MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE
LA RECHERCHE SCIENTIFIQUE
UNIVERSITE ABDELHAMID IBN BADIS - MOSTAGANEM

Faculté des Sciences Exactes et de l'Informatique
Département de Mathématiques et d'Informatique
Filière : Informatique

MEMOIRE DE FIN D'ETUDES
Pour l'Obtention du Diplôme de Master en Informatique
Option : **Ingénierie des Systèmes d'Information**

THEME :

La sûreté de fonctionnement des services Web SOA

Etudiant(e) : « **HAMDI Khaled** »

Encadrant(e) : « BELKACEM Imad »

Année Universitaire 2015/2016

Remerciement

Tout d’abord je tiens à remercier monsieur BELKACEM Imad pour l'encadrement de ce mémoire et pour la confiance qu'il m'a accordée, sa disponibilité et ses bons conseils.

Je voudrais remercier toutes les personnes qui, de près ou de loin, ont contribué à l'accomplissement de ce travail.

Dédicace

A mes parents.

A mes frères.

A tous ceux qui me sont chers.

Résumé

Les logiciels actuels deviennent de plus en plus distribués et opèrent dans des environnements hautement dynamiques. L'utilisation de l'architecture SOA (Service Oriented Architecture) permet de rendre les applications complètement distribuées. La création d'une application qui utilise l'architecture SOA à grande échelle d'entreprise est une tâche difficile qui nécessite une connaissance approfondie. L'architecture orientée services (Service Oriented Architecture, ou SOA) constitue un style d'architecture basée sur le principe de séparation de l'activité métier en une série de services. Un service web SOA défaillant ou malveillant pourrait fournir à nos applications distribuées des données erronées et influencer sur les décisions finales prises à partir des données collectées. L'étude de la sûreté de fonctionnement de l'architecture orientée services est alors un sujet important.

Mots clés: Services Web; SOA; Sûreté de fonctionnement.

Table des matières

Chapitre II : Les services Web et les architectures SOA.

Chapitre III: Système décisionnel distribué pour la tolérance aux fautes des services web SOA.

Liste des figures

Chapitre I: La sûreté de fonctionnement des systèmes informatiques.

Chapitre II: Les services Web et les architectures SOA

Chapitre III : Système décisionnel distribué pour la tolérance aux fautes des services web SOA.

Liste des tableaux

Liste des Abréviations

API	Application Programming Interface.
BMP	Bean-Managed Persistence.
CMP	Container-Managed Persistence
CMEC	Contruction Mterial enegineering Council.
CSS	fueille de Style en Cascade.
DETC	The Distance Education and Training Council.
EJB	Entity Java Beans.
MDB	Message-Driven Beans.
DETC	Distance Education and Training Council.
DOM	Document Object Model.
DTD	Definition Type de Données
EJB	Entreprise Java Bean.
FTP	File Transfer Protocol
HTML	Hypertext Mark-up Language.
HTTP	Hyper Test Transfer Protocol
IDE	Integrated development Environment.
J2EE	Java 2 EEnterprise Edition.
JDBC	Java Data base Connectivity.
JDOM	Java Document Object Model.
JRE	Java runtime Environment
JSF	Java Server Faces.
JSP	Java Server Pages.
JVM	Java Virtual Machine.
LCD	Langage de Contrôle de Données.
LDD	Langage de Définition de Données.
POO	Programmation Oriented Objects.
SOA	Services Oriented Architecture.

Introduction générale

Apparus dès la fin des années 1990, les services web ont provoqué une forte évolution dans le monde de l'informatique distribuée et un bouleversement majeur dans la façon de concevoir des architectures.

Un des intérêts de l'informatique distribuée est de faciliter l'interconnexion entre applications distantes, indépendamment des plates-formes et des langages utilisés. Les trois derniers standards CORBA/IIOP (Common Object Request Broker Architecture / Internet Inter-ORB Protocol), DCOM (Distributed Component Object Model) et RMI (Remote Method Invocation) ont été créés dans ce but. Cependant, ces modèles, de par leur complexité, leur aspect fortement couplé sont en fait, incapables de passer à l'échelle.

Les services web représentent une tendance dans le monde de l'informatique distribué pour le développement d'une infrastructure logicielle flexible et fédératrice. Alors en reprenant l'idée de services Web, les composants logiciels fournissent leur fonctionnalité en utilisant la description des interfaces uniformes et des protocoles d'invocation et de communication d'Internet.

Basés sur les protocoles XML, les Services Web (SW) comme une mise en œuvre de l'architecture orientée services (Services Oriented Architecture ou SOA) permettent de mettre en place des applications faiblement couplées avec un fort degré de configuration dynamique. Elles se basent sur la notion de relations de "service" formalisée par un contrat qui unit le client et le prestataire de services.

Une caractéristique importante dans un système distribué est que les services peuvent apparaître, disparaître, et lié dynamiquement lors de l'exécution. Le déploiement et l'utilisation des services Web entraîne également un niveau variable de la qualité. L'accès à des services Web peut impliquer l'Internet et donc se fonde sur ses protocoles qui sont connus pour leur robustesse. Cependant, ils ne garantissent pas que la connexion puisse être établie et maintenu à des conditions données.

Donc, les services web ne présentent pas une solution magique pour tous les problèmes d'applications distribuées. D'abord comme les autres technologies distribuées. Les middlewares de service Web reposent sur les protocoles de d'Internet, donc ils héritent les problèmes de sûreté de fonctionnement existant dans le web. Par exemple, l'interaction entre les clients et les prestataires repose sur le Web ou d'autres réseaux, qui sont non fiables et qui peuvent causer une perte, un retard ou un entrave de messages ; en outre les services Web sont déployés sur des serveurs d'applications, qui ne peuvent pas fournir un service continue, en raison de la maintenance du système ou d'autres activités internes; aussi que la procédure

de la conception ou la mise en œuvre du services web peut contenir des fautes et peut influencer sur leur comportement. Donc la sureté de fonctionnement est une question vitale pour les services web, alors que dans les systèmes distribués qui sont composé de plusieurs services, la défaillance d'un service peut nuire à la sureté de fonctionnement d'autres services et donc la défaillance de tout un système. Il est logique que la sureté de fonctionnement des services Web comme un domaine de recherche ait suscité un intérêt actif dans les dernières années.

Alors avec leur architecture complexe, les applications de service Web sont menacées à un taux élevé de défaillances. Cela appelle une variété de méthodes pour être conçu pour minimiser les défaillances survenant dans les services Web et dans leur interaction avec les clients. À cet égard, l'utilisation de techniques de la tolérance aux fautes est essentielle pour améliorer la sûreté de fonctionnement des applications de service Web.

Dans ce travail, l'approche que nous développons consiste à concevoir un système de décision tolérant aux fautes et distribué qui détecte et localise les défaillances en comparant les résultats des différents services Web homogènes en utilisant des techniques statistiques non paramétriques sur les résultats obtenues. Ce qui permet à l'application de détecter le service Web défectueux, corriger les fautes et continuer son exécution en présence des entraves de la sûreté de fonctionnement.

Notre objectif est d'assurer la sûreté de fonctionnement de notre système en développant des modules d'analyse des données statistiques précisément dans le module de filtrage et de collection.

Pour réaliser la tolérance aux fautes d'une manière extensible et distribué, nous proposons un schéma de détection dans lequel nous utilisons la technique de la redondance matérielle passive, en analysant les données dans un nœud principal. Si la valeur obtenue pour un service Web n'appartient pas à un intervalle de confiance calculé, ce capteur est considéré défectueux. Dans le nœud principal, les modules de décision peuvent prendre une décision binaire concernant l'événement étudié et détecter les services Web défectueux.

Ce mémoire, témoin de ce développement, est composé de trois chapitres :

- Le chapitre I est consacré en particulier à la définition de la notion de la sûreté de fonctionnement, la première section présente les concepts de la sûreté de fonctionnement. La deuxième section représente les techniques de la tolérance aux fautes.
- Le chapitre II est une représentation des services Web et l'architecture orientée services, dans la première section, nous avons détaillé les technologies utilisés dans les services Web en précisant les protocoles de base pour construire un service Web. En deuxième section, nous avons parlé sur les architectures orientées services, en premier temps, nous avons défini la notion du service, nous avons montré la différence entre une architecture orientés services et une architecture orientée objets, et nous avons expliqué les concepts fondamentaux de la SOA. Dans la dernière section, nous avons fait une

recherche sur les travaux de la sûreté de fonctionnement des services Web à base de la SOA, en considérant que les services Web est une nouvelle technologie qui nécessite une intégration des mécanismes de la sûreté de fonctionnement spécialement pour l'automatisation des processus métier critiques, qui est à terme, la cible des technologies de services Web.

- Dans le chapitre III, nous avons présenté notre cas d'étude, notre approche comme une solution, qui nous permet d'utiliser la tolérance aux fautes en donnant une modélisation UML de notre système, et expliquant notre approche ave un exemple illustratif.

Problématique

Dans ce mémoire, nous allons uniquement nous intéresser à certaines caractéristiques de la sûreté de fonctionnement vis-à-vis des Services Web. La sûreté de fonctionnement d'un système informatique est la propriété qui permet à ses utilisateurs de placer une confiance justifiée dans le service qu'il leur délivre. Selon la, ou les applications auxquelles le service Web est destiné, l'accent peut être mis sur différentes facettes de la sûreté de fonctionnement, ce qui revient à dire que la sûreté de fonctionnement peut être vue selon des propriétés différentes mais complémentaires, qui permettent de définir les attributs suivant:

- ✓ Le fait d'être prêt à l'utilisation conduit à la disponibilité;
- ✓ La continuité du service conduit à la fiabilité;
- ✓ La non-occurrence de conséquences catastrophiques pour l'environnement conduit à la sécurité-innocuité;
- ✓ La non-occurrence de divulgations non-autorisées de l'information conduit à la confidentialité;
- ✓ La non-occurrence d'altérations inappropriées de l'information conduit à l'intégrité;
- ✓ L'aptitude aux réparations et aux évolutions conduit à la maintenabilité.

La maturité des travaux concernant la confidentialité et l'intégrité étant beaucoup plus aboutie que les autres, nous ne nous intéressons pas dans ce mémoire aux travaux basés sur cette thématique. On peut tout de même mentionner qu'actuellement, sur un réseau IP, la confidentialité des échanges, 1'intégrité des messages et l'authentification des agents participant à l'échange peuvent être gérées au travers d'une session sécurisée, via l'utilisation de protocoles de sécurité bas niveau tel que SSL (Transport Layer Security). Cependant, bien que la gestion de la sécurité au niveau transport soit nécessaire aux architectures de services Web, elle est insuffisante pour mettre en œuvre des architectures orientées services complexes sur Internet. Il a donc fallu réaliser des technologies d'infrastructure spécialement conçues pour la gestion de la sécurité des Services Web : WS Security, spécification édictée par OASIS, couvre l'ensemble des problématiques de sécurité d'un processus transactionnel comme l'authentification des utilisateurs et la gestion de leurs droits, la gestion de l'intégrité des messages par le biais de certificats ou encore le chiffrement des flux de données.

Ce mémoire se concentre donc sur la problématique plus ouverte de la disponibilité et de la fiabilité des Services Web. Nos travaux portent sur l'étude des mécanismes permettant surtout d'améliorer la sûreté de fonctionnement des services web. Cette thématique fait partie des recherches actuelles dans ce domaine.

Aujourd'hui nous savons beaucoup plus sur la disponibilité et la fiabilité et leurs implications pour Les services Web et leur importance dans le travail passé sur des systèmes

informatiques critiques pour des grandes entreprises tels que pour les marchés boursiers et les systèmes de contrôle de circulation aérienne.

Un haut degré de fiabilité est nécessaire tout en échangeant des messages entre le client et le prestataire où pour des transactions importantes qui sont menées au cours de web pour des systèmes critiques ou décisionnelles. Alors la nature des problèmes de fiabilité qui peuvent menacer la sûreté de fonctionnement des services Web est diversifiée. Parmi les problèmes qui peuvent survenir lors de l'exécution, et qu'un système lui-même devra faire face à eux d'une manière automatisée: est la concentration sur la phase de développement et de débogage d'un système en ignorant la phase de la configuration en cas de défaillance causée qui nuire un système et qui influence sur son comportement.

La disponibilité des services Web est une question cruciale pour la construction d'un système robuste, alors pour assurer ce critère il faut faire face à La défaillance qui peut se produire entre les serveurs, les réseaux et les baies de stockage ; Cela vise à assurer la communication entre le client et le prestataire en prenant en considération que le prestataire est disponibles et qu'il garantit la qualité de service en tout moment.

Chapitre I :
La sûreté de fonctionnement des systèmes informatiques.

I.1 Introduction

Nous présentons dans cette section les principaux concepts de la sûreté de fonctionnement. Nous introduisons d'abord la notion de sûreté de fonctionnement dans les systèmes informatiques. Ensuite, nous décrivons les différents concepts de base et les principales approches de sûreté de fonctionnement et de tolérance aux fautes. Enfin nous montrons les différentes techniques de tolérance aux fautes que nous pouvons appliquer sur un système réparti.

I.2 Notion de la sûreté de fonctionnement informatique

L'expression sûreté de fonctionnement est également employée pour désigner l'ensemble des moyens permettant la maitrise des causes et des effets, c'est-à-dire l'obtention de produits et de systèmes **sûr de fonctionnement**. Cette discipline couvre de multiple théories et techniques. Dans cette section, parmi plusieurs définitions, nous allons présenter la définition du chercheur Jean Claude Laprie qui a contribué à l'essor théorique de la sûreté de fonctionnement des systèmes informatiques, et aussi la définition de la société d'IBM (International Business Machines) qui présente dans les domaines du matériel informatique, du logiciel et des services informatiques.

I.2.1 Définition1

La sûreté de fonctionnement d'un système informatique est la propriété qui permet de placer une confiance justifiée dans le service qu'il leur délivre. Le service délivré par un système est son comportement tel que perçu par son, ou ses utilisateurs; un utilisateur est un autre système (humain ou physique) qui interagit avec le système considéré [1].

I.2.2 Définition2

La sûreté de fonctionnement consiste à évaluer les risques potentiels, prévoir l'occurrence des défaillances et tenter de minimiser les conséquences de situations catastrophiques lorsqu'elles se présentent [2].

I.3 Concepts de sûreté de fonctionnement informatique

La sûreté de fonctionnement informatique s'articule en trois principaux axes: les attributs qui la caractérisent, les entraves qui empêche sa réalisation et enfin les moyens de l'atteindre. La figure ci-dessous (Figure I. 1) résume les concepts de la sureté de fonctionnement.

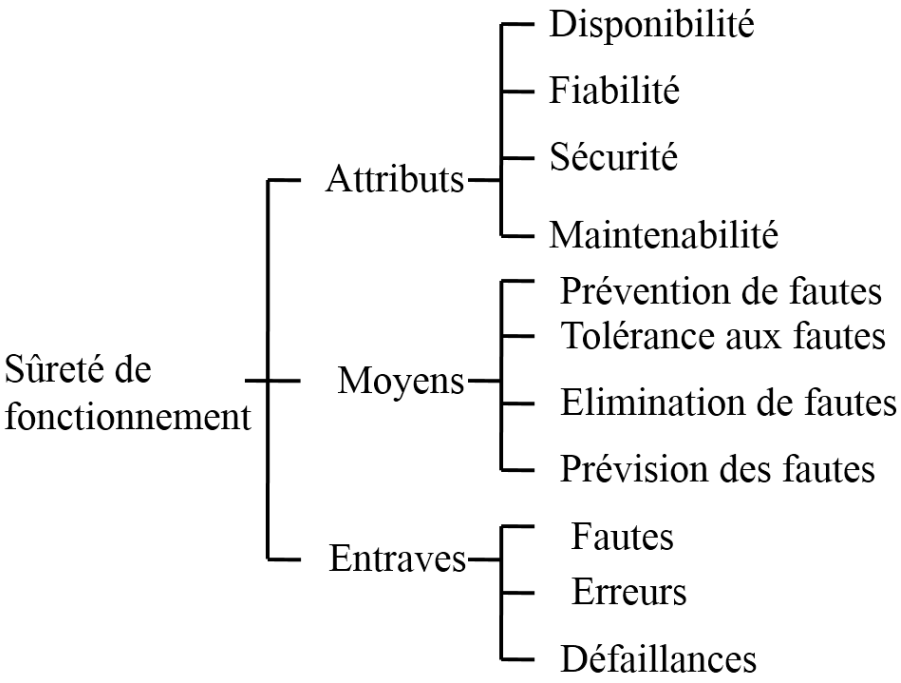


Figure I. 1. Concepts de la sûreté de fonctionnement des systèmes informatiques.

Cette sous-section définie la sûreté de fonctionnement comme une propriété attendue des services des systèmes informatiques et présente quelques techniques de sûreté de fonctionnement et de tolérance aux fautes.

I.3.1 Les attributs de la sûreté de fonctionnement

Les attributs de sûreté de fonctionnement sont définis pour exprimer les propriétés de sûreté de fonctionnement qui sont attendues du système et apprécier la qualité du service à délivrer [1].

- **Disponibilité:** C'est la capacité du système à être prêt de délivrer un service dans des conditions données, à un instant donné ou pendant un intervalle de temps donné.
- **Fiabilité:** Elle est définit par rapport à la continuité de service.
- **Sécurité-innocuité:** Elle est définit par rapport l'écartement des conséquences catastrophiques sur l'environnement du système.
- **Maintenabilité:** Elle mesure la facilité de réparation d'un système défaillant.

I.3.2 Les entraves de la sûreté de fonctionnement

Les entraves à la sûreté de fonctionnement sont les circonstances indésirables mais non inattendues, causes ou résultats de la non-sûreté de fonctionnement.

Une faute peut être définie pouvant provoquer une erreur. Comme une faute est activée, la partie du code affectée est exécutée, une erreur alors est créée. Une erreur est un état (ou partie de l'état) du système susceptible de provoquer une défaillance. Une défaillance du système apparaît lorsque le service rendu par le système ne correspond pas la réalisation de la

fonction du système. C'est une transition d'un service correct vers un service incorrect. On peut aller de la faute à la défaillance comme suit : quand une faute est activée, elle provoque une erreur. En propageant, l'erreur engendre une défaillance. La conséquence de la défaillance est une faute pour le système (Figure I. 2).

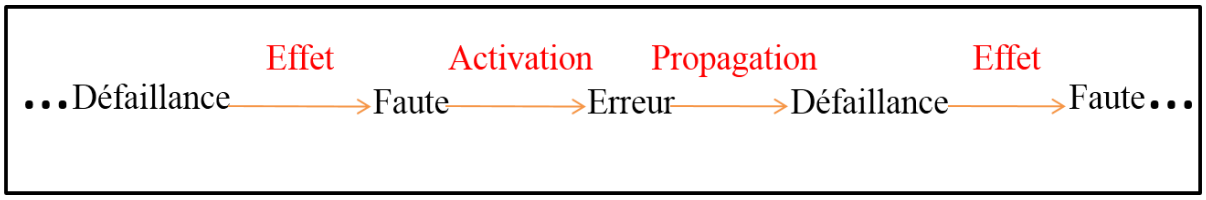


Figure I. 2. La chaîne fondamentale des entraves à la sûreté de fonctionnement (extraite de [2]).

I.3.2.1 Propagation d'erreurs

Une erreur affectant un service interne va se propager à travers la structure du système, en gré de son activité en suivant un ou plusieurs chemins de propagation d'erreurs. Elle va gagner de proche en proche certains services pour finalement s'exprimer en défaillance [3].

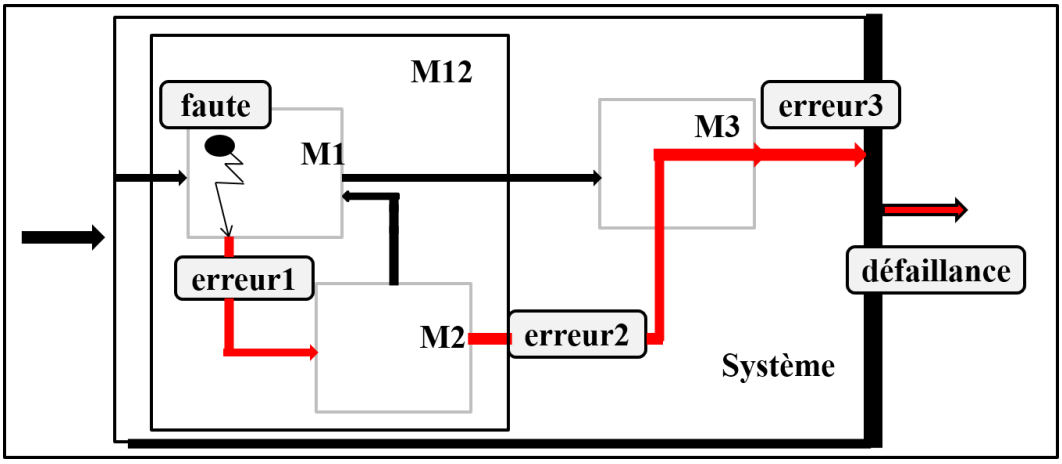


Figure I. 3. De la faute à la défaillance (extraite de [6]).

I.3.3 Les moyens de la sûreté de fonctionnement

Il s'agit des méthodes et techniques permettant de fournir au système l'aptitude à délivrer un service conforme à l'accomplissement de sa fonction, et de donner confiance dans cette aptitude [2]. Les moyens de la sûreté de fonctionnement[4] sont classés en quatre familles:

- **Prévention de fautes:** comment empêcher l'occurrence ou l'introduction des fautes pendant la phase de développement.
- **Tolérance aux fautes:** comment faire un service à même de remplir la fonction du système en dépit des fautes. L'objectif de la tolérance est d'éviter la défaillance en utilisant des techniques de détection ou masquage d'erreurs.
- **Elimination des fautes:** comment réduire le nombre et la sévérité des fautes.
- **Prévision des fautes:** estime la présence, la création et les conséquences des fautes par des tests prenant en compte leurs activités et leurs occurrences.

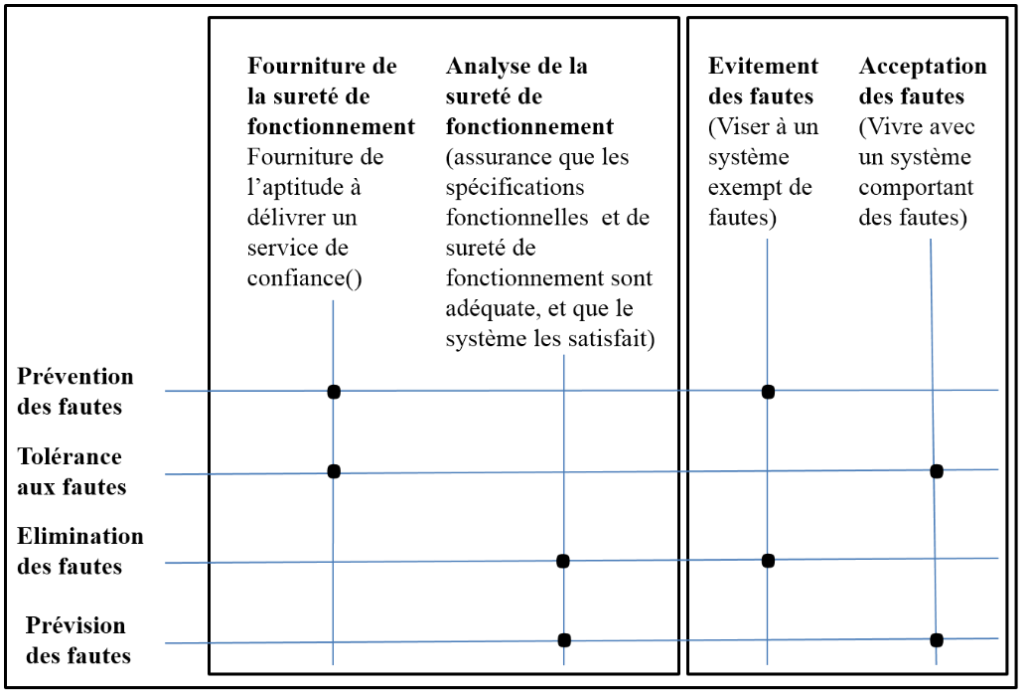


Figure I. 4. Groupement des moyens pour la sûreté de fonctionnement.

La prévention des fautes et la tolérance aux fautes peuvent être vues comme des moyens d'obtention et de la fourniture de la sûreté de fonctionnement (Figure I. 4).L'élimination des fautes et la prévision des fautes peuvent être vues comme constituant les moyens de la validation de la sûreté de fonctionnement (Figure I. 4).

Parmi ces méthodes, la tolérance aux fautes peut être mise en œuvre par le traitement des erreurs et des fautes. Le traitement d'erreur est destiné à éliminer les erreurs, si possible avant qu'une défaillance ne survienne. Le traitement de faute est destiné à éviter qu'une, ou des fautes ne soient activées à nouveau [2].

I.4 La tolérance aux fautes

La tolérance aux fautes vise à éviter les défaillances et est mise en œuvre par la détection des erreurs et le rétablissement du système [4]. Elle fournit un fonctionnement continu des services d'un système en présence des fautes, en évitant leur activation et qu'une erreur n'entraine pas une défaillance. La technique la plus utilisée pour traiter les erreurs consiste à détecter l'existence d'un état incorrect (erreur) puis remplacer l'état incorrect par un état correct (conforme aux spécifications).

La sous-section suivante présente les différents types de moyens utilisés pour détecter l'erreur.

I.4.1 Détection des erreurs

Il y a différents types de moyens de contrôle utilisés pour détecter l'erreur. Parmi ces types, nous citons:

- Redondance;
- Codes détecteurs d'erreurs;
- Contrôle de vraisemblance;
- Contrôle de données structurées;
- Testes en lignes.

La sous-section suivante présente les techniques de contrôle utilisées pour détecter l'erreur.

I.4.1.1 Redondance

Dans cette section, nous allons nous focaliser sur une technique trop utilisée et qui assure la sûreté de fonctionnement: **la redondance**. Son principe est d'ajouter des ressources, de l'information, ou du temps au-delà de ce qui est nécessaire pour le fonctionnement du système. L'objectif ici est de permettre la détection des fautes et la tolérance aux fautes. Il existe quatre formes de redondance:

- Matérielle;
- Logicielle;
- Temporelle;
- D'information.

Nous utilisons dans notre approche (dans le chapitre suivant) la redondance matérielle et redondance logicielle. Nous présentons par la suite ces deux formes de redondances.

I.4.1.1.1 La redondance matérielle

La Redondance Matérielle est la forme la plus répandue de redondance en raison de la diminution de coût du matériel et de la minimisation de la taille des composants. Il existe trois types de techniques:

1) Les techniques passives;
2) Les techniques actives ou dynamiques;
3) Les techniques hybrides.

Dans la redondance passive, certains éléments peuvent être supprimés sans que cela modifie le fonctionnement. Elle sert à masquer les effets internes des fautes par détection et correction ou réplication et vote. L'objectif de la redondance matérielle passive est de masquer les fautes et éviter leur propagation en erreurs [5]. Son mécanisme de base est le vote majoritaire. Un système tolérant aux fautes emploie les éléments redondants fonctionnant en parallèle. À tout moment, toutes les reproductions de chaque élément devraient être dans le même état. Les mêmes données sont fournies à chaque reproduction, et les mêmes sorties sont prévues. Les sorties des reproductions sont comparées utilisant un module de vote. Un système avec deux

reproductions de chaque élément se nomme Dual Modular Redundant (DMR). Le circuit de vote peut alors seulement détecter une erreur et la récupération se fonde sur d'autres méthodes. Un exemple est la redondance modulaire triple (Triple Modular Redundancy ou TMR) (Figure I. 5).

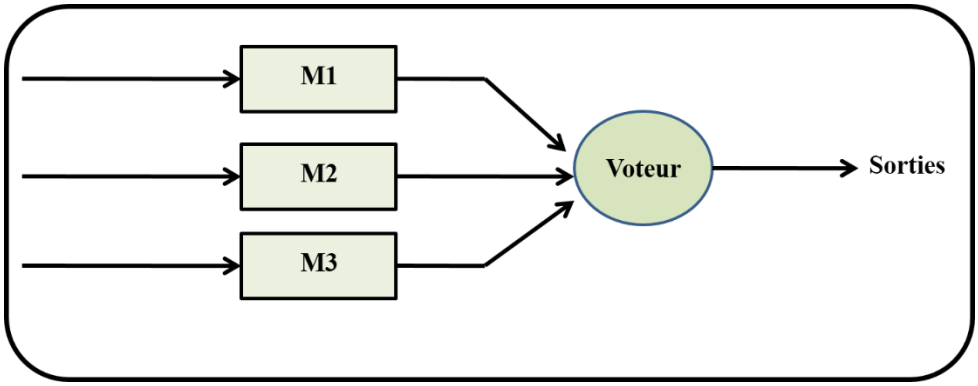


Figure I. 5. TMR (Triple Modular Redundancy).

TMR est une forme de la redondance modulaire multiple (N-modular redundancy), dans laquelle trois systèmes effectuent un processus et ce résultat est traité par un système de vote pour produire un à sortie unique. Si un des trois systèmes échoue, les deux autres systèmes peuvent corriger et masquer la faute. Le circuit de vote peut déterminer quelle réplication est en erreur quant à un vote deux-à-un est observé. Dans ce cas, le circuit de vote peut produire le résultat correct, et jette la version incorrecte. Après ceci, on assume que l'état interne de la reproduction incorrecte est différent de celui des autres deux, et le circuit de vote peut commuter à un mode de DMR (Dual Modular Redundant). Si le voteur échoue alors le système complet échouera. Cependant, dans un bon système TMR, le voteur est beaucoup plus digne de confiance que les autres composants du TMR.

La figure précédente (Figure I. 5) présente le TMR. M_1, M_2 et M_3 sont trois copies du même module. Le voteur permet de produire la donnée finale. Si une réplication de la donnée est en désaccord avec les autres deux, elle est détectée comme échouée. La production de sortie des autres deux est traitée comme correcte.

Nous généralisons le cas de la figure précédente (Figure I. 5) à N modules : Redondance N-modulaire (N-modular redundancy ou NMR). N modules identiques sont utilisés avec N impair.

I.4.1.1.2 La redondance logicielle

La redondance logicielle consiste à écrire plusieurs versions du programme. Elle est réalisée par les techniques suivantes:

I.4.1.1.2.1 N-self-checking programming

Elle consiste à écrire N versions d'un même programme (Figure I. 6). Chaque version possède son propre jeu de test. Une logique de sélection choisit les résultats d'une version ayant passé avec succès l'ensemble de ses tests.

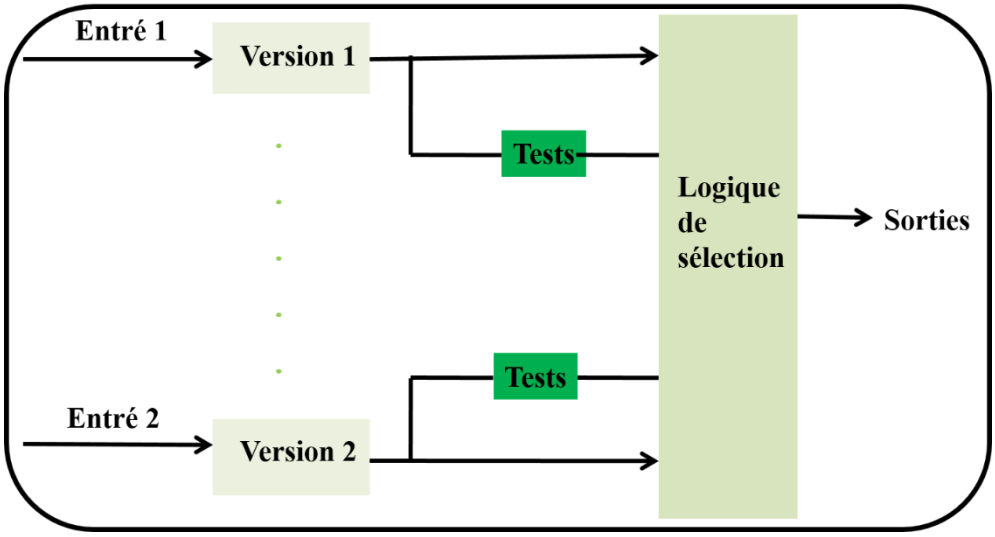


Figure I. 6. N-self-checking programming.

I.4.1.1.2.2 N-Version programming

Elle consiste à concevoir et coder le logiciel N fois et puis effectuer un vote majoritaire sur les N résultats obtenus. Néanmoins, cette approche présente des inconvénients:

- Les concepteurs et programmeurs peuvent faire les mêmes erreurs.
- S'il y a des erreurs dans les spécifications, elles se propagent sur toutes les versions.

I.4.1.2 Codes détecteurs d'erreurs

Les codes détecteurs d'erreurs permet de diagnostiquer qu'une erreur a été commise dans un système mais pas de corriger cette erreur. La détection est basée sur la redondance dans la représentation binaire de l'information soit en ajoutant des bits de contrôle aux données pour la classe de codes séparables, soit en utilisant une nouvelle représentation incluant la redondance pour la classe des codes non séparables [5].

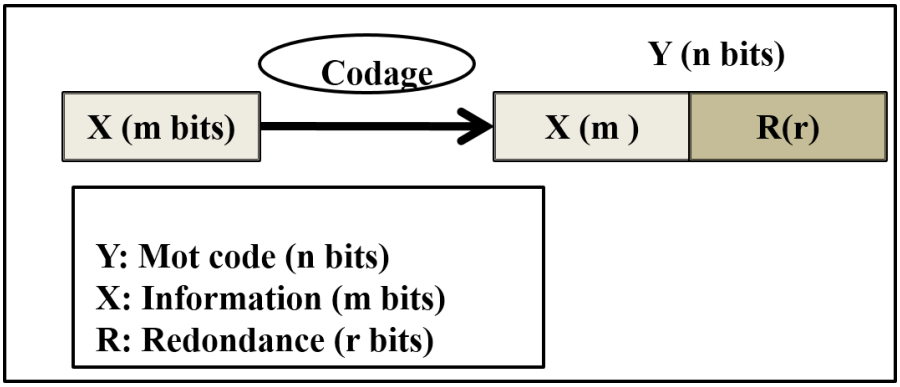


Figure I. 7. Codes séparables.

I.4.1.3 Contrôle de vraisemblance

Les contrôles de vraisemblance peuvent être mis en œuvre par: 1) du matériel spécifique pour détecter des valeurs erronées (instruction illégale, adresse mémoire inexistante) ou des violations de protections de segments mémoire. Et 2) du logiciel spécifique pour vérifier la conformité des entrées ou des sorties du système à des invariants [4].

I.4.1.4 Contrôle de données structurées

Deux types de contrôle peuvent être appliqués aux structures complexes de données d'un système informatique. Les contrôles peuvent porter sur l'intégrité sémantique des données ou sur l'intégrité structurelle de la structure de données. Les contrôles de l'intégrité sémantique consistent à vérifier la cohérence des informations contenues dans la structure de données en utilisant des tests de vraisemblance. Les contrôles structurels sont particulièrement applicables à des données structurées dont les différents éléments sont liés par des pointeurs[4].

I.4.1.5 Tests en ligne

L'objectif principal du test en ligne est la détection des erreurs apparaissant au cours du fonctionnement du système, nous définissons deux classes différentes de techniques de test en ligne [3]:

1) Le test en ligne discontinu: on exploite les redondances temporelles naturelles du système pour le tester de manière discontinue avec une périodicité fixe ou variable. Et
2) Le test en ligne continu: le système est testé en permanence.

I.4.2 Rétablissement du système

Deux formes de rétablissement du système ont été identifiées: la reprise et la poursuite.

I.4.2.1 Reprise

Elle consiste en la sauvegarde périodique de l'état du système de façon à pouvoir après avoir détecté une erreur, ramener le système dans un état antérieur appelé « point de reprise ».

I.4.2.2 Poursuite

Le rétablissement par poursuite constitue une approche alternative ou complémentaire à la reprise après avoir détecté une erreur et après éventuellement tenté une reprise. La poursuite consiste en la recherche d'un nouvel état acceptable pour le système à partir duquel celui-ci pourra fonctionner.

I.5 La tolérance aux fautes pour les systèmes répartis

Un système réparti est une collection de processus indépendants qui apparaît à ces utilisateurs comme un système unique et cohérent. L'objectif principal d'un système réparti est de faciliter l'accès à des ressources distantes pour les utilisateurs (et applications) et de les partager d'une manière contrôlée et efficace. Un autre objectif important d'un système réparti est de cacher le fait que ses processus et les ressources sont physiquement distribués sur plusieurs ordinateurs, c'est la transparence [4].

L'abstraction de processus communicants offre un cadre intéressant pour aborder la tolérance aux fautes, c'est celui de l'algorithmique répartie. La correction des algorithmes est argumentée dans le cadre d'un modèle logique qui est en fait constitué d'au moins deux sous-modèles : un modèle temporel décrivant une connaissance a priori sur les aspects temporels et un modèle de faute définissant les types de fautes pris en compte au niveau de processus et des canaux [4].

I.5.1 Techniques de tolérance aux fautes pour un système réparti

Dans sa forme la plus simple, la tolérance peut se limiter au rétablissement d'un processeur ou nœud de traitement après une défaillance temporaire. Cependant, la continuité de service malgré la défaillance ou l'inaccessibilité de nœuds de traitement nécessite la réplication de processus ou de données sur des nœuds multiples. Nous abordons successivement la reprise locale (d'un seul processus), la reprise répartie (impliquant plusieurs processus), la réplication de processus et la réplication de données.

I.5.1.1 La reprise locale

La forme minimale de tolérance aux fautes dans un système réparti vise à minimiser l'influence négative de la défaillance temporaire d'un processus serveur en facilitant un redémarrage rapide. Cela peut être facilité si le serveur est conçu de façon à ne conserver aucun état entre deux requêtes et que toutes ses opérations soient idempotentes. L'absence d'état interne élimine la nécessité même de le restaurer lors d'une reprise. L'idempotence des opérations du serveur permet à chaque client de pouvoir répéter une requête tant qu'elle n'est pas satisfaite.

I.5.1.2 La reprise répartie

Une reprise répartie correspond au cas où les dépendances induites par la communication interprocessus ont pour effet que la reprise d'un processus conduit à reprendre aussi d'autres

processus ayant interagi avec le premier. Les processus doivent reprendre leur exécution depuis un ensemble de points de reprise qui constituent un état global cohérent. Généralement l'état global cohérent antérieur est l'état initial du système. On distingue généralement trois approches pour la reprise répartie:

- **la création indépendante des points de reprise:** autorise à chaque processus à créer des points de reprise au moment favorables pour lui.
- **la création coordonnée des points de reprise:** permet d'assurer que seules des lignes de reprise cohérentes sont créées. Chaque processus maintient un ou deux points de reprise : un permanent, et un autre temporaire. La transformation de points de reprise temporaires en points permanents s'effectue au moyen d'un protocole à deux phases gérées par le processus qui a déclenché la création coordonnée d'une ligne de reprise.
- **la création de points de reprise induite par la communication:** est un compromis entre les deux approches précédentes.

I.5.1.3 Processus répliqués

Un service tolérant aux fautes peut être mis en œuvre en coordonnant un groupe de processus répliqués sur deux ou plusieurs processeurs différents. On distingue généralement trois stratégies: la réplication passive, active et semi-active.

Avec la réplication passive, une seule copie (la copie primaire) traite tous les messages reçus, met à jour son état interne et effectue l'envoi de messages de sortie. La copie primaire met à jour régulièrement une copie de son état interne, qui constitue ainsi un point de reprise. Lorsque la copie primaire défaille, une des copies de secours est élue pour prendre sa place.

La réplication active traite toutes les copies sur un pied d'égalité: chacune traite tous les messages reçus, met à jour son état interne de façon autonome, et génère les messages de sortie. Les messages de sortie sont choisis au moyen d'une fonction de décision Pour des arrêts simples.

La réplication semi-active est similaire à la réplication active dans le sens que toutes les copies reçoivent les messages d'entrée et peuvent ainsi les traiter. Cependant, comme pour la réplication passive, le traitement est asymétrique car une copie privilégiée (la meneuse) assume la responsabilité de certaines décisions.

I.5.1.4 Données répliquées

La réplication des données sert à améliorer à la fois la disponibilité de données et la performance des opérations de lecture. D'une part, une donnée dupliquée peut être accédée même si certaines copies résident sur des nœuds défaillants ou inaccessibles. D'autre part, il est plus rapide de lire une copie proche qu'une copie distante. Les protocoles de gestion de données dupliquées sont qualifiés d'optimistes ou de pessimistes selon s'ils négligent ou non la probabilité des conflits d'accès en cas de partitionnement des copies

Chapitre II :

Les services Web et les architectures SOA.

II.1 Introduction

Les services Web sont, à l'heure actuelle, le moyen le plus utilisé pour mettre en place des architectures orientées services. Dans la première sous-section, nous allons nous définir la notion du service. Ensuite dans la deuxième sous-section, nous allons nous parler un peu sur l'aspect fonctionnel et technique des services Web, nous étudierons les différents protocoles de base permettant de mettre en place une architecture à base de Service Web. Enfin, dans la troisième sous-section, nous allons nous introduire les concepts des architectures orientées services et la relation de services et les rôles de clients et de prestataires joués par les applications participantes. Il est important de noter que nous avons choisi le terme « prestataire » pour marquer une différence avec la terminologie des architectures client/serveur, qui ne sont qu'une forme spécifique et limitée des architectures client/prestataire [7].

II.2 Les services Web

La technologie des services Web [8] est un moyen rapide de distribution de l'information entre clients, fournisseurs, partenaires commerciaux et leurs différentes plates-formes. Les services Web sont basés sur le modèle SOA. Pour ce faire, Ils s'appuient sur un ensemble de protocoles internet très répandus, afin de communiquer. Cette communication est basée sur le principe des demandes et réponses effectuées avec les messages XML.

Les services Web sont décrits par des documents WSDL (Web Service Description Language), qui précise les opérations pouvant être invoquées, leurs signature est le point d'accès du service. Les services Web sont accessibles via SOAP, les requêtes et les réponses sont des messages XML transportés sur HTTP.

II.2.1 Les protocoles de base des services Web

La pile des technologies de services Web commence à proprement parler avec les protocoles de base : WSDL et SOAP. Ces protocoles imposent un format des messages XML. WSDL (Web Service Description Language) est le langage de description des services Web, même s'il n'est pas formellement imposé par l'architecture de référence du W3C. On peut cependant considérer aujourd'hui qu'une description WSDL est nécessaire pour qu'une application puisse revendiquer la qualification de service Web. SOAP (Simple Object Access Protocol) est, quant à lui, le protocole standard d'interaction, d'échange d'informations entre un client et un prestataire.

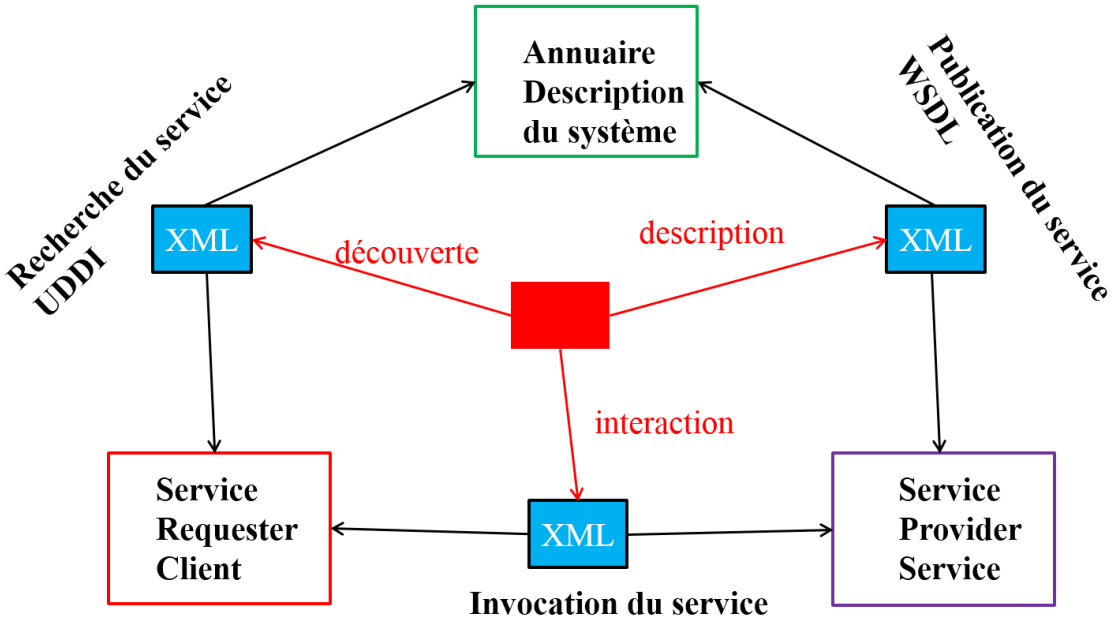


Figure II. 1. Les protocoles de base des services Web.

II.2.1.1 XML

XML (Extensible Markup Language) est un langage de balisage extensible [9]qui a été mis au point par le XML Working Group sous l'égide du World Wide Web Consortium (W3C). XML est un standard qui sert de base pour créer des langages balisés spécialisés, c'est un « méta langage ». Il est suffisamment général pour que les langages basés sur XML, puissent être utilisés pour décrire toutes sortes de données et de textes. Il s'agit donc partiellement d'un format de données. Son objectif est, dans un échange entre systèmes informatiques, de transférer, en même temps, des données et leurs structures. Permettant de coder n'importe quel type de donnée, depuis l'échange EDI (Electronic Data Interchange ou Echange de Données Informatisées) [10] jusqu'aux documents les plus complexes, son potentiel est de devenir le standard universel et multilingue d'échange d'informations.

II.2.1.2 WSDL

WSDL (Web Service Description Language)[11] est l'outil pivot de la technologie des services Web car il permet véritablement de donner une description d'un Service Web indépendante de sa technologie d'implémentation. Dans la mise en place des architectures de services Web aujourd'hui, la fonction de contrat de service est portée par le document WSDL.

II.2.1.3 SOAP

SOAP (Simple Object Access Protocol)[13]–[17] fournit un mécanisme qui permet d'échanger de l'information structurée et typée entre applications dans un environnement réparti et décentralisé.

SOAP ne véhicule pas de modèle de programmation ou d'implémentation, mais fournit les outils nécessaires pour définir des modèles opérationnels d'échange (styles d'échange) aussi

diversifiés que les systèmes de messagerie asynchrone et l'appel de procédure distante (RPC). Le message SOAP est un document XML. Un message SOAP présente une structure normalisée.

II.2.1.4 UDDI

UDDI (Universal Description Discovery and Integration) [18] est le support d'un système réparti d'annuaires répliqués qui permettent la publication et la découverte de services sur Internet.

Un annuaire UDDI est accessible par l'intermédiaire du protocole SOAP. L'API UDDI est un service Web décrit au format WSDL qui permet d'accéder à un annuaire via l'utilisation du protocole SOAP. L'annuaire UDDI est accessible soit via un navigateur Web qui dialogue avec une application Web dédiée, soit par un programme, en utilisant l'API (Application Programming Interface) définie par la spécification.

L'API comporte deux groupes de fonctions,

1) les fonctions de recherche (Inquiry API): navigation, recherche et consultation des informations de l'annuaire;
2) les fonctions de publication (Publishing API)**:** publication, création, modification ou suppression des informations de l'annuaire.

II.3 Les Architectures Orientées Services

L'architecture orientée service (Service Oriented Architecture ou SOA) [7] est le terme utilisé pour désigner un modèle d'architecture pour l'exécution d'applications logicielles réparties.

Ce modèle d'architecture prend forme au cours de l'activité pluriannuelle de spécification des architectures de systèmes répartis, développée dans des contextes aussi variés que ceux de: l'Open Group (Distributed Computing Environment ou DCE); l'Object Management Group (Object Management Architecture/ Common Object Request Broker Architecture ou OMA/ CORBA); et l'éditeur de logiciel Microsoft (Distributed Component Object Model ou DCOM).

Les deux derniers modèles (CORBA et DCOM) relèvent l'architecture par composants logiciels répartis plutôt que l'architecture orientée services, et le terme «service» est généralement absent de leur terminologie.

De plus, un des avantages des SOA réside dans le fait que les applications réparties n'ont plus besoin d'un système de middleware réparti commun pour communiquer, mais seulement des protocoles et des technologies de communications interopérables sur Internet.

II.3.1 Qu'est-ce qu'un service

La notion de service est le produit d'une démarche d'abstraction par rapport au logiciel qui l'implémente, au processus qui l'exécute et au port qui le localise. Cependant, le service reste un objet très concret et technique.

La réalisation d'un service passe par des messages échangés, des transitions d'état et des actions que l'application cliente suppose assurés de la part de l'application prestataire du service.

Les caractéristiques fonctionnelles, opérationnelles et d'interface d'un service sont consignées dans un contrat. Un Service Web n'est donc rien d'autre que la réalisation supposée de la prestation définie dans le contrat de service.

Tout simplement, un service est un ensemble de fonctions défini par une interface, "contrat" entre le fournisseur et l'utilisateur du service [4].

II.3.2 Définition du l'architecture orientée services

Le concept SOA est une évolution de l'informatique distribuée, conçue pour permettre l'interaction de composants logiciels appelés "services" sur l'ensemble d'un réseau. Les applications sont créées à partir d'une combinaison de ces services, qui peuvent être partagés entre plusieurs applications [16].

L'architecture orientée services (Service Oriented Architecture, ou SOA) est une forme d'architecture de médiation qui est un modèle d'interaction applicative qui met en œuvre des services (composants logiciels) :

- avec une forte cohérence interne (par l'utilisation d'un format d'échange pivot, le plus souvent XML),
- et des couplages externes faibles (par l'utilisation d'une couche d'interface interopérable, le plus souvent un service web).

Le service est une action exécutée par un « prestataire » à l'attention d'un « client », cependant l'interaction entre prestataire et client est faite par le biais d'un médiateur responsable de la mise en relation des composants.

L'objectif d'une SOA est donc principalement de décomposer une fonctionnalité en un ensemble de fonctions basiques, appelées services, fournies par des composants et de décrire finement le schéma d'interaction entre ces services. L'idée sous-jacente est de cesser de construire la vie d'une architecture autour d'applications classiques mais plutôt faire en sorte de construire une architecture logicielle globale décomposée en services correspondant aux processus métiers gérant cette application [17].

La SOA est un moyen étendu de réutiliser les modèles existants et de les transformer en des services plus agiles [18]. La base d'une SOA repose sur des services répondant, notamment, aux critères:

- **Faiblement couplés:** Les applications traditionnelles incluent dans leur code les données métiers. Elles sont complètement liées aux systèmes pour lesquels elles ont été conçues. Toute demande de modification, qu'elle concerne l'accès aux modèles, les règles de gestion de leur simulation ou celles de présentation des résultats de la

simulation, nécessitera une refonte du code entraînant coûts, délais, compétences spécifiques et surtout manque de réactivité et risque de déstabilisation des applications.

- **Distribués:** Les modèles et les services qui composent l'application peuvent être physiquement répartis sur différents systèmes dans l'environnement de simulation, mais aussi au-delà. Ils permettent une approche distribuée, à l'opposé des systèmes traditionnels centralisés. Là où l'environnement devait maintenir certaines fonctions et données comme les tarifs des fournisseurs de modèles, il pourra interroger directement, via son applicatif, le service du fournisseur de modèles sans se préoccuper de sa mise à jour.
- **Invocables et publiables:** Les services doivent être invocables et publiables quels que soient les systèmes utilisés.
- **Orientés métier:** Les services permettent de gérer l'application avec une approche fonctionnelle, par l'intermédiaire de processus métiers intégrés à l'architecture, permettant de la piloter sans développement conséquent.

II.3.3 Quelle différence entre une architecture orientée objet et une SOA ?

Nous présentons les deux architectures pour montrer l'intérêt de la SOA.

II.3.3.1 Modèle orienté objets (POO)

La Figure II. 2 présente une architecture à trois couches classiques avec un modèle objet:

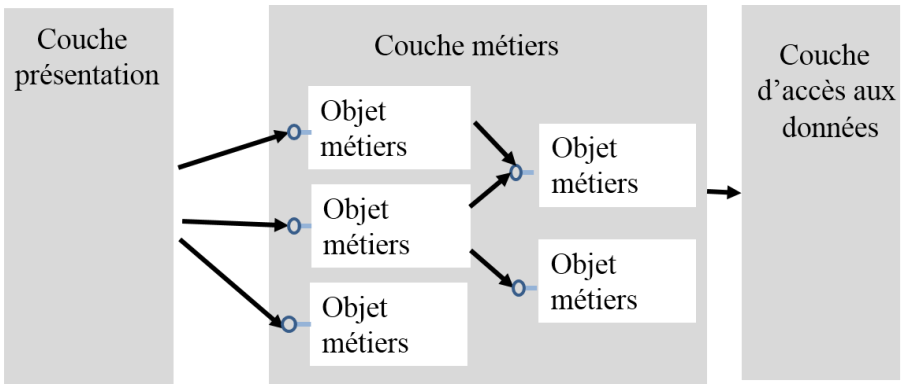


Figure II. 2. Modèle orienté objets.

Nous remarquons tout de suite le nombre de liens entre la couche présentation et les objets métiers. Le code client doit jongler directement avec le modèle objet de la couche métier, ce qui a pour conséquence de lier celle-ci très fortement à un modèle spécifique et requiert un nombre d'appels important entre les deux couches.

II.3.3.2 Modèle orienté services (POO)

La Figure II. 3 présente une architecture orientée services qui reposerait sur les mêmes objets métiers :

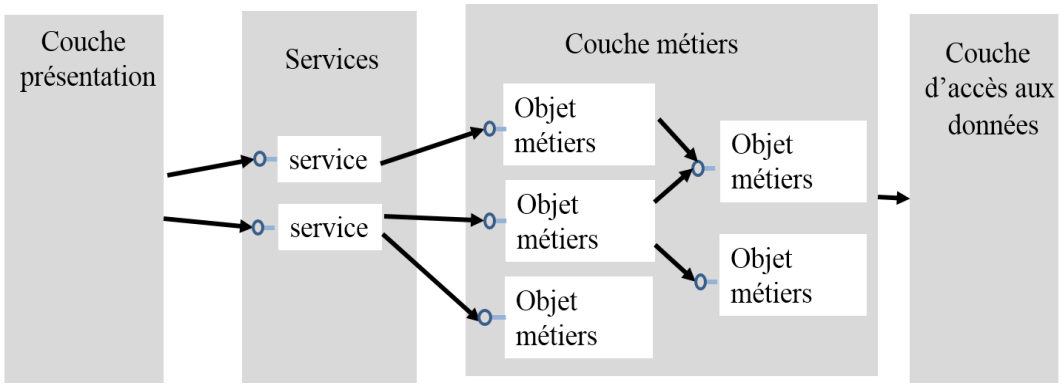


Figure II. 3. Modèle orienté services.

Nous remarquons qu'il est introduit un niveau d'indirection supplémentaire sous la forme de services. La couche présentation ne manipule plus directement les objets métiers, mais passe par des services. Les objets métiers se trouvent dans des bibliothèques de classes directement chargées dans le même processus que les services, le coût des appels aux objets métiers est alors très faible. Les services agissent comme des boites noires faisant abstraction de la complexité du modèle objet, présentant un ensemble de fonctionnalités restreint et permettant de réduire les échanges entre les couches.

II.3.4 Le contrat du service

Le contrat de service du modèle des SOA s'inspire directement du modèle des contrats professionnels de service. Il s'agit d'un document qui développe l'ensemble des points permettant de décrire et donc de définir la relation de service.

Dans le monde des SOA, les éléments du contrat de service sont organisés en six thèmes majeurs: l'identification des parties, la description des fonctions du service, la description de l'interface du service, la description de la qualité de service, la description du cycle de vie du service et du contrat et la description des termes de l'échange.

II.3.5 Conception d'architecture orientée services

Une architecture orientée services peut être conçue par une approche incrémentale, résultat de la combinaison de deux démarches de base:

1) L'agrégation du service;
2) la dissémination de services.

II.3.5.1 Approche par agrégation de services

L'agrégation de services se fait par la mise en œuvre de processus métier à l'aide d'outils tel que BPEL4WS[19]. Il s'agit ici d'établir la coopération entre les services agrégés, à savoir : la division du travail et l'échange d'informations nécessaires pour que les activités de ces services agrégés contribuent efficacement à la réalisation du service agrégeant. Il est également nécessaire de coordonner les services agrégés en synchronisant les étapes de leurs activités.

L'implémentation du service agrégeant organise la coopération et coordonne la réalisation des services agrégés, en faisant éventuellement appel à des services spécialisés de coordination avec des protocoles tels que WS-Coordination [20] et WS-Transaction [21].

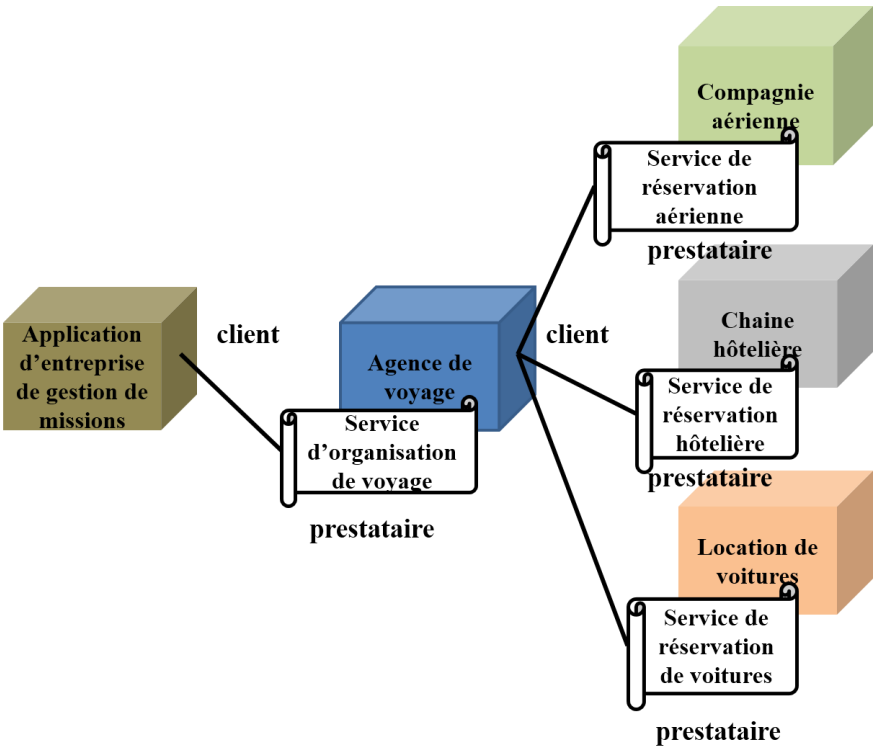


Figure II. 4. Agrégation de services.

II.3.5.2 Approche de dissémination de services

La dissémination de services permet d'effectuer la décentralisation des données. Comme on peut le voir sur la figure (Figure II. 5), plusieurs applications sont prestataires du même service sur des données différentes. Une entreprise peut donc implémenter un ensemble de services modulaires à partir d'applications différentes. Le point essentiel est que le client d'un des services issus d'une démarche de dissémination bénéficie de la modularité du service sans que la question de la modularité de l'implémentation ne soit même posée. En informatique, on parle traditionnellement, à ce propos, de « boîte noire » et d'information « hiding ».

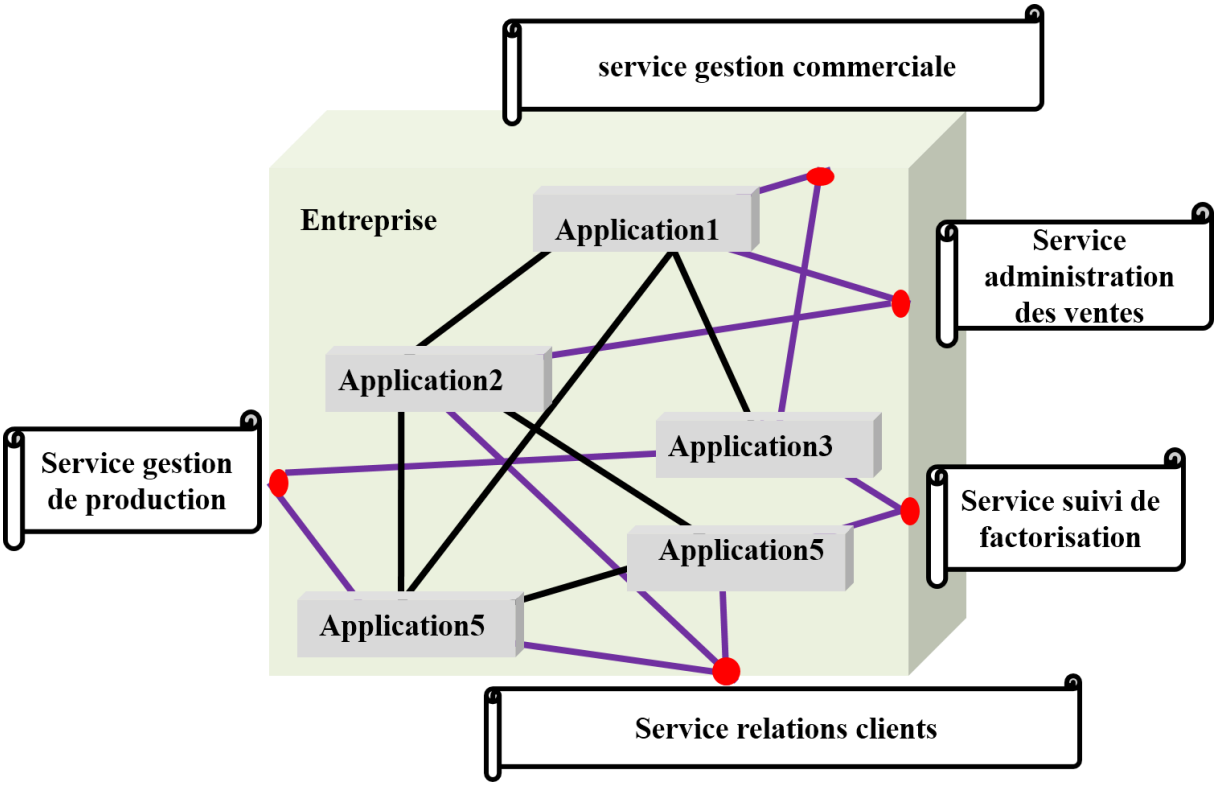


Figure II. 5. Dissémination de services.

II.3.6 Les architectures dynamiques

Dans une architecture dynamique, les services qui la composent, les applications prestataires qui interviennent, ainsi qu'un certain nombre de propriétés opérationnelles des prestations de services ne sont pas définies avant sa mise en place, mais sont composés, configurés, établis, voire négociés, au moment de l'exécution.

Les avantages en termes de qualité de service, de configuration dynamique, de facilité de maintenance et d'évolution des architectures faiblement couplées sont évidents. En revanche, leurs conceptions et leurs mises en œuvre sont beaucoup plus complexes que celles des architectures fortement couplées. En effet, les SOA possèdent également beaucoup d'inconvénients du fait de leur complexité de mise en œuvre. L'aspect « boite noire » d'un service pose beaucoup de problèmes pour caractériser sa sûreté de fonctionnement. L'intégrateur de service ne connaît pas du tout l'implémentation du service auquel il accède, il ne fait que se connecter à un port derrière lequel se cache une prestation supposée définie par le contrat associé.

II.3.7 Les architectures « boites noires »

Dans une architecture orientée services, les relations de service qui lient les applications participantes sont régies par des contrats de service. Les prestations, issues des contrats, doivent impérativement être décrites au niveau fonctionnel, et il est incorrect d'inclure les modèles d'implémentation des applications prestataires dans les contrats de service.

L'implémentation de l'application prestataire est donc une boîte noire par rapport au contrat de service, sauf pour ce qui touche l'implémentation de la communication.

II.4 Travaux sur la sûreté de fonctionnement des services Web

II.4.1 Caractérisations des services Web

Pour analyser la sûreté de fonctionnement des services Web, des expériences sont mises en place avec le monde réel des services Web, WSsDAT (Web Services Dependability Assessment Tool) est un outil qui est développé pour évaluer la sûreté de fonctionnement des services Web. L'outil peut contrôler dans une durée un certain nombre de services Web et de générer des métriques de résultats de contrôle afin de présenter les caractéristiques de sûreté de fonctionnement des services.

Nous décrivons brièvement une expérience[22] sur deux services Web, couramment utilisé dans la recherche bio-informatique, qui offrent des fonctionnalités similaires. Dans l'expérience, l'outil WSsDAT est utilisé pour contrôler les services Web à partir de trois emplacements simultanément. Voici la liste des deux services Web:

1) EBI Web Service, déployé par l'Institut européen de bio-informatique (EBI), Cambridge, Royaume-Uni [41].
2) DDBJ service Web, hébergé par la Banque de données génétiques, le Japon.

Deux outils WSsDAT ont été localisés à Newcastle en Royaume-Uni: le premier a été déployé à partir du réseau de campus à l'Université de Newcastle, tandis que l'autre a été hébergé sur un ordinateur connecté à haut débit avec 1 Mo via prestataire de services local (Telewest Broadband Royaume-Uni). Les outils WSsDAT restants ont été déployés à CERNET (the China Education and Research Network) en Chine.

Afin d'observer les variations de la sûreté de fonctionnement et les mesures de performance sur des différentes périodes les deux Web services ont été contrôlés dans une durée de 72 heures. Toutes les mesures ont été stockées dans une base de données.

Pendant l'expérience, le service Web EBI est comporté d'une manière irrégulière. Ci-dessous un rapport sur les résultats recueillis concernant le service:

	Temps de réponse moyen (s)	Taux de défaillance(%)
Telewest Broadband	842.1	58.3
Réseau de campus Université de Newcastle	764.6	62.9
CERNET (chine)	945.7	43.2

Tableau II. 1 . Résultats d'évaluation d'EBI.

La sûreté de fonctionnement du service DDBJ était très bonne lors de l'expérience, sans défaillance enregistrée. Il y avait deux retards enregistrés à chacune des trois racines, en indiquant les états inconnus du service ou une partie du réseau.

	Temps de réponse moyen (s)	Taux de défaillance(%)
Telewest Broadband	103	0
Réseau de campus Université de Newcastle	97	0
CERNET (chine)	130	0

Tableau II. 2 Résultats d'évaluation d'DDBJ.

Ces deux tableaux représentent les résultats de l'expérience qui montre que la sûreté de fonctionnement d'un service Web peut varier considérablement. Cette conclusion empirique peut être généralisée pour tous les services Web et les architecture SOA, où la sûreté de fonctionnement des services est tous différents selon le point de vue de l'utilisateur. Alors en basant sur les caractéristiques de sûreté de fonctionnement pour les services d'EBI et DDBJ, le service de DDBJ devrait être le premier choix pour les utilisateurs. En outre, le fait de déployer et d'utiliser les services Web dans des différents emplacements peut donner une idée comment le réseau peut affecter la sûreté de fonctionnement et la performance des services Web, en faisant un contrôle de sûreté de fonctionnement du côté de client sur les emplacements du service Web.

II.4.2 Mécanismes de sûreté de fonctionnement sur les services Web

Parallèlement aux recherches sur la caractérisation et la qualité de services, la mise en place de mécanismes sûrs de fonctionnement est un aspect prépondérant dans la bonne évolution des services Web. Elle constitue à l'heure actuelle un frein à leurs développements. De gros efforts de recherche sont effectués dans ce sens-là.

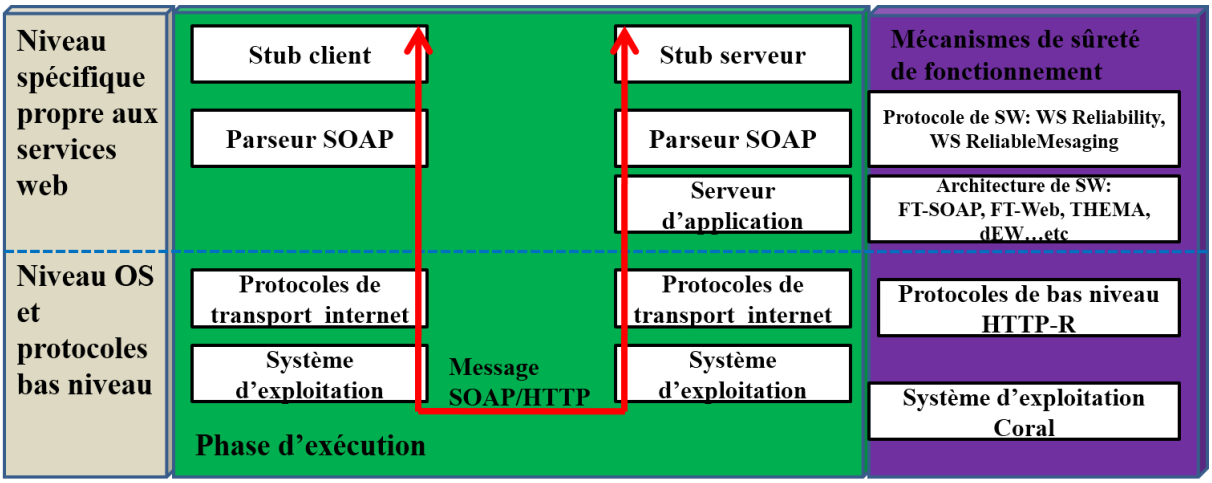


Figure II. 6. Mécanismes de sûreté de fonctionnement dans les services Web.

La Figure (Figure II. 6) présente à quel niveau se place les différents travaux de sûreté de fonctionnement que nous présentons ci-dessous. Comme on peut le voir, ces travaux peuvent être classifiés en deux catégories:

1) les travaux génériques au niveau OS et protocole bas niveau qui permettent de renforcer la fiabilité des couches basses de communication. Ces travaux ne sont pas nécessairement spécifiques aux SOA mais peuvent être parfaitement agrégés à ce type d'application.
2) les travaux spécifiques propres aux Services Web. Ces recherches ont permis la création de plates-formes spécifiques pour la mise en place de certains types de recouvrement ainsi que de nouvelles extensions aux protocoles SOAP pour assurer un niveau de sûreté de fonctionnement plus élevé.

La section qui suit parcourt toutes les couches fonctionnelles présentées dans la Figure (Figure II. 6) et expose les travaux de recherche qui s'y trouvent.

Le modèle de la plate-forme **CORAL** (Connection Replication and Application-Level logging) proposé dans [23] effectue des changements sur le noyau du système d'exploitation (linux) et du serveur Web pour fournir des mécanismes de tolérance aux fautes qui soient transparents pour le client. Les mécanismes implémentés ici sont la redondance active et le « log » de messages.

Le protocole **HTTP-R** [24] est un projet ayant pour but de garantir la fiabilité de l'échange entre un client et un prestataire. En effet, HTTP ne prévoit pas de mécanisme permettant de s'assurer qu'un message est bien arrivé à destination. De fait, avec HTTP, des paquets d'informations peuvent être perdus sans qu'il soit donné suite à cette défaillance. Ces pertes sont sans gravité quand il s'agit simplement de délivrer à un utilisateur une page d'information. En revanche, une telle défaillance peut avoir des conséquences fâcheuses si l'émetteur est une application qui attend une confirmation pour valider une transaction, ce qui suppose d'en avoir conservé une copie quelque part; en outre, cette réexpédition doit, par exemple, respecter un créneau horaire... Pour garantir une telle intégrité, un émetteur et récepteur doivent être informés avec précision du statut (de l'état) du traitement. Utiliser le Web comme un canal d'échange de messages entre applications demande donc de lui apporter certains mécanismes. Telle est la vocation de HTTP-R qui intègre, dans l'entête des messages, beaucoup plus d'informations de statut que ne le fait HTTP. En s'appuyant sur ce protocole, les gestionnaires de messages et les différents agents chargés de stocker le statut des messages pourront appliquer des règles afin de réagir aux éventuelles défaillances, qu'elles proviennent d'un serveur ou d'un réseau.

L'architecture **DeW** (A Dependable Web Service Framework) [25] a été proposée pour améliorer la sûreté de fonctionnement des Services Web. Celle-ci peut être comprise comme registre contenant l'adresse des différentes copies d'un Service Web. Ce registre garantit l'indépendance de localisation physique. Ceci permet à une application basée sur les Web Service de continuer à s'exécuter tant que la référence d'une copie disponible est accessible

par le registre. Cette architecture permet la continuation d'une opération en présence de fautes affectant les Services Web.

FT-SOAP [26] propose une plate-forme permettant à un prestataire d'effectuer une redondance passive sur un groupe de Services Web répliqués. L'utilisation d'intercepteurs dans la couche SOAP du client permet la redirection des requêtes vers les répliques en cas d'une défaillance du primaire. Sur le serveur, des intercepteurs s'ajoutent sur les composants pour effectuer le log, la détection des fautes et la gestion des répliques.

FTWeb [27] est une infrastructure capable de réaliser la réplication active entre les clients et les prestataires. Ce projet s'appuie sur un algorithme d'ordonnancement pour effectuer l'ordonnancement des messages et assurer la cohérence des répliques.

Thema [28] est un middleware pour Services Web capable de tolérer les fautes byzantines. Une couche applicative est insérée sur le client et dans chaque prestataire. Cette couche est une librairie permettant d'introduire le canal de communication adéquat et les politiques de consensus. Les problèmes de consensus sont également abordés dans [29].

WS-Reliability [30], créé par Sun, est le concurrent direct de HTTP-R. L'objectif de WSReliability est la prise en compte de la fiabilité de l'échange au niveau message. Le constat de départ est que la simple liaison SOAP/HTTP n'est pas suffisante lorsqu'un protocole de message au niveau applicatif doit aussi se conformer à des contraintes de sécurité et de fiabilité. Ainsi, WS-Reliability garantit la livraison, la non duplication et l'ordonnancement des messages. **WSReliableMessaging** [31] de Microsoft est une spécification analogue qui permet de garantir les mêmes propriétés que WS-Reliability.

WSCAL (Web Service Composition Action Language) est un langage de composition de service basé sur XML qui a été conçu pour la tolérance aux fautes et notamment le rétablissement par poursuite (Forward Error Recovery).Le rétablissement par poursuite s'appuie sur les mécanismes d'exception de chaque service Web. Le langage WSCAL crée un coordinateur possédant un arbre d'exception. Le coordinateur englobe tous les Services Web de la composition. C'est un Service Web qui va jouer le rôle de proxy (d'intermédiaire) entre le client et les différents services de la composition. Dans le cas d'une levée d'exception sur un service, il sera capable, grâce à l'arbre d'exceptions, de déterminer si cette exception a un impact sur les autres services de la composition ou non. Si c'est le cas, il pourra effectuer des transactions compensatoires ou autres actions de recouvrement, sur les services impactés.

II.5 Conclusion

En conclusion, les Services Web sont le résultat d'une évolution technologique majeure, et les bases d'une révolution organisationnelle et économique. Le changement est considérable : les Services Web vont modifier en profondeur l'organisation du travail et les processus métier des organisations (entreprises, administrations, associations). En outre, ils vont changer les pratiques des professionnels de l'informatique.

En fait, au cœur de l'évolution technologique portée par les Services Web, il y a la montée en puissance du concept de service et l'architecture orientée services.

SOAP, WSDL et UDDI sont des protocoles de base qui constituent une infrastructure de base qui permet la mise en œuvre de services et d'architectures orientées services. Cette base est en revanche insuffisante pour l'automatisation des processus métier critiques, qui est à terme, la cible des technologies de services Web.

Les nombreux travaux de recherche que nous avons exposé dans ce chapitre confirment la jeunesse des Services Web et tente de ce fait de renforcer la sûreté de fonctionnement de cette technologie.

Chapitre III:
Système décisionnel distribué pour la tolérance aux fautes des services web SOA.

III.1 Introduction

Parmi les propriétés attendues des services Web est la sûreté de fonctionnement. Malheureusement, les mesures retournées à partir des services Web sont inévitablement soumises aux erreurs. Une faute peut être vue comme une valeur arbitraire d'une mesure incohérente d'un service Web qui ne peut pas compensé systématiquement. L'une des techniques de la sûreté de fonctionnement que traite ce mémoire est la tolérance aux fautes. Elle peut être définit comme la : « méthode qui permet à un système de remplir ses fonctions en dépit des fautes pouvant affecter ses composants, sa conception, ou ses interaction avec autres systèmes» [32]. Quelles que soit les précautions prises, l'occurrence de fautes est inévitable. Cependant, il faut assurer la fourniture du service malgré l'occurrence des fautes [4].

L'approche que nous souhaitons développer est basée sur deux idées:

1) Concevoir un système de décision distribué tolérant aux fautes en utilisant la technique de redondance passive.
2) Comparer les résultats des différents services Web en utilisant des techniques statistiques non paramétriques sur les résultats obtenues pour identifier les mesures qui ne sont pas corrigeables retournées à partir les services Web.

Nous traitons dans la section suivante l'architecture générale de notre système et la solution proposé qui base sur la tolérance aux fautes pour des données retournées par les services web à partir des capteurs environnementaux simulés.

III.2 Notre cas d'étude : stations de captage de température

L'objectif de notre approche est de concevoir un système de décision tolérant aux fautes à partir les données retournées de services web. Nous considérons pour cela des nœuds qui sont des stations à lequel sont connectées des capteurs environnementaux de température et des modules de décision qui permettent de traiter les données retournées à partir des services Web.

Nous avons N nœuds qui sont des stations et qui contiennent N services Web (N WSDLs identiques ou équivalents) pour observer une hypothèse inconnue. Chaque station transmet ses résultats des services Web à un nœud principal qui contient un module qui permet de comparer les valeurs pour chaque opération de détection par les stations et pour chaque mesure de des services Web. Enfin, nous réalisons l'analyse statistique sur les données pour chaque station. Si les valeurs obtenues pour un service Web n'appartiennent pas à un intervalle de confiance calculé, ce service Web est considéré comme défectueux. Les services Web voisins sont susceptibles d'avoir des résultats semblables.

Les résultats retournés sont combinées par le nœud principal pour permettre la prise de décision globale pour le phénomène étudié. Pour réaliser la tolérance aux fautes, nous proposons un schéma de détection dans lequel nous utilisons la technique de redondance

logicielle passive avec l'utilisation d'un module qui analyse les données provenant des services Web.

A partir des résultats qui viennent des stations, le nœud principal peut détecter les services Web défectueux en faisant une analyse statistique sur les données retournées et prendre une décision binaire concernant l'évènement étudié.

Nous voulons que la décision finale prise par notre système soit fiable malgré la présence des défaillances au niveau de quelques services Web de notre système. La décision finale pour chaque type d'observation n'est pas prise à partir des résultats d'un seul capteur. Les capteurs qui donnent des résultats non-conformes aux résultats des autres capteurs sont considérées défectueux et leurs résultats sont éliminés dans l'analyse : donc nous avons une correction des fautes qui peuvent influencer sur les résultats finaux du nœud principal.

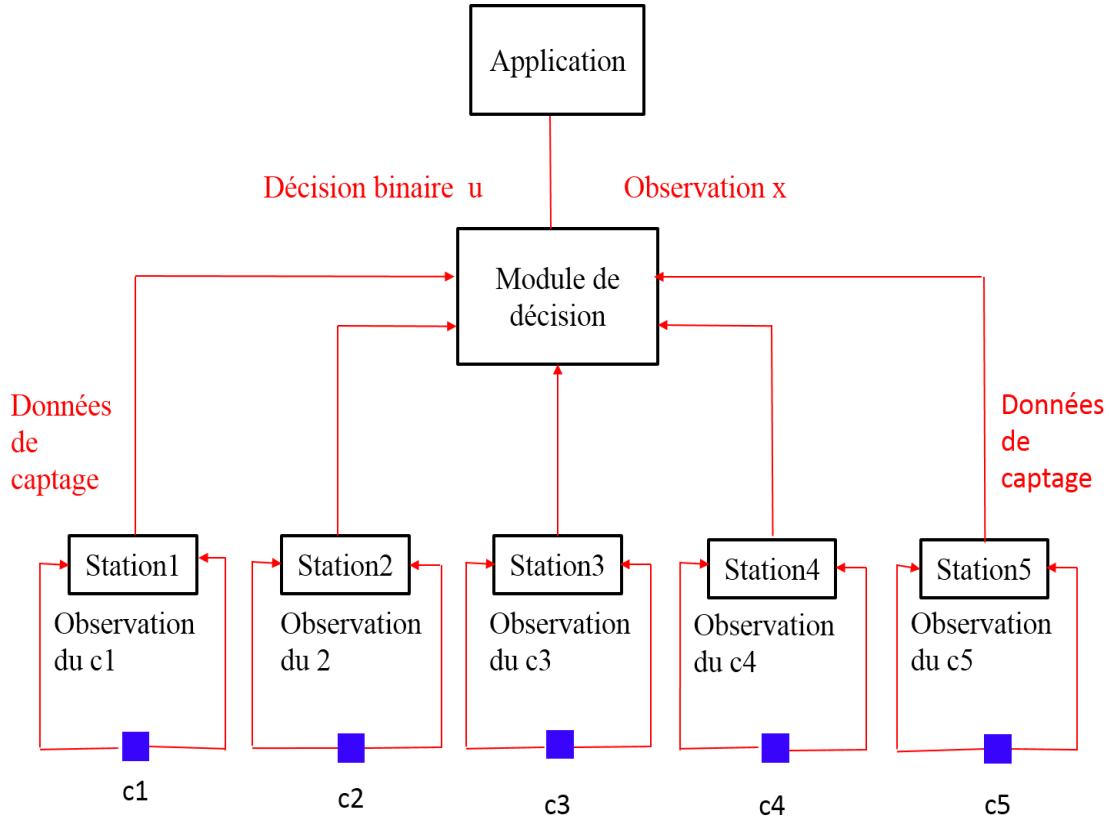


Figure III. 1. Redondance matérielle passive: duplication des capteurs et des stations.

III.3 Solution proposée: la redondance vers un système distribué tolérant aux fautes

III.3.1 Algorithme

Nous décrivons dans cette section notre méthodologie pour détecter les services Web défectueux pour avoir une décision globale plus fiable.

a. En premier pas nous invoquons N services Web qui sont identiques (ils ont la même description WSDL) et qui retournent des données générés par des capteurs simulés.
b. Le nœud principal récupère les résultats et les données des différents services Web (de chaleur par exemple).
c. Le nœud principal fait l'analyse statistique sur les résultats des inventaires en choisissant un percentile approprié (95% par exemple). Le choix de percentile est décisif pour notre étude. Si le résultat d'un capteur n'appartient pas à l'intervalle de confiance IC (ici à 95%) ce service Web est considéré défectueux. Alors le résultat pris par ce service Web n'est pas pris en compte dans l'analyse statistique.
d. Après l'élimination des valeurs prises par les services Web défectueux, le nœud principal prend une observation finale pour chaque type de service Web et à partir de laquelle une décision binaire globale u (évènement ou non évènement) est prise.

$$u = \begin{cases} 1, & Si\ NE\ (Non\ Evénement) \\ 0, & Si\ E\ \ (\ \ Evénement\ \ \ ) \end{cases}$$

III.3.2 Notion: Intervalle de confiance [33]

En statistiques, et en particulier dans la théorie des sondages, lorsqu'on cherche à estimer la valeur d'un paramètre, on parle d'intervalle de confiance lorsque l'on donne un intervalle qui contient, avec un certain degré de confiance, la valeur à estimer. Le degré de confiance est en principe exprimé sous la forme d'une probabilité. Par exemple, un intervalle de confiance à 95% (ou au seuil de risque de 5% et degré de confiance de 95%) a une probabilité égale à 0,95 de contenir la valeur du paramètre que l'on cherche à estimer. Ainsi, lorsqu'on effectue un sondage (tirage au hasard d'un sous-ensemble d'une population), l'estimation d'une quantité d'intérêt donnée est soumise au hasard et correspond rarement d'une manière exacte à la valeur de la quantité que l'on cherche à estimer. En présentant pour l'estimation non pas une valeur mais un encadrement, on quantifie d'une certaine manière l'incertitude sur la valeur estimée.

Plus l'intervalle de confiance est de taille petite, plus l'incertitude sur la valeur estimée est petite.

Les bornes de l'intervalle de confiance sont des percentiles qui coupent l'ensemble de données. Par exemple, pour l'intervalle de confiance IC95%, 95% des données devraient être au-dessus de la borne inférieure et au- dessous de la borne supérieure.

Pour bien interpréter l'intervalle de confiance et son étendue nous sommes allés à la référence [34].

III.3.3 Etendue de l'intervalle de confiance (IC) [34]

L'étendue de l'intervalle de confiance est égale à la borne supérieure moins la borne inférieure. Elle quantifie la précision de l'estimation : plus l'étendue est faible plus l'estimation est précise.

Plus la taille d'échantillon est importante, plus l'étendue de l'IC95% est faible. Pour réduire l'étendue de moitié, il faut inclure 4 fois plus de sujets. Pour diviser l'étendue par k, il faut inclure k^2 fois plus de sujets.

Plus l'échantillon est hétérogène (i.e. plus la variance ou l'écart-type sont élevés), plus l'étendue est importante. L'étendue de l 'IC% est directement proportionnelle à l'écart-type de la variable mesurée.

L'usage le plus simple des intervalles de confiance concerne les populations à distribution normale (en forme de cloche) dont on cherche à estimer la moyenne *X* [35]. Si on connaît l'écart type σ(*X*) (ou si on en connaît une estimation assez fiable) de cette distribution, et si on mesure la moyenne $\overline{x}$ sur un échantillon de taille n pris au hasard, alors :

- L'intervalle

$$\left[\bar{x} - 1\frac{\sigma(x)}{\sqrt{n}}; \bar{x} + 1\frac{\sigma(x)}{\sqrt{n}}\right]$$

est un intervalle de confiance à environ 68% ;

- L'intervalle

$$\left[\bar{x} - 2\frac{\sigma(x)}{\sqrt{n}}; \bar{x} + 2\frac{\sigma(x)}{\sqrt{n}}\right]$$

est un intervalle de confiance à environ 95% ;

- L'intervalle

$$\left[\bar{x} - 2\frac{\sigma(x)}{\sqrt{n}}; \bar{x} + 2\frac{\sigma(x)}{\sqrt{n}}\right]$$

est un intervalle de confiance à environ 99.7%.

III.4 Exemple illustratif

La figure illustre notre exemple. Nous avons les données suivantes:

Nous avons 5 stations qui contiennent chacune d'elles contient 5 services Web et qui retournent 5 observations de température.

Les services Web ont mis pour contrôler des produits alimentaires, nous faisons la collection des données des différents services Web d'une façon périodique (chaque quart d'heure par exemple).

Au niveau du nœud principal, nous avons une décision globale : le module décide après la récupération des résultats des services Web quels sont les services défectueux et il prend aussi :

- Une observation finale de chaque type de service.
- La décision binaire globale concernant le produit alimentaire: événement ou non-événement.

Si Température > 40° alors le produit alimentaire est altéré (événement) sinon le produit alimentaire est non altéré (non-événement).

III.4.1 Résultats des observations des capteurs

Les observations des différents capteurs sont données comme suit :

- ✓ Station 1: 35
- ✓ Station 2: 36
- ✓ Station 3: 25
- ✓ Station 4: 35.2
- ✓ Station 5: 35.8

III.4.2 Analyse des données

Nous appliquons la formule suivante pour calculer la moyenne arithmétique :

$$\bar{x} = \frac{x1 + x2 + \ldots + xn}{n} = \frac{1}{n}\sum_{i=1}^{n}(x_i)$$

Avec n=5, x1=35, x2=36, x3=25, x4=35.2, x5=35.8, alors :

$\bar{x}\ (moyenne) = 33$

Pour calculer l'écart-type σ, nous appliquons la formule suivante:

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

Avec n=5, $\bar{x} = 33$, alors σ (écart-type)=5.344155686

L'écart-type est important ce qui signifie que les valeurs retournées par les capteurs ne sont pas centrés autour de la moyenne [36].

Si nous calculons l'intervalle de confiance de $\bar{x}$ (la moyenne) à environ 95%, l'intervalle de confiance à 95% est :

$$IC95\% = \left]\bar{x} - 2\frac{\sigma(x)}{\sqrt{n}}; \bar{x} + 2\frac{\sigma(x)}{\sqrt{n}}\right[$$

(σ: écart-type, $\bar{x}$: moyenne, n : taille de l'échantillon).

Avec n=5 ; $\bar{x}$ (la moyenne)=33 et σ (écart-type)=5.344155686.

L'intervalle de confiance IC95% = [28.21, 37.79].

Alors la valeur de la station 3 n'appartient pas à l'intervalle de confiance, donc on considère ce service Web comme un capteur défectueux.

L'observation finale de température sans pris en compte les résultats provenant des services Web défectueux en calculant la moyenne arithmétique: $\bar{x} = 35.5$.

La décision finale concernant l'événement: u=0 produit non altéré (non-événement) car la température<40°.

Pour réaliser une meilleure correction de la faute, notre schéma de décision devrait prendre en compte la faute de services Web en choisissant un meilleure intervalle de confiance (si l'intervalle de confiance à 95% génère beaucoup de valeurs erronées nous pouvons utiliser par exemple l'intervalle de confiance 99.7%).

III.5 Modélisation UML : Diagramme de classes

Toutes les données et les informations qui englobent le travail du notre système sont exprimées par les différentes tables (Capteur, Station, observation, Décision, moyenne_temperature, stations_défectueux) qui sont présentés sous forme de diagramme. Le diagramme suivant explique les relations entre les différentes tables de modèle. Notre base de données est organisé d'une manière tel que chaque station comporte un seul capteur qui peut générer une ou plusieurs observations de température qu'on peut la récupérer à partir de la table observation, calculer la moyenne pour un ensemble de stations, enregistrer cette moyenne dans la table moyenne_temperature, et en basant sur le résultat ,une décision finale se produit.

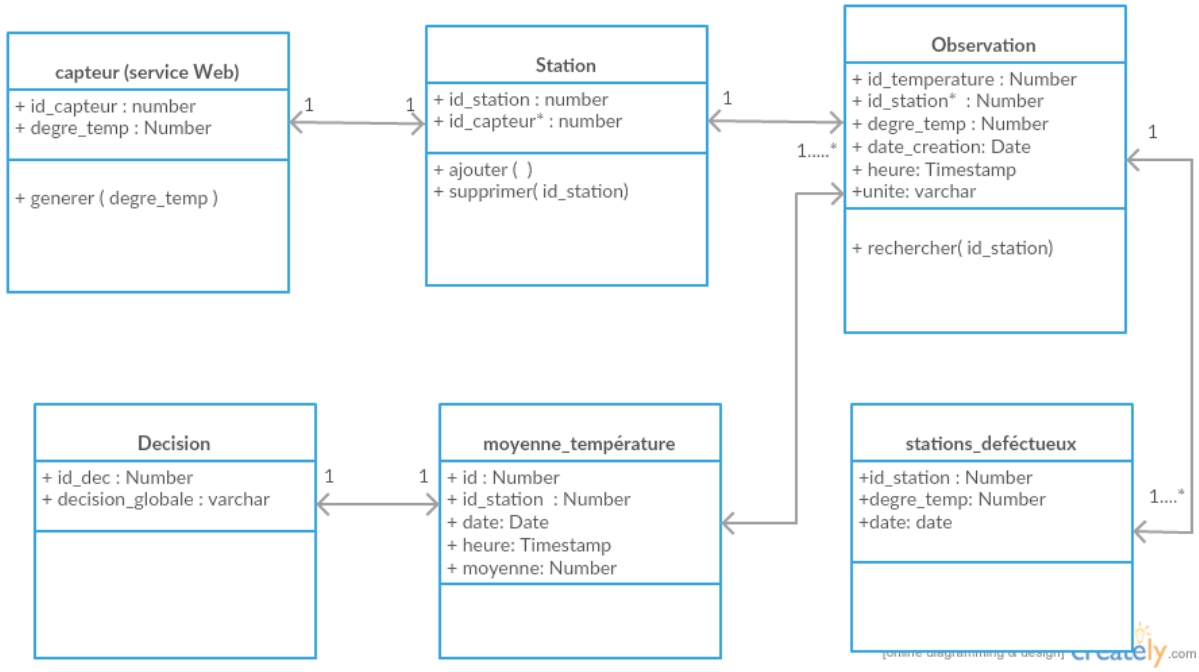


Figure III. 2. Diagramme de classes de notre système de captage.

III.6 Modélisation UML : Diagramme de séquences

Ce diagramme explique les processus utilisés entre les service Web (capteurs), les stations et les modules de décision en basant sur les observations retournées par les différents services Web, qui seront analysés, filtrés et collectés par les modules de décision en but d'une prise d'une décision finale et qui montre le comportement de notre système. Alors notre système sera exécuté à partir la page d'accueil en générant plusieurs capture en même temps à partir plusieurs services Web. Ensuite, et en passant par le processus d'enregistrement qui nous permet de récupérer les observations des services Web et enregistrer dans notre base de données. L'étape suivante nous permet d'analyser les données et comparer les observations récupérées à partir la base de données. Ensuite une décision sera prise selon les résultats d'analyse de données. Finalement les résultats d'analyse et la décision seront montrés dans un rapport de décision (Figure III. 3).

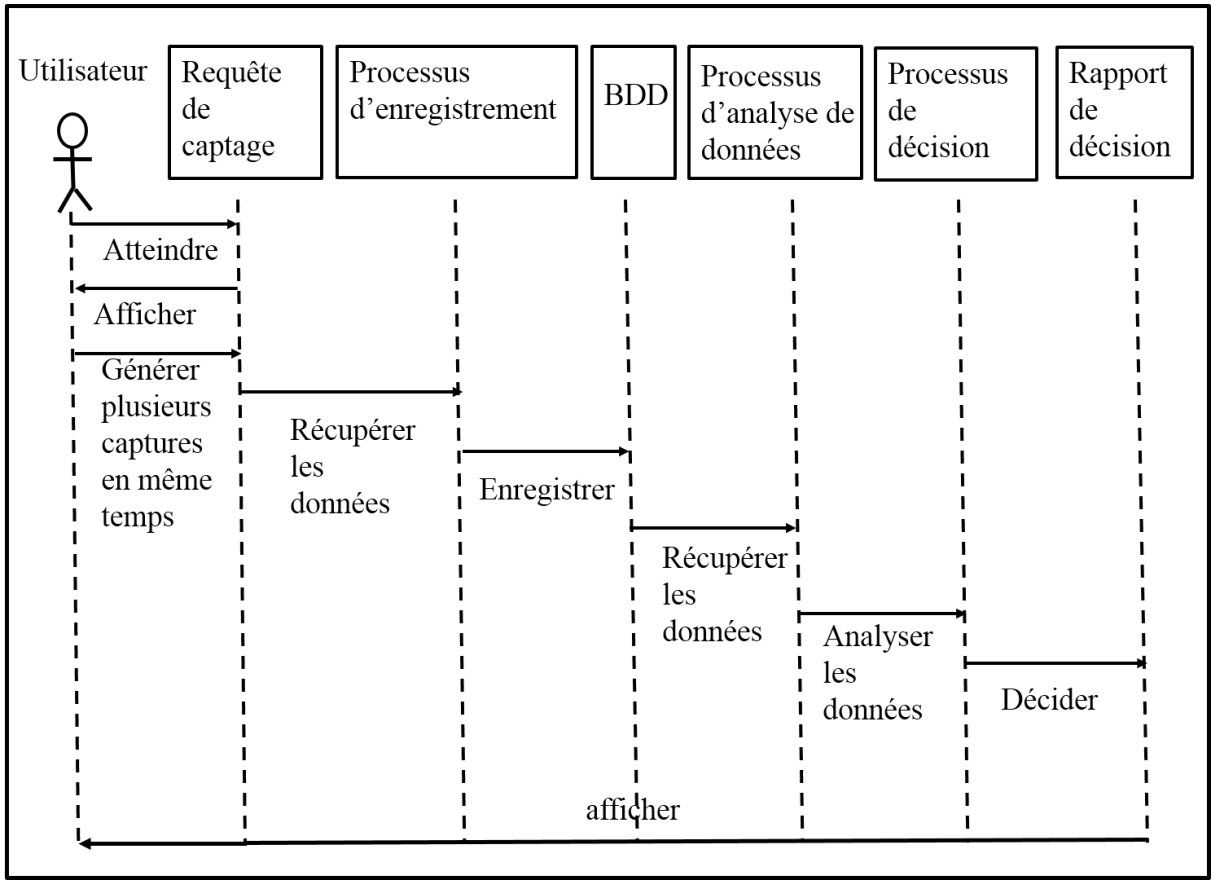


Figure III. 3. Diagramme de séquences pour le processus d'une prise de décision.

III.7 Implémentation

Nous allons essayer, dans cette section, de présenter pas à pas le processus complet du développement du notre système. Alors nous allons débuter avec les outils utiles pour mettre en œuvre notre système distribué: JDeveloppeur, J2EE, et JSF. En outre nous allons présenter l'architecture générale de l'application. Enfin nous allons définir les procédures de détection de fautes qui permettent de valider notre approche pour réaliser un système sûr et tolérant aux fautes.

Notre projet s'intitule la sûreté de fonctionnement des services Web SOA, c'est une application qui représente notre approche qui se base sur une intégration des modules d'analyse des données retournées à partir de plusieurs services Web qu'ils ont la même description WSDL(observations de température provenant des capteurs simulés).

III.7.1 Le langage SGBD Oracle

L'Oracle est un système de gestion de base de données (SGBD) relationnel fourni par Oracle Corporation et couramment utilisé dans les applications sur différentes plateformes. Il a été développé par Larry Ellison, accompagné d'autres personnes telles que Bob Miner et Ed Oates. Il s'agit du SGBD le plus vendu au monde.

III.7.2 JDeveloppper

Nous utilisons principalement l'environnement de développement Oracle JDeveloppper (Figure III. 4).

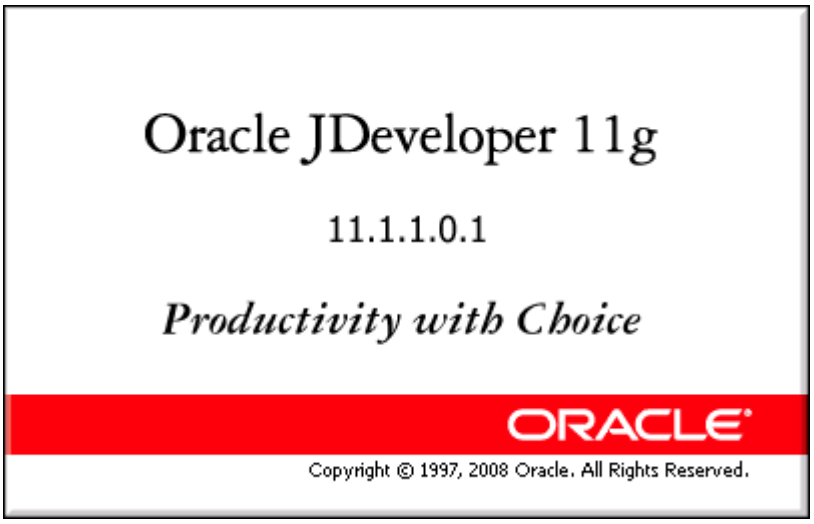


Figure III. 4. JDeveloppper.

Oracle JDeveloppper, l'un des composants d'Oracle fusion Middleware, est un environnement de développement intégré (IDE) complet pour le développement java et SOA. Optimisé pour s'exécuter avec Oracle Application Server et la base de données Oracle. JDeveloppper repose sur ses normes et plates-formes ouvertes en prenant en charge tous les principaux serveurs d'applications J2EE et bases de données [37].Un environnement de développement intégré commercial entièrement gratuit, Oracle s'attache à offrir aux développeurs un plus vaste éventail d'environnements et technologies de développement. Dans ce but, Oracle propose gratuitement JDeveloppper, intégrant divers outils graphiques et déclaratifs pour JSP (Java Server pages), Struts, JSF (Java Server Faces) et Business Process Execution Language (BPEL), comblant ainsi les attentes des développeurs jugeant insuffisantes les fonctions offertes par la plupart des IDE Java gratuits.

III.7.3 L'Infrastructure J2EE

La plate-forme J2EE (Java 2 Entreprise Edition) désigne l'ensemble constituée des services (API) accessibles et de l'infrastructure d'exécution [38].

J2EE comprend notamment :

- Les spécifications du serveur d'application, c'est-à-dire l'environnement d'exécution : J2EE définit finement les rôles et les interfaces pour les applications ainsi que l'environnement dans lequel elles seront exécutées.
- Des services, au travers d'API, c'est-à-dire des extensions Java indépendantes permettant d'offrir en standard certaines fonctionnalités. Sun fournit une implémentation minimale de ces API appelée J2EE SDK (J2EE Software Development Kit).

III.7.4 Entreprise Java Beans «EJB » [39]

Les EJB sont des composants et en tant que tel, ils possèdent certaines caractéristiques comme la réutilisabilité, la possibilité de s'assembler pour construire une application ... etc. Les EJB et les Beans n'ont en commun que d'être des composants. Les java Beans sont des composants qui peuvent être utilisés dans toutes les circonstances. Les EJB doivent obligatoirement s'exécuter dans un environnement serveur dédié. Les EJB sont parfaitement adaptés pour être intégrés dans une architecture trois tiers ou plus.

III.7.5 Java Server Faces « JSF » [40]

Java Server Faces est une technologie qui développe des applications Web en Java basé sur des composants côté présentation. Il bénéficie des apports du Framework STRUTS et des concepts J2EE (Swing, modèle événementiel, JSP, Servlets). JSF s'appuie sur un des technologies suivantes : Génération en Servlet, utilisation des composants JSF dans les pages JSP ou l'exposition des composants JSF aux JSP grâce aux balises personnalisés.

III.7.6 Architecture générale de notre Application

La figure suivante montre l'architecture générale qui présente les interfaces de notre système:

- L'interface A : représente notre application qui permet la communication avec notre système décisionnel. Nous avons utilisés principalement à ce niveau l'un des composants de l'infrastructure J2EE: les pages JSF (Java Service Faces). Il s'agit de la partie chargée de l'interface avec l'utilisateur qui permet de :
  - créer des capteurs virtuels et générer des captures d'observations de température.
  - Lancer le processus d'analyse des données.
  - Visualiser les résultats d'analyse.

- L'interface B: représente notre nœud principal, qui se compose de trois modules :
  - Module de Communication avec le Matériel de communication (capteurs).
  - Module de Filtrage et de Collection (MFC).
  - Module de Gestion de la Base de Données (MGBD).
- L'interface C: représente les capteurs simulés (services Web), nous allons insérer les fautes dans les capteurs pour pouvoir montrer le but de notre approche. Alors nous avons utilisé un service Web qui permet la communication avec les capteurs : le module de communication envoie une requête vers les services Web qui lui répondent en générant une capture de température.
- L'interface D: représente notre base de données sous Oracle qui permet d'enregistrer les résultats datées des différentes analyses. Des évènements seront déduit directement après la dernière analyse d'autres seront dédiées après plusieurs analyses.

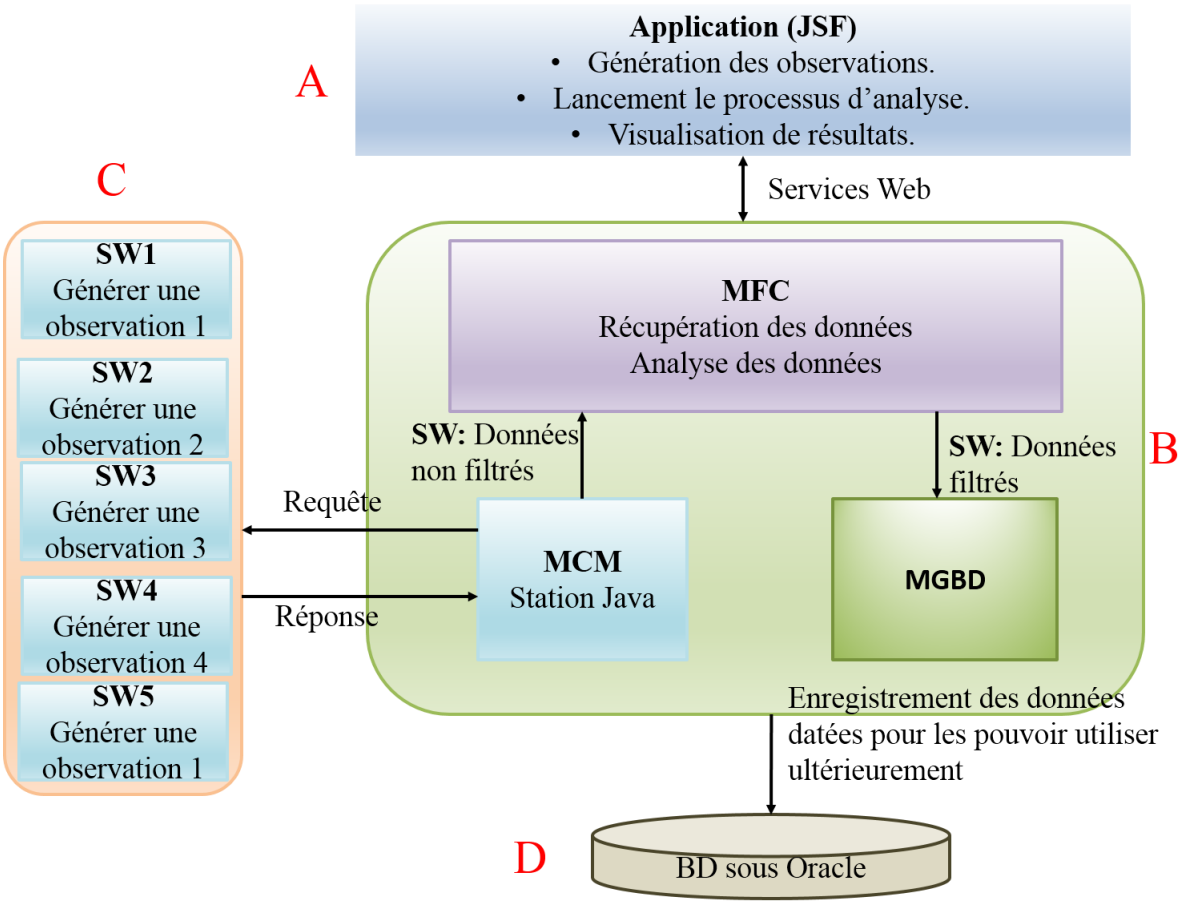


Figure III. 5. Architecture de notre logiciel.

III.7.7 Procédure de la détection de la faute

Nous donnons ici notre algorithme pour détecter la faute au niveau des services Web (capteurs), pour la manipulation des services Web défectueux (capteurs défectueux).

La procédure Analyser_Vecteur(M) permet de fournir la liste des capteurs défectueux et prendre une décision globale sur un évènement en prenant en considération seulement les observations des services Web qui sont fiables (les données qui sont dans l'intervalle de confiance).

```
Analyser_Vecteur (vecteur M)
    int nb=nombre de lignes de vecteurs M ;
    float tmp=0 ; / * observation de la température initiale */
    créer_vecteur (capteurs_défectueux) ; /* créer le vecteur des capteurs défectueux */
    Vecteur v= new vecteur (new float [n]) ; /* créer le vecteur qui contient toutes les
    valeurs*/
    V= vecteur(M) ;
    float bi=v.getBorne_inf_95 () ; /* la borne inférieure de l'intervalle de confiance
    IC95% */
    float bi=v.getBorne_sup_95 () ; /* la borne supérieure de l'intervalle de confiance
    IC95% */
    Pour  int i allant de 0 à n-1
        int compteur=0 ; /* pour calculer le nombre totale des capteurs fiables*/
        tmp= M.getValue(i))
        si (tmp>bi && tmp<bs)
            compteur++ ;
            tmp= (temp+M.getValue(i))/compteur;
            compteur++ ;
        else
            capteurs_défectueux.add (new integer (num)) ; /* insérer le capteur
            dans la liste des capteurs défectueux  */
        finSi
    finPour
Fin
```

Nous avons montré comment détecter l'erreur en analysant les résultats de captage pour détecter les services Web et prendre une décision finale en calculant la moyenne des observations obtenues à partir des services Web fiables.

III.8 Conclusion

Dans ce chapitre nous avons expliqué notre cas d'étude pour notre système qui se base principalement sur des données retournées à partir des services Web (capteurs

environnementaux simulés) et distribués (plusieurs services web qu'ils ont le même objectif) et qu'à partir d'eux une décision finale sera produite. Ensuite nous avons montré notre méthodologie choisi qui aller bien avec les concepts la sûreté de fonctionnement qui sont définit précédemment, en utilisant l'une des techniques de la tolérance aux fautes en basant sur le principe de la redondance passive qui permet d'assurer le fonctionnement de notre système en prenant en compte le cas où les fautes peuvent se produisent et peuvent être inattendue. Enfin nous avons présenté les algorithmes et les outils pour réaliser notre système et en précisant le processus de l'implémentation de notre logiciel.

Conclusion Générale

La sûreté de fonctionnement des services Web basés sur l'architecture orientée services est un domaine de recherche actif et important. L'architecture distribuée faiblement couplée des services Web a apporté des avantages pour le développement d'applications d'e-Science et d'e-commerce. Les applications développées à partir de services existants reposent sur des standards comme SOAP, WSDL et UDDI qui constituent une infrastructure de base permettant la mise en œuvre d'architectures orientées services. Cependant une telle architecture est fondamentalement peu fiable.

Alors dans notre mémoire, dans le premier chapitre nous avons fait une présentation des concepts de base de la sûreté de fonctionnement dans un système informatique, et de quelques techniques de sûreté de fonctionnement et de tolérance aux fautes. Ensuite dans le deuxième chapitre, d'une part, nous avons fait une présentation des services Web comme une mise en place de l'architecture orientée services, nous avons parlé un peu sur l'aspect fonctionnel et technique des services Web, et nous avons étudié les différents protocoles de base permettant de mettre en place une architecture à base de Service Web. D'une autre part, dans la deuxième sous-section, nous avons expliqué les différents concepts de la SOA en basant sur la notion de **service,** les différentes interactions entre le client et le prestataire en utilisant les architectures dynamiques basées sur l'agrégation et la dissémination des services. Enfin, dans le troisième chapitre, nous avons présenté notre cas d'étude, notre approche comme une solution, qui nous permet d'utiliser la technique de la tolérance aux fautes en donnant une modélisation UML de notre système et expliquant notre approche ave un exemple illustratif.

L'effort fourni dans ce travail nous permet de développer des solutions pour le problème de tolérance aux fautes dans les systèmes distribués à base des services Web en respectant les concepts de l'architecture SOA. La sûreté de fonctionnement de ces systèmes est donc indispensable pour permettre la livraison de services corrects aux utilisateurs du système. Dans ce contexte, cette mémoire avait pour objectif global de contribuer à la tolérance aux fautes dans les systèmes distribués en concevant un système de décision qui détecte et localise les défaillances en comparant les résultats des différents capteurs homogènes en utilisant des techniques statistiques sur les résultats obtenus et permettre à l'application de détecter l'élément défectueux, corriger les fautes et continuer son exécution en présence de ces entraves.

Références

[1] C. Pagetti, "Module de la sureté de fonctionnement," 2012. [Online]. Available: docplayer.fr/1398365-Module-de-surete-de-fonctionnement.html. [Accessed: 03-Dec-2015].

[2] J. C. Laprie, Ed., *Dependability: Basic Concepts and Terminology*, vol. 5. Vienna: Springer Vienna, 1992.

[3] J.-C. Geffroy and G. Motet, *Design of Dependable Computing Systems*, SPRINGER-S. Springer Netherlands, 2002.

[4] J. Arlat, Y. Crouzet, and Y. Deswarte, "Tolérance aux fautes," *Encyclopédie de l'informatique et des systèmes d'information. Vuibert, Paris, France*. pp. 1–32, 2006.

[5] C. Rousseau and Y. Saint-Aubin, *Mathématiques et Technologie*. New York: Springer Science+ Business Media, LLC, 2009.

[6] L. Maesano, C. Bernard, and X. Le Galles, *Services Web avec J2EE et .NET (Conception et Implementation)*, Eyrolles. Paris, 2003.

[7] "Extensible Markup Language (XML) 1.0 (Fifth Edition)." [Online]. Available: https://www.w3.org/TR/REC-xml/. [Accessed: 18-Jan-2016].

[8] W.Houser, C.Hage, and J.Griffin, "EDI Meets the Internet." [Online]. Available: http://www.ietf.org/rfc/rfc1865.txt. [Accessed: 18-Jan-2016].

[9] C. Roberto, M. Gudgin, J.-J. Moreau, and S. Weerawarana, "Web Services Description Language (WSDL) Version 1.2 Part 1: Core Language," 2003. [Online]. Available: https://www.w3.org/TR/2003/WD-wsdl12-20030611/. [Accessed: 18-Jan-2016].

[10] "SOAP Specifications." [Online]. Available: https://www.w3.org/TR/soap/. [Accessed: 18-Jan-2016].

[11] N. Mitra and Y. Lafon, "SOAP Version 1.2 Part 0: Primer (Second Edition)," 2007. [Online]. Available: https://www.w3.org/TR/2007/REC-soap12-part0-20070427/. [Accessed: 18-Jan-2016].

[12] H. Haas, I. Oisin Hurley, A. Karmarkar, J. Mischkinsky, M. Jones, L. Thompson, and Richard Martin, "SOAP Version 1.2 Specification Assertions and Test Collection (Second Edition)," 2007. [Online]. Available: https://www.w3.org/TR/2007/REC-soap12-testcollection-20070427/. [Accessed: 18-Jan-2016].

[13] M. Gudgin, M. Hadley, N. Mendelsohn, J.-J. Moreau, H. F. Nielsen, A. Karmarkar, and Y. Lafon, "SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)," 2007. [Online]. Available: https://www.w3.org/TR/2007/REC-soap12-part1-20070427/. [Accessed: 18-Jan-2016].

[14] Martin Gudgin, M. Hadley, N. Mendelsohn, J.-J. Moreau, H. F. Nielsen, A. Karmarkar, and Y. Lafon, "SOAP Version 1.2 Part 2: Adjuncts (Second Edition)," 2007. [Online]. Available: https://www.w3.org/TR/2007/REC-soap12-part2-20070427/. [Accessed: 18-Jan-2016].

[15] L. Clement, A. Hately, C. von Riegen, and Tony Rogers, "OASIS and UDDI Version 3.0.2," 2004. [Online]. Available: http://www.uddi.org/pubs/uddi_v3.htm. [Accessed: 18-Jan-2016].

[16] N. D. M. Hadi Valipour, Bavar AmirZafari, Kh. Niki Maleki, "A Brief Survey of Software Architecture Concepts and Service Oriented Architecture .," in *InProceedings of 2nd IEEE*

International Conference on Computer Science and Information Technology, 2009, p. pp 34–38.

[17] C. K. Arsanjani A., Liang-Jie Zhang, Ellis M., Allam A, *A Service-Oriented Reference Architecture .*, . IT Profe. 2007.

[18] Z. C. et K. Djelloul, "Approche de simulation DEVS à base de services web," Oran, 2007.

[19] I. Trickovic and S. Weerawarana., *Business Process Execution Language for Web Services, Version 1.1.* .

[20] L. F. Cabrera, G. Copeland, M. Feingold, R. W. Freund, T. Freund, J. Johnson, and D. Langworthy, *Web Services Coordination (WSCoordination), Version 1.0*. 2005.

[21] "WS-Transaction," 2004. [Online]. Available: 128.ibm.com/developerworks/library/specification/ws-tx/. [Accessed: 19-Jan-2016].

[22] Y. Chen, "WS-Mediator for Improving Dependability of Service Composition Thesis by," 2008.

[23] N. Aghdaie and Y. Tamir, "Client-Transparent Fault-Tolerant Web Service," *20th IEEE Int. Performance, Comput. Commun. Conf.*, 2001.

[24] S. Todd, F. Parr, and M. H. Conner, "A Primer for HTTPR," *IBM Internet Softw. Austin*, 2002.

[25] E. Alwagait and S. Ghandeharizadeh, "DeW : A Dependable Web Services Framework," *14th Int. Work. Res. Issues Data Eng. Web Serv. E-Commerce E Gov. Appl. (RIDE'04), Boston, Massachusetts*, 2004.

[26] D. Liang, C.-L. Fang, and C. Chen, "FT-SOAP: A Fault-tolerant web service," *Tenth Asia-Pacific Softw. Eng. Conf. Chiang Mai, Thail.*, 2003.

[27] G. T. Santos, L. C. Lung, and C. Montez, "FTWeb: A Fault Tolerant Infrastructure for Web Services," *Proc. 2005 Ninth IEEE Int. EDOC Enterp. Comput. Conf.*, 2005.

[28] A. Avizĭenis and L. Chen, "On the Implementation of N-Version Programming for Software Fault Tolerance During Execution," *Proc. IEEE Ann. Int'l Comput. Softw. Appl. Conf chwicago*, 1977.

[29] M. Corporation, "NET Framework." [Online]. Available: http://msdn2.microsoft.com/en-gb/netframework/default.aspx. [Accessed: 24-Jan-2016].

[30] J. Fraga, F. Siqueira, and F. Favarim, "An Adaptive Fault-Tolerant Component Model," *Proc. Int. Work. Object- Oriented Real-Time Dependable Syst.*, 2003.

[31] S. M. Inc, "Java EE at a Glance." [Online]. Available: http://java.sun.com/javaee/index.jsp. [Accessed: 26-Jan-2016].

[32] L. G. et C. R. a.Thi-Quynh Bui, Oum-El-Kheir Aktouf, Michel Dang, "Diagnosis Service for Software Component and Its Application to a Heterogeneous Sensor Data Management System.," in *Second International Conference on Dependability.*, 2009, pp. 143–149.

[33] J. . Zar, "Prentice Hall International," in *Biostatistical Analysis*, 1984, pp. 43–45.

[34] Costantini G, "Estimation ponctuelle- Estimation par intervalle de confiance." [Online]. Available: www.sante.univ-nantes.fr/med/IntervalleDeConfiance.ppt. [Accessed: 08-May-2016].

[35] R. D.G, *Essential Statistics. 4th Edition (CRC/C&h Texts in Statistical Science)*, Chapman an. 2003.

[36] H. Walker, *Studies in the History of Statistical Method with Special Reference to Certain*

Educational Problems. USA: Williams & Wilkins, 1931.

[37] "JDeveloper. Site officiel de Oracle JDeveloper." [Online]. Available: http://www.oracle.com/technetwork/developer-tools/jdev/overview/index.html.. [Accessed: 14-May-2016].

[38] A. Y. M. Guedda, "Une approche J2EE pour piloter une simulation sur le web.," oran, 2002.

[39] J. M. DOUDOUX, *développons en Java*. .

[40] F.-X. SENNESAL, *java server faces (jsf) avec eclipse-conception d''applications web exploitant des composantes jsf.*, Eni. france, 2009.

Annexe (A) : Création de la base de données

A.1 Présentation des tables

Toutes les données et les informations qui englobent le travail du notre plateforme sont exprimés par des différents tables : Capteurs, Stations, Observations, Stations_defectueux, Decision, moyenne_temperature.

A.2 Présentation des scripts

A.2.1 Le script tables.sql

Permet de créer les tables de l'application, la figure montre uniquement quelques tables.

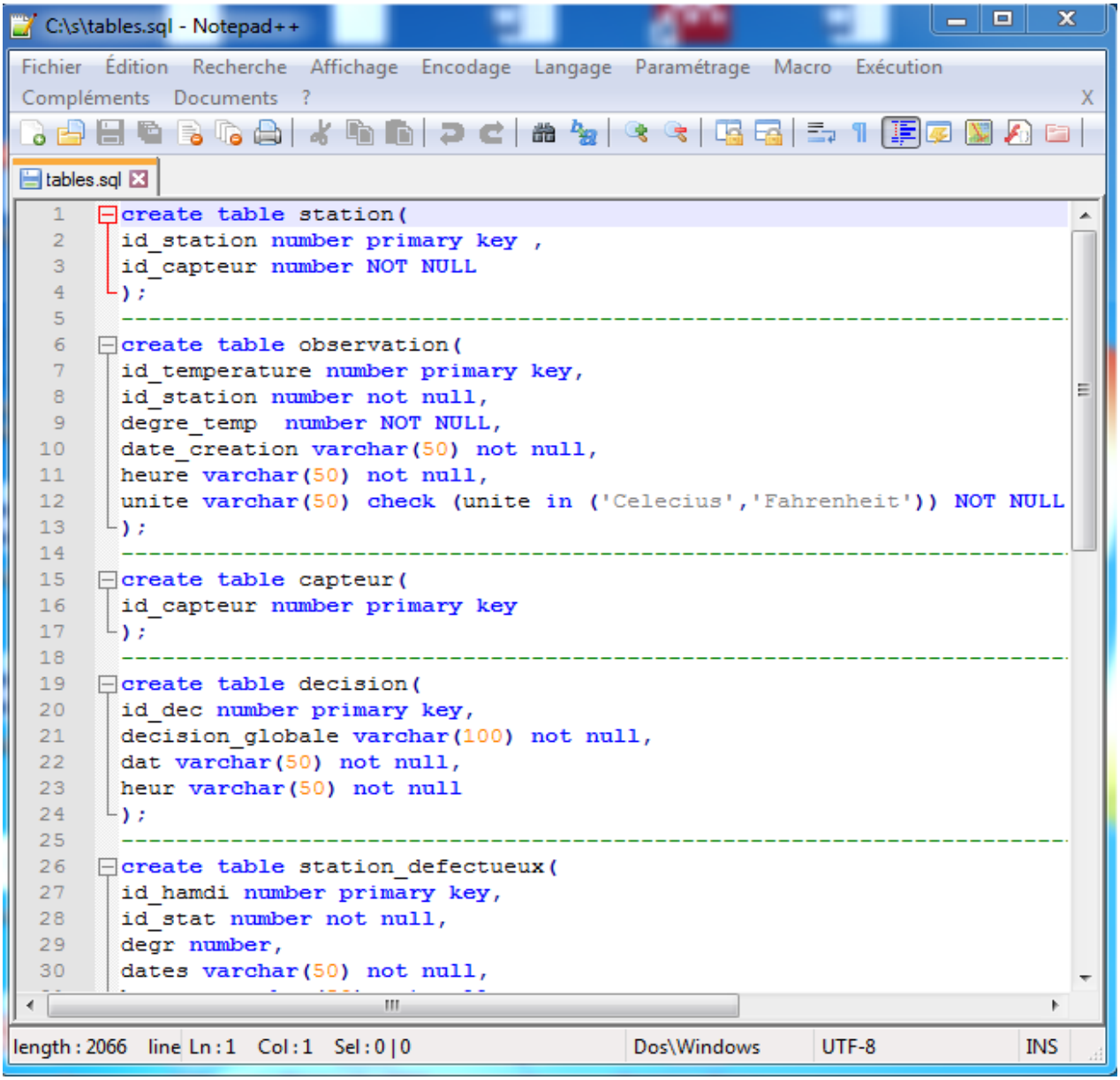


Figure A. 1. Création des tables sous forme SQL.

- Après la conception des tables, la deuxième étape consiste à réaliser les scripts de création des tables au niveau de la base de données Oracle 10g (Express Edition). Ces scripts sont donnés par la figure ci-dessous :

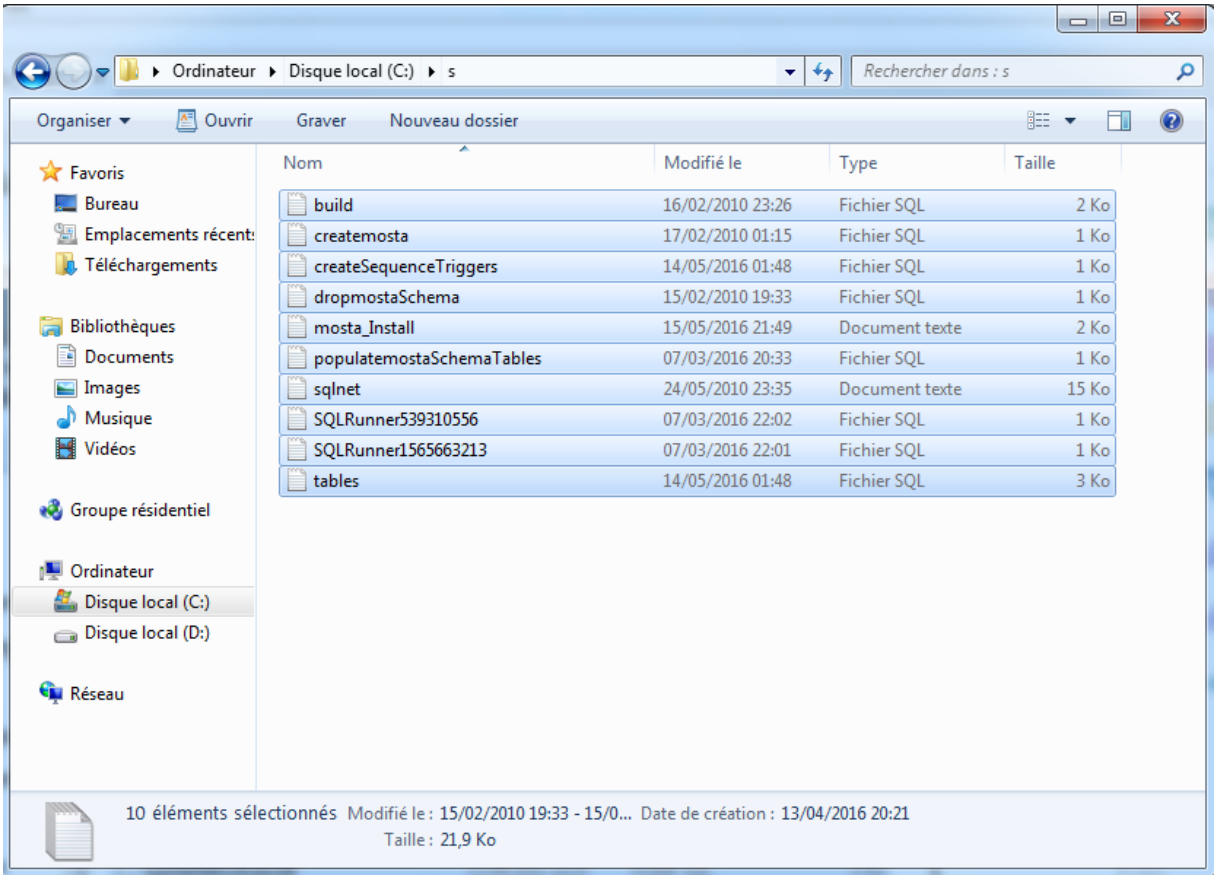
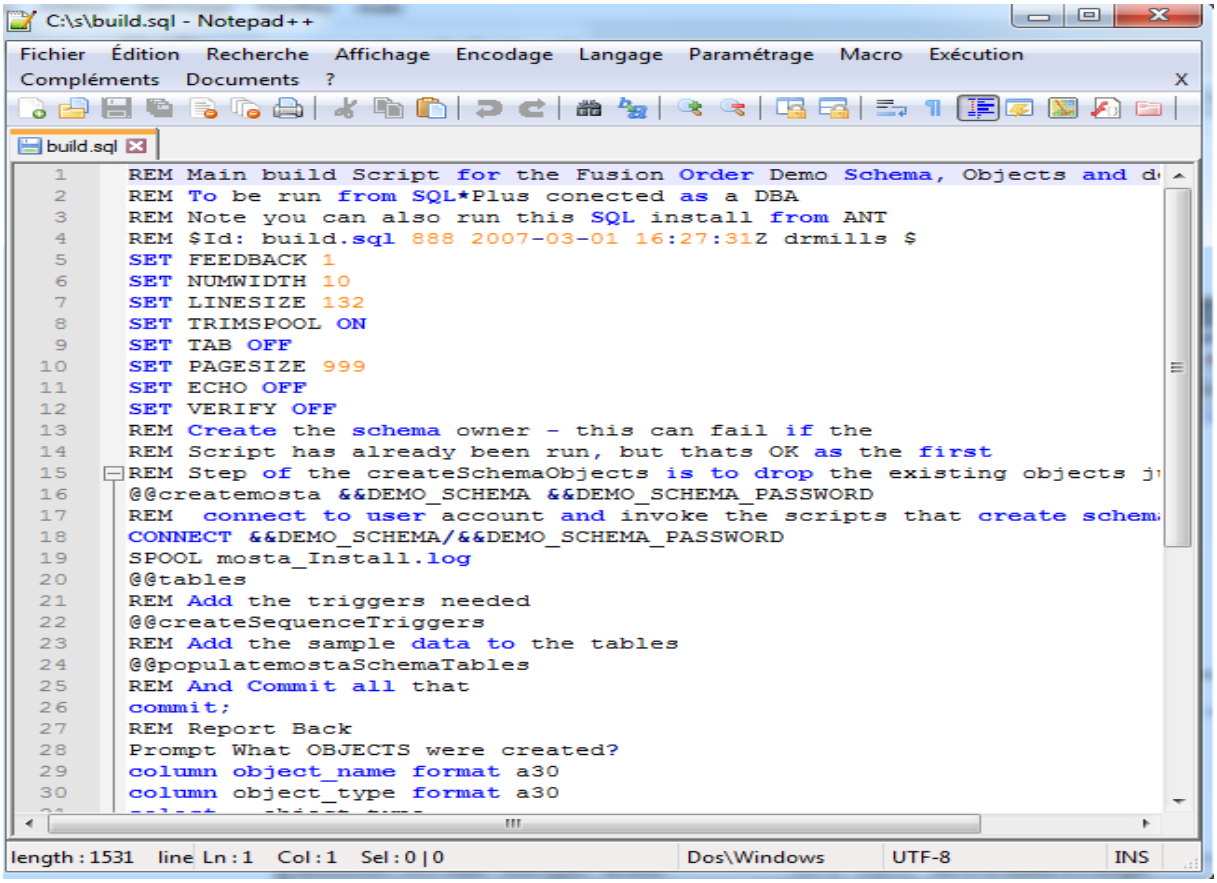


Figure A. 2. Les scripts.



A.2.2 Le script build.sql

Permet de lancer tous les scripts nécessaires.

Figure A.3. Le script : build.sql.

A.2.3 Le script createmosta.sql

Permet de créer l'utilisateur de ces tables. Un compte sur la base de données.

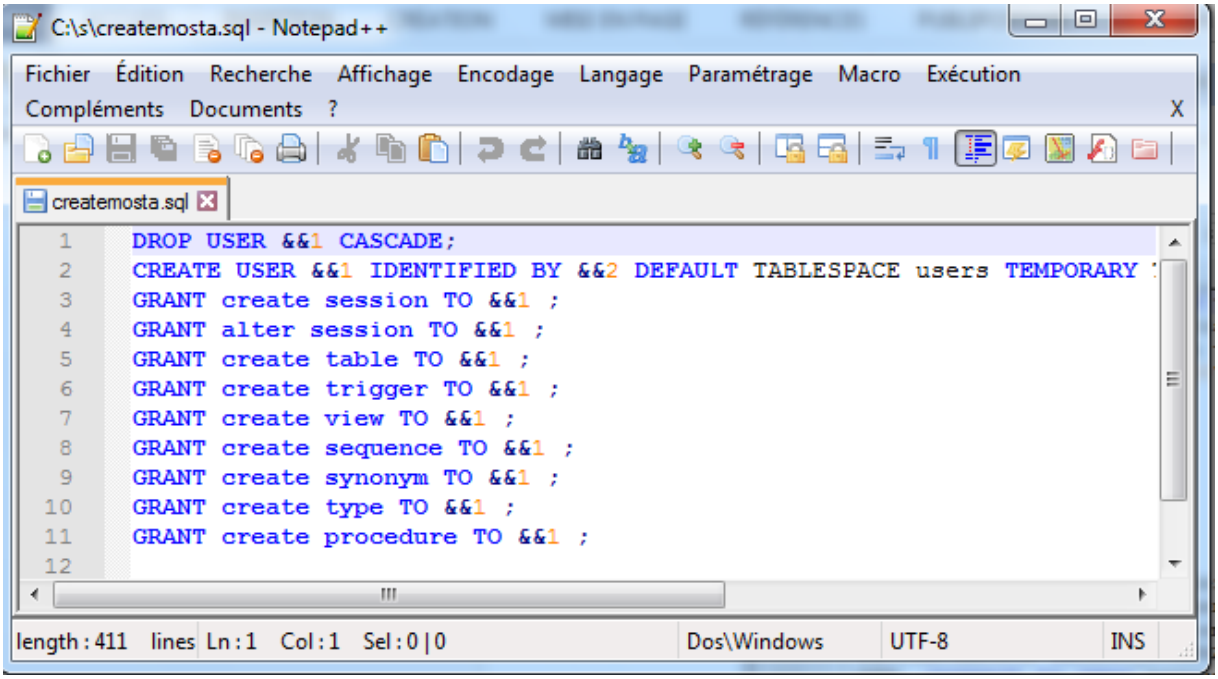


Figure A. 4. La table createmosta.sql.

- La figure suivante démontre l'identification du mot de passe et le nom d'utilisateur du compte de la base de données.

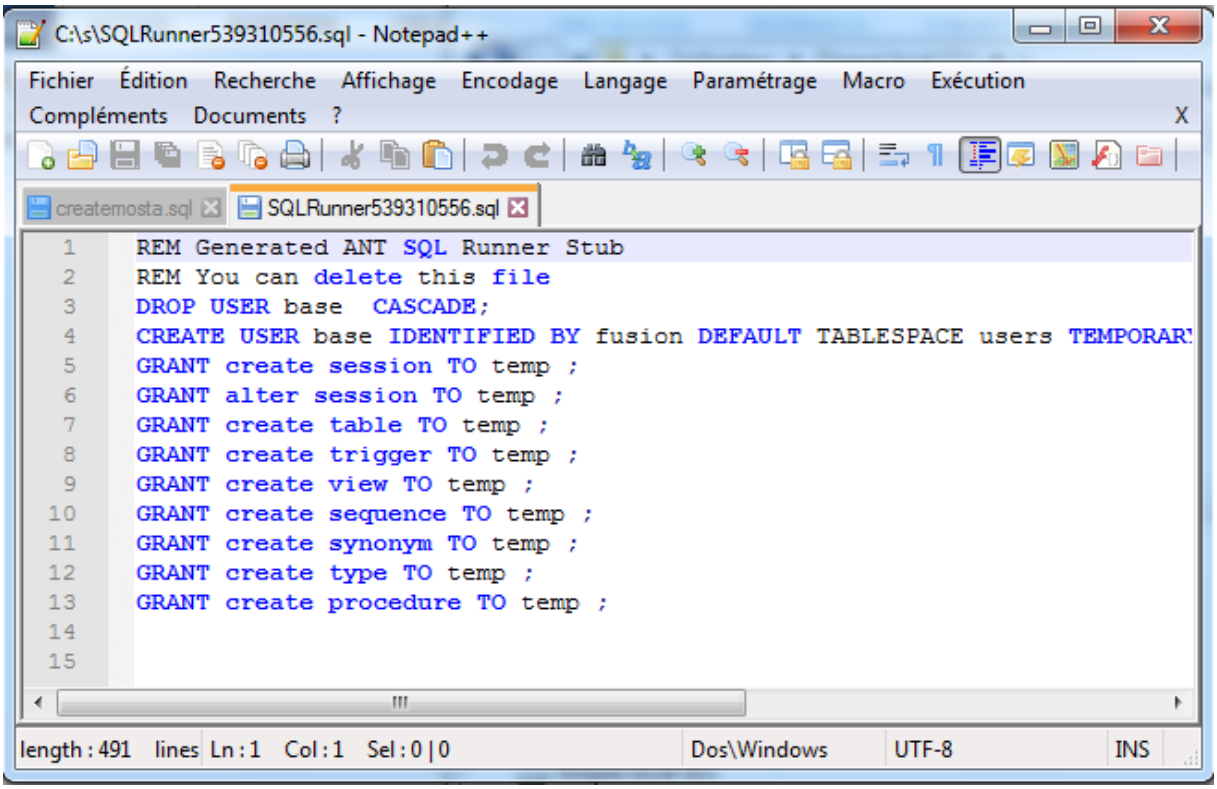


Figure A. 5. La table SQLRunner539310556.sql.

A.2.4 Le script create Sequence et triggers.sql

Permet de créer tous les triggers des tables (les triggers sont des déclencheurs).

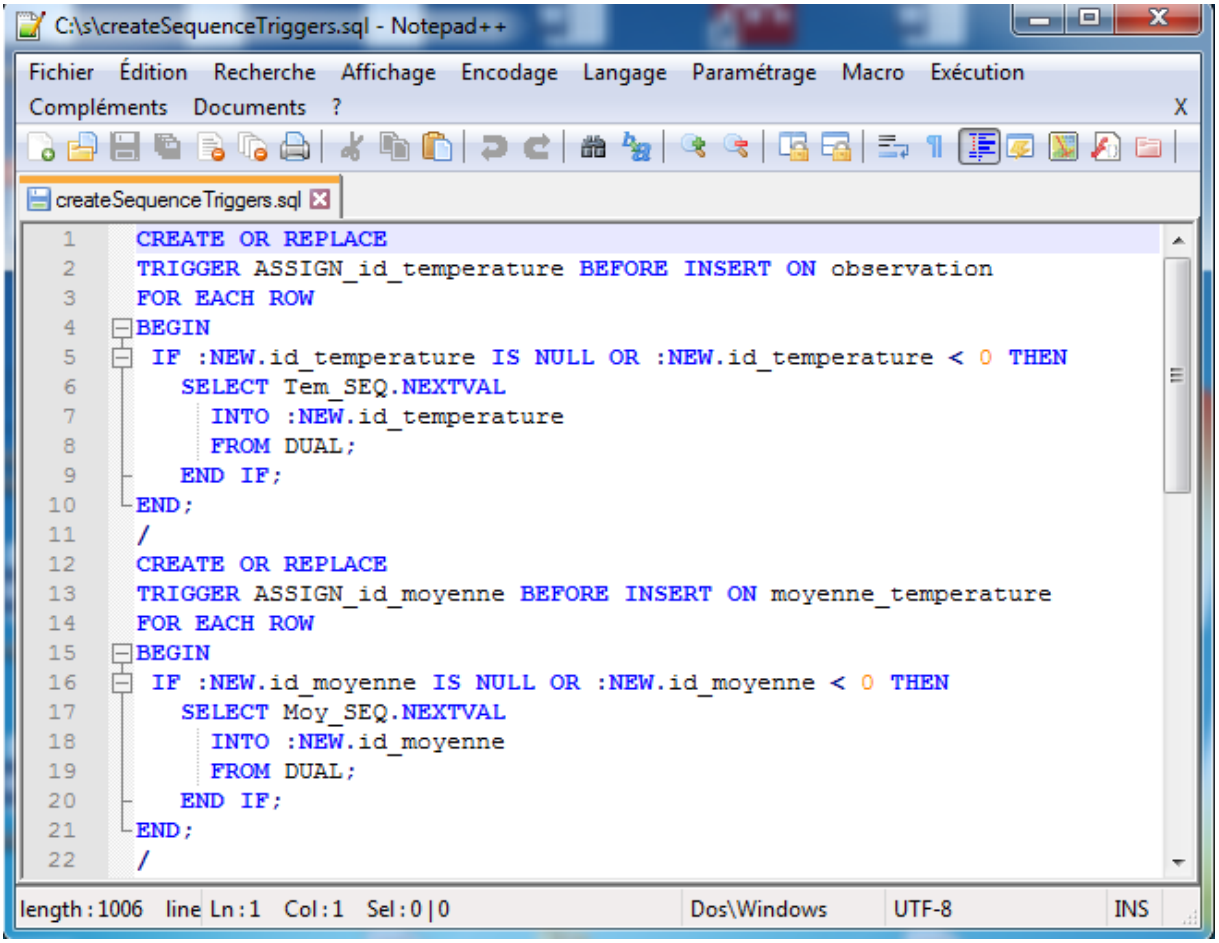


Figure A. 6. La table create Sequence et Triggers.sql.

A.2.5 Chargement des tables à partir le DOS

Après la réalisation de tous les scripts nécessaire pour la création de la base de données, il ne reste que lancer des commandes permettant de charger les tables sur l'oracle. La figure suivante montre ces commandes :

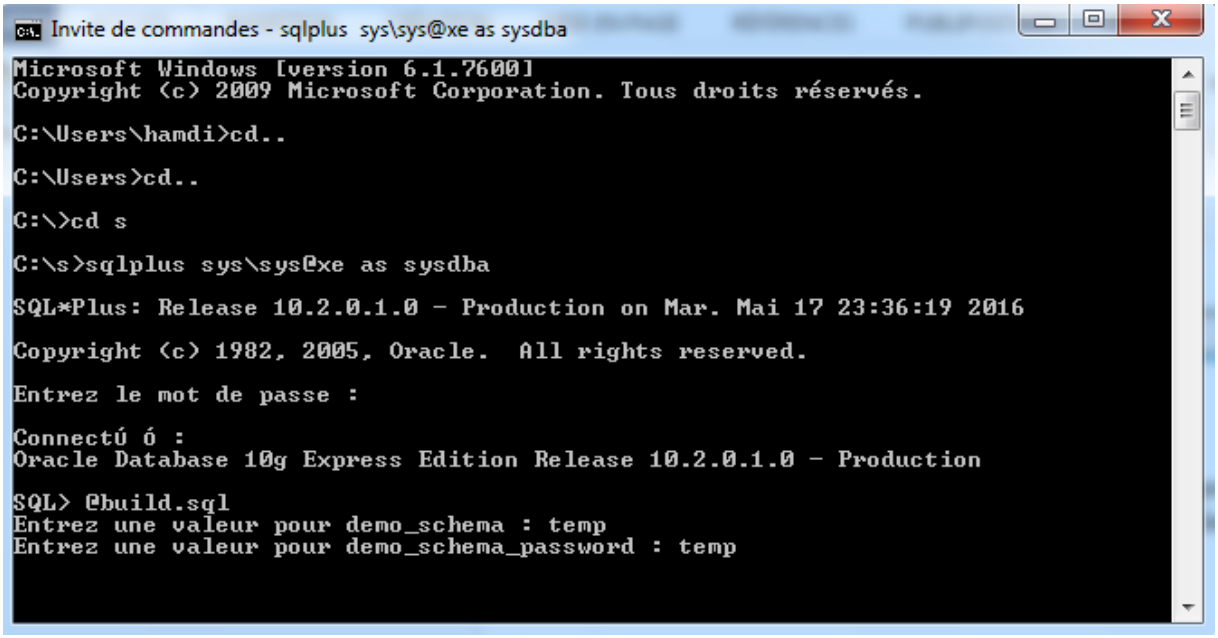


Figure A. 7. Les commandes en DOS.

A.2.6 Connexion à la base de données via Oracle

Après le chargement des tables à partir le DOS on peut consulter notre Base en utilisant le nom d'utilisateur et mot de passe qu'ils nous sommes déjà les mentionner dans les scripts pour notre compte de la base de données.

N.B : nom utilisateur = temp

Mot de passe = temp

Ensuite appuyer sur bottons connexion.

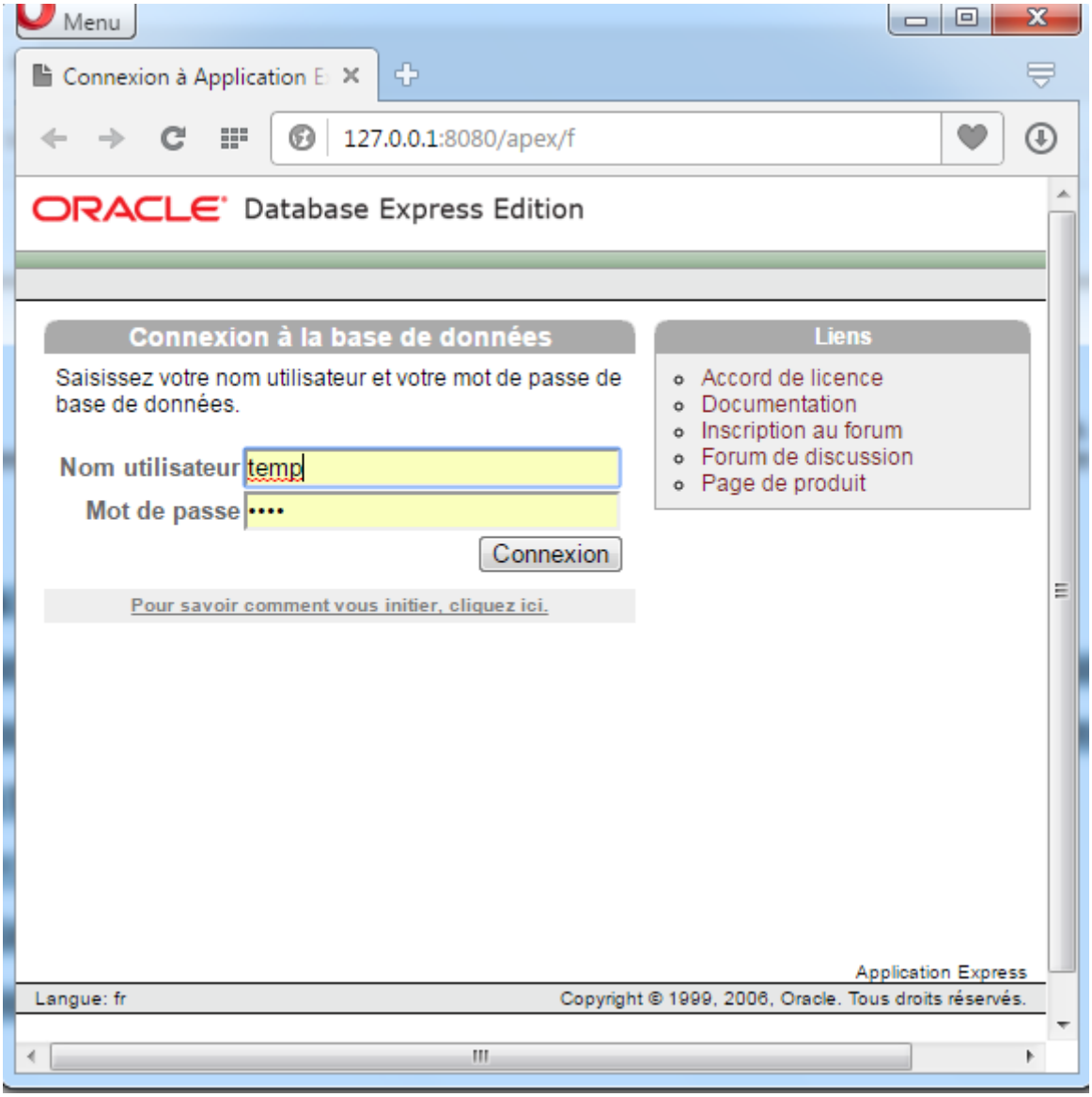


Figure A. 8. Connexion à la base de données.

Annexe (B) : Création de Projet

B.1 Construction l'application

On a créé une application général **« Generic Application »** qui contient plusieurs projets.

B.1.1 Création des EJB Entities

A ce niveau on a créé les **« EJB Entities from Tables »** à partir les tables de notre base de données stockées au niveau d'Oracle.

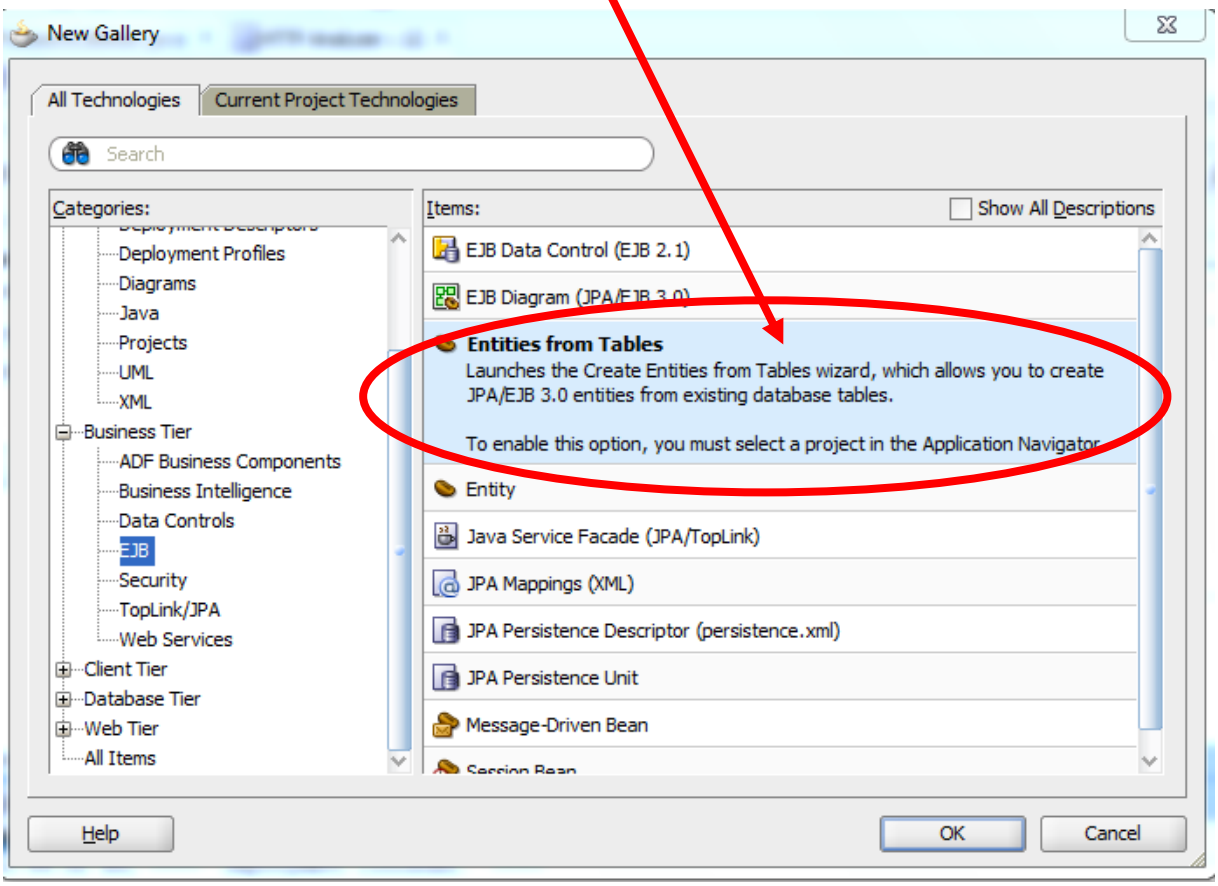


Figure B. 1. Création des EJB à partir des tables.

Pour créer les EJB from tables on doit connecter à la base de données par le nom d'utilisateur et le mot de passe du compte.

NB: Username=temp.

Password=temp.

On fait le test de la connexion s'il s'affiche **succès** alors la connexion est établie.

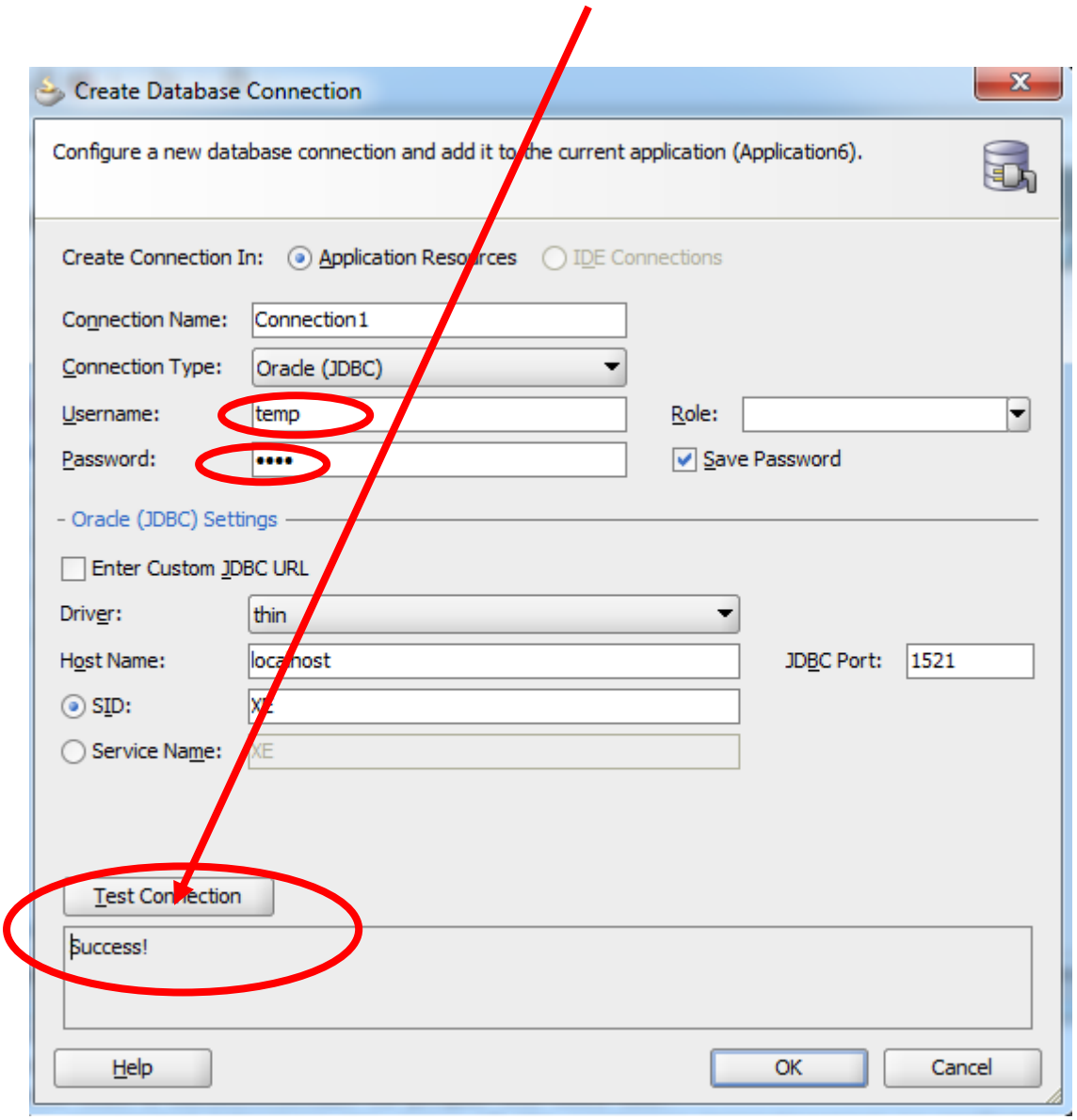


Figure B. 2. Etablir une connexion à la base de données.

B.1.2 Le choix de package et les tables

A ce niveau on donne un nom au package et on select les tables qu'on va les manipuler.

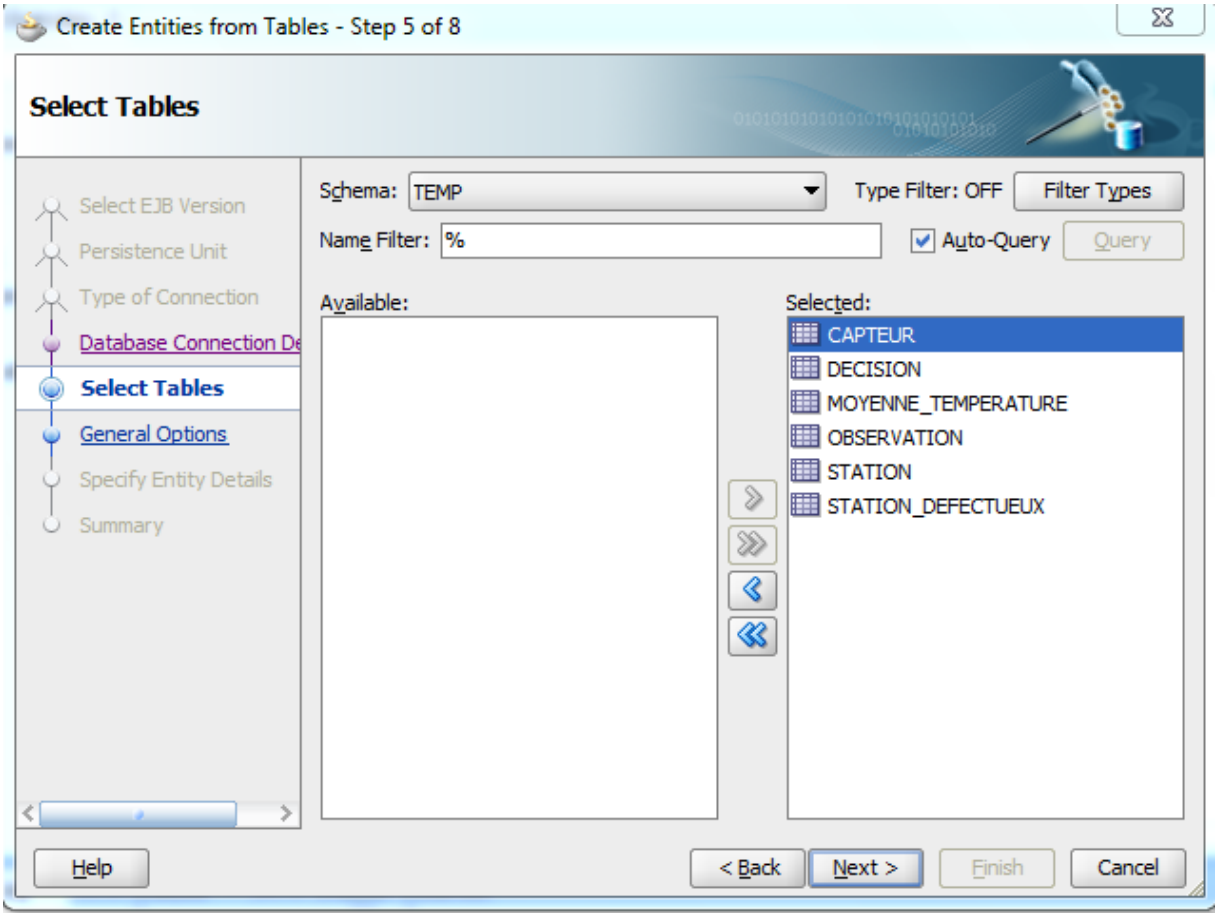


Figure B. 3. Sélection des tables utilisées

B.1.3 Requêtes de manipulation

On atteint chaque EJB Entities et on saisit dans son code les **requêtes** qui nous permettent de manipuler les données selon le besoin.

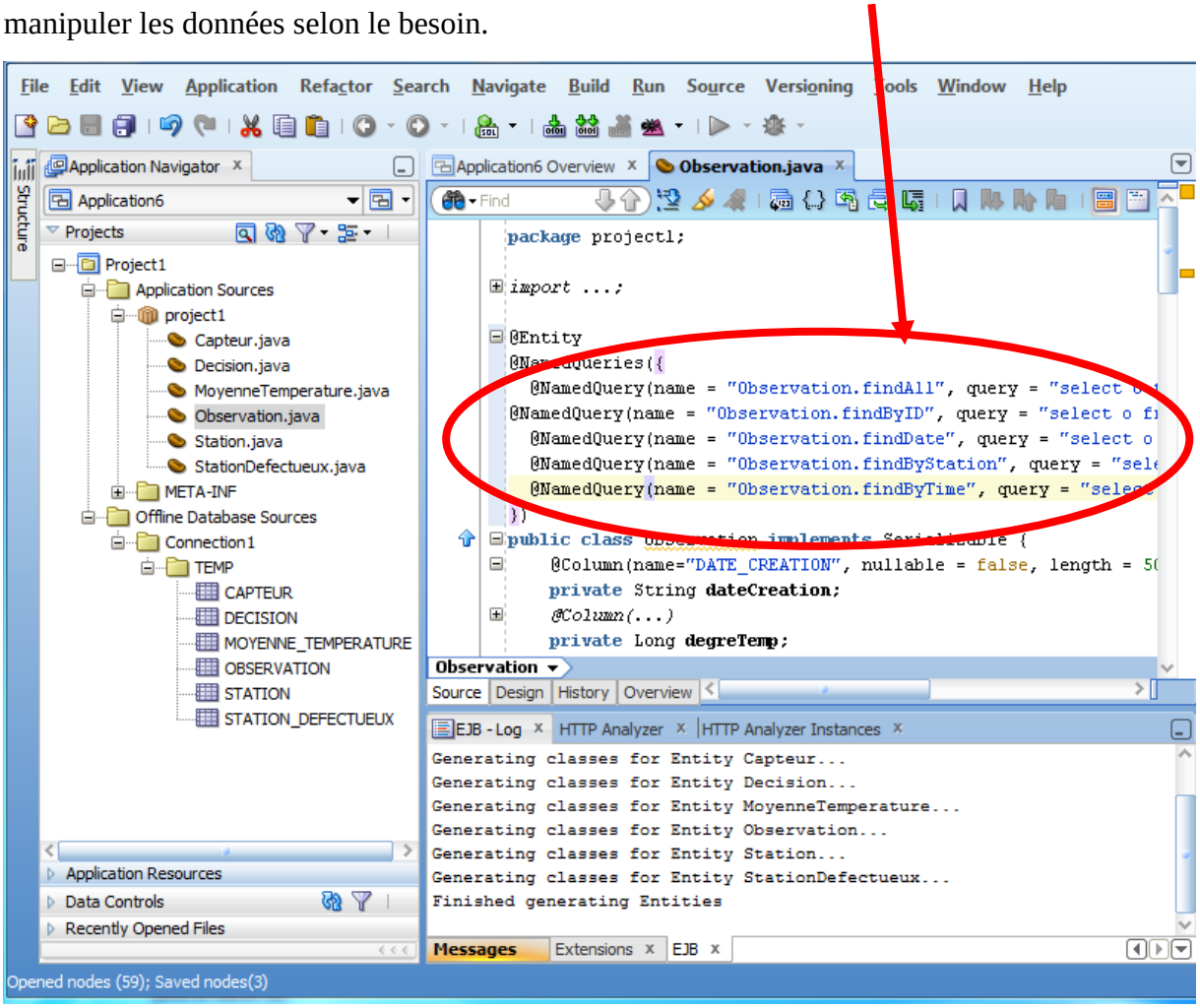


Figure B. 4. Syntaxe des requêtes.

B.1.4 Création des EJB Session

Pour mettre une façade entre le modèle et le client et aussi pour éviter la saturation du web service avec les clients on utilise des **EJB Session**.

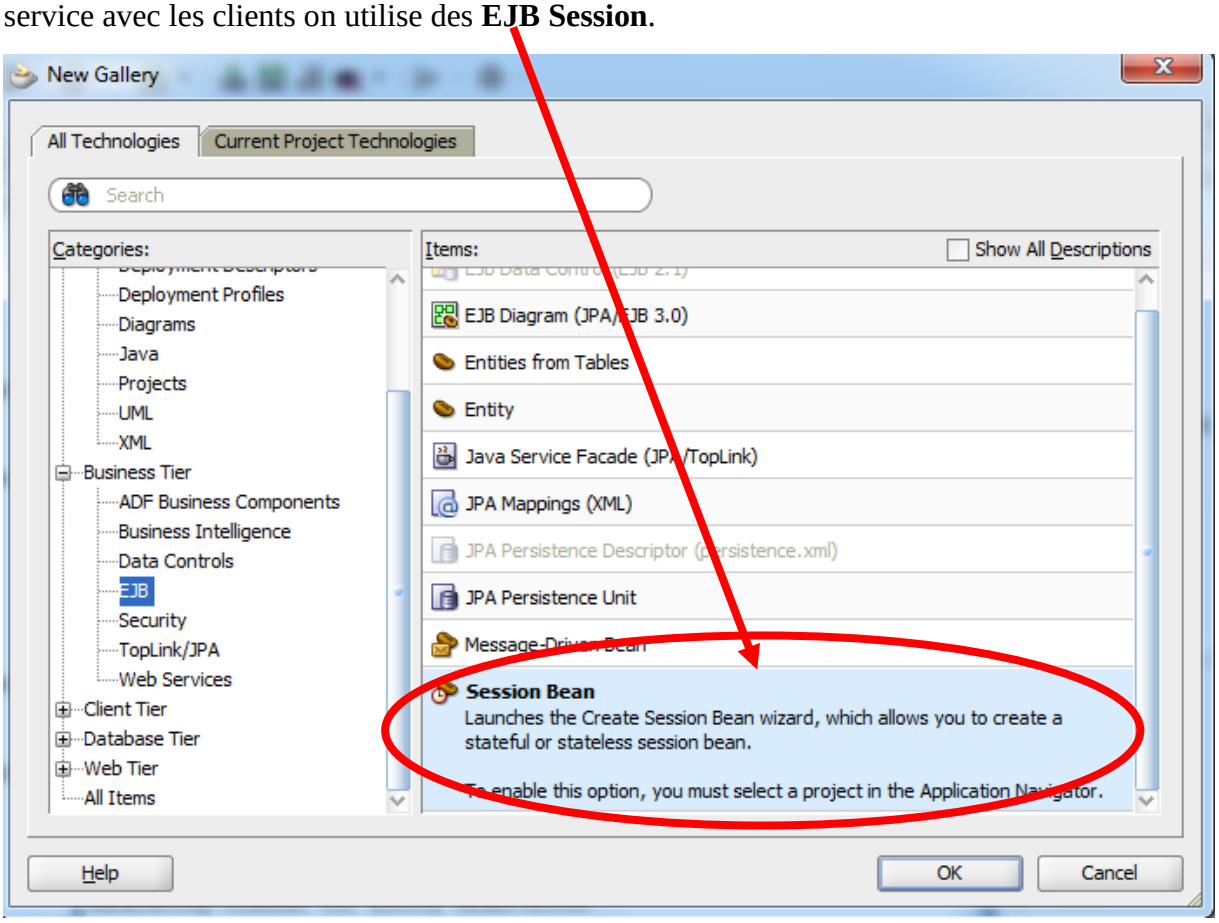


Figure B. 5. Création d'un Session Bean.

B.1.5 L'ajout des méthodes

Pour chaque EJB Session on ajout des **méthodes** qu'ils seraient des fonctions.

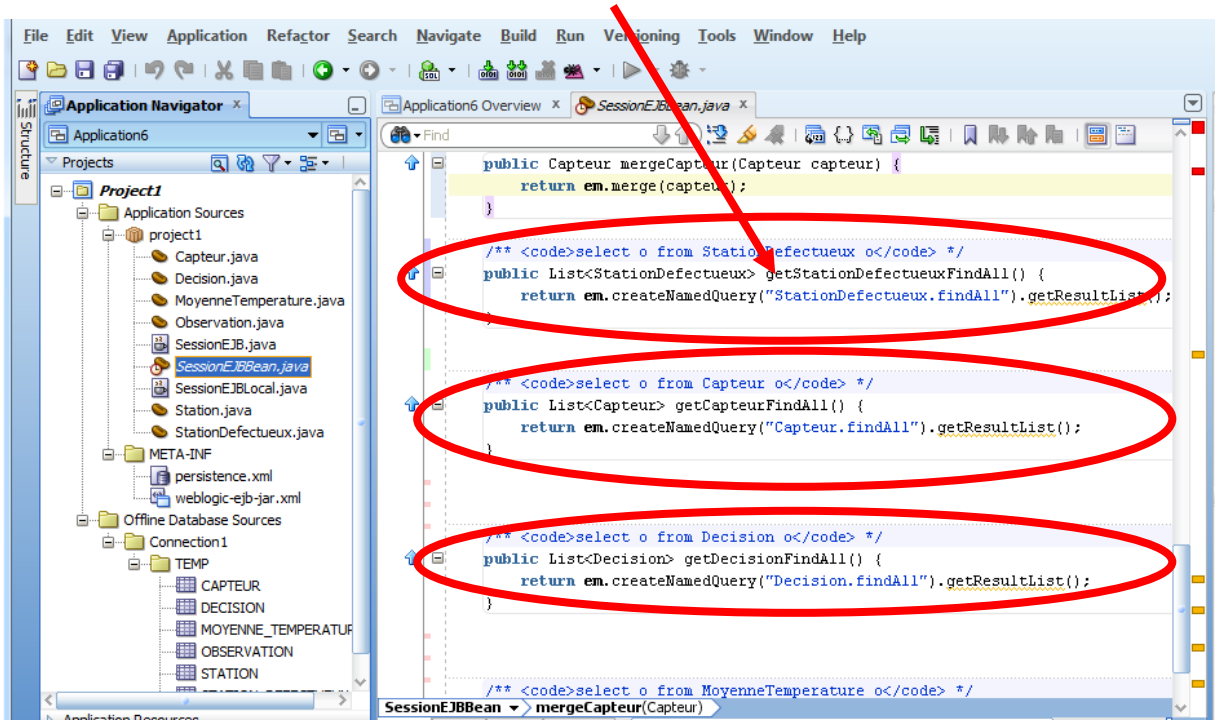


Figure B. 6. Syntaxe des méthodes.

B.2 Création de Web Service

Pour chaque EJB Session on cria un Web Service.

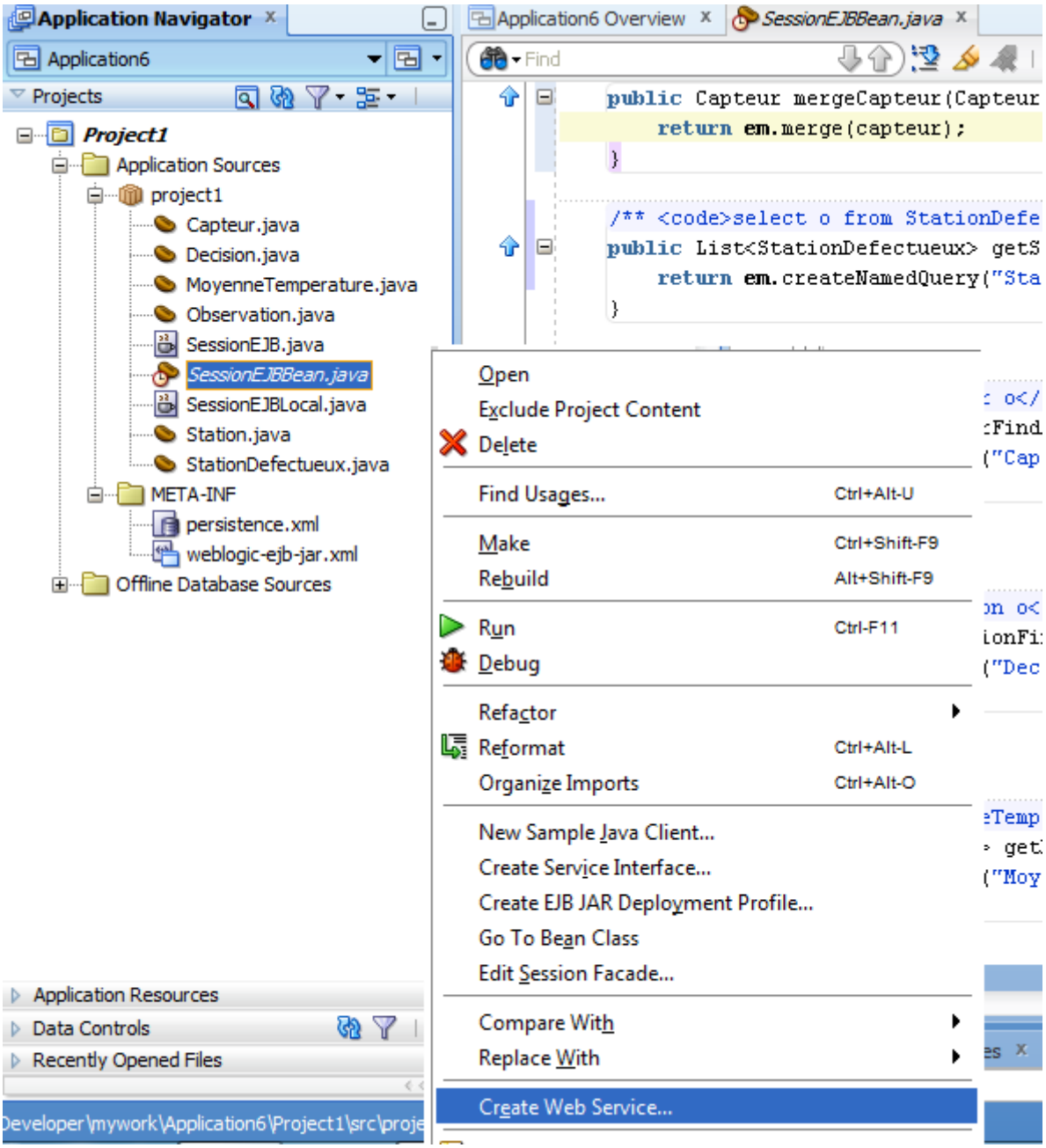


Figure B. 7. Création des Services Web.

B.2.1 Test de Web Service

Après la création du Web Service on fait le test de ce dernier, et le résultat de test est un **lien de WSDL** qu'il sera utilisé pour le proxy et le DataControl afin de permettre aux clients d'exploiter les méthodes d'EJB Session.

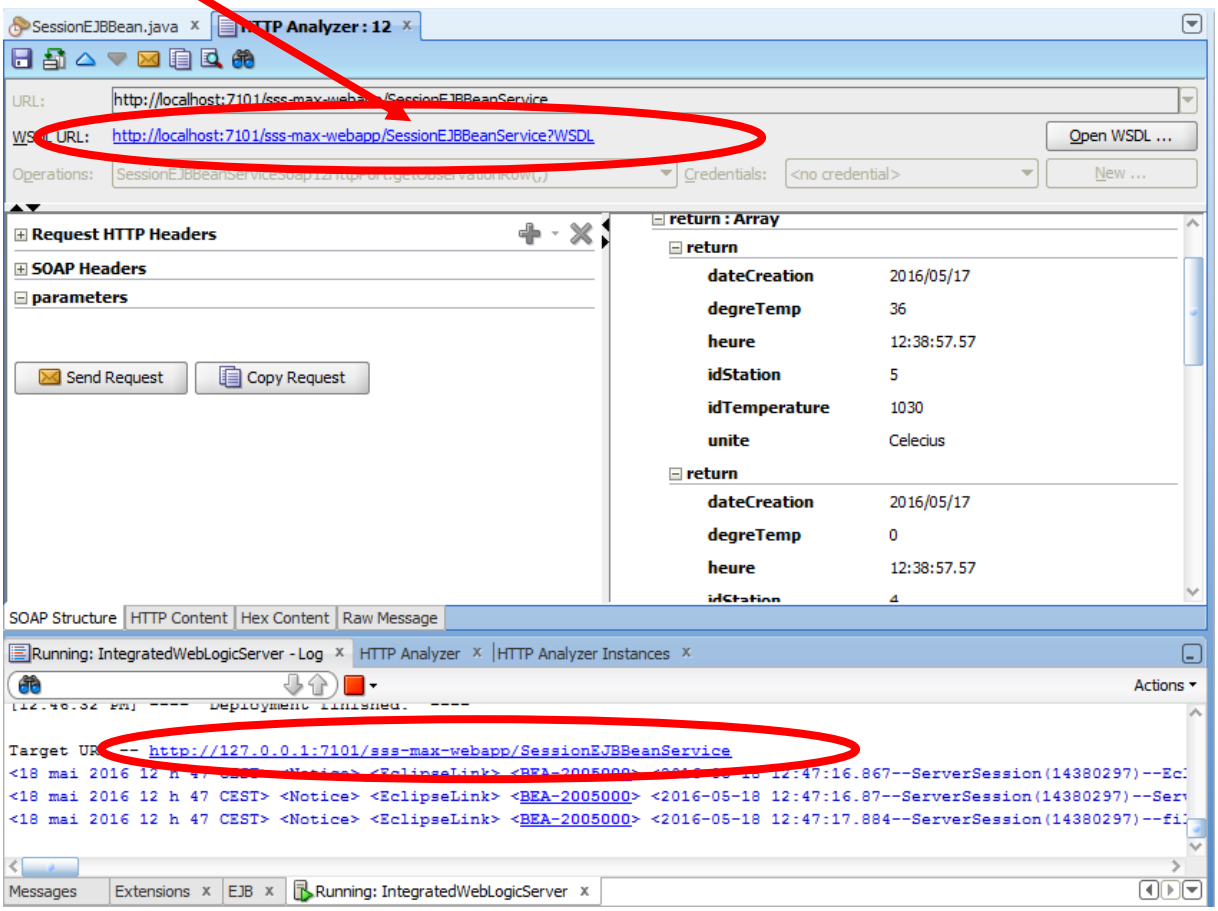


Figure B. 8. Résultat d'exécution de WSDL.

B.2.2 Création de Web Service Proxy

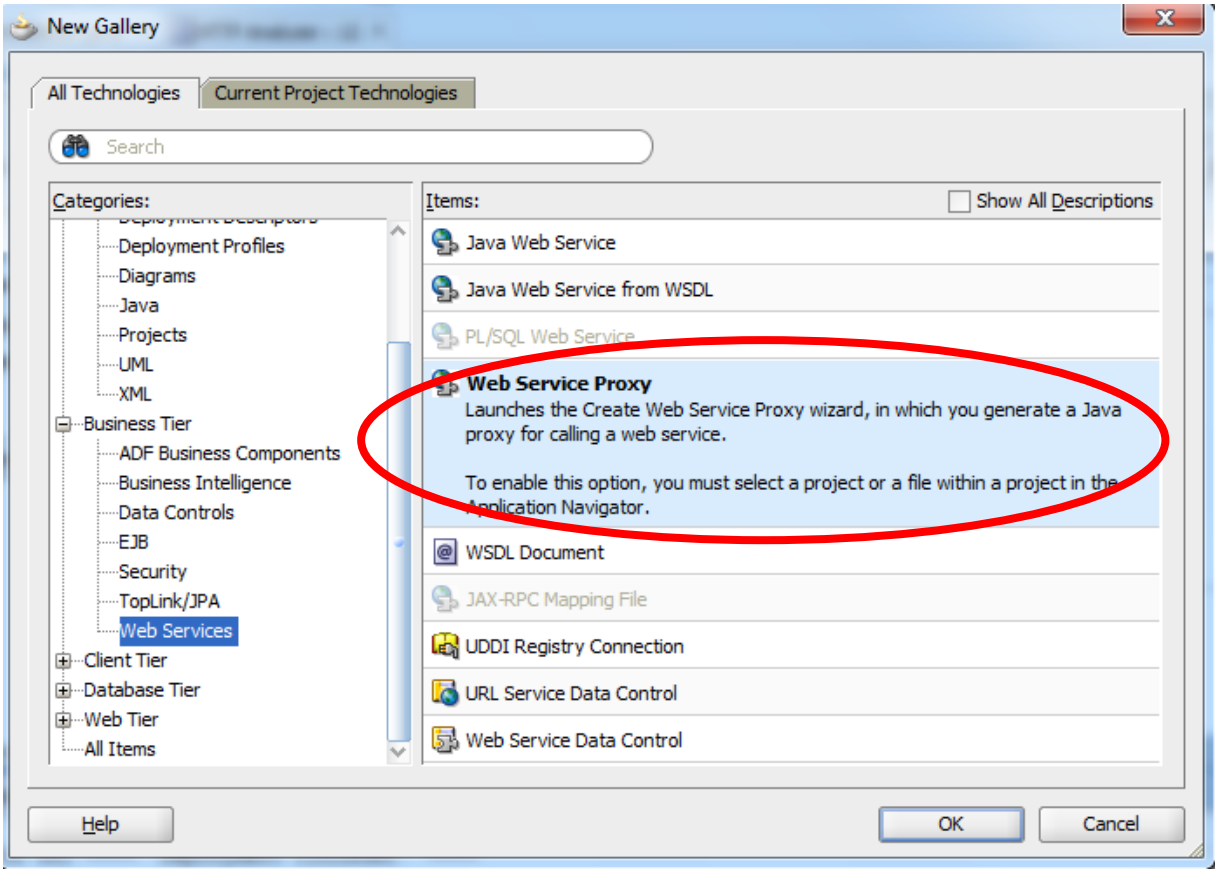


Figure B. 9. Création de Web Service Proxy.

B.2.3 Création de Service Data Control

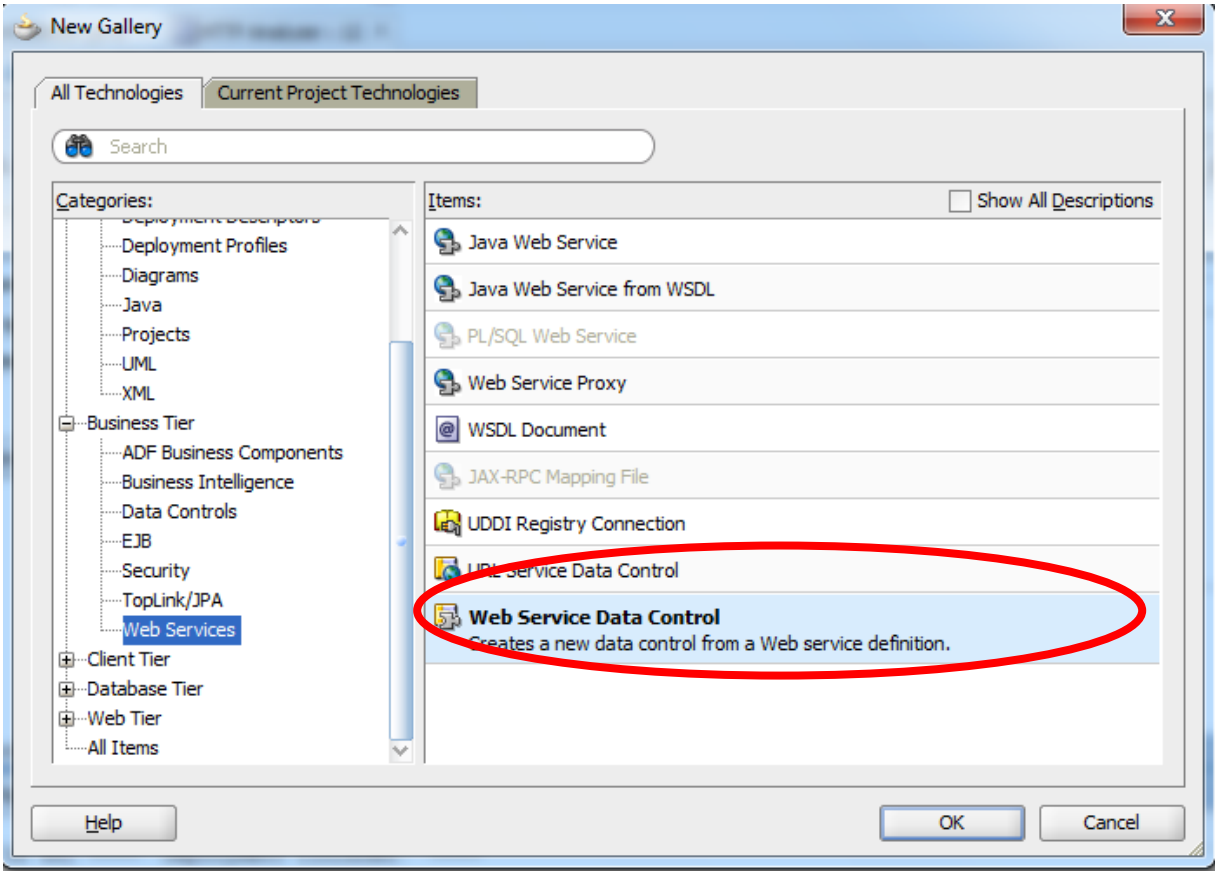
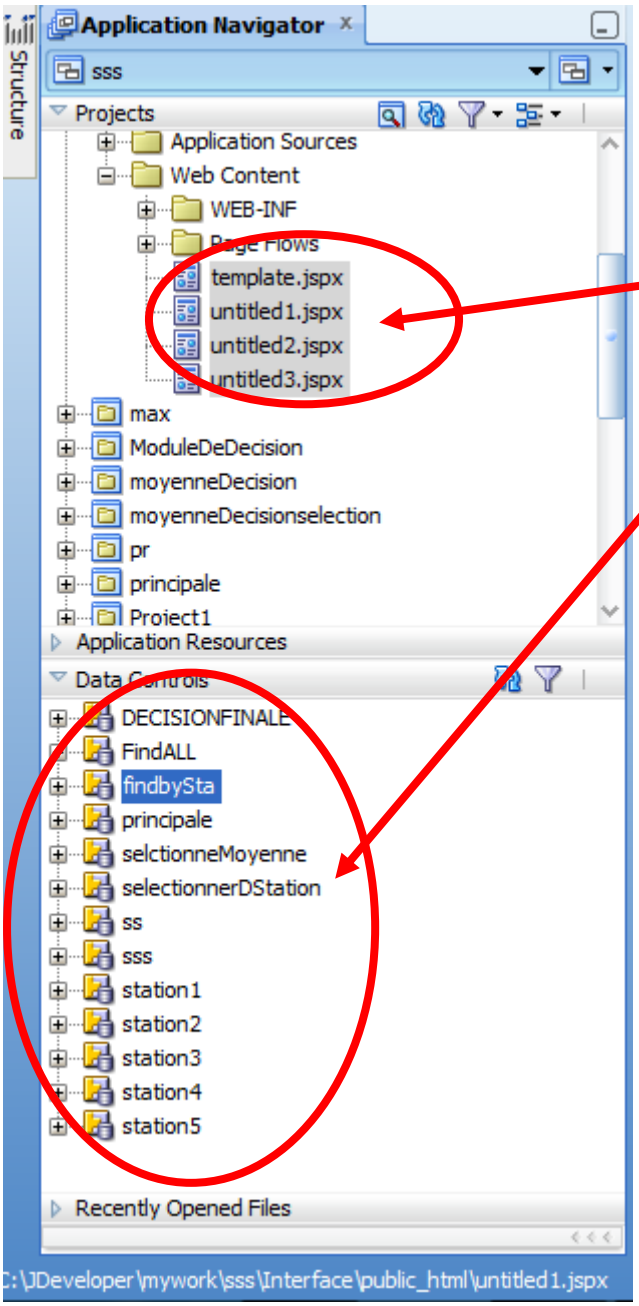


Figure B. 10. Création de Web Service Data Control.

B.3 Création de JSF



Les **fonctions** qui sont dans les data Contrôles sont utilisées par notre page **JSf**

Figure B. 11. Création d'un JSF.