques concernant plusieurs aspects non-fonctionnels: autorisation, QOS, qualité de protection, la fiabilité des messages, les options spécifiques aux applications et bien sûr, la privacité. La syntaxe de ce langage est un sous-ensemble de XACML standard d'OASIS. Il permet en particulier de fusionner deux politiques en une seule qui respecte les deux.
Dans [TuDT03], les auteurs utilisent une approche voisine de celle de [Weix06] , mais plus formelle. En effet ils utilisent un automate d'états finis pour exprimer la privacité dans le business protocole du web service.

2.6 Conclusion

Dans ce chapitre nous avons présenté la notion de privacité (d'un point de vue littéraire ce mot est un néologisme) mais nous avons insisté sur son aspect informatique. Nous avons présenté deux grandes implémentations de préservation de la privacité (P3P et EPAL) mais dans un cadre général. Nous avons aussi présenté la privacité dans le cadre d'une architecture SOA/Web Services telle qu'elle est abordée dans certains travaux de recherches.

Chapitre 3

Le Modèle de Privacité

3 LE MODÈLE DE PRIVACITÉ

3.1 Introduction

Dans ce chapitre nous présentons un modèle formel de privacité. Il utilise un paradigme de structures hiérarchiques. Comme dans P3P on distingue les deux concepts de politique et de préférence. Formellement, les deux concepts sont équivalents, ce qui nous permet de les comparer pour déterminer s'il y a concordance (totale ou partiale) entre les deux. Le modèle utilise aussi 3 ontologies (elle-même des structures hiérarchiques) : hiérarchie des buts (*Purpose hierarchy*), hiérarchie des utilisateurs (*User hierarchy*), hiérarchie des données (*data hierarchy*), ensuite il installe un ordre de restriction ou de spécialisation pour les 3 entités citées plus hauts (*but*, *utilisateur* et *données*). Le modèle de privacité sera ensuite utilisé pour annoter un business protocole dans le prochain chapitre.

3.2 Le modèle

On distingue deux types de règles:
– Les règles spécifiées par le fournisseur qu'on appelle *politiques privées*.
– Les règles spécifiées par le client appelées *préférences privées*.
Le modèle proposé pour modéliser une politique/préférence consiste à considérer une politique/préférence comme étant un ensemble indiquant les différents termes (clauses) d'utilisation des données. Chaque terme d'utilisation de données désigne le but pour lequel la donnée a été collectée. En outre, chaque but peut en découler deux types d'actions:

1. *Les droits* : Ce sont les actions que le fournisseur a le choix de faire ou de ne pas faire.
2. *Les obligations* : Ce sont les actions que le fournisseur doit réaliser afin d'assurer la sécurité des données.

Pour un but, nous spécifions l'entité qui le réalise, le délai dans lequel le but doit être réalisé, les droits impliqués ainsi que les obligations. Le droit est spécifié par l'entité autorisée à le réaliser, le délai dans lequel il est valide et le délai dans lequel il doit être accompli. Des obligations peuvent être attachées à la réalisation d'un droit et ce pour assurer la sécurité des données privées. Les obligations sont spécifiées de la même manière que les droits, sauf qu'une obligation n'implique aucune autre action ultérieure.

3.2.1 Les politiques

Comme le montre la figure ci-dessous, une politique définit l'ensemble des termes d'utilisation de données *TUD*.

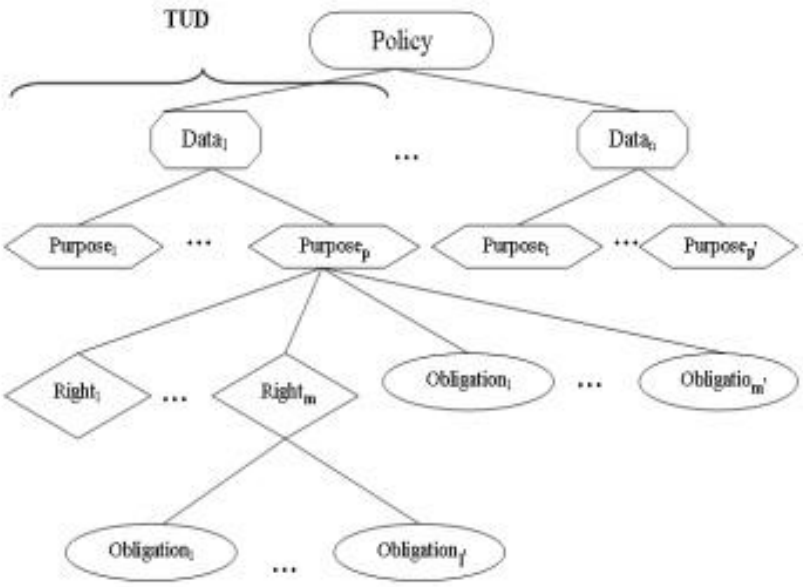


Figure 11 - Modèle de Privacité

Un $tud \in TUD$ est un couple (d, p) composé d'une donnée privé et d'un *but* (purpose) pour lequel la donnée a été collectée. Un *but* est une action représentant les besoins d'un client et qui est exécuté par une entité. Les entités utilisent la donnée pour satisfaire la requête (le but) du client dans une fenêtre temporelle. Le service prestataire peut exiger d'avoir le choix d'exécuter ou de ne pas exécuter certaines

actions appelées des *droits*. Puisque la satisfaction des buts et des droits suppose l'utilisation des données privées du client, le service prestataire doit garantir leurs sécurités. Pour ce faire, il doit spécifier des actions appelés *obligations* destinées à la sécurisation des données. Pour résumer, un but implique deux genres d'actions:

– *Les Droits* : Un droit est une action que le service prestataire a l'autorisation d'effectuer. Pour chaque droit, on spécifie des entités autorisées à l'exécuter, un délai durant lequel le droit doit être effectué une fois activé. De plus un droit peut entrainer un ensemble d'obligations.

– *Les Obligations* : Une obligation est une action que le service prestataire doit effectuer après une collecte de donnée privée pour assurer sa sécurité. Une obligation est spécifiée comme un droit mais sans faire intervenir d'autres actions.

Définition formelle

Définissant les ensembles suivants:

- *U*: ensemble des entités.
- *D*: ensemble des données.
- *A*: ensemble des actions.
- *P*: ensemble des buts (*purposes*).
- *O*: ensemble des obligations.
- *R*: ensemble des droits (*rights*).
- *I*: ensemble des intervalles.

Les ensembles *U*, *D*, et *A* sont représentés comme une hiérarchie par une ontologie utilisée pour comparer les niveaux de restriction de 2 politiques (voir Figure 13).

1. Une politique de privacité est un ensemble de termes de donnée (tud) avec $tud \in D \times P$
2. Un but (purpose) p est défini par le tuple

$$(a, u, \mu, S^R, S^O) \in A \times U \times I \times 2^R \times 2^O,$$

où a est une action identifiant le but, u est une entité réalisant le but, μ est un intervalle pendant lequel u doit réalisé le but a, S^R est un ensemble de droits et S^O est un ensemble d'obligations associés avec le but.

3. Un droit r est défini par le tuple

$$(a', u', v', \mu', S^{O\prime}) \in A \times U \times I \times I \times 2^O,$$

où a' est une action identifiant le droit, u' est une entité autorisée à réaliser l'action ′ , v' est un délai pendant lequel l'entité u' est autorisée à réaliser l'action a', μ' est un délai pendant lequel l'action doit être réalisée une fois activée.

4. Une obligation o est un tuple

$$(a", u", v", \mu") \in A \times U \times I \times I,$$

tel que $a"$ est une action identifiant l'obligation, $u"$ est une entité réalisant l'obligation, $v"$ est un intervalle de validité de l'action $a"$, $\mu"$ est un délai pendant lequel l'action $a"$ doit être réalisée une fois activée.

Puisque le service prestataire peut divulguer les données qu'il collecte à une partie tierce (un autre service prestataire), nous faisons la distinction entre deux types d'entités:

1. ***ours*** spécifie les entités du service collectant la donnée privée.
2. ***others*** spécifie des entités tierces à qui le service collecteur peut leurs divulguer des données privées.

Exemple illustrateur

Soit un service de réservation d'hôtel par internet. Un client spécifie la destination souhaitée (la ville). Le service de réservation suggère une liste d'hôtels. Une fois que le client a fait son choix, il est sollicité pour fournir un numéro de carte crédit afin de compléter la transaction. Puisque cette donnée ccn (*credit card number*) est très

sensible, le client peut exiger une utilisation sécurisée de cette dernière. Les restrictions sur ccn sont les suivantes:

Le service de réservation collecte ccn pour payer la réservation d'hôtel correspondante (p_1) dans un délai de 20mn après avoir reçu ccn. La banque qui est une entité externe possède le droit (r_1) de vérifier la validité de ccn dans un délai de 15mn après sa réception. Si le droit (r_1) est déclenché, l'action correspondante doit être réalisée dans un délai de 5mn. Le service de réservation doit détruire ccn (o_1) 10 mn après l'obtention du but (la fin de la transaction). En plus la banque doit crypter ccn (o_2) dans un délai de 60mn après sa réception. L'effacement (o_1) et le cryptage (o_2) doivent être exécutés immédiatement après leurs déclenchements, spécifiés par un intervalle vide $[0,0]$. Formellement nous écrivons:

- $plcy = \{(ccn, p_1)\}$ tel que
  $p_1 = (PayReservation, ours : FinancialService, \mu : [0, 20mn], \{r_1\}, \{o_1\})$
- $r_1 = (ValidityVerification, others : Bank, v : [0, 15mn], \mu : [0, 5], \{o_2\})$
- $o_1 = (Destruction, ours : FinancialService, v : [t(p_1), t(p_1) + 10mn], \mu : [0, 0])$ $tel\ que\ t(p_1) \in [0, 20mn]$ est la durée allouée pour l'obtention du but (p_1).
- $o_2 = (Encrypt, others : Bank, v : [0, 60mn], \mu : [0, 0])$

3.2.2 Les Préférences

Un service client peut envoyer ces propres données privées à un service prestataire, sous la forme de règles appelées préférences de privacité locales, qui décrivent comment il souhaite que le service prestataire utilise ses données privées. En plus, le service prestataire, dans le rôle d'un service client, peut envoyer les données privées collectées à un service tiers. Dans ce cas, il doit spécifier par des règles appelées préférences externes, comment le service tiers doit utiliser les données privées. Les préférences locales et externes constituent les préférences du client. Une préférence est similaire à une politique de privacité. En effet, elle est définie par un ensemble fini de *tud*. Cependant, nous ne faisons pas de distinction entre les entités internes (*ours*)

et externes (*others*). En fait, le client ignore si des entités sont considérées comme tierces (*others*) ou non (*ours*) par le service collectant les données privées.

3.2.3 Extraction des préférences externes à partir des politiques

Puisque un service Q peut invoquer d'autres services, il peut divulguer des données privées de ses clients à des services tierces comme illustré par la figure ci-dessous:

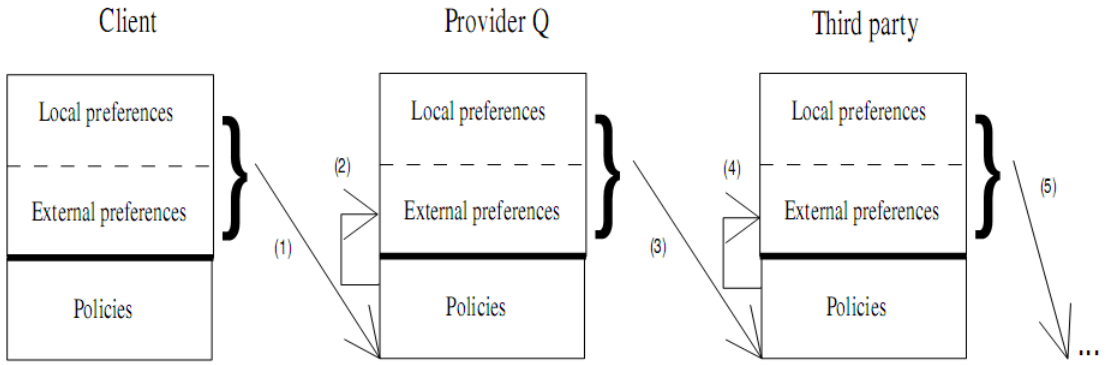


Figure 12 - Extraction des préférences

Les interactions entre un service Q et un service tiers sont basées sur les préférences du service Q et la politique du service tiers (3). Pour pouvoir informer le service tiers des restrictions du client original, à travers le service Q, nous ajoutons ces restrictions aux préférences externes du service Q (2) comme indiquée sur la figure. Ainsi nous propageons l'ensemble *tud* de la politique de Q en des préférences externes (2). Chaque *tud* de la politique est ajouté à l'ensemble *TUD* des préférences lorsqu'une entité responsable de la réalisation du but (droit, obligation respectivement) est une entité externe (others).

3.2.4 Raisonnement sur les politiques de privacité

Un des raisonnements que nous pouvons faire sur les politiques privés c'est la remplaçabilité. Pour comparer deux politiques nous nous proposons de définir un ordre entre elles, basé sur leurs niveaux de restrictions.

Formalisation de comparaison des politiques

Dans cette section, nous donnons une idée intuitive de la façon de comparer les niveaux de restriction de 2 politiques. La figure ci-dessous montre des parties des hiérarchies disponibles: hiérarchie des buts (*Purpose hierarchy*), hiérarchie des utilisateurs (*User hierarchy*), hiérarchie des données (*data hierarchy*).

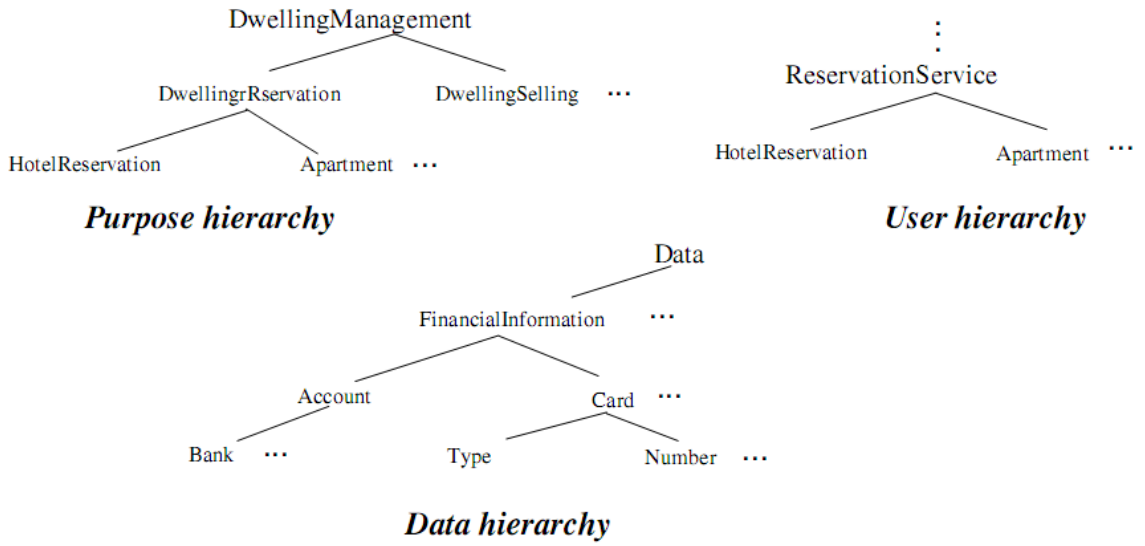


Figure 13 - les hiérarchies (ontologies)

Les hiérarchies sont utilisées pour comparer les données, les actions et les utilisateurs par rapports aux buts (droits, obligations respectivement) comme expliqué dans l'exemple suivant:

- $plcy_1 = \{(FinancialInformation, p_1)\}$ tel que
  $p_1 = (ReserveDwelling, ours : ReservationService, \mu: [0, 30mn], \{r_1\}, \{o_1\})$
- $r_1 = (ValidityVerification, others : FinancialPartener, v : [0, 15mn], \mu : [0, 5mn], \{o2\})$
- $o_1 = (Destruction, ours : ReservationService, v : [t(p_1), t(p_1) + 10mn], \mu: [0, 0])$ tel que $t(p1) \in [0, 30mn]$
- $o_2 = (Encrypt, others : FinancialPartner, v[0, 60mn], \mu : [0, 0])$

- $plcy_2 = \{(NumberCard, p_2)\}\ tel\ que$
  $p_2 = (ReserveHotel, ours : HotelService, \mu : [0, 20mn], \{r'_1\}, \{o'_1\})$
- $r'_1 = (ValidityVerification, others : Bank, v[0, 15mn], \mu : [0, 5], \{o'_2\})$
- $o'_1 = (Destruction, ours : FinancialService, v : [t(p_2), t(p_2) + 10mn], \mu :$
  $[0, 0]$ tel que $t(p_2) \in [0, 20mn]$
- $o'_2 = (Encrypt, others : Bank, v : [0, 60mn], \mu : [0, 0])$

En étudiant les 2 politiques ci-dessus, nous remarquons que la politique *plcy₂* est plus restrictive que la politique *plcy₁,* puisque la donnée privée concernée dans *plcy₁* est *FinancialInformation*, et comme on le voit dans la figure précédente, *FinancialInformation* est une généralisation de *Account* et de *NumberCard*. Ainsi la donnée collectée par *plcy₂* (*NumberCard*) est plus restrictive (plus spécifique) que la donnée de *plcy₁* (*FinancialInformation*).

En plus le but spécifie dans la politique *plcy₁* concerne *DwillingReserevation* (voir figure), qui est une généralisation de *HotelReservation* et de *Appartment*. Le but de la politique *plcy₂* est donc plus restrictif que celui de la politique *plcy₁*. Encore plus, l'entité qui doit obtenir le but de la politique *plcy₁* est *ReservationService* qui est une généralisation de *HotelService* l'entité de la politique *plcy₂*. On remarque aussi que les intervalles de la politique *plcy₂* sont contenus dans ceux de la politique *plcy₁*. Conclusion la politique *plcy₂* est plus restrictive que politique *plcy₁*.

Comparaison d'Obligations $\preccurlyeq_O$

Une obligation $o_2 = (a"_2, u"_2, v"_2, \mu"_2)$ est plus restrictive qu'une obligation $o_1 = (a"_1, u"_1, v"_1, \mu"_1)$ notée par

$$o_1 \preccurlyeq_O o_2$$

ssi

$$a"_2 \sqsubseteq a"_1,\ u"_2 \sqsubseteq u"_1, v"_2 \subseteq v"_1, \mu"_2 \subseteq \mu"_1.$$

La notation $X \sqsubseteq Y$ indique que : Y subsume X (Y englobe X).

Comparaison de Droits $\preccurlyeq_R$

Puisque un droit référence un ensemble d'obligations, la comparaison d'obligations définie plus haut nous permet de définir celle des droits. Un droit $r_2 = (a'_2, u'_2, v'_2, \mu'_2, S'^{O'}_2)$ est plus restrictif qu'un droit $r_1 = (a'_1, u'_1, v'_1, \mu'_1, S'^{O'}_1)$, notée par

$$r_1 \preccurlyeq_R r_2$$

ssi

$$a'_2 \sqsubseteq a'_1, u'_2 \sqsubseteq u'_1, v'_2 \subseteq v'_1, \mu'_2 \subseteq \mu'_1, \forall o'_1 \in S_1^{O'}, \exists o'_2 \in S_2^{O'}$$

tel que

$$o'_1 \preccurlyeq_O o'_2.$$

Comparaison de buts $\preccurlyeq_P$

Puisque un but référence des ensembles de droits et d'obligations, les comparaisons de droit et d'obligation définies précédemment, nous permettent de définir celle des buts.

Un but $p_2 = (a_2, u_2, v_2, S_2^R, S_2^O)$ est plus restrictif qu'un but $p_1 = (a_1, u_1, v_1, S_1^R, S_1^O)$ noté par

$$p_1 \preccurlyeq_P p_2$$

ssi

$$a_2 \sqsubseteq a_1, u2 \sqsubseteq u1, v2 \subseteq v1, \forall r_2 \in S_2^R, \exists r_1 \in S_1^R$$

tel que

$$r_1 \preccurlyeq_R r_2, \forall o_1 \in S_1^O, \exists o_2 \in S_2^O \; tel\ que\ o_1 \preccurlyeq_O o_2$$

Comparaison de tud $\preccurlyeq_{TUD}$

Puisque un *tud* définit un but pour lequel une donnée privé est collectée, la comparaison précédente des buts nous permet de définir celle des *tud*. Un terme d'utilisation de donnée $tud_2 = (d_2, p_2)$ est plus restrictif qu'un terme d'utilisation de donnée $tud_1 = (d_1, p_1)$ notée par

$$tud_1 \preccurlyeq_{\mathbf{TUD}} tud_2$$
$$ssi$$
$$d_2 \sqsubseteq d_1, p_1 \preccurlyeq_{\boldsymbol{P}} p_2$$

Comparaison des politiques $\preccurlyeq_{Private}$

Puisque une politique est définie par un ensemble fini de *tud*, la comparaison précédente des *tud* nous permet de définir celle des politiques.

Une politique $plcy_2$ est plus restrictive qu'une politique $plcy_1$ notée par

$$plcy_1 \preccurlyeq_{\mathbf{Private}} plcy_2$$
ssi
$$\forall tud_1 = (d_1, p_1) \in plcy_1, \exists tud_2 = (d_2, p_2) \in plcy_2$$
tel que
$$tud_1 \preccurlyeq_{\mathbf{TUD}} tud_2$$

Remarque: Une préférence *pref* est cohérente avec une politique *plcy* si la politique *plcy* est plus restrictive que *pref*

$$(pref \preccurlyeq_{\mathbf{Private}} plcy).$$

Ceci est possible parce qu'une préférence est définie comme une politique.

3.3 Conclusion

Ce chapitre introduit un modèle formel de privacité. Ce dernier est basé sur le paradigme du couple « politique/préférence » déjà utilisé dans d'autres modèles de privacité comme dans P3P. Son but est d'annoter un modèle de Business Protocole

pour le doter de la capacité de gérer la privacité lors des conversations entre web services. Ce sera l'objet du prochain chapitre.

Chapitre 4

Les Business Protocoles

4 LES BUSINESS PROTOCOLES

4.1 Introduction

Les protocoles standards de base de la spécification des web services, tels que définis par le WC3 (*SOAP* et *WSDL*), installent un cadre minimum pour faire fonctionner un web service de base. D'autres aspects encore, doivent être adressés pour le fonctionnement normal d'un web service "réel" (c'est-à-dire un web service déployé dans un système reparti et accédé par des millions d'utilisateurs via internet). Parmi ces aspects on peut citer la sécurité, la gestion des transactions, la tolérance aux pannes et l'interaction avec d'autres web services (dans le cadre d'une composition ou autres).

Ce dernier aspect, s'est avéré crucial dans beaucoup de domaines notamment pour les systèmes d'informations du commerce électronique. En effet, le fichier *WSDL* expose l'interface des services proposés, mais une information importante manque pour le faire interagir correctement avec les autres partenaires. Cette information c'est la séquence correcte des appels des fonctions (méthodes, procédures,…) représentés par les messages *WSDL*.

Pour résoudre ce problème la communauté de recherches sur les web services, propose le concept du business protocole. L'idée générale c'est d'utiliser un modèle formel pour représenter la séquence correcte de l'envoi des messages d'entrée/sortie d'un web service, et de convoyer cette information avec le fichier *WSDL* pour que les autres web services (clients ou prestataires) puissent interagir correctement avec lui. L'un des modèles formels utilisé souvent par la communauté de recherches pour représenter le Business Protocol (BP) c'est le FSM ("finite state machine" ou "automaton"). Par la suite nous allons passer en revues certains modèles de BP,

chacun d'eux essaye d'adresser un aspect ou un autre des fonctionnalités que les Web Services traitent.

4.2 Business Protocol [BbFF04]

Le premier Business Protocol étudié c'est le business Protocol proposé dans [BbFF04]. Le but de ce BP est de pouvoir:

1. Déterminer si un WS client peut interagir correctement avec un WS prestataire. C'est l'analyse de compatibilité. On considère alors leurs business protocoles respectifs

    a. Une compatibilité partielle : un protocole P_1 est partiellement compatible avec un protocole P_2 (noté par $\boldsymbol{P - Compat(P_1, P_2)}$) s'il existent des exécutions de P_1 qui interagissent correctement avec P_2 (c'est-à-dire qu'il y a au moins une conversation valide entre P_1 et P_2).

    b. Une compatibilité totale : un protocole P_1 est complètement compatible avec un protocole P_2 (noté par $\boldsymbol{F - Compat(P_1, P_2)}$), si toutes les exécutions de P_1 interagissent correctement avec P_2 (c'est-à-dire toutes les conversations de P_1 peuvent être comprises par P_2).

2. Déterminer si un WS prestataire peut remplacer un autre WS prestataire. C'est l'analyse de remplaçabilité. Cette notion a été traitée avec plus de détails puisqu'on définit:

    a. Equivalence de protocoles: 2 protocoles P_1 et P_2 sont équivalents (notée par $\boldsymbol{Equiv(P_1, P_2)}$) s'ils supportent le même ensemble de conversations. L'échange du protocole P_1 par P_2 et vis-versa sera complètement transparent pour un client.

b. Subsumption de protocoles : un protocole P_2 est subsumé par un protocole P_1 (noté par $\boldsymbol{Subs}(\boldsymbol{P_2}, \boldsymbol{P_1})$) si toutes les conversations de P_2 sont supportés par P_1. Dans ce cas P_1 peut complètement remplacer P_2, mais le contraire n'est pas forcément vrai.
c. Une remplaçabilité par rapport à un client particulier : Dans ce cas, on considère une notion de remplaçabilité moins forte que les deux précédentes où le protocole P_1 peut remplacer le protocole P_2 par rapport à un protocole client P_c, si toute conversation légale entre P_2 et P_c est aussi une conversation légale entre P_1 et P_c. On note alors $\boldsymbol{Repl}_{[P_c]}(\boldsymbol{P_1}, \boldsymbol{P_2})$. P_1 peut remplacer P_2, lorsque P_1 interagit avec P_c ce changement est transparent à P_c.
d. Une remplaçabilité par rapport à un rôle : C'est une notion de remplaçabilité encore plus faible que la précédente. Un standard industriel est défini par un protocole P_R. P_1 est conforme à P_R (c'est-à-dire qu'il peut reproduire tout ou uniquement certaines conversation de P_R). Est-ce-que P_2 peut remplacer P_1 lorsque P_1 se comporte comme P_R ? On note alors $\boldsymbol{Repl\_role}_{[P_R]}(\boldsymbol{P_2}, \boldsymbol{P_1})$.

4.2.1 Syntaxe formel d'un Business Protocol

Un business protocole est un tuple *P=(S, s_0, F, M, R)* dont les éléments sont définis comme suit :

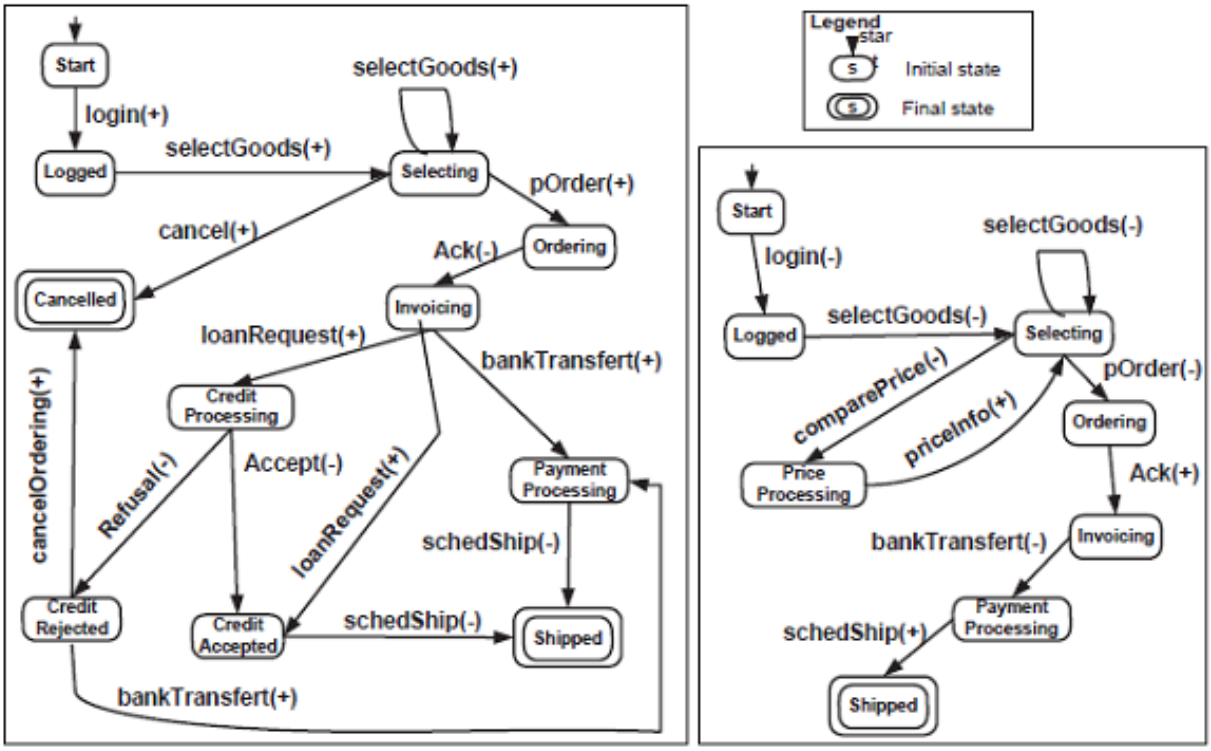
- S est un ensemble fini d'états.
- $s_0 \in S$ est l'élément initial.
- $F \subseteq S$ est un ensemble d'états finaux. Si $F = \emptyset$ alors *P* est un protocole vide.
- M est un ensemble fini de messages. Pour chaque message $m \in M$, on définit une fonction $Polarity(P, m)$ qui sera positive (+) si m est un message d'entrée dans P et négative (-) si m est un message de sortie de P. Dans le

reste de ce document nous utiliserons la notation $m(+)$ (respectivement $m(-)$) pour indiquer la polarité du message m.

- Un ensemble fini $R \subseteq S \times S \times M$ de transitions. Chaque transition (s, s', m) identifie un état source s, un état cible s' et soit m un message d'entrée ou un message de sortie, qui est soit consommé soit produit durant cette transition. Par la suite nous noterons par $R(s, s', m)$ la transition au lieu de (s, s', m). La figure ci-dessous montre deux Business Protocoles d'un Web service client est d'un Web service prestataire.

Remarque :

L'automate qui décrit un business protocole doit être un automate déterministe, c'est-à-dire, à partir d'un état quelconque on ne doit pas transiter vers 2 (ou plus) états différents avec le même message. Formellement: si (s, s_1, m), (s, s_2, m) sont deux transitions à partir de s alors on a obligatoirement $s_1 = s_2$. Le non-déterminisme dans le contexte d'un business protocole impliquerait qu'un protocole client ne peut pas déterminer dans quel état serait le protocole serveur, ce qui enlèverait tout intérêt aux business protocoles.



(a) The conversation protocol P_1 of a store service (b) The conversation protocol P_2 of a client

Figure 14 - Business protocole client P_2 et prestataire P_1

4.2.2 Sémantique des Business Protocoles

Dans la littérature spécialisée, la sémantique des processus (dont les business protocoles sont un sous-ensemble) est souvent spécifiée soit en « linear time semantic», soit en «branching time semantic». Dans « linear time semantic » un processus est déterminé par l'ensemble de toutes ces « exécutions », qui dans notre cas (business protocole), c'est la séquence « **état/message** » comme $Selecting.pOrder(-).Ordering.Ack(+)$ de la figure précédente. Dans «branching time semantic» la structure est aussi importante et doit être prise en compte. Cette dernière sémantique nous permet de déduire un arbre qui représente l'arbre de conversation d'un business protocole, c'est-à-dire un arbre qui spécifie toutes les conversations valides d'un business protocole. Remarquons en fin, que dans le cas

d'un automate déterministe (ce qui est notre cas), les sémantiques «branching time semantic» et « linear time semantic» sont équivalentes.

4.2.3 Simulation de protocoles

La Simulation est une relation de pré-ordre des systèmes de transitions étiquetées, qui permet de déterminer si deux systèmes donnés ont la même structure de branchement. Dans ce qui suit nous adaptons la notion de simulation aux business protocoles dans le but de comparer 2 protocoles vis-à-vis de leurs arbres exécution complets.

Définition 1 (**simulation de protocoles)**

Soient $P = (S, s_0, F, M, R)$ et $P' = (S', s_0', F', M', R')$ deux business protocoles,

- Une relation $\Gamma \subseteq S \times S'$ est une simulation de protocole entre P et P' si quel que soit $(s_1, s_1') \in \Gamma$ alors on a :
  - $\forall R(s_1, s_2, m)$ alors il existe un s_2' tel que$R'(s_1', s_2', m)$, $Polarity(P, m) = Polarity(P', m)$ et $(s_2, s_2') \in \Gamma$
  - $\forall (s, s') \in \Gamma$, si $s \in F$ alors $s' \in F'$
- On utilise la notation $s_1 \lesssim s_1'$ pour dire qu'il existe une simulation de protocole Γ tel que $(s_1, s_1') \in \Gamma$.
- On dit que le protocole P est simulé par P' (noté par $P \lesssim P'$) ssi $s_0 \lesssim s_0'$.
- On dit que deux protocoles P et P' sont similaires (noté par $P \cong P'$) ssi $P \lesssim P'$ et $P' \lesssim P$.

4.3 Business Protocol Privé

Dans cette section nous allons étudier une variante du business protocole qui permet de prendre en charge la gestion des données privées sensibles lors de l'échange de messages entre web services. Le modèle de privacité abordé dans le chapitre 3, sera utilisé pour annoter un business protocole pour exprimer la manipulation des données privées selon les souhaits des utilisateurs. Les objectifs d'un business protocole privé sont les suivants :

- Annoter les préférences et les politiques dans le business protocole
- Vérifier la conformité des politiques avec les préférences de privacité.

Dans un business protocole privé une interaction entre un service client et un service prestataire se déroule comme suit : le client spécifie ses données privées et comment ces données doivent être utilisées. Cette spécification représente *la préférence de privacité* du client. Ensuite, en utilisant un ensemble de *politiques de privacité*, le prestataire va montrer comment interagir avec le client en conformité avec les préférences du client. Une conversation ne pourra se réaliser uniquement si les préférences sont consistantes avec les politiques, c'est-à-dire si les préférences sont moins restrictives que les politiques.

Dans ce qui suit, nous montrerons comment étendre un business protocole pour accommoder la privacité.

La première extension consiste dans l'extension de la fonction de polarité pour prendre en compte les différents aspects de la privacité. La deuxième extension concerne les définitions des transitions et d'états.

Polarité : Pour spécifier qu'un message d'entrée (respectivement un message de sortie) importe (respectivement exporte) une donnée privée, nous étendons la

définition de la fonction de polarité en deux variantes selon la nature du message comme suit :

- Polarité des messages entrants dans le service: indique si le message importe une donnée privée du client.
- Polarité des messages sortants du service: indique si le message exporte sa propre donnée privée ou celle de ces clients.

Transition : Les politiques de privacité doivent être associées avec les messages entrants contenants une donnée privée. Par conséquent, les transitions avec messages privés entrants seront annotées avec la politique de privacité associée.

Etat : Un état peut être le point de départ de messages sortants privés. Par conséquent les préférences seront annotées dans les états.

4.3.1 Business protocole Privé :

Un Business protocole privé Q est un tuple $Q = (S, s_0, F, M, PREF, \vartheta, PLCY, T)$ dont les composants sont définis comme suit:

- S est un ensemble fini d'états, avec $s_0 \in S$ est l'état initial.
- $F \subseteq S$ est un ensemble d'états finaux. Si $F = \emptyset$, alors Q est une protocol vide.
- M est un ensemble fini de messages. Pour chaque message $m \in M$, on définit deux variantes de la fonction de polarité $Polarity(Q, m)$:

    - La polarité des messages entrants a la forme
      $IPolarity(Q, m) = (+, ClientPData)$
      tel que:
      $ClientPData = \begin{cases} 1 & \text{si le messag importe des donnes prive des clients} \\ 0 & sinon \end{cases}$
    - La polarité des messages sortants a la forme
      $OPolarity(Q, m) = (-, ClientPData, ServicePData)$

tel que :

$$ServicePData = \begin{cases} Y & \text{si le message } m \text{ exporte des données privées du service} \\ N & sinon \end{cases}$$

- $PREF$ est un ensemble fini de préférences.
- $\vartheta : S \rightarrow 2^{PREF}$ une fonction qui affecte un ensemble de préférences aux états.
- $PLCY$ est un ensemble fini de politiques.
- $T \subseteq S^2 \times M \times (PLCY \cup PREF)$ est un ensemble fini de transition. Chaque transition $(s, s', m, plcy)$ identifie un état source s, un état cible s', un message m , les politiques correspondantes $plcy \subseteq PLCY$ (si m est un message privé entrant) et les préférences correspondantes $\vartheta(s)$ affectés à l'état s (si m est un message privé sortant). Dans ce cas, nous disons que le message m est activé à partir de l'état s:
  - Si m est un message privé entrant (+) alors $PLCY \neq \emptyset$
  - Si m est un message privé sortant (-) alors $\vartheta(s) \neq \emptyset$

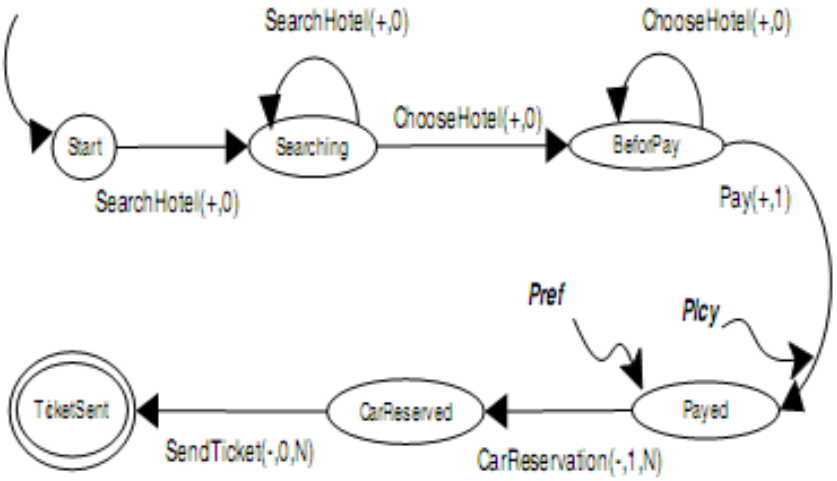


Figure 15 - Business Protocole Privé

La figure ci-dessus est une représentation graphique d'un business protocole privé Q d'un service qui permet la réservation d'hôtel, que nous avons déjà rencontré plus haut. Le message $pay(+,1)$ est un message entrant (+) qui importe une donnée privé du client (1). Donc nous annotons la transition correspondante avec la politique

appropriée. Le message $CarReservation(-,1,N)$ est un message sortant (-) qui exporte une donnée privé du client (1) et qui n'exporte pas une donnée propre au service (N). Nous annotons l'état source de cette transition avec la préférence correspondante:

- $plcy = \{(CCN, p_1)\}$ tel que:
  - $p_1 = (PayReservation, ours : FinancialService, \mu : [0, 20mn], \{r_1\}, \{o_1\})$
  - $r_1 = (ValidityVerification, others : Bank, v : [0, 15mn], \mu : [0, 5], \{o_2\})$
  - $o_1 = (Destruction, ours : FinancialService, v : [t(p_1), t(p_1) + 10mn], \mu : [0, 0])$
  - $o_2 = (Encrypt, others : Bank, v : [0, 60mn], \mu : [0, 0])$

- $pref = \{(Name, p'_1)\}$ tel que :
  - $p'_1 = (CarReservation, u : ReservationAgency, \mu : [0, 20mn], \{o'_1\})$
  - $o'_1 = \big(Destruction, u : ReservationAgency, v : \big[t(p'_1), t(p'_1) + 10mn\big], \mu : [0, 0]\big)$, $t(p'_1) \in [0, 20mn]$ est le temps pendant lequel le but p'_1 a été réalisé.

4.3.2 Les différentes classes de remplaçabilité de politiques privées

Nous distinguons 3 classes de remplaçabilité de politiques privées: (1) remplaçabilité privée, (2) équivalence privée, et (3) remplaçabilité privée partielle.

Remplaçabilité privée

Une politique $plcy_1$ peut être remplacée par une autre politique $plcy_2$, notée par $plcy_1 \preccurlyeq plcy_2$ si toutes les restrictions supportées par $plcy_1$sont aussi supportées par $plcy_2$. En plus on doit s'assurer que $plcy_2$ ne viole pas $plcy_1$. La violation se produit lorsque pour la même donnée d, la politique $plcy_2$ spécifie des buts (respectivement des droits, et obligations) qui ne sont pas spécifiés par $plcy_1$.

Définition 2 (remplaçabilité privée $\preccurlyeq$)

Une politique $plcy_1$ peut être remplacée par une autre politique $plcy_2$, notée par

$$plcy_1 \preccurlyeq plcy_2$$

ssi:

- $plcy_1 \preccurlyeq_{\mathbf{Private}} plcy_2$
- $\forall tud_2 = (d_2, p_2) \in plcy_2, \nexists tud_1 = (d_1, p_1) \in plcy_1 \; tel \; que$ $(d_2 \sqsubseteq d_1) \wedge \neg(p_1 \preccurlyeq_{\boldsymbol{P}} p_2)$.

Equivalence privée

Deux politiques sont équivalentes si elles ont le même niveau de restriction.

Définition 3 (équivalence privée ≡)

Une politique $plcy_1$ est dite équivalente à une autre politique $plcy_2$, notée par:

$$plcy_1 \equiv plcy_2$$

Ssi

$$(plcy_1 \preccurlyeq plcy_2) \wedge (plcy_2 \preccurlyeq plcy_1)$$

Remplaçabilité privée partielle

Lorsque la politique $plcy_2$ ne remplace pas la politique $plcy_1$, mais qu'au moins un tud peut être remplacé, nous disons que la politique $plcy_2$ peut partiellement remplacer la politique $plcy_1$.

Définition 4 (Remplaçabilité privée partielle)

La politique $plcy_2$ partiellement remplace la politique $plcy_1$ ssi les conditions suivantes sont vérifiées:

- $\exists tud_1 \in plcy_1, \exists tud_2 \in plcy_2 \; tel \; que \; tud_1 \preccurlyeq_{\mathbf{TUD}} tud_2$
- $\exists tud_1 \in plcy_1, \nexists tud_2 \in plcy_2$ tel que $tud_1 \preccurlyeq_{\mathbf{TUD}} tud_2$
- $\forall tud_2 = (d_2, p_2) \in plcy_2, \nexists tud_1 = (d_1, p_1) \in plcy_1$

$$tel \; que$$

$$d_2 \sqsubseteq d_1 \wedge \neg(p_1 \preccurlyeq_{\boldsymbol{P}} p_2)$$

La 1ere condition dit que la politique $plcy_2$ doit remplacer au moins un tud de la politique $plcy_1$. La 2eme condition dit qu'il existe au moins un tud de la politique $plcy_1$ qui ne peut pas être remplacé par la politique $plcy_2$. La 3eme condition s'assure que la politique $plcy_2$ ne viole pas la politique $plcy_1$.

Opérateur Différence privée

En plus des opérateurs déjà définis plus haut, nous introduisons un opérateur spécial business protocole privé. C'est l'opérateur *différence privé*. Cet opérateur dénoté par $\sqrt{^{Privé}}$ permet de trouver les éléments qui manquent dans une politique $Plcy_2$ pour remplacer une politique $Plcy_1$. Il prend en entrée deux politiques $Plcy_1$ et $Plcy_2$ et retourne une politique nommée *politique de différence*. Cette dernière décrit tous les éléments de la politique $Plcy_1$ qui ne sont pas communs avec ceux de la politique $Plcy_2$.

Définition 5 *(racine* $\sqrt{^{Privé}}$)

Soit $Plcy_1(TUD_1)$ et $Plcy_2(TUD_2)$ deux politiques. La différence $Plcy_1 \sqrt{^{Privé}} Plcy_2$ est une politique $Plcy_3(TUD_3)$ tel que : $TUD_3 \subseteq TUD_1$:
$tud = (d, p) \in TUD_3$ si :

- $d \in D_{Pr_1} \wedge \nexists d_2 \in D_{Pr_2}$ tel que $d_2 \sqsubseteq d$
- $p(op_p, u_p, delay_p, O_p, R_p) \in P_1, \nexists p_2(op_{p2}, u_{p2}, delay_{p2}, O_{p2}, R_{p2}) \in P_2$

tel que :

$$op_{p2} \sqsubseteq op_p,\ u_{p2} \sqsubseteq u_p, delay_{p2} \subseteq delay_p$$

  - $\exists r(op_r, u_r, v_{vr}, v_{rr}, O_r) \in R_P, \nexists r_2(op_{r2}, u_{r2}, v_{vr2}, v_{rr2}, O_{r2}) \in R_2$

tel que

$$op_{r2} \sqsubseteq op_r, u_{r2} \sqsubseteq u_r,\ v_{vr2} \sqsubseteq v_{vr},\ v_{rr2} \sqsubseteq v_{rr}$$

    - $\exists o'(op'_o, u'_o, v'_{vo}, v'_{ro}) \in O_r, \nexists o_2(op_{o2}, u_{o2}, v_{vo2}, v_{ro2}) \in O_2$

tel que

$$op_{o2} \sqsubseteq op'_o, u_{o2} \sqsubseteq u'_o,\ v_{vo2} \sqsubseteq v'_{vo},\ v_{ro2} \subseteq v'_{ro}$$

  - $\exists o(op_o, u, v_{vo}, v_{ro}) \in O_{\mathrm{p}}, \nexists o_2(op_{o2}, u_{o2}, v_{vo2}, v_{ro2}) \in O_2$

tel que

$$op_{o2} \sqsubseteq op_o, u_{o2} \sqsubseteq u,\ v_{vo2} \sqsubseteq v_{vo},\ v_{ro2} \subseteq v_{ro}$$

Cet opérateur nous permet d'obtenir les propositions suivantes :

- Si $Plcy_1 \sqrt{}^{Privé} Plcy_2 = \emptyset$ alors $Plcy_1 \sqsubseteq Plcy_2$
- Si $Plcy_1 \sqrt{}^{Privé} Plcy_2 \neq \emptyset$ alors $Plcy_2$ remplace partiellement $Plcy_1$.
- Si $Plcy_1 \sqrt{}^{Privé} Plcy_2 = \emptyset$ et Si $Plcy_2 \sqrt{}^{Privé} Plcy_1 = \emptyset$ alors $Plcy_1 \equiv Plcy_2$

4.4 Simulation des Business protocoles Privés

Comme dans le business protocole précédent, on définit une notion de simulation pour les business protocoles privé. Cette simulation est une extension de la précédente pour tenir compte des politiques privées.

Soient

$$Q = (S, s_0, F, M, PREF, \vartheta, PLCY, T)$$

et

$$Q' = (S', s'_0, F', M', PREF', \vartheta', PLCY', T'),$$

deux business protocoles privés. Le protocole Q' simule le protocole Q, noté par

$$Q \precsim Q'$$

S'il existe une relation $\Gamma \subseteq S \times S'$ telle qu'on a :

- $\forall (s_1, s'_1) \in \Gamma$ et $\forall T(s_1, s_2, m, plcy)$, $\exists\, s'_2$ tel que $T'(s'_1, s'_2, m', plcy')$, $m \sqsubseteq m'$, $polarity(Q, m) = polarity(Q', m')$, $(s_2, s'_2) \in \Gamma$, $plcy \preccurlyeq plcy'$
- $\forall (s, s') \in \Gamma$, si $s \in F$ alors $s' \in F'$

Remarques :

- Pour remplacer le service décrit par le protocole Q le 1er pas serait de découvrir un ensemble de web services. Supposons que parmi tous ces web services aucun ne remplace complètement Q en prenant en compte sa politique de privacité. Alors, il serait intéressant de choisir parmi ces services celui qui possède la politique de privacité la plus proche de celle de Q (celle qui offre le plus haut degré de restriction par rapport à Q). Mais comme cette remplaçabilité de proximité serait partielle, elle ne sera pas automatique, et réclamerait une autorisation explicite du client.

- Lorsque on remplace un service Q par Q', Q' utiliserait ces propres données privées, et donc on ne serait pas obligé de vérifier les préférences locales, mais Q' peut aussi utiliser des données privées des clients de Q, et donc logiquement nous devons vérifier la consistance des préférences externes de Q avec celle de Q', mais vu que les préférences externes sont extraites à partir des politiques déjà vérifiés dans la remplaçabilité privée entre Q et Q', elles sont déjà vérifiées implicitement.

4.5 Conclusion

Dans ce chapitre nous avons étudié 2 modèles formels de Business protocoles. Le premier [BbFF04] est le modèle original à partir duquel le 2ème est dérivé. La contribution du modèle de [GBCH07a], c'est la gestion de la privacité. Ce dernier modèle utilise le modèle de privacité du chapitre 3 pour annoter le business protocole ce qui permet de décrire d'une manière claire les préférences de privacité des clients d'un web service, afin qu'il respecte lui-même ces préférences et impose le respect de ces préférences à d'éventuels prestataires.

Chapitre 5

Business Protocole Privé Temporisé – BPPT

5 BUSINESS PROTOCOLE PRIVE TEMPORISE – BPPT

Dans ce chapitre nous commençons par introduire les automates à états finis temporisés (ou TA pour Timed Automata) de (Dill et Alur) [RaDl94]. Les TA et leur sous-classe ECTA (Event Clock Timed Automata) sont « le socle formel » sur lequel notre business protocole est bâti. Le business protocole que nous proposant permet de traiter les contraintes temporelles.

5.1 Introduction aux Automates Temporisés (AT)

Les Automates Temporisées sont une extension des Machines à états finies (FSM – Finite State Machine), appelés aussi Automates, introduite en 1990 [AlDi90] par Alur et Dill, et révisée en 1994 [RaDl94]et 1996 [DiAl96]. Ce formalisme garde toutes les caractéristiques originales des automates mais introduit une notion de temps à l'aide d'un ensemble d'horloges.

Un AT est une machine à états finis (un graphe contenant un ensemble fini de nœuds et un ensemble fini de transitions étiquetées) augmenté avec des variables de type réel. Un tel automate peut être considéré comme un modèle abstrait d'un système temporel. Les variables modélisent les horloges logiques du système, qui sont initialisées à 0 lorsque le système démarre et qui augmentent d'une manière synchrone avec le même rythme. Des contraintes sur les horloges (appelées les Gardes) sont utilisées pour restreindre le comportement du système (et donc le contrôler). Une transition (représentée par un arc) peut se produire lorsque les valeurs des horloges respectent la condition du garde étiqueté avec la transition. Après transition, des horloges peuvent être remis à zéro (reset). L'exemple ci-dessus montre un AT. Son comportement temporel est contrôlé par deux variables d'horloges x et y.

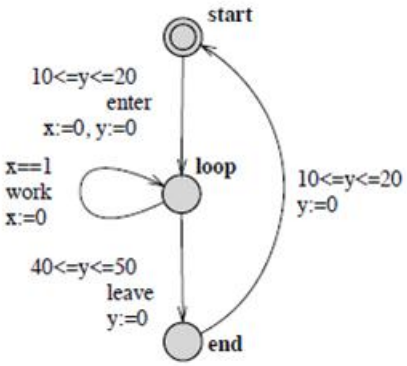


Figure 16 - (a) AT

L'horloge x est utilisée pour contrôler la boucle à partir du nœud "$loop$". La transition qui boucle (étiquetée "$work$") peut se produite lorsque $x = 1$. L'horloge y contrôle l'exécution globale du AT. Ce dernier peut quitter l'état initiale "$start$" à n'importe quel moment lorsque la valeur de y est comprise entre $10\ et\ 20$. Il peut aussi transiter entre "$loop$" et "end" avec la transition "$leave$" lorsque la valeur de y et entre $40\ et\ 50$. Après la transition "$enter$" les horloges x et y sont mis à zéro $(x := 0, y := 0)$. De même après la transition "$work$" x est remise à zéro $(x := 0)$ etc...

Remarque: Dans le paragraphe précédent, nous avons utilisé la phrase "Une transition (représentée par un arc) peut se produire" si les gardes sont respectés, mais cela n'implique pas forcément que la transition "doit" se produire. Le garde d'une transition (arc) n'est qu'un autorisateur mais en aucun cas il doit forcer la transition. En effet dans le AT de l'exemple précédent il peut rester dans l'état "$start$" indéfiniment même si les horloges x et y satisfassent le garde. Pour résoudre ce problème Alur et Dill dans leurs papier original de 1990 [AlDi90], introduisaient la notion de condition de Buchi dite aussi condition « d'acceptance »: un sous ensemble des nœuds est désigné (et marqué) comme étant "acceptant".
Ensuite, ne prendre en considération que les exécutions du TA qui passe par un de ces "nœuds acceptants". Dans l'exemple précédent, en supposant que le nœud "end"

est marqué "acceptant", impliquera que toutes les exécutions du TA de la figure.(a) doivent passer par "end". Ceci imposera implicitement des conditions sur "$start$" et "$loop$". Le AT doit quitter "$start$" au plus tard lorsque $y = 20$, sinon le AT sera bloqué sur "$start$" et ne pourra jamais passer par "end". De même le AT doit quitter "$loop$" au plus tard lorsque $y = 50$ pour pouvoir passer par "end". Plus tard Alur et Dill, modifièrent leur modèle en introduisant la notion de "safety" (Timed safety Automata). C'est une façon plus intuitive pour exprimer la notion de progression. Au lieu de la notion de "Acceptance-Buchi", on définit dans une partie des nœuds ("location" dans le jargon de Dill et Alur) des contraintes temporelles locales, appelées "invariants" ou "location invariants".

Un AT peut rester dans un nœud (location) tant que la condition "invariant" est vraie sinon la transition doit se produire. L'exemple suivant (figure. (b)) est l'équivalent "safety" de l'exemple "Buchi-acceptance" (c'est celui qui correspond à la figure (a)).

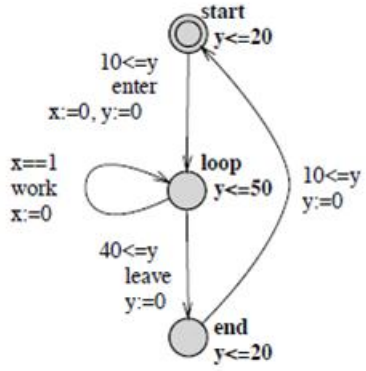


Figure 17 - (b) AT

Les invariants définissent des conditions locales sur "$start$" et "end". On "doit" quitter ces derniers lorsque $y > 20$. De même on doit quitter "$loop$" lorsque $y > 50$. Les invariants ont l'avantage de donner une vue locale des conditions temporelles d'un AT dans chaque nœud (location). Par la suite, tous les ATs considérés dans ce document, seront des "safety Timed Automata".

5.1.1 Syntaxe formelle:

- Soit C un ensemble fini d'horloges désignées par $x, y, z \ldots$
- Soit Σ un ensemble fini d'alphabets désignés par a, b, c pour exprimer les actions (les étiquettes des transitions).
- Soit l'expression sous forme d'une formule conjonctive de contraintes d'horloges atomiques de la forme :

$$(x \sim n)\ ou\ (x\ -\ y \sim n)\ o\grave{u}\ x, y \in C\ et \sim \in \{\leq, <, =, >, \geq\}\ et\ n\ \in \mathbb{N}.$$

Les contraintes d'horloges seront utilisées dans les AT pour les gardes et les invariants. Nous utiliserons la notation $\beta(C)$ pour désigner l'ensemble des contraintes d'horloges.

5.1.2 Définition d'un AT

Un Timed Automata (TA) est un tuple $A(V, V_0, V_f, E, I)$ où:

- V est un ensemble de nœuds (locations).
- $V_0 \subset V$ est un ensemble de nœuds Initiaux.
- $V_f \subset V$ est un ensemble de nœuds finaux.
- $E \subseteq V \times \beta(C) \times \Sigma \times 2^C \times V$ est l'ensemble des arcs (transitions ou edges).
- $I : V \rightarrow \beta(C)$ une fonction qui affecte des invariants aux noeuds (locations).

On utilisera la notation $l \xrightarrow{(g,a,r)} l'$ lorsque $(l, g, a, r, l') \in E$. avec:

- l : location (noeud) source.
- l': location (noeud) cible.
- a: une action $\in \Sigma$ (ε (chaıne nulle) est acceptée).
- g : garde (une contrainte d'horloge (atomique ou complexe)).
- r : un ensemble d'horloge à initialiser.

5.1.3 Sémantique d'un AT

- Pour un AT noté A, on définit un système de transition de configuration infini $S(A)$.
- Configuration q: une configuration q est un couple (l, v), où l est un nœud (location) et v est un vecteur d'horloge qui réalise une bijection de C dans $\mathbb{R}_+$, satisfaisant $I(l)$.
- (l, v) est la configuration initiale si $l \in V_0$ et $v(x) = 0$.
- Transition de configuration par écoulement du temps:

  $\forall (d > 0), (l, v) \xrightarrow{d} (l, v + d)$ si v et $v + d$ satisfassent $I(l)$.
- Transition de configuration par changement de nœuds:

$$(l, v) \xrightarrow{a} (l', v')$$

s'il y a un arc (l, g, a, r, l') / v satisfait g et $v' = v[r := 0]$

5.1.4 Déterminisme et Non-déterminisme

Dans la théorie des automates d'état finis (classique, intemporel), on démontre qu'un automate non-déterministe est équivalent à un autre automate déterministe. Pour les AT ce n'est pas toujours le cas. Par conséquent, les AT déterministes (deterministic timed automata- DTA) sont une sous-classe à part des AT. En effet beaucoup de problèmes classiques sont indécidables dans les AT (exemple la clôture de l'opération de complément) mais décidables dans les DTA. Un DTA est un AT qui possède les restrictions suivantes :

1. Il n'existe qu'un seul état initial : (V_0) est un singleton.
2. Si deux actions avec la même étiquette sortent d'un même nœud, alors leurs gardes doivent être différents. C'est-à-dire

$$\forall l \in V, \forall a \in \Sigma, \text{si } (l, g_1, a, r, l_1) \in E \text{ et } (l, g_2, a, r, l_2) \in E$$

alors $g_1 \neq g_2$

A cause de ces restrictions, les DTA sont moins expressifs que les AT. C'est-à-dire qu'ils existent des langages reconnus par des AT, qui ne peuvent pas être reconnus pas les DTA.

5.1.5 Event-Clock Timed Automata

Une autre sous-classe des ATs, particulièrement pertinente à ce travail, c'est Event-Clock Timed Automata (ECTA). Cette sous-classe n'est pas forcément déterministe, mais elle possède la particularité d'être « déterminisable», c'est-à-dire à chaque ECTA non-déterministe on peut calculer un équivalent déterministe. La principale caractéristique des ECTA c'est à chaque alphabet $a \in \Sigma$ on associé une horloge $x_a \in C$ qui sera initialisée chaque fois que a est reconnu. Le domaine du temps est $\mathbb{R}_{\geq 0} \cup \{\perp\}$. Le symbole $\perp$ indique qu'un alphabet n'a pas encore été reconnu. Au début toutes les horloges seront affectées avec $\perp$. Par conséquent la valeur des horloges dépend uniquement des alphabets.

5.2 Business Protocole Privé Temporisé -BPPT

Dans le chapitre 4 nous avons présenté un modèle d'un Business Protocole Privé selon le modèle du papier [GBCH07b]. Dans ce chapitre nous proposons un modèle de business protocole appelé "**Business Protocol privé Temporisé - BPPT**" qui améliore le business protocole privé en le dotant de la possibilité de traiter les contraintes temporelles qu'on rencontre dans les scénarios réels d'une conversation entre web services. Puisque le modèle précédent était basé sur un automate d'états finis, le nouveau modèle est aussi basé sur un automate d'états finis mais sur la variante qui prend en compte les contraintes temporelles, c'est-à-dire le TA (Timed Automata), et précisément l'ECTA (Event clock Timed Automata).

Dans la suite de cette section nous verrons comment appliquer les formalismes de l'ECTA pour augmenter le business protocole privé en Business protocole privé temporisé.

5.2.1 Contraintes d'horloges

Nous dotons le BPPT avec un ensemble de n variables d'horloges [MBSA12], où n est le nombre de transitions, de telle sorte que chaque transition a une horloge associée. Chaque horloge est une variable qui prend ses valeurs dans le domaine du temps $\mathbb{T}$. Les valeurs de ces variables changent simultanément. Les horloges n'ont pas les mêmes valeurs à un instant donné. Pour un ensemble d'horloges C, l'ensemble des contraintes d'horloge $\Phi(C)$ est défini par la grammaire suivante:

$$\varphi 1, \varphi 2 \ := \ c \ \alpha \ k \mid \neg \varphi 1 \mid \varphi 1 \ \wedge \ \varphi 2 \mid \varphi 1 \ \vee \ \varphi 2 \mid true$$

Avec

$$c \in C, k > 0, et \ \alpha \ \in \{<, \leq, >, \geq, =\}.$$

Exemple: Soit $C = \{c1, c2\}$ un ensemble d'horloges, les expressions suivantes expriment des contraintes d'horloges:

$$c1 \ = \ 4, (c1 \ < \ 10) \wedge (c2 \ = \ 5)$$

5.2.2 Ensemble Remise à Zéro C_0

Chaque transition dans un BPPT est associée à une horloge. Cette horloge doit être incrémentée lorsque la transition se produit la 1ère fois. Elle est remise à zéro lorsque la transition est déclenchée une autre fois.

Formellement nous écrivons:

$$C \supset C_0 = \{c_i \mid \forall (1 \leq i \geq n), n_{tr_i} > 1\}$$

où c_i est la variable d'horloge associée à la transition tr_i et n_{tr_i} est le nombre d'exécutions de la transition tr_i.

5.2.3 Valuation d'horloge

Une valuation d'horloge est une fonction $v : C \to \mathbb{T}$. Nous notons par $\mathbb{T}^C$ l'ensemble de toutes les valuations. Ces dernières sont soit des opérations du temps qui passe soit des remises à zéro.

Temps qui passe:

Soit $t \in \mathbb{T}$, on note par: $v + t$ la valuation d'horloge telle que:

$$(v + t)(c) = v(c) + t, \forall c \in C$$

Remise à zéro:

Soit $r \subseteq C$, on note par: $v[r \leftarrow 0]$ la valuation d'horloge telle que:

$$v[r \leftarrow 0](c) = \begin{cases} 0, si\ c \in C_0 \\ v(c), sinon \end{cases} \forall c \in C$$

5.2.4 Définition Formelle d'un BPPT

Un BPPT est un tuple de dimension 10, dont la définition est:

$$(S, s_0, F, M, T, \vartheta, C, C_0, PLCY, PREF)$$

où :

- S est un ensemble fini d'états et s_0 est l'état initial.
- F est un ensemble d'états finaux.
- M est un ensemble fini de messages. Pour chaque message $m \in M$ on définit deux types de polarité:
  - $IPolarity(BP, m) = (+, DataPrivacyClient)$, spécifique aux messages entrants (+) où:

    $$DataPrivacyClient = \begin{cases} 1, si\ le\ message\ contient\ une\ donnée\ privé\ du\ client \\ 0, sinon \end{cases}$$

  - $Opolarity(BP, m) = (-, DataPrivacyClient, DataPrivacyService)$, spécifique aux messages sortants (-) où $DataPrivacyClient$ est définie comme précédemment et $DataPrivacyService$ est définie comme suit:

$$DataPrivacyService = \begin{cases} Y, si\ le\ message\ contient\ une\ donnée\ privée\ du\ Service \\ N, sinon \end{cases}$$

- $T \subseteq S \times \varphi(C) \times M \times S$ est ensemble fini de transitions

$$tr = (s(pref), \varphi, m, plcy, s') \in T$$

représentant une transition de s vers s', $m \in M$ est un message de la transition, φ est une contrainte d'horloge associée à la transition tr,
$plcy \in PLCY$ est une politique, $pref \in PREF$ est la préférence associée à l'état s. Cette transition est représentée par:

$$s(pref) \xrightarrow{m,\varphi,plcy} s'.$$

- $\vartheta : S \to 2^{PREF}$ affecte un ensemble de préférences aux états.
- C est un ensemble fini d'horloges (des réels positifs).
- $C_0 \subseteq C$ est un sous ensemble de remises à zéro.
- $PLCY$ est un ensemble fini de politiques.
- $PREF$ est un ensemble fini de préférences.

- Si le message m est un message entrant, et que ses paramètres contiennent des données privées des clients, alors on a $plcy \neq \emptyset, pref = \emptyset$ et la transition est définie par: $s \xrightarrow{\varphi,m,plcy} s'$.

- Si le message m est un message entrant, et que ses paramètres ne contiennent pas des données privées des clients, alors on a :

$$plcy = \emptyset, pref = \emptyset,$$

et la transition est définie par: $s \xrightarrow{\varphi,m} s'$

- Si le message m est un message sortant, et que ses paramètres contiennent des données privées des clients et/ou des données privées du service, alors on a :

$$plcy = \emptyset, pref \neq \emptyset ,$$

et la transition est définie par: $s(pref) \xrightarrow{\varphi,m} s'$

- Si le message m est un message sortant, et que ses paramètres ne contiennent pas des données privées des clients et/ou des données privées du service, alors on a :

$$plcy = \emptyset, pref = \emptyset,$$

et la transition est définie par $s \xrightarrow{\varphi,m} s'$

5.3 Sémantique d'un BPPT

Une configuration d'un BPPT est définie par:
$(s, v) \in S \times \mathbb{T}_+^C$, où s représente le nœud (état) courant, et v la valeur des horloges courantes. La configuration (s, v) est considérée comme une configuration initiale d'un BPPT si $s = s_0$ et $v(c) = 0$ pour chaque horloge de C. L'exécution d'un BPPT est une séquence de paires (message, temps) :

$$\sigma = (s_0, v_0) \xrightarrow{m_1,t_1} (s_1, v_1) \xrightarrow{m_2,t_2} (s_2, v_2) \xrightarrow{m_3,t_3} \dots (s_n, v_n)$$

avec

$$t_i \in \mathbb{R}_+ \text{ et } t_{i+1} > t_i \; \forall \, i \geq 0.$$

Le temps t_i correspond au temps de passage du message, et $(v_i)_{i\geq 0}$ les valeurs des horloges tels que:

- $v_0(c) = 0, \forall c \in C$
- $\forall (i > 0), v_{i-1} + (t_i - t_{i-1}) \vDash \varphi_i$
- $\forall (i > 0)\, et\, \forall c \in C,\; v_i = \begin{cases} 0 & , si\ c \in C_0 \\ v_{i-1} + (t_i - t_{i-1})\ sinon \end{cases}$

La figure ci-dessous représente un BPPT de 5 nœuds et 6 transitions

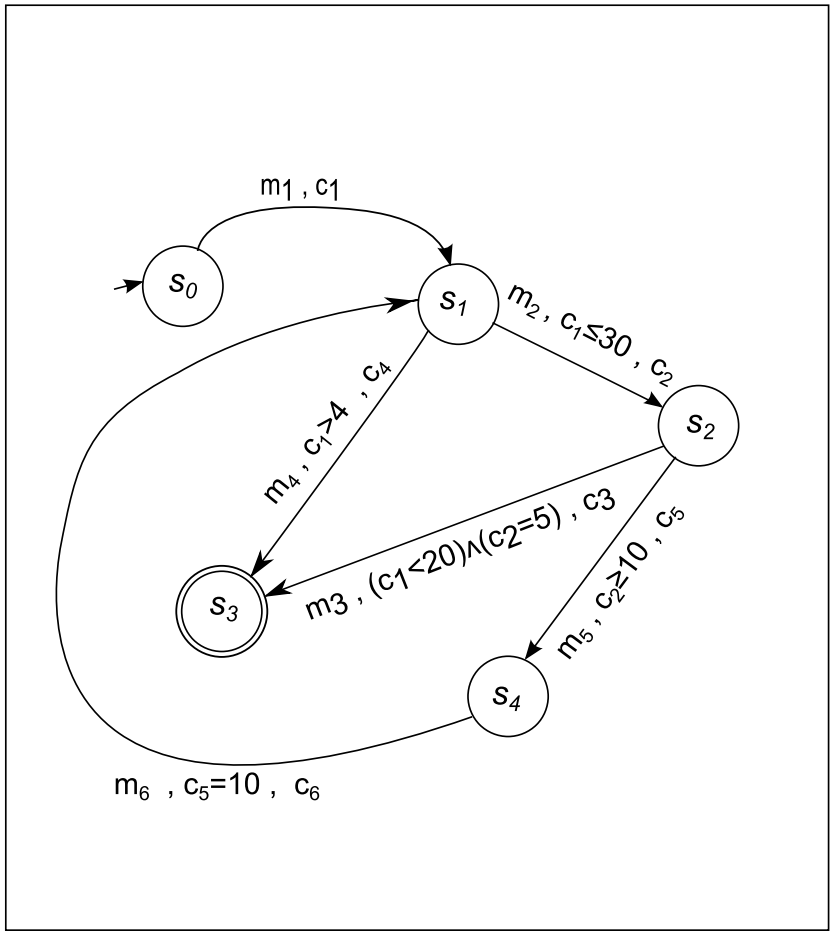


Figure 18 – Exemple d'un BPPT

Exemple: Soit le service dont le protocole est représenté par la Figure 18, on peut obtenir les conversations suivantes :

1. $(m_1, t_1 = 2)(m_2, t_2 = 7)(m_3, t_3 = 10)$. Les valuations d'horloges pour chaque état de ce chemin sont :

configuration	Valuations		
	c_1	c_2	c_3
s_0, v_0	0	0	0
s_1, v_1	2	0	0
s_2, v_2	7	5	0
s_3, v_3	10	8	3

- s_0 est un état initial, alors $v_0(c_1) = 0$.
- $v_2(c_1) = v_1(c_1) + (t_2 - t_1) = 2 + (7 - 2) = 7$.

2. $(m_1, t_1 = 3)(m_2, t_2 = 5)(m_5, t_3 = 14)(m_6, t_4 = 15)(m_2, t_5 = 18)$
$(m_3, t_6 = 20)$.
Les valuations d'horloges pour chaque état dans ce chemin sont :

configuration		Valuations			
	c_1	c_2	c_3	c_5	c_6
s_0, v_0	0	0	0	0	0
s_1, v_1	3	0	0	0	0
s_2, v_2	5	2	0	0	0
s_4, v_3	14	11	0	9	0
s_1, v_4	15	12	0	10	1
s_2, v_5	18	$0 \longrightarrow 3$	0	13	4
s_3, v_6	20	5	2	15	6

- $v_4(c_2) = v_3(c_2) + (t_4 - t_3) = 11 + (15 - 14) = 12.$

- si le message m_2 est exécuté une deuxième fois, alors la valuation de c_2 est initialisée à zéro et incrémentée en même temps:

$v_5(c_2) = 0 \; et \; v_5(c_2) = v_4(c_2) + (t_5 - t_4) = 0 + (18 - 15) = 3.$

5.4 Conclusion

Dans ce chapitre nous avons introduit un outil formel : les automates temporisés. Une sous-classe bien particulière des AT : ECTA (Event Clock Timed Automata) possède la caractéristique (essentielle pour nous) d'être « déterminisable ». En effet comme un Business protocole ne peut pas être non-déterministe (voir [7]), les AT « généraliste » ne conviennent pas pour sa modélisation. ECTA est donc utilisés comme fondement de notre Business Protocole Privé Temporisé.

Le BPPT est ensuite présenté en détail (syntaxe et sémantique). Enfin quelques exemples d'exécution de BPPT sont déroulés.

Chapitre 6

Vérification des BPPT

6 VÉRIFICATION DES BPPT

Dans ce chapitre, nous allons utiliser les capacités formelles des BPPT (elles même héritées des capacités formelles des TA et du modèle de privacité (TUD)) pour vérifier la conformité de la privacité et des contraintes temporelles lors des conversations entre web services. Pour ce faire, nous définissons une catégorisation des BPPT et nous identifions les propriétés à vérifier.

6.1 Catégorisation des BPPT

Nous classons les BPPT en deux catégories:

6.1.1 BPPT Courant

Noté $BPPT_C$, il collecte les données privées des clients de son web service associé. Les données privées citées dans la politique concernent l'opération courante d'un état (nœud). C'est donc le protocole qui est associé au web service en contact direct avec les clients. Ces derniers peuvent consulter (dans sa forme humainement lisible) la politique de privacité de ce web service, dans le cas où ils accordent une attention à ce sujet. Mais ils ne sont pas obligés de le faire systématiquement puisque il suffit d'exprimer leurs préférences une seule fois pour toute, ensuite ces dernières seront codées dans le $BPPT_C$ qui se chargera de comparer ces préférences automatiquement et systématiquement.

6.1.2 BPPT Transversal

Noté $BPPT_T$, utilise des données privées collectées par un $BPPT_C$. Exemple : les protocoles $BPPT_2$, et $BPPT_3$, de la figure ci-dessous, sont 2 protocoles transversaux qui utilisent deux données privées ($email$ et ccn respectivement) collectées par le protocole courant $BPPT_c$. Ce dernier se comporte comme un client vis-à-vis de $BPPT_2$, et $BPPT_3$. Notons que ces derniers n'ont aucune connaissance du client de $BPPT_C$

mais ils doivent quand même respecter ses préférences de privacité telles qu'il (le client) les a soumises à $BPPT_c$.

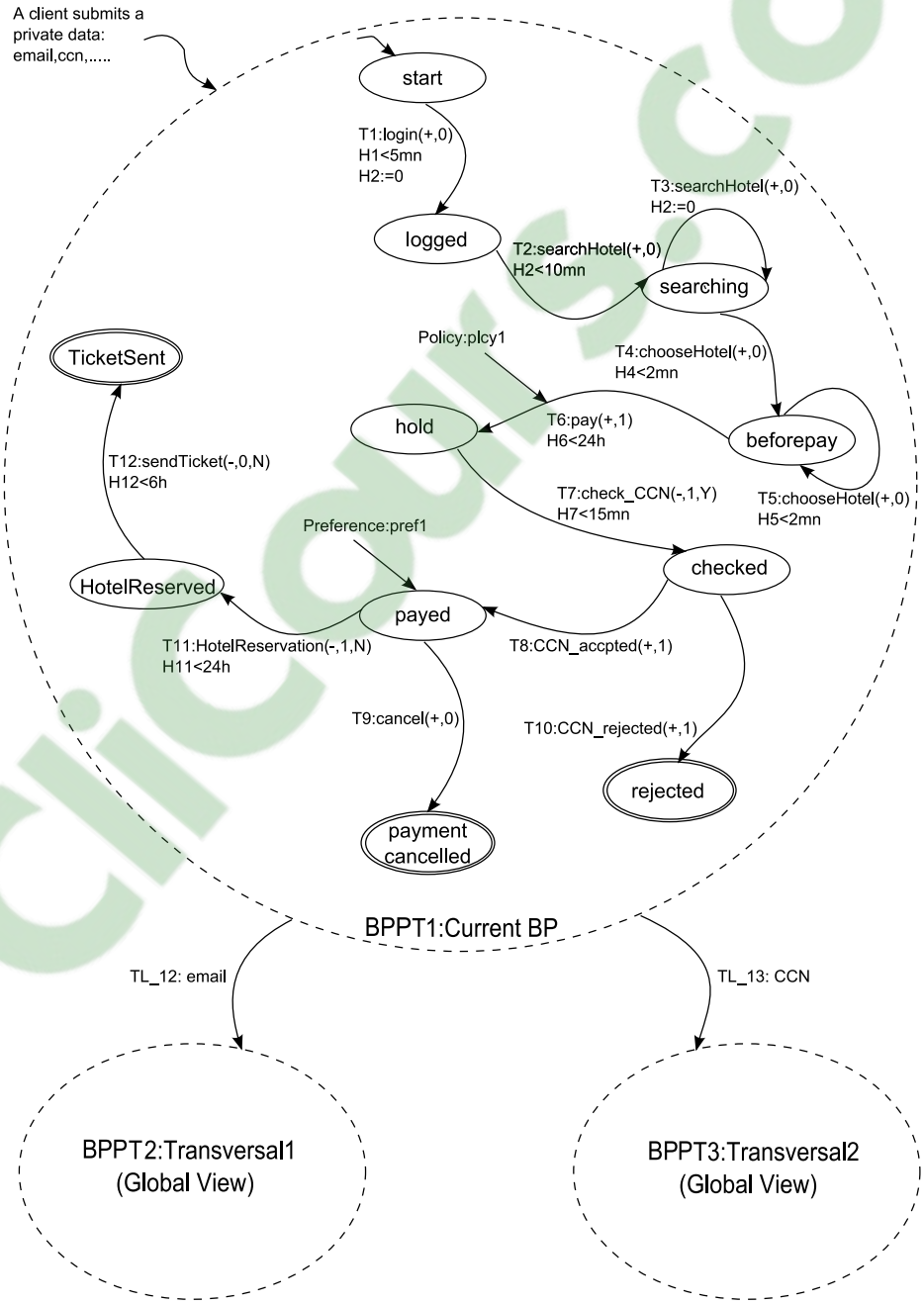


Figure 19 Type de BPPT

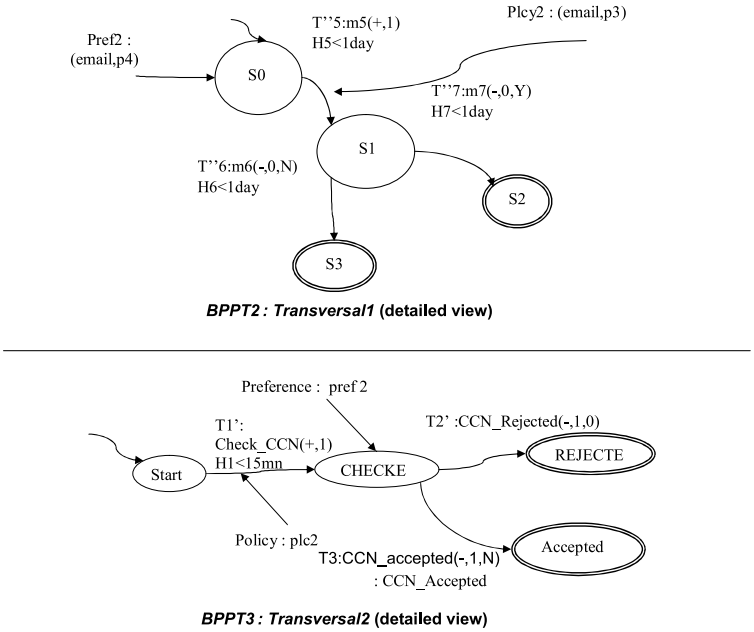


Figure 20 Business protocoles transversaux

La relation entre les catégories de protocoles est définie comme suit :

6.2 Liens de transition (Transition links)

Soit $BPPT_C$ un protocole courant, et $BPPT_T$ un transversal qui lui correspond, le lien entre eux est défini par :

$$TL_{ij} : \begin{matrix} BPPT_C \longrightarrow BPPT_T \\ s_i \longrightarrow q_{rj} \end{matrix}$$

$o\grave{u}\ s_i$ est un état (nœud) de $BPPT_C$ et q_{r_J} est un état pour lequel le droit rj est imposé dans $BPPT_T$. Nous désignons par :

$$L_i = \{TL_{ij}, \forall\ (1 \leq j \leq k)\}$$

l'ensemble des liens de transitions entre un état s_i de $BPPT_C$ collectant une donnée privée d et les états q_{r_j} de $BPPT_T$ utilisant la donnée privée d.

6.3 Vérification des Contraintes des BPPT

Nous définissons deux types de contraintes à vérifier: les contraintes d'horloges et les contraintes de privacité.

6.3.1 Vérification des Contraintes d'horloges

Les contraintes d'horloges peuvent être interprétées naturellement par les valuations:

si v est une fonction de valuation et $(v(c))_{c\in C}$ satisfait la contrainte d'horloge φ, nous disons que $(BPPT, s, v)$ vérifie φ et on note

$$(BPPT, s, v) \vDash \varphi ,$$

tel que :

- $(BPPT, s, v) \vDash c\ \alpha\ K\ ssi\ v(c)\ \alpha\ K$
- $(BPPT, s, v) \vDash \neg\varphi\ ssi\ (BPPT, s, v) \nvDash \varphi$
- $(BPPT, s, v) \vDash \varphi_1 \wedge \varphi_2\ ssi\ (BPPT, s, v) \vDash \varphi_1\ et\ (BPPT, s, v) \vDash \varphi_2$
- $(BPPT, s, v) \vDash \varphi_1 \vee \varphi_2\ ssi\ (BPPT, s, v) \vDash \varphi_1\ ou\ (BPPT, s, v) \vDash \varphi_2$

6.3.2 Vérification des Contraintes de privacité

Soit $T(s_{i-1}, m, s_i), \forall\ i > 0$ une transition dans $BPPT$, où m est un message entrant contenant une donnée privée. Par conséquent, une politique $plcy$ est associée à la transition. Cette transition pointe vers un état cible (s_i). Nous définissons δ_i comme une horloge associée à cette politique, est initialisée à 0. Elle est réinitialisée si la

transition T est déclenchée une autre fois. La valuation de l'horloge de la politique est définie par: $v(\delta_i)$.

6.3.3 Vérification de But (purpose)

Soit $\{op_1, \ldots, op_k\}$ un ensemble d'opérations relatives au but de l'état courant s_i. Soit $f : S \rightarrow 2^{but}$ une fonction d'étiquetage qui associe à chaque état s_i , $f(s_i)$ l'ensemble des opérations du but réalisés dans l'état s_i, $\forall s_i \in S$ alors:

$$(BPPT, s_i) \vDash op_1 \wedge op_2 \ldots . \wedge op_k$$
$$ssi$$
$$op_j \in f(s_i) \, et \, Inf(bound) \leq v(\delta_i) \leq sup(bound)$$

où $Inf(bound)$ est la borne inf. de l'intervalle du temps de validité de l'opération, et $Sup(bound)$ est la borne sup. de l'intervalle du temps de validité de l'opération, $v(\delta_i)$ est une valuation de l'horloge de la politique à l'instant de l'exécution de l'opération op_j.

6.3.4 Vérification d'obligation

Soit $O_i = \{o_1, o_2, o_3, \ldots, o_l\}$ un ensemble d'obligations associées au but p_k dans l'état s_i. $OP_{O_i} = \{op(o_1), op(o_2), op(o_3), \ldots, op(o_l)\}$ est l'ensemble des opérations relatives à l'obligation O_i. Soit $\eta : S \rightarrow 2^{obligation}$ une fonction d'étiquetage qui associe à l'état s_i, $\eta(s)$ l'ensemble des opérations relatives aux obligations vérifiées dans s_i . Alors nous aurons:

$$(BPPT, s_i.) \vDash op(o_1) \wedge op(o_2) \ldots \wedge op(o_l)$$

$$ssi$$

$$op(o_j) \in \eta(s_i) \; et \; Inf(\lambda_i) \leq v(\delta_i) \leq Sup(\lambda_i)$$

où $Inf(\lambda_i)$ et $Sup(\lambda_i)$ sont respectivement les bornes inf. et sup. de λ_i (l'intervalle de validité de l'action $op(o_j)$), $v(\delta_i)$ est une valuation de l'horloge de la politique à l'instant de l'exécution de l'opération op_j.

6.3.5 Vérification de droit (right)

Soit $OP_{R_i} = \{op(r_1), op(r_2), op(r_3), \ldots, op(r_k)\}$ un ensemble d'opérations associées aux droits.

Soit $g : S_T \rightarrow 2^{droit}$ une fonction d'étiquetage qui associe à chaque état $s_i \in S_T$, $g(s)$ un ensemble d'opérations associées au droit et satisfaits dans s_i. Alors nous aurons:

$$(BPPT_C, s_i) \vDash op(r_1), \wedge op(r_2) \ldots \wedge op(r_k)$$
$$ssi$$
$$\left(BPPT_{T(r_i)}, q(r_i)\right) \vDash op(r_i)$$
$$et$$
$$\left(BPPT_{T(r_i)}, q(r_i)\right) \vDash op(r_i)$$
$$ssi$$
$$op(r_i) \in g(q(r_i)) \; et \; Inf(\lambda) \leq v(\delta_i) \leq Sup(\lambda) \;,$$

où $op(r_i)$ est une opération associée au droit r_i, $Inf(\lambda)$ et $Sup(\lambda)$ sont respectivement les bornes inf. et sup. de λ qui est le délai pendant lequel l'entité associée au droit est autorisée à réaliser l'action associée à $op(r_i)$, $v(\delta_i)$ est une valuation de l'horloge de la politique à l'instant de l'exécution de l'opération $op(r_i)$.

Lemme

Soit s_i un état dans $BPPT_C$, $q(r_j)$ un état dans $BPPT_{T(r_j)}$, pour lequel le droit r_j est associé, et $op(r_j)$ une opération associée au droit r_j , alors on a:

$$(BPPT_C, s_i) \vDash op(r_j)$$

$$ssi$$

$$(\ BPPT_{T(r_j)}\ , q(r_j)\ )\ \vDash\ op(r_j)\ o\grave{u}\ q(r_j)\ =\ TL_{ij}(s_i).$$

6.4 Les étapes de la vérification

Avant d'aborder la vérification des contraintes, nous introduisons les définitions utiles suivantes:

6.4.1 Chemin

Un Chemin p dans un BPPT est défini en termes d'états et de transitions comme suit:

$$p\ =\ s_0 \xrightarrow{\varphi_1, m_1} s_1 \xrightarrow{\varphi_2, m_2} s_2 \xrightarrow{\varphi_3, m_3} \dots s_n$$

où s_0 est l'état initial, s_n est un état final et $\forall\ (0 < i \leq\ n)$, $(s_{i-1}, \varphi_i, m_i, s_i)\ \in\ T$

6.4.2 Exécution

Une exécution x de BPPT dans le chemin p est défini par:

$$x\ =\ (s_0, v_0) \xrightarrow{\varphi_1, m_1, t_1} (s_1, v_1) \xrightarrow{\varphi_2, m_2, t_2} (s_2, v_2) \xrightarrow{\varphi_3, m_3, t_3} \dots (s_n, v_n)$$

$$avec$$

$\forall\ (0 <\ i\ \leq\ n), v_i\ \in \mathbb{T}^C\ et\ (t_i)_{i>0}$ une séquence temporale.

Nous désignons par $x(s_0)$ l'ensemble des exécutions commencées à partir de l'état initial s_0.

La vérification des contraintes consiste dans les étapes suivantes:

Phase 1: Cette phase concerne la vérification des états le long du chemin d'exécution. Si m_i est un message entrant dans s_i contenant une donnée privée, les éléments suivants ont besoin d'être vérifiés:

- le but p associé à l'état s_i comme indiqué plus haut.
- les obligations $(o_1, o_2, \ldots, o_l)$ associées à p,
- les droits $(\mathrm{r}_1, \mathrm{r}_2, \ldots, \mathrm{r}_k)$ associés à p,
- les obligations $(\mathrm{o}_{\mathrm{j1}}, \mathrm{o}_{\mathrm{j2}}, \ldots., \mathrm{o}_{\mathrm{jq}})$ associées aux droits $\mathrm{r}_j, \forall\, (1 \leq j \leq k)$. Le comportement d'un état est vérifié *ssi*:

$$\big((BPPT_C, s_i) \vDash op(p)\big) \wedge \big((BPPT_C, s_i) \vDash op(o_1) \wedge op(o_2) \wedge \ldots \wedge op(o_l)\big) \wedge$$
$$(\forall\, 1 \leq j \leq k, ((BPPT_C, s_i) \vDash op(\mathrm{r_j})) \wedge$$
$$((BPPT_C, s_i) \vDash op(\mathrm{o_{j1}}) \wedge op(\mathrm{o_{j2}}) \wedge \ldots op(\mathrm{o_{jq}}))$$

Phase 2: Cette phase concerne la vérification des contraintes d'horloge associées aux transitions le long des chemins d'exécution. Une contrainte d'horloge est vérifiée ssi $(BPPT_C, s_i, v_i) \vDash \varphi_{\mathrm{i}}$, où φ_{i} est la contrainte d'horloge associée à la transition et s_i l'état cible de la transition.

Phase 3: Cette phase considère la conversation dans le business protocole courant. En effet, étant donné les états initial et final s_0 et s_n dans le chemin d'exécution:

$$r = (s_0, v_0) \xrightarrow{\varphi_1, m_1, t_1} (s_1, v_1) \xrightarrow{\varphi_2, m_2, t_2} (s_2, v_2) \xrightarrow{\varphi_3, m_3, t_3} \ldots (s_n, v_n),$$

la conversation est vérifiée ssi:

s_0 est l'état initial,

et

$$s_n \in F \; et \; \forall \, (1 \leq i \leq n), (BPPT_c, s_i, v_i) \vDash \varphi_i \wedge (BPPT_c, s_i) \vDash op(r_i) \, .$$

6.5 Conclusion

Dans ce chapitre nous abordons la vérification d'un protocole BPPT (et donc de son web service sous-jacent). Les éléments à vérifier sont les contraintes temporelles et les éléments du modèles TUD de sa privacité. Le but étant de détecter (pendant la conception et l'exécution) les anomalies qui empêchent le fonctionnement correct d'une conversation entre web services. Ces anomalies sont dues soit à l'incompatibilité temporelle, soit à l'inadéquation des politiques BPPT Courant (service prestataire) avec les préférences de privacité d'un client ou avec la politique de privacité d'un BPPT Transversal (service sous-traitant).

Conclusion

& Perspectives

CONCLUSION ET PERSPECTIVES

Dans le domaine des recherches sur les web services, on constate un intérêt croissant pour les technologies de la privacité en relation avec plusieurs types d'application e-business. Par exemples WS-POLICY aborde la question de l'imposition des politiques métiers entres les nœuds intermédiaires et finaux, mais cette spécification (dans sa version actuelle) ne traite pas les règles des politiques en détail. En effet bien que WS-POLICY définisse un modèle pour les préférences et les règles métiers, elle n'est pas encore implémentée [HaHo06].

Autres spécifications: EPAL (The Enterprise Privacy Authorization Language) est utilisée pour formaliser l'imposition réelle et effective des règles de privacité au sein d'une entreprise ou inter-entreprises dans des interactions business-to-business [28]. Une autre spécification: P3P (The Platform for Privacy Preferences Project) [AKSX03] développée par le World Wide Web Consortium (W3C) permet aux sites web de publier leurs politiques de privacité dans un format standard qui peut être téléchargé et traité automatiquement par des agents utilisateurs comme les browsers. Une spécification complémentaire APPEL [CLMM02] (A P3P Preference Exchange Language) permet aux agents utilisateurs de définir les préférences de privacité comme un ensemble de règles qui peuvent être comparées aux politiques des sites et prendre des décisions automatiques ou semi-automatiques concernant l'acceptation (ou non) de ces politiques. P3P/APPEL et EPAL ne sont malheureusement pas pertinents dans le contexte d'une architecture SOA.

Dans notre travail nous avons proposé un modèle formel qui permet de décrire la conversation qui s'opère entre web services (aspects fonctionnel) et gérer la privacité (aspect non-fonctionnel), même si cette conversation s'opère sur plusieurs « couches » d'appels entre web services. En effet notre protocole BPPT propage la

prise en compte des exigences de privacité d'un client de web service vers tous les autres web services qui traiteront une ou plusieurs de ses données privées.

Une autre caractéristique de notre BPPT, c'est qu'il traite les contraintes temporelles des règles métiers (business rules) que les web services implémentent.
Le fondement formel du BPPT (Les automates temporisés ECTA), procure à ce dernier une « solidité » similaire à ce que procure l'algèbre relationnelle aux systèmes SGBD. A partir de cette base on peut implémenter divers outils de génie logiciel pour les phases de développement (conception et simulation) et d'exploitation (monitoring et vérification).

Comme perspective à ce travail, nous suggérons deux axes de recherches :

- Améliorer la « résolution » de la vérification des caractéristiques temporelles du BPPT. C'est-à-dire, étudier et vérifier la conformité des variables temporelles combinées du modèle fonctionnel et de la privacité.
  En effet, pour l'instant 2 types de contraintes temporelles coexistent dans un BPPT, l'un exprimé par les horloges, l'autre par les tuples qui expriment les buts (*purpose*), droit (*right*) et obligation du modèle de privacité. Il serait intéressant de voir comment ces 2 types de contraintes temporelles « s'imbriquent les uns aux autres ». Déterminer s'ils entrent en conflit, et proposer éventuellement, de résoudre ce conflit automatiquement ou semi automatiquement.

- Prendre en compte non seulement les préférences de privacité d'un client, mais aussi d'autres informations contextuelles qui permettent de le caractériser comme les informations stockées dans les systèmes CRM (customer relationship managment). Par exemple, un system CRM d'une firme du téléphone peut retarder de couper la ligne d'un client si son « indice

d'assiduité de paiement » de ses anciennes factures (donnée contextuelle) est supérieur à un seuil donné.

Il serait intéressant d'annoter le business protocole avec ce type de données, étant donné qu'ils influent sur la logique métier d'un web service.

Bibliographie

Réferences

[AAAB07] ALVES, Alexandre; ARKIN, Assaf; ASKARY, Sid; BARRETO, Charlton; BLOCH, Ben; CURBERA, Francisco; FORD, Mark; GOLAND, Yaron; et al.: Web Services Business Process Execution Language Version 2.0 (2007)

[AAFJ02] ARKIN, Assaf; ASKARY, Sid; FORDIN, Scott; JEKELI, Wolfgang; KAWAGUCHI, Kohsuke; ORCHARD, David; POGLIANI, Stefano; RIEMER, Karsten; et al.: Web Service Choreography Interface (WSCI) 1.0 (2002)

[ACKM04] ALONSO, Gustavo; CASATI, Fabio; KUNO, Harumi; MACHIRAJU, Vijay: Web Services: Concepts, Architectures and Applications. In: M J CAREY; S CERI (eds.) *P Bernstein U Dayal C Faloutsos JC Freytag G Gardarin W Jonker V Krishnamurthy H Lu M A Neimat P Valduriez G Weikum J Widom*, Springer (2004) — ISBN 3540440089

[AHKP03] ASHLEY, Paul; HADA, Satoshi; KARJOTH, Günter; POWERS, Calvin; SCHUNTER, Matthias: Enterprise Privacy Authorization Language (EPAL 1.2). In: *Submission to W3C* (2003)

[AHKP04] ASHLEY, P; HADA, S; KARJOTH, G; POWERS, C; SCHUNTER, M; CORPORATION, I B M: Enterprise Privacy Authorization Language (EPAL). In: *IBM Research Report* (2004), Nr. RZ 3485 (#93951)

[AKSX03] AGRAWAL, R; KIERNAN, J; SRIKANT, R; XU, Y: Implementing P3P Using Database Technology. In: *Proceedings of the 19th International Conference on Data Engineering(ICDE'03)*. Bangalore, India, IEEE Computer Society (2003)

[AlDi90] ALUR, Rajeev; DILL, David L: A theory of timed automata. In: *17th ICALP, LNCS 443* (1990)

[AlFH99] ALUR, Rajeev; FIX, Limor; HENZINGER, Thomas A: Event-clock automata: a determinizable class of timed automata. In: *Theoretical Computer Science* vol. 211 (1999), Nr. 97

[Anne04] ANNE H. ANDERSON: An Introduction to the Web Services Policy Language (WSPL). In: *5th IEEE International Workshop on Policies for Distributed Systems and Networks,* (2004)

[BbFF04] B.BENATTALLAH; F.CASATI; F.TOUMANI: Analysis and management of Web Services Protocols. In: *ER'04: International Conference on Conceptual Modeling* vol. 3288. Shanghai, China, Springer (2004) — ISBN 3-540-23723-2

[BeCT04] BENATALLAH, B; CASATI, F; TOUMANI, F: Web service conversation modeling: a cornerstone for e-business automation. In: *IEEE Internet Computing* vol. 8, IEEE Educational Activities Department (2004), Nr. 1

[BeCT06] BENATALLAH, Boualem; CASATI, Fabio; TOUMANI, Farouk: Representing, analysing and managing Web service protocols. In: *Data Knowl. Eng.* vol. 58 (2006), Nr. 3

[BeMH07] BENBERNOU, Salima; MEZIANE, Hassina; HACID, Mohand-Said: Run-Time Monitoring for Privacy-Agreement Compliance. In: *ICSOC* (2007)

[BeMo06] BENATALLAH, Boualem; MOTAHARI-NEZHAD, Hamid Reza: ServiceMosaic Project : Modeling , Analysis and Management of Web Services Interactions. In: *Reproduction* vol. 53, Australian Computer Society, Inc. (2006), Nr. 2 — ISBN 192068235X

[Bera05] BERARDI, Daniela: Automatic Service Composition. Models, Techniques and Tools. In: *International Journal of Foundations of Computer Science* vol. 19, La Sapienza (2005), Nr. 02

[BeWJ02] BETTINI, Claudio; WANG, X Sean; JAJODIA, Sushil: Temporal Reasoning in Workflow Systems. In: *Distributed and Parallel Databases* vol. 11, Springer (2002), Nr. 3

[BFFB04] B.BENATTALLAH; F.CASATI; F.TOUMANI; BENATALLAH, B; CASATI, F; TOUMANI, F: Analysis and Management of Web Service Protocols. In: *ER'04: International Conference on Conceptual Modeling* vol. 3288. Shanghai, China, Springer (2004).

— From Duplicate 1 (Analysis and Management of Web Service Protocols - Benatallah, B; Casati, F; Toumani, F) — ISBN 3-540-23723-2

[BRSM03] BERARDI, D; DE ROSA, F; DE SANTIS, L; MECELLA, M: Finite state automata as conceptual model for e-services. In: *Contract*, Citeseer (2003)

[CKPM04] CANTOR, Scott; KEMP, John; PHILPOTT, Rob; MALER, Eve: Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0 (2004)

[CLMM02] CRANOR, L; LANGHEINRICH, M; MARCHIORI, M; M.MARCHIORI; L.CRANOR; M.LANGHEINRICH: A P3P Preference Exchange Language 1.0 (APPEL1.0), W3C Working Draft. In: *Technical report*, W3C Working Draft (2002), Nr. 15 April. — From Duplicate 1 (A P3P Preference Exchange Language 1.0 (APPEL1.0) - L.Cranor; M.Langheinrich; M.Marchiori) And Duplicate 2 (A P3P Preference Exchange Language 1.0 (APPEL1.0), W3C Working Draft - Cranor, L; Langheinrich, M; M.Marchiori) From Duplicate 3 (A P3P Preference Exchange Language 1.0 (APPEL 1.0) - Cranor, L; Langheinrich, M; Marchiori, M) www.w3.org/TR/P3P-preferences

[CrLM02] CRANOR, L; LANGHEINRICH, M; MARCHIORI, M: A P3P Preference Exchange Language 1.0 (APPEL 1.0) (2002). — www.w3.org/TR/P3P-preferences

[DaFH04] DALY, Janet; FORGUE, Marie-Claire; HIRAKAWA, Yasuyuki: World Wide Web Consortium Publishes First Public Working Draft of Web Services Choreography Description Language 1.0 - W3C's WS-CDL Targets Peer-to-Peer Web Services Collaboration (2004)

[Davi00] DAVIES, Jeff: The Definitive Guide to SOA, Apress — ISBN 9781590597972

[DCPV06] DYAZ, G; CAMBRONERO, M E; PARDO, J J; VALERO, V; CUARTERO, F: Automatic generation of Correct Web Services Choreographies and Orchestrations with Model Checking Techniques. In: *Telecommunications 2006 AICTICIW 06 International Conference on Internet and Web Applications and ServicesAdvanced International Conference on*, IEEE Computer Society (2006)

[DeCa06] DESWARTE, Yves; CARLOS AGUILAR MELCHOR: Current and future privacy enhancing technologies for the Internet. In: *Annals of Telecommunications* vol. 61 (2006)

[DiAl96] DILL, David; ALUR, Rajeev: Automata-theoretic verification of real-time systems. In: *Formal Methods for Real-Time Computing, Trends in Software Series, John Wiley & Sons Publishers* (1996)

[DiRe08] DIERKS, T.; RESCORLA, E.: RFC 5246 The Transport Layer Security (TLS) Protocol-Version 1.2 (2008)

[EFGS01] E.CHRISTENSEN; F.CURBERA; G.MEREDITH; S.WEERAWARANA: Web services description language (WSDL) 1.1. http://www.w3.org/TR/wsdl. (2001)

[GBCH07a] GUERMOUCHE, N; BENBERNOU, S; COQUERY, C E; HACID, M; N.GUERMOUCHE; S.BENBERNOU; E.COQUERY; M.S.HACID: Privacy-Aware Web Service Protocol Replaceability. In: *IEEE International Conference on Web Services ICWS'07*. Salt Lake City, Utah, USA, IEEE Computer Society (2007). — From Duplicate 2 (Privacy-Aware Web Service Protocol Replaceability - Guermouche, N; Benbernou, S; Coquery, C E; Hacid, M)

[GBCH07b] GUERMOUCHE, N; BENBERNOU, S; COQUERY, C E; HACID, M: Privacy-Aware Web Service Protocol Replaceability. In: *IEEE International Conference on Web Services ICWS'07*. Salt Lake City, Utah, USA, IEEE Computer Society (2007)

[Grou11] GROUP OMG: Catalog of OMG® Specifications (2011)

[HaHo06] HALLAM-BAKER, Phillip; HONDO, Maryann: Web Services Policy Framework (WS-Policy). In: *Structure* (2006), Nr. March

[HaPB07] HAMADI, Rachid; PAIK, Hye-young; BENATALLAH, Boualem: Conceptual Modeling of Privacy-Aware Web Service Protocols. In: *Banking* (2007)

[HeKi09] HEWETT, Rattikorn; KIJSANAYOTHIN, Phongphun: On Securing Privacy in Composite Web Service Transactions. In: *Electronics* (2009)

[HHKM02] HALLAM-BAKER, Phillip; HONDO, Maryann; KALER, Chris; MARUYAMA, Hiroshi; NADALIN, Anthony; NAGARATNAM, Nataraj; PRAFULLCHANDRA, Hemma; SHEWCHUK, John: WS-Security Profile for XML-based Tokens (2002)

[Hmah04] H. MAHMOUD QUSAY: Middleware for communications, John Wiley & Sons (2004) — ISBN 9780470862063

[HuFC04] HUNG, P C K; FERRARI, E; CARMINATI, B: Towards Standardized Web Services Privacy Technologies. In: *Proceedings of the IEEE International Conference on Web Services (ICWS'04)*. San Diego, California, USA, IEEE Computer Society (2004)

[Ibmc02] IBM CORPORATION: Security in a Web Services World: A Proposed Architecture and Roadmap (2002). — www-106.ibm.com/developerworks/library/ws-secroad/

[ImDS02] IMAMURA, Takeshi; DILLAWAY, Blair; SIMON, Ed: XML Encryption Syntax and Processing -W3C Recommendation 10 December 2002 (2002)

[KaPP06] KAZHAMIAKIN, R; PANDYA, P; PISTORE, M: Timed modelling and analysis in Web service compositions. In: *ARES 2006*, Ieee (2006) — ISBN 0769525679

[KBRF04] KAVANTZAS, Nickolas; BURDETT, David; RITZINGER, Gregory; FLETCHER, Tony; YVES LAFON: Web Services Choreography Description Language Version 1.0 - (2004)

[Khal07] KHALAF, R: From RosettaNet PIPs to BPEL processes: A three level approach for business protocols. In: AALST, W M P Van Der; BENATALLAH, B; CASATI, F; CURBERA, F (eds.) *Data & Knowledge Engineering* vol. 61, Elsevier Science Publishers B. V. (2007), Nr. 1 — ISBN 3540282386

[LGMM05] L.CRANOR; G.HOGBEN; M.LANGHEINRICH; M.MARCHIORI; M.PRESLER-MARSHALL; J.REAGLE; M.SCHUNTER: The Platform for Privacy Preference 1.1({P3P} 1.1) Specification. In: *Technical report*, W3C Working Draft (2005)

[Maco05] MACOVEI, G I: Timed Workflow Nets. In: *Proceedings of Sevents International Symposium on Symbolic and Numeric Algorithms for Scientific ComputingSYNASC 2005 IEEE Computer Society*, Citeseer (2005)

[MaMZ06] MARIA, Elisabetta De; MONTANARI, Angelo; ZANTONI, Marco: An automaton-based approach to the verification of timed workflow schemas. In: *Time*, IEEE Computer Society (2006) — ISBN 0769526179

[MBRH09] MOKHTARI, K; BENBERNOU, S; ROUACHED, M; HACID, M S; LEYMAN, F: *Privacy Time-Related Analysis in Business Protocols* : Ieee, 2009 — ISBN 9780769537092

[MBSA12] MOKHTARI-ASLAOUI, KARIMA; BENBERNOU, SALIMA; SAHRI, SOROR; ANDRIKOPOULOS, VASILIOS; LEYMANN, FRANK; HACID, MOHAND-SAID: TIMED PRIVACY-AWARE BUSINESS PROTOCOLS. In: *International Journal of Cooperative Information Systems* 21 (2012), Nr. 02, pp. 85–109

[MBSC08] MOKHTARI, K; BENBERNOU, S; SAID, M; COQUERY, E; HACID, M S; LEYMANN, Frank: Verification of Privacy Timed Properties in Web Service Protocols. In: *Proceedings of the International Conference on Services Computing SCC 2008*. vol. 2 : IEEE Computer Society Press, 2008 — ISBN 9780769532837, pp. 593–594

[Mitr03] MITRA, Nilo: SOAP Version 1.2, partie 0: Préliminaire - Recommandation W3C 24 Juin 2003 (2003)

[NGGB09a] NADALIN, Anthony; GOODNER, Marc; GUDGIN, Martin; BARBIR, Abbie; GRANQVIST, Hans: WS-Trust 1.4 OASIS Standard 2 February 2009 (2009)

[NGGB09b] NADALIN, Anthony; GOODNER, Marc; GUDGIN, Martin; BARBIR, Abbie; GRANQVIST, Hans: WS-SecureConversation 1.4 OASIS Standard 2 February 2009 (2009)

[NKMH06] NADALIN, Anthony; KALER, Chris; MONZILLO, Ronald; HALLAM-BAKER, Phillip: Web Services Security: SOAP Message Security 1.1 (WS-Security 2004) OASIS Standard Specification, 1 February 2006 (2006)

[NSEM00] N.GUERMOUCHE; S.BENBERNOU; E.COQUERY; M.S.HACID: Privacy-Aware Web service protocole replaceability. In: *IEEE International Conference on Web Services (ICWS 2007), July 9-13, 2007, Salt Lake City, Utah, USA*, IEEE Computer Society

[PaGe03] PAPAZOGLOU, M P; GEORGAKOPOULOS, D: Service-Oriented Computing Special Section: Introduction. In: *Communications of the ACM* vol. 46 (2003)

[PaHe07] PAPAZOGLOU, Mike P; HEUVEL, Willem-Jan: Service oriented architectures: approaches, technologies and research issues. In: *The VLDB Journal* vol. 16, Springer-Verlag New York, Inc. (2007), Nr. 3 — ISBN 106688880949877X

[PaLL11] PARDUCCI, Bill; LOCKHART, Hal; LEVINSON, Rich: OASIS eXtensible Access Control Markup Language (XACML) TC (2011)

[PoBC10] PONGE, Julien; BENATALLAH, B; CASATI, Fabio: Analysis and applications of timed service protocols. In: *ACM Transactions on* (2010), Nr. i

[Pong06] PONGE, Julien: A New Model For Web services Timed Business Protocols. In: *Informatique des organisations et systèmes d'information et de décision* (2006)

[PTDL07] PAPAZOGLOU, M P; TRAVERSO, P; DUSTDAR, S; LEYMANN, F: Service-Oriented Computing: State of the Art and Research Challenges. In: *Computer* vol. 40, IEEE Computer Society Press (2007), Nr. 11

[QaAb10] QASAIMEH, Malik; ABRAN, Alain: Privacy Protection Mechanisms for Web Service Technology. In: *2010 Eighth ACIS International Conference on Software Engineering Research Management and Applications*, Ieee (2010) — ISBN 9781424473366

[RaDl94] R.ALUR; D.L.DILL: A Theory of Timed Automata. In: *Journal of Theoretical Computer Science* vol. 2 (1994), Nr. 126

[Sara07] SARANG, Matjaz & B. Juric & Ramesh Loganathan & Dr. P. & G: SOA Approach to Integration, Packt Publishing (2007) — ISBN 9781904811176

[SSCI03] SYSTEMS, BEA; SOFTWARE, BMC; CA, Inc.; IBM; TECHNOLOGIES, Layer 7; MICROSOFT; NOVELL; VERISIGN: Web Services Federation Language (2003)

[Sunm88] SUN MICROSYSTEMS: rfc1057 - RPC: Remote Procedure Call Protocol Specification Version 2 (1988)

[Tber96] T. BERNERS-LEE, R. FIELDING, H. Frystyk: RFC 1945 : Hypertext Transfer Protocol -- HTTP/1.0 (1996)

[TFHJ03] T.ANDREWS; F.CURBERA; H.DOLAKIA; J.GOLAND; J.KLEIN; F.LEYMANN; K.LIEU; D.ROLLER; et al.: Business Process Execution Language for Web Serices (version 1.1) (2003)

[That01] THATTE, Satish: XLANG Web Services for Business Process Design. In: *Microsoft* (2001)

[TJCE00] TIM BRAY, Textuality and Netscape <tbray@textuality.com>; JEAN PAOLI, Microsoft <jeanpa@microsoft.com>; C. M. SPERBERG-MCQUEEN, W3C <cmsmcq@w3.org>; EVE MALER, SUN MICROSYSTEMS, Inc. <eve.maler@east.sun.com>; FRANÇOIS YERGEAU: Extensible Markup Language (XML) 1.0 (Fifth Edition)

[TuDT03] TUMER, Arif; DOGAC, Asuman; TOROSLU, Ismail Hakki: A Semantic-Based User Privacy Protection Framework for Web Services. In: *ITWP* (2003)

[Vino04] VINOSKI, S: WS-nonexistent standards. In: *IEEE Internet Computing* vol. 8, IEEE Educational Activities Department (2004), Nr. 6

[VOHH07] VEDAMUTHU, Asir S; ORCHARD, David; HIRSCH, Frederick; HONDO, Maryann; YENDLURI, Prasad; BOUBEZ, Toufic; YALÇINALP, Ümit: Web Services Policy 1.5 - Framework - W3C Recommendation 04 September 2007 (2007)

[Weix06] WEI XU V.N. VENKATAKRISHNAN R. SEKAR, I V Ramakrishnan: A Framework for Building Privacy-Conscious Composite Web Services. In: *IEEE International Confernce on Web Serviced (ICWS'06)* (2006)

[Wesl02] WESLEY, Ajamu: WSFL in action, Part 1 (2002)

[West69] WESTIN, Alan F.: Privacy and Freedom (1969) — ISBN 978-0370013251

[Wiki11] WIKILIVRE: Les réseaux TCP/IP (2011)

[Wwwo02] WWW.OASIS-OPEN.ORG: UDDI Version 2.04 API Specification (2002)

Specification (2002)