Université Abou Bekr Belkaid
Tlemcen Algérie
جامعة أبي بكر بلقايد
تلمسان الجزائر

République Algérienne Démocratique et Populaire

Université Abou Bakr Belkaid– Tlemcen

Faculté des Sciences

Département d'Informatique

Mémoire de fin d'études

pour l'obtention du diplôme de master en Informatique

Option : Modéle d'information et de désicion (M.I.D)

Thème

Élaboration automatique d'une stratégie de tolérance aux fautes de services web basée sur la diversité

Réalisé par :

- Melle BOUAKKAZ khadidja
- Melle BELHARIZI Asmaa

Présenté le : 21 Juin 2015 devant le jury composé de.

- M. BENAMMAR .A (Président)
- Mme. ABDELDJELIL.H (encadreur)
- M. BENMOUNA.Y (Examinateur)
- M. KHELASSI.A (Examinateur)

Année universitaire : 2014-2015

REMERCIEMENTS

Louange à Dieu Miséricorde et Miséricordieux qui nous a donné la force, la volonté et la patience durant toutes nos années d'études. Nous remercions nos très chers parents qui nous ont accordé le courage, la volonté et le soutien pour bien réaliser ce travail.

Nous tenons à exprimer nos vifs remerciements à Mme ABDELDJELIL.H, notre encadreur ; c'est sous sa direction scientifique que ce travail de mémoire à été réalisé. Nous tenons à la remercier pour ses conseils, ses encouragements et sa orientation. Merci Madame pour votre gentillesse et votre patience. Qu'elle trouve aussi ici l'expression de notre profonde gratitude d'avoir initié et dirigé ce travail.

Nous adressons toute notre gratitude à Mr BENMOUNA.Y et Mr KHELASSI.A pour avoir très gentiment acceptés d'examiner ce travail.

Nous remercions également Mr BENAMMAR. A, chef de département d'informatique à l'université de Tlemcen. Il nous a fait un plaisir d'être examinateur de ce travail. Veuillez trouver ici de notre haute considération.

Enfin nos remerciements vont à toutes les personnes qui de près ou de loin nous ont aidés par leurs encouragements.

DÉDICACES

A nos chers parents

A nos familles

A tous nos amis

spécialement à

CHATER.M et BOUCHEKARA.M

Nous dédions ce mémoire.

Résumé

La capacité d'une application à continuer l'exécution en présence de fautes de composants réfère à la Tolérance aux fautes (TF). La TF qui est une des méthodes de sûreté de fonctionnement vise à accepter la faute avec une certaine déviation, mais elle ne permet pas de tolérer les défaillances causées par cette faute. Cette dernière repose sur des stratégies principales afin de garantir la fiabilité de ces composants, dans notre cas des services Web. Les services Web similaires c.à.d qui fournissent la même fonctionnalité mais qui sont implémentés différemment, sont regroupés au sein d'un même espace virtuel appelé Groupe de Diversité (GD). La détermination automatique de la stratégie de TF dans un GD et en présence de plusieurs critères conflictuels représente un grand challenge. Nous cherchons à prendre en considération des données plus complètes sur le mode de défaillance d'un service Web, de l'origine de fautes, des exigences clients et les paramètres de QoS. Les méthodes multicritères d'aide à la décision permettent de faciliter la tâche. L'approche proposée dans ce mémoire est basée sur Electre I.

Mots clés : Tolérance aux fautes, Services Web, Groupe de diversité, stratégies de TF, décision multi critères, Electre I.

Abstract

The Fault Tolerance (FT) is a dependability methods designed to accept fault with some deviation, but it does not tolerate failures caused by this fault. This is based on a set of strategies that ensure the reliability of these components, in our case Web services. Similar Web services that provide the same functionality but implemented differently, are grouped within the same virtual space called Diversity Group (DG). The automatic determination of FT strategy in a DG and in the presence of several conflicting criteria represents a great challenge. We aim to take into account complete data in terms of failure mode for Web service, faults origin, customer's requirements and QoS parameters. The methods of multi-criteria decision support can facilitate the task. The approach proposed in this paper is based on Electre I.

Keywords : Faults tolerant FT, Web Services, diversity group, FT strategies, multi-criteria decision, ElectreI

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction Générale

Sommaire

Ce chapitre est une introduction générale. Il décrit le contexte dans lequel s'inscrit ce travail et résume ensuite la problématique traitée et les contributions scientifiques proposées et mise en œuvre.

1.1 Contexte et Motivation

L'évolution d'Internet et la compétitivité entre les entreprises ont été les facteurs de l'explosion des services Web. En effet, les services web peuvent constituer un apport de rapidité et d'efficacité pour le e-business. La notion de service web désigne essentiellement une application (un programme) mise à disposition sur Internet par un fournisseur de service, et accessible par des clients à travers des protocoles d'Internet standards. Leurs particularités par rapport aux autres technologies de l'informatique répartie résident dans le fait qu'ils offrent un modèle de composants à couplage faible en utilisant la technologie Internet comme infrastructure pour la communication. Les services Web reposent sur une architecture orientée service. Celle-ci fait intervenir trois catégories d'acteurs : les fournisseurs de services (i.e. les entités responsables du service Web), les clients qui servent d'intermédiaires aux utilisateurs de services et les annuaires qui offrent aux fournisseurs la capacité de publier leurs services et aux clients le moyen de localiser leurs besoins en termes de services.

La dynamique entre ces trois acteurs inclut donc les opérations de publication, de recherche et de liens (binding) d'opérations. Cette dynamique est normalisée à travers 3 standards : un protocole abstrait de description et de structuration des messages (Simple Object Access Protocol SOAP), une spécification XML (Extensible Markup Language) qui permet la publication et la localisation des services dans les annuaires (Universal Description, Discovery and Integration of Web Services UDDI) et un format de description des services Web (Web Services Description Language WSDL). Un service WSDL est composé d'un ensemble d'opérations élémentaires, chacune décrite par un flux de messages échangés entre le client et le service.

Les architectures orientées services (AOS) sont mises en œuvre basé essentiellement sur des protocoles et langages standards. De telles architectures offrent plusieurs avantages dont le couplage faible et l'interopérabilité standardisée des applications. Le couplage faible permet à une application d'invoquer des services autonomes et déployés sur des infrastructures différentes grâce à des protocoles de communication. Il n'y a nul besoin de disposer d'un système de middleware réparti commun comme c'est dans le cas des approches orientées objets. L'interopérabilité permet à une application cliente d'interagir avec un service distant malgré l'hétérogénéité des plateformes et des langages dans lesquels les services ont été développés.

Les applications orientées services peuvent nécessiter une haute sûreté de fonctionnement[2].

Celle-ci est définie comme la caractéristique qui vise à pouvoir placer une confiance justifiée dans les services offerts par une application. Il est tout d'abord important de souligner la différence entre les concepts de défaillance, erreur et faute. Une défaillance se produit lorsque le service délivré dévie de ce qui est attendu. La partie du système susceptible d'avoir causée la défaillance est appelée erreur. Une faute est la cause adjugée ou supposée d'une erreur. La sûreté de fonctionnement [2] d'une application est définie selon plusieurs dimensions : la disponibilité, la fiabilité, la sécurité, et la confidentialité. Dans notre travail, nous nous limiterons aux deux premières dimensions. La disponibilité d'une application correspond à sa capacité d'être atteint pour délivrer des résultats. La fiabilité d'une application correspond à sa capacité de continuer son fonctionnement en présence de fautes. La sûreté de fonctionnement est généralement accomplie par différentes méthodes, nous nous sommes intéressés aux aspects de tolérance aux fautes des applications à base de services. Il est naturellement envisageable que la défaillance au sein d'un de leurs composants non convenablement traitée peut se propager et peut par conséquent perturber le service fourni par l'application. De plus, chaque service web participant à une application donnée introduit un potentiel point de défaillance.

La conception et le déploiement de services tolérants aux fautes constituent donc notre principale motivation en vue d'aboutir à des applications à haute sûreté de fonctionnement.

1.2 Problématique

Assurer la disponibilité et la fiabilité des applications à base des services constitue un réel challenge. La tolérance aux fautes des services, atomiques ou composite, constitue une des principales approches de sûreté fonctionnement. La méthode de tolérance aux fautes est basée sur la duplication, soit par la réplication qui réalise des copies exactes du service d'origine, soit par la diversité qui consiste à recourir à des services différents mais sémantiquement équivalents. La réplication d'un service est simple à mettre en œuvre mais malheureusement elle ne permet pas de faire face à certaines fautes inhérentes au service lui-même. Ces fautes se retrouveront naturellement dans toutes les répliques du service concerné. La diversité présente un avantage majeur par rapport à la réplication due à l'absence des fautes répliquées. Cependant, contrairement à la réplication où les différents stratégies de réplication (active et passive) sont bien définis et formalisés, très peu de travaux ont été consacrés aux mécanismes de tolérance aux fautes à base de diversité dans les applications à base de services. Notre travail adopte donc la notion de diversité et vise à déterminer d'une façon automatique la meilleure stratégie à utiliser dans un groupe de diversité. Les stratégies variées entre séquentielle, parallèle avec vote et parallèle sans vote (voir cette sélection est basée sur le comportement de service Web dans des exécutions précédentes (historique) :

– Mode de défaillances (défaillances de type Byzantine, défaillances de type crash).

– Qos (coût, Temps)

En plus du comportement de SW dans l'historique, nous devrons prendre en considération les exigences clients en terme de chaque critère.

Plus particulièrement, nous devons répondre aux questions suivantes :

$\mathbf{Q}_1$: Comment réalisé un compromis entre les différents critères conflictuels ?

$\mathbf{Q}_2$: Étant donné des services Web similaires regroupés au sein d'un même espace virtuel appelé groupe de diversité, Quelle est la meilleure stratégie à utiliser dans ce groupe ?

1.3 Contributions

Les contributions de ce mémoire se présentent comme suit :

- $\mathbf{C}_1$ **: Modélisation de la problématique par une méthode de décision multi critères ElectreI (ÉLimination Et Choix Traduisant la REalité)** :

L'élaboration d'un mécanisme de sélection automatique de la stratégie de tolérance aux fautes (recouvrement) représente un vrai challenge. Particulièrement, si on est dans un environnement conflictuel, où plusieurs critères sont à prendre en considération. Dans ce cadre, notre première contribution est la modélisation de notre problématique avec une des méthodes de décision multi critères à savoir, Electre I.

- $\mathbf{C}_2$ **: Proposition d'un algorithme de sélection de la meilleure stratégie de tolérance aux fautes (TF) dans un environnement conflictuel** :

Il s'agit essentiellement de déterminer d'une manière automatique pour un groupe de diversité la stratégie nécessaire de tolérance aux fautes. Nous avons proposés un algorithme qui permet de choisir une stratégie parmi trois stratégies, cette détermination répond aux comportement de SW dans les exécutions précédentes et aux exigences clients, en se basant sur Electre I.

1.4 Organisation du mémoire

Le manuscrit est structuré comme suit :

- Le Chapitre 2 décrit dans un premier temps des notions fondamentaux relatives aux domaines des architectures orientées services et de la tolérance aux fautes. Il présente ensuite les principales approches de la sûreté de fonctionnement dans les services web. Et enfin, nous présentons quelques travaux existants concernant les stratégies de réplication.

- Le Chapitre 3 est lié à notre première et deuxième contribution, nous présentons notre approche qui vise à assurer le bon fonctionnement d'un ensemble de services Web équivalents, tout en déterminant la stratégie de TF à utiliser pour un groupe, avec une implémentation de l'algorithme.

Pour finir, une conclusion générale reprendra nos contributions dans ce mémoire. Nous proposerons enfin quelques perspectives de recherche de ce travail.

Chapitre 2

Concepts fondamentaux et État de l'art

Sommaire

2.1 Introduction

Ce chapitre a pour objectif de présenter le cadre de départ de nos travaux. Dans un premier temps, nous introduisons les concepts de base des services web et de la sûreté de fonctionnement. Nous présentons ensuite l'état de l'art des travaux réalisés sur la tolérance aux fautes dans les services web. Enfin, nous discutons les différentes approches pour se positionner par rapport aux solutions existantes.

2.2 Généralités

2.2.1 Les Architectures Orientées Services (AOS)

Une architecture orientée service (en anglais Service Oriented Architecture) fait référence à un style de conception de systèmes distribués qui exposent leurs fonctionnalités en services en se focalisant d'avantage sur la notion du faible couplage entre les services [27]. L'AOS est un modèle d'exécution des applications logicielles distantes. Cette architecture se caractérise par le concept de service entre applications qui échangent à travers des messages. Un service est le plus petit traitement qu'une application offre à une autre, généralement à la demande de celle-ci. L'application qui fournit un service est appelée fournisseur ou prestataire de service et l'application qui utilise le service est appelée cliente de service. Une application peut jouer à la fois le rôle de fournisseur de service et le rôle de client de service. L'accomplissement d'un service par un fournisseur de service à la demande d'un client de service est régi dans un contrat de service. Le contrat de service est spécifié par le fournisseur de service. Il s'agit d'un document qui définit :

- Les fonctions du service.
- L'interface du service.
- La qualité du service.

On désigne par fonction de service une description abstraite de l'offre de service. L'interface de service est une description des mécanismes et des protocoles de communication avec le prestataire de service. On désigne par qualité de service les détails de fiabilité, de disponibilité et de robustesse, etc. [13]

2.2.2 Services Web

Plusieurs définitions des services Web ont été mises en avant par différents auteurs. Ci-dessous, nous citons une définition généralement acceptée et fournie par le consortium W3C :

2.2.2.1 Définition

"A Web service is a software system designed to support interoperable machine-to machine interaction over a network. It has an interface described in a machine processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards".[32]
Une étude plus approfondie des points communs partagés par les différentes définitions et par les usages qui sont faits des services Web, permet de dégager au moins deux principes fondamentaux :

- Les services Web interagissent au travers d'échanges de messages encodés en XML.

- Les interactions dans lesquelles les services Web peuvent s'engager sont décrites au sein d'interfaces.

La définition W3C [31] restreint la portée des interfaces des services Web aux aspects fonctionnels et structurels, en utilisant le langage WSDL qui essentiellement ne permet de décrire que des noms d'opérations et des types de messages.

2.2.2.2 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 de message XML. WSDL (Web Services 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 et d'échange d'informations entre un client et un prestataire.

Les prestataires des Services Web, leurs interfaces et leurs points d'accès, peuvent être enregistrés, découverts et localisés via des technologies d'annuaire comme UDDI (Universal Description, Discovery and Integration of Web Services). Autant un standard ouvert (non propriétaire) sur les annuaires de services semble indispensable, surtout pour la mise en œuvre d'architectures dynamiques, autant la technologie UDDI, qui est clairement une technologie de Services Web, n'est pas encore formellement considérée aujourd'hui comme le standard des annuaires.

WSDL, SOAP et UDDI (FIGURE 2.1) constituent l'ensemble des technologies clés de Services Web, sur lesquelles d'autres technologies plus proches de la problématique applicative peuvent être spécifiées et mises en œuvre. [7]

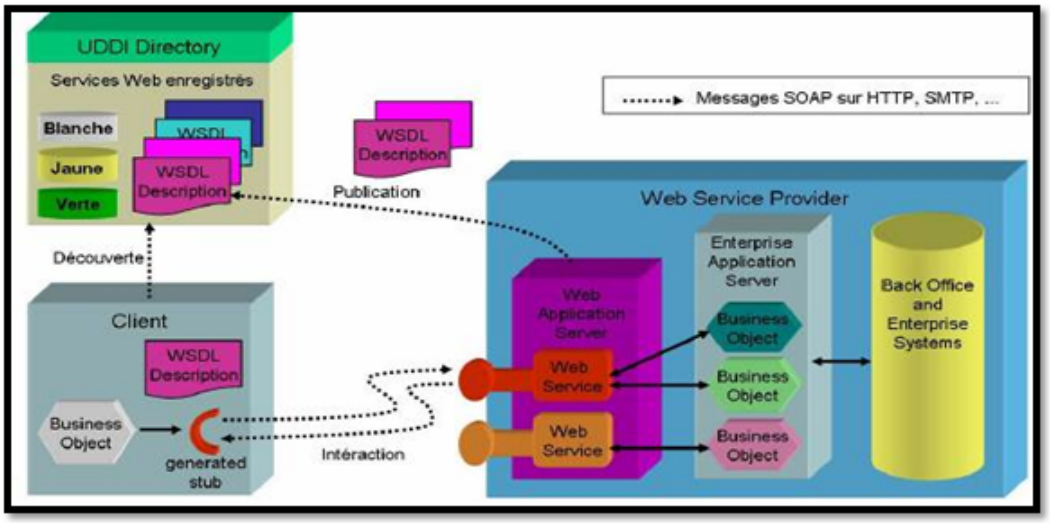


FIGURE 2.1 – Les protocoles de base des Services Web

2.2.2.3 Les principaux protocoles

a. XML

XML (Extensible Markup Language) est un langage de balisage extensible [30] 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, appelés aussi dialectes 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) jusqu'aux documents les plus complexes, son potentiel est de devenir le standard universel et multilingue d'échange d'informations.

XML Namespaces est une extension de la recommandation XML qui permet de créer des espaces de nommages. Les espaces de noms d'XML permettent de qualifier de manière unique des éléments et des attributs. On sait alors à quel domaine de définition se rapporte un objet et comment il doit être interprété, selon sa spécification. Différencier des espaces de noms permet de faire coopérer, dans un même document, des objets ayant le même nom, mais une signification différente, souvent liée à un modèle de contenu différent. Cette spécification est une avancée importante car, à partir du moment où beaucoup de formats s'expriment selon XML, les risques de "collision de noms" deviennent plus importants et cette spécification prend alors toute son importance. [12]

b. WSDL

Le WSDL est un langage qui permet de décrire les services web, et en particulier, leurs interfaces. Ces descriptions sont des documents XML. WSDL décrit un service web en deux étapes fondamentales : une abstraite et une concrète. Dans chaque étape, la description utilise un nombre de constructions pour favoriser la réutilisation de la description et pour séparer les préoccupations de conception indépendantes (Voir La FIGURE 2.2)

Au niveau abstrait, WSDL décrit un service Web en termes des messages qu'il envoie et reçoit ; les messages sont décrits de façon indépendante d'un format spécifique en utilisant un système de types, typiquement un schéma XML. La partie abstraite est composée de définitions de "port type" qui sont analogues aux interfaces des "middleware" traditionnel. Chaque "port type" est une collection logique d'opérations. Une "opération" associe un modèle d'échange de message à un ou plusieurs messages. Un message est une unité de communication avec un service web. Il représente les données échangées dans une unique transmission logique. Un modèle d'échange de messages identifie l'ordre et la cardinalité des messages envoyés et/ou reçus. Une interface regroupe un ensemble d'opérations. Au niveau concret, un "binding" indique des détails de format de transport pour une ou plusieurs interfaces. Un "endpoint" (point final) associe une adresse de réseau à un "binding" (attache). Finalement, un service groupe un ensemble d'endpoints qui implémente une interface commune. [31]

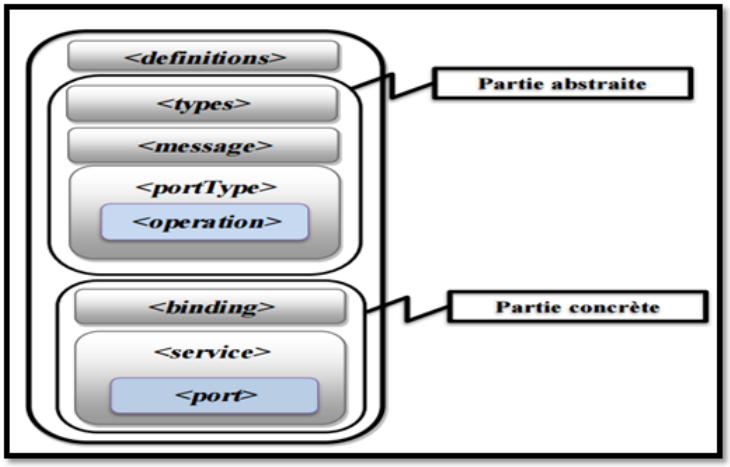


FIGURE 2.2 – Les éléments de fichier WSDL

c. SOAP

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

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. Il est toujours constitué d'une enveloppe (SOAPENV : Enveloppe) munie d'une en-tête (SOAP-ENV : Header) optionnelle et d'un élément (SOAP-ENV : Body) obligatoire contenant le corps du message, suivis d'éventuels éléments applicatifs spécifiques. La spécification considère explicitement que les en-têtes SOAP sont destinés à la mise en œuvre de couches supérieures et transversales de la technologie des services Web, comme la gestion des transactions, la gestion de la sécurité, etc [35]. La FIGURE 2.3 présente la structure d'un message SOAP.

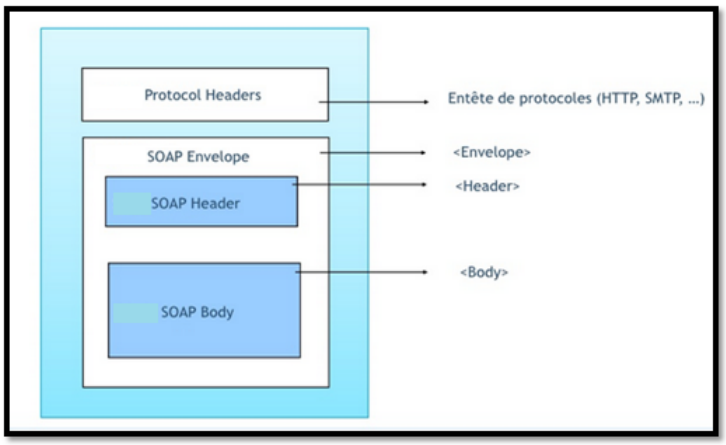


FIGURE 2.3 – Structure d'un message SOAP

d. UDDI

UDDI (Universal Description, Discovery and Integration) 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(interface spécifique à l'annuaire accédé), soit par programme en utilisant l'API (Application Programming Interface) définie par la spécification.Un annuaire UDDI peut être public ou privé.

De manière standard, la spécification prévoit qu'un annuaire est distribué sur plusieurs nœuds. Ces nœuds sont synchronisés au moyen du mécanisme de réplication interne. La réplication n'est pas obligatoire, notamment dans un cadre privé, mais est fortement recommandée pour des raisons

évidentes de disponibilité. Cette fonctionnalité est, bien sûr, mise en œuvre par l'annuaire public UDDI, dont les implémentations des opérateurs (IBM, Microsoft, NTT Communications et SAP) se répliquent entre elles. [29]

2.2.2.4 Cycle de vie des services Web

Chaque service Web est défini par un fournisseur. Le fournisseur de services déploie et publie la description de son service dans des registres en vue d'être localisé par des clients.

- Les clients localisent leurs besoins en terme de services en effectuant des recherches sur les registres de services Web .

- Une fois le service localisé, le client extrait sa description du registre .

- Sur la base des informations définies dans la description du service, le client entreprend une interaction (FIGURE 2.4).

- SOAP : « assure la communication avec et inter services Web »

- WSDL : « offre un schéma formel de description des services Web »

- UDDI : « offre une manière uniforme de définir des registres des services Web et aussi un schéma uniformément extensible de descriptions des services Web. »

Actuellement, SOAP, WSDL et UDDI sont les trois standards qui constituent l'architecture Services Web. Ils résolvent les problèmes de l'hétérogénéité des systèmes pour l'intégration d'applications en ligne.

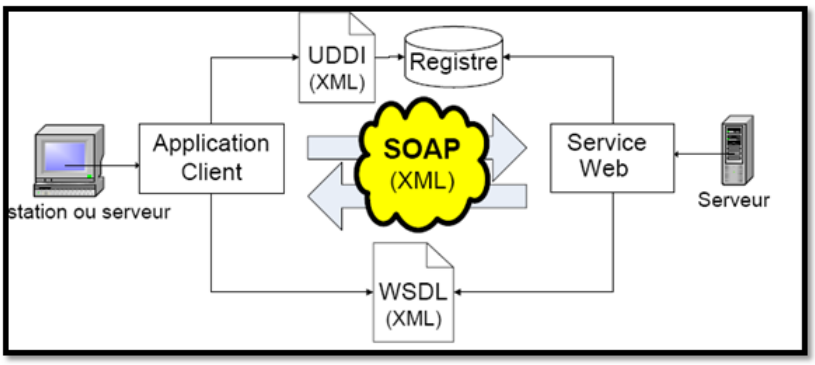


FIGURE 2.4 – Cycle de vie de service Web

L'architecture orientée service nécessite une haute sûreté de fonctionnement qui nous décrivons par la suite :

2.3 La sûreté de fonctionnement

Cette section à pour objectif d'introduire les concepts liés au domaine de la sûreté de fonctionnement permettant de cerner plus précisément la tolérance aux fautes. Selon Avizienis et al. [2] La sûreté de fonctionnement est la capacité de placer une confiance justifiée dans le service délivré. La notion de service délivré par un système correspond à son comportement perçu par ses utilisateurs (i.e. autre système en interaction avec celui-ci).

Dans la suite nous présentons les concepts liés à la sûreté de fonctionnement (attributs, entraves et moyens) illustrés par la FIGURE 2.5 avant de s'intéresser à l'un de des moyens en particulier qui est : La tolérance aux fautes.

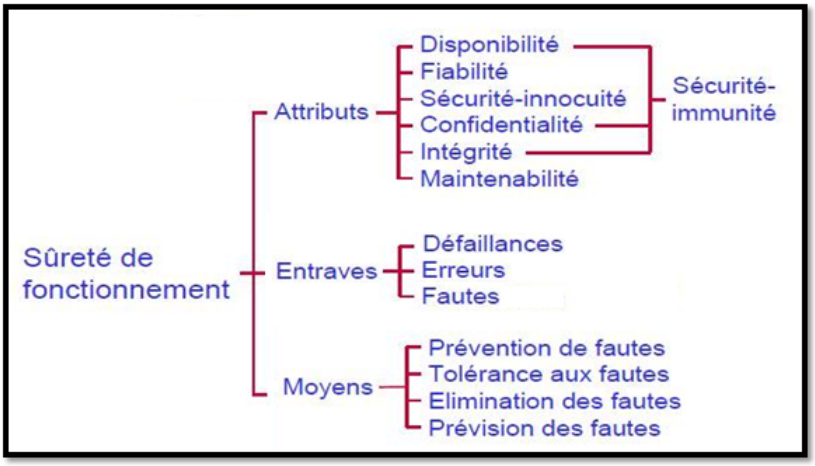


FIGURE 2.5 – L'arbre de la sûreté de fonctionnement

2.3.1 Attributs de la sûreté de fonctionnement

Les attributs de sûreté de fonctionnement sont définis pour exprimer les propriétés de sûreté de fonctionnement du système. L'importance de chacun de ces attributs est relative aux applications auxquelles le système est destiné. On peut distinguer :

- La disponibilité : le fait que le système soit prêt à l'utilisation à tout moment.
- La fiabilité : la continuité du service en dépit de fautes.
- La sécurité-innocuité : la non-occurrence de conséquences catastrophiques pour l'environnement.
- La confidentialité : la non-occurrence de divulgations non-autorisées de l'information.
- L'intégrité : la non-occurrence d'altérations inappropriées de l'information.
- La maintenabilité : l'aptitude aux réparations et aux évolutions. [17]

2.3.2 Entraves à la sûreté de fonctionnement

Les entraves de la sûreté de fonctionnement sont les circonstances indésirables mais non inattendues c.à.d causes ou résultats de la non-sûreté de fonctionnement. Dans l'ensemble des entraves, on distingue **la défaillance** du système qui survient lorsque le service délivré dévie de l'accomplissement de la fonction du système, c.à.d à quoi le système est destiné. **Une erreur** est la partie de l'état du système qui est susceptible d'entraîner une défaillance, c'est-à-dire qu'une défaillance se produit lorsque l'erreur atteint l'interface du service fourni, et la modifie. **Une faute** est la cause adjugée ou supposée d'une erreur. On voit donc qu'il existe une chaîne causale entre faute, erreur et défaillance, représentée dans la FIGURE 2.6.

Notons que dans les systèmes auxquels nous nous intéressons, qui sont formés par l'interaction de plusieurs sous-systèmes, les notions de faute, d'erreur et de défaillance sont récursives : la défaillance d'un sous-système devient une faute pour le système global la FIGURE 2.7.

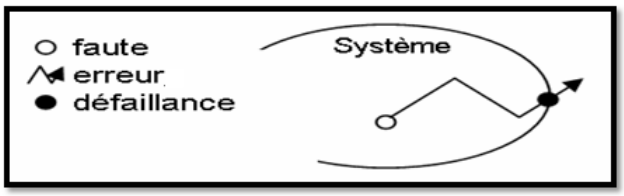


FIGURE 2.6 – Chaîne causale entre faute, erreur et défaillance

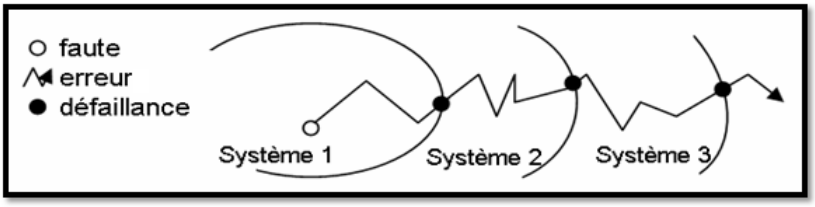


FIGURE 2.7 – Récursivité de la chaîne causale faute => erreur => défaillance

2.3.3 Moyens pour 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. Le développement d'un système sûr de fonctionnement passe par l'utilisation combinée de ses méthodes qui peuvent êtres classées en :

-**La prévention des fautes :** qui permet de limiter l'introduction de fautes pendant la phase de

développement.

-**La tolérance aux fautes :** qui a pour objectif d'éviter la défaillance du système en utilisant des techniques de détection ou de recouvrement d'erreur.

-**L'élimination des fautes :** qui réduit la présence (nombre, sévérité) des fautes. Pendant la phase de développement, elle consiste à vérifier, diagnostiquer et corriger les fautes. Lors de l'utilisation, il s'agit de maintenance corrective ou préventive.

-**La prévision des fautes :** qui estime la présence et les conséquences de fautes par des tests prenant en compte leurs activités et leurs occurrences.

La prévention des fautes et la tolérance aux fautes peuvent être vues comme des moyens d'obtention de la sûreté de fonctionnement, c'est-à-dire comment procurer au système l'aptitude à fournir des services conformes aux fonctions attendues.

L'élimination des fautes et la prévision des fautes peuvent être vues comme constituant les moyens de la validation de la sûreté de fonctionnement, c'est-à-dire comment avoir confiance dans l'aptitude du système à fournir des services conformes à l'accomplissement de sa fonction. [17]

2.3.4 Classification des défaillances

Avizienis et al, [2] classifient les défaillances selon plusieurs points de vue (voir FIGURE 2.8) :

• **Domaine :** Il permet de spécifier quatre types de défaillances selon le comportement du composant :

-*Défaillance franche (crash) :* Le composant cesse toute interaction avec les autres composants du système en dehors des fenêtres temporelles attendues. La notion d'interaction dépend du modèle de système considéré (e.g. appel de procédure, envoi de message, écriture dans une mémoire partagée).

-*Défaillance temporelle (Omission) :* Le composant intéragit avec les autres composants du système en dehors des fenêtres temporelles attendues. Cela concerne aussi bien des interactions ayant lieu trop tard (i.e. échéance manquée) ou trop tôt. La défaillance par crash est une défaillance temporelle, dans le sens ou toutes les interactions seront réalisées trop tard (ou jamais en l'occurrence).

-*Défaillance en valeur :* Le composant interagit avec les autres composants du système avec des valeurs incorrectes.

-*Défaillance Byzantine :* Le composant défaille de manière arbitraire. Ce défaillance regroupe tous les modes de défaillance ainsi que leurs combinaisons (e.g. valeur erronée trop tard).

• **Détectabilité :** Elle réfère à la propriété qu'une défaillance soit signalée ou non à L'utilisateur du système. Un mécanisme de détection permet par exemple de vérifier l'exactitude du résultat délivré. La signalisation d'une défaillance peut être communiquée à tord à l'utilisateur. La signalisation est alors dite fausse alerte.

• **Consistance (cohérence) :** Les défaillances sont dites consistantes si le service incorrect est perçu de façon identique par tous les utilisateurs du système, sinon elles sont inconsistantes. Par exemple, les défaillances d'omission sont consistantes contrairement aux défaillances byzantines.

• **Conséquence :** Les Conséquences de défaillance sont classées selon leur niveau de gravite (ou sévérité) dépendant largement du type d'application. En général, deux niveaux sont distingués (i.e. mineur et catastrophique) et se définissent par rapport au bénéfice fourni par le service rendu en absence de défaillance, et les conséquences de défaillances. Les défaillances sont dites mineures lorsque leurs conséquences ont un coût similaire à celui de la prestation du service correct. Les défaillances sont dites catastrophiques lorsque le coût de la défaillance est conséquent (e.g. perte humaine, crash financier).

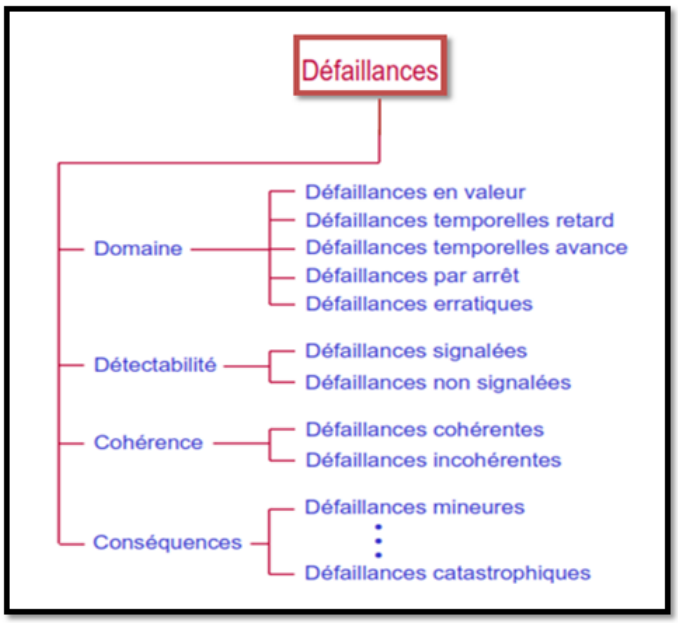


FIGURE 2.8 – Classification des défaillances dans les services web

2.3.5 Classification des fautes

Avizienis et al, [2] classifient les fautes élémentaire selon huit point de vue : Phase de création, frontière du système, cause et persistance de la faute, dimension du système, objectif du concepteur/ développeur selon son intention et sa compétence (voir FIGURE 2.9).

Les fautes peuvent être soit introduites au moment de la conception et du développement du sys-

tème (appelées fautes de conception) soit apparaitre au moment de l'exécution du système (appelées fautes opérationnelles).

Les fautes sont internes (resp. externes) si elles sont produites à l'intérieur (resp. extérieur) du système. la cause des fautes est naturelle si elle sont provoquées par un phénomène naturel. Elle est non-naturelle si l'être humain intervient dans le processus de création / occurrence de fautes. Les fautes persistent soit temporairement (dites transitoires) soit définitivement (dites permanentes). Par exemple, les composants défaillants ré-fonctionnent normalement après la disparition des causes de la faute.

Les fautes peuvent être aussi définies par rapport à l'environnement de développement ou d'exécution (appelé dimension) du système. Ainsi, les fautes peuvent être soit de nature logicielle (e.g. composant défaillants) soit matérielle (e.g., machines défaillance).

Les fautes introduites par des humains sont malicieuses si l'objectif de ces derniers est de nuire au système tel que : accéder à des données confidentielles, ou dégrader le fonctionnement du système. Elles sont non-malicieuses dans le cas contraire.[2]

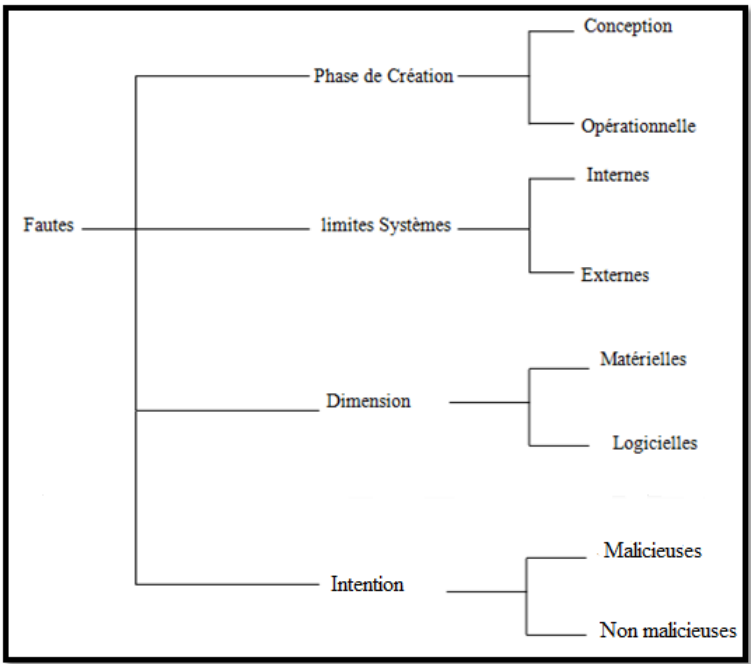


FIGURE 2.9 – Classification des Fautes

2.3.6 La tolérance aux fautes

Nous nous sommes intéressé dans ce mémoire à l'une de méthode de sûreté de fonctionnement qui est la tolérance aux faute (TF). Elle consiste à éviter la défaillance du système en utilisant des mécanismes de détection et recouvrement/compensation d'erreur causée par la défaillance d'un

ou plusieurs composants (considérée comme des fautes pour système globale). Il est essentiel de bien cerner la nature des faute que l'on cherche à tolérer afin de construire des mécanismes de tolérance qui contiennent la diversité appropriée pour être efficace. Dans ce mémoire, nous nous intéressons aux défaillances de type crash et de type byzantine que nous détaillons dans le chapitre 3.

2.3.6.1 Principe de la tolérance aux fautes

La tolérance aux fautes s'effectue le plus souvent en deux temps :

• La détection d'erreurs provoque la levée d'un signal ou d'un message d'erreur à l'intérieur du système.

• Le rétablissement du système, déclenché par le signal ou message, substitue un état jugé exempt d'erreurs et de fautes à l'état erroné détecté.

2.3.6.2 Techniques de 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. La FIGURE 2.10 liste les techniques de tolérance aux fautes (Détection d'erreurs et Rétablissement du système).

a. Détection d'erreurs

Ils permettent de détecter la présence d'erreurs dans le système après activation d'une faute. Il existe deux catégories de mécanismes de détection d'erreurs :

•*La détection concomitante (concurrente) :* elle a lieu pendant l'exécution normale du système. Elle a pour objectif de détecter la présence d'une erreur afin d'y remédier. Dans un système compositionnel, les erreurs détectées sont le plus souvent les défaillances des composants constitutifs du système.

•*La détection préemptive (préventive) :* elle a lieu pendant des phases particulières de l'activité du Système, comme les phases de démarrage ou de maintenance par exemple : effectuée lors de suspensions du service, destinée à révéler d'éventuelle présence d'erreurs latentes ou de fautes dormantes.

b. Rétablissement du système

Le rétablissement du système peut être assuré par deux méthodes complémentaires : le traitement d'erreur, qui vise à corriger l'état erroné du système, et le traitement de faute, qui vise à

empêcher la faute à l'origine de l'erreur d'être à nouveau activée. Parmi les techniques de traitement d'erreur, on distingue :

•*La reprise (rollback) :* qui consiste à sauvegarder périodiquement l'état du système pour le ramener dans un état antérieur à la détection d'erreur ; les principaux inconvénients de la reprise sont la taille des sauvegardes, la difficulté d'effectuer des sauvegardes cohérentes, et le surcoût temporel nécessaire à leur établissement.

•*La poursuite (rollforward) :* qui consiste à créer un nouvel état à partir duquel le système peut continuer à fonctionner de façon acceptable, éventuellement en mode dégradé ; une forme limite de cette technique est la mise du système dans un état sûr, par exemple, le train est mis à l'arrêt.

•*La compensation :* qui consiste à utiliser des redondances présentes dans le système pour fournir un service correct en dépit des erreurs ; cette compensation peut être consécutive à une détection d'erreur (détection et compensation), ou appliquée systématiquement (masquage d'erreur).

Le traitement de faute est composé de quatre phases successives :

• Le diagnostic a pour but d'identifier la localisation et le type de la faute responsable de l'état erroné du système.

• L'isolation consiste à exclure la participation du composant erroné de la délivrance du service, par moyen physique ou logiciel.

• La reconfiguration cherche à compenser l'isolement du composant défaillant, soit en basculant sur des composants redondants, soit en réassignant ses tâches à d'autres composants.

• La réinitialisation vérifie et met à jour la nouvelle configuration du système.[20]

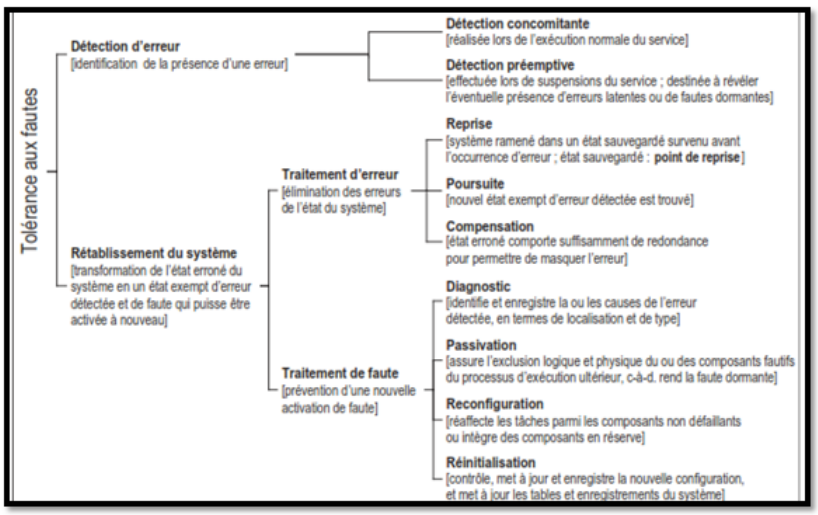


FIGURE 2.10 – Techniques de tolérance aux fautes

2.4 Mécanismes de tolérance aux fautes dans AOS

Dans cette section, nous nous intéressons aux mécanismes à la mise en œuvre de stratégies de tolérance aux fautes : recouvrement et compensation d'erreur dans les environnement de AOS.

2.4.1 Recouvrement d'erreur

Le recouvrement d'erreur peut être réalisé à travers deux méthodes, à savoir la réplication et la diversité.

2.4.1.1 Recouvrement par réplication

La réplication des services Web correspond à des copies exactes de service Web d'origine dépoliés sur différentes machines. La réplication de service web avec état peut être réalisée de plusieurs manières : active(ou machine à états) ; passive (ou copie primaire) ; semi active (i.e. combinaison des deux).

a. Réplication active

La réplication active se définie par la symétrie des comportements des copies d'un composant répliqué. Chaque copie joue un rôle identique à celui des autres.
PRINCIPE : La réplication active est définie ainsi :
- Réception des requêtes : toutes les copies reçoivent la même séquence.
- Traitement des requêtes : toutes les copies traitent les requêtes de manière déterministe.
- Émission des réponses : toutes les copies émettent les mêmes séquences de réponses. La FIGURE 2.11 illustre ce principe à l'aide d'un diagramme temporel. Les flèches horizontales représentent l'exécution de quatre composants : Client, S_1, S_2, S_3.

Les S_i sont les copies du composant répliqués S. Elles appartiennent à un même groupe. Lorsque le client invoque S, il envoie une requête (traits en flèche continu) à tous les S_i à l'aide d'un ABCAST assurant l'ordre total et la propriété d'atomicité. Chaque Si traite la requête(petit rectangle horizontal) et renvoie une réponse au client. [33]

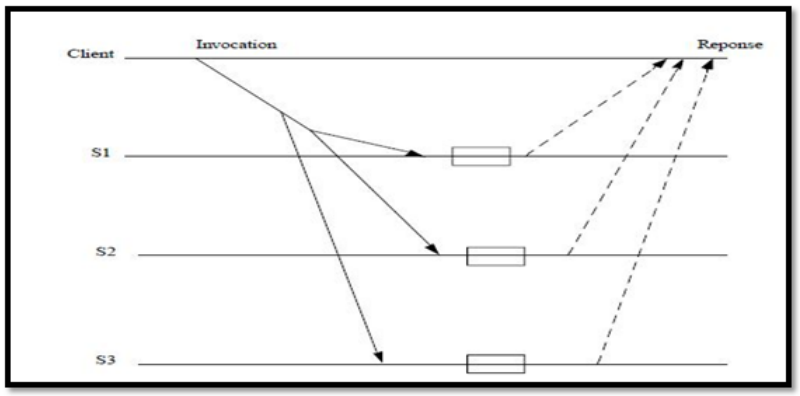


FIGURE 2.11 – Principe de la réplication active

b. Réplication passive

La réplication passive distingue deux comportements d'un composant répliqué : la copie primaire (primary copy) et les copies secondaires (backups). La copie primaire est la seule à effectuer tous les traitements. Les copies secondaires, surveillent la copie primaire. En cas de défaillance de la copie primaire, une copie secondaire devient la nouvelle copie primaire.

PRINCIPE : La réplication passive est définie ainsi :

- Réception des requêtes : la copie primaire est la seule à recevoir les requêtes.
- Traitement des requêtes : la copie primaire est la seule à traiter les requêtes.
- Émission des réponses : la copie primaire est la seule à émettre les réponses.

La FIGURE 2.12 illustre ce principe. Le client envoie la requête uniquement à la copie primaire S_1.

Celle-ci traite la requête, construit un point de reprise et l'envoie à l'aide d'un multicast fiable assurant l'ordre FIFO, aux copies secondaires S_2 et S_3 en précisant le changement de son état (message update). Après la mise à jour de leurs états, les copies secondaires envoie un ack à la copie primaire. La copie primaire envoie la réponse au client après réception des "ack" de toutes les copies secondaires. Le point de reprise permet de synchroniser l'état des copies secondaires avec celui de la copie primaire puisque celle-ci est la seule qui communique avec le reste du système. [33]

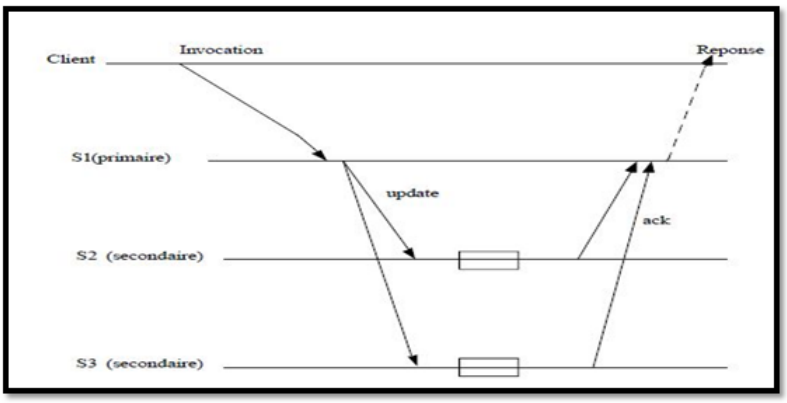


FIGURE 2.12 – Principe de la réplication passive

c. Réplication semi-active

La réplication semi-active se situe à mi-chemin entre la réplication active et la réplication passive. Contrairement à la réplication passive, les copies secondaires ne sont pas oisives. La copie primaire est appelée leader et les copies secondaires sont appelées suiveurs.

PRINCIPE : La réplication semi-active est définie ainsi :

- Réception des requêtes : toutes les copies reçoivent le même ensemble de Requêtes.
- Traitement des requêtes : toutes les copies traitent toutes les requêtes. La copie primaire traite une requête dès qu'elle la reçoit. Par contre, une copie secondaire doit attendre une notification de la copie primaire pour pouvoir traiter une requête.
- Émission des réponses : la copie primaire est la seule à émettre les réponses.

La FIGURE 2.13 illustre ce principe. Le client envoie une requête à tous les Si. La copie primaire S_1 envoie une notification "notify" aux copies secondaires et commencent le traitement de la requête. Les copies secondaires S_2 et S_3 ne commencent à traiter la requête qu'après avoir reçu la notification de la copie primaire. Sitôt le traitement terminé, S_1 envoie la réponse au client.[34]

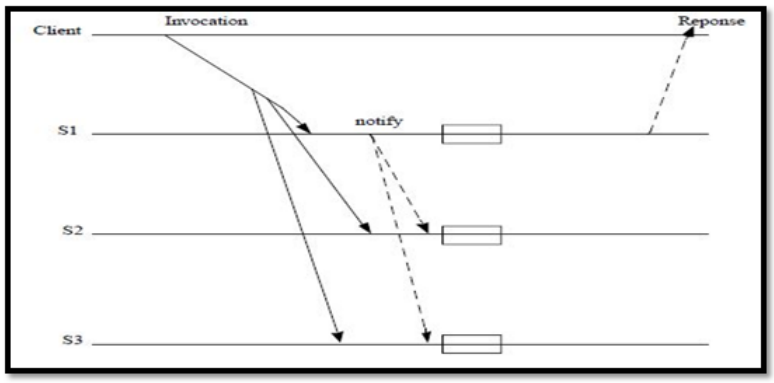


FIGURE 2.13 – Principe de la réplication semi-active

Et comme nous avons montré dans le chapitre précédent (section problématique), les avantages de la diversité par rapport à la réplication nous nous intéressons à la compensation par diversité décrit par la suite :

2.4.1.2 Compensation par diversité

Traditionnellement, la recherche en tolérance aux fautes logicielle s'articule autour de deux approches : N-versioning (en anglais, N-version programming (NVP)) de Avizienis [3] et blocs de recouvrement (en anglais, Recovry Blocks(RB)) de Randell [22].

Dans la première approche, N-versions d'un programme exécutent de multiples implémentations d'une même fonctionnalité mais ayant des conceptions diverses en **parallèle** (i.e., différentes équipes de développement, différents outils). Un vote majoritaire est effectué pour obtenir les résultats final à retourner à l'utilisateur (ou application appelante).

Dans la seconde approche, les RB's invoquent des implémentations redondantes de manière **séquentielle**, si la sortie d'un RB ne réussit pas le test de conformité / validité. La technique de NVP semble être une bonne candidate aussi bien à des objectifs de sûreté de fonctionnement que de performance, malgré le coût de développement de versions multiples. Dans le domaine des services Web, Looker et Munro [18] ont été les premiers à introduire NVP. Sommerville [25] a suivi le travail avec son approche de tolérance aux fautes à base de conteneur configurable, avec une politique spécifiant le type de mécanisme de tolérance aux fautes appliqué aux services le contenant. Dobson [11] a utilisé quant à lui les mécanismes fournis par WS-BPEL [21] pour supporter NVP et RB.

2.5 Travaux existants sur la tolérance aux fautes

Dans cette section, nous allons nous focaliser sur les études s'intéressant aux quelques aspects de la technique de compensation sous la forme : Stratégies de services web. Durant la dernière décade, un grand nombre de travaux ont mis l'accent sur la forme, laissant ainsi un champ fertile de recherche à explorer pour l'application du concept de diversité à la conception de services web tolérants aux fautes.

2.5.1 Approches basées sur les stratégies de réplication

2.5.1.1 Empreinte réflexive de la réplication

TAIANI et al. [28] présente une empreinte réflexive de la réplication pour résoudre de nombreux problèmes de sûreté de fonctionnement des services web. Le projet DELTA- 4 décrit trois stratégies de réplication d'une application client/serveur communiquant par messages : la réplication active, la réplication passive et la réplication semi-active. En réplication active, les différentes répliques ont des rôles symétriques, et traitent toutes les requêtes simultanément. Ce mode de réplication nécessite d'une part que les mêmes messages d'entrée soient fournis dans le même ordre à toutes les répliques (cohérence des entrées) ; et d'autre part que, partant du même état initial, les répliques produisent les mêmes messages de sortie, aussi dans le même ordre (déterminisme des répliques). Lorsque ces hypothèses sont respectées, la réplication active autorise un recouvrement très rapide des défaillances, puisque chacune des répliques peut à tout moment assumer seule le service répliqué. La réplication active permet aussi de tolérer les fautes en valeur d'une minorité de répliques par la mise en place d'un système de comparaison et de vote sur les réponses produites. On s'assure ainsi qu'une réponse valide sera toujours retournée à l'utilisateur. Le prix à payer pour ces différentes propriétés reste néanmoins une forte utilisation des ressources matérielles, tous les traitements devant être répliqués en permanence. La réplication passive élimine ce fort sur-coût de traitement en introduisant une dissymétrie dans le rôle des répliques. Une seule réplique, la réplique primaire ou primary, traite les requêtes en entrée et produit des sorties. Les autres répliques, les répliques de secours ou backups, sont maintenues synchronisées avec la réplique primaire par un transfert périodique de captures d'état (checkpointing).
Le choix des instants de capture est déterminant pour assurer un masquage des défaillances complet à l'utilisateur. Il s'agit en fait du même problème que celui à la base des protocoles de capture (checkpointing protocols), les informations d'état transmises par la réplique primaire doivent permettre en cas de défaillance un basculement transparent du service sur les répliques secondaires. Dit autrement, les répliques secondaires doivent pouvoir assurer, à partir des captures d'état, une reprise qui soit cohérente avec le comportement perçu jusqu'alors par les clients du service répli-

qué. En plus de ses avantages en termes de sur-coût de traitement en mode sans défaillances, la réplication passive, au contraire de la réplication active, ne requiert pas le déterminisme de l'application à laquelle elle s'applique. Cependant, les transferts de capture peuvent nécessiter des coûts de communication élevés, et les temps de recouvrement en cas de défaillance de la réplique primaire sont souvent plus longs, du fait de traitements qui n'étaient pas compris dans la dernière capture d'état et qui doivent être ré effectués. Enfin, la réplication passive ne permet pas de tolérer les fautes en valeur, une seule réplique produisant des réponses. La troisième forme de réplication, la réplication semi-active, a pour objectif de lever la contrainte du déterminisme qui vaut pour la réplication active. Elle ajoute des mécanismes de « transfert », inspirés de la réplication passive, au mécanisme classique de la réplication active pour assurer un consensus entre les différentes répliques vis-à-vis de leurs décisions non-déterministes. Une réplique est choisie comme « meneuse » et est responsable des décisions non-déterministes de l'application (ordonnancement des

Stratégie	Fautes tolérées	Déterminisme	**Sur-coût**		
active	fautes incontrôlées	requis	important	faible	faible
passive	silence sur défaillance	non-requis	faible	important	important
semi-active	fautes incontrôlées	non-requis	important	faible	faible

TABLE 2.1 – Propriétés et hypothèse des stratégies de réplication considérées

brins d'exécution, des messages, d'événements asynchrones) qui sont alors communiquées sous forme de notifications aux autres répliques, « suiveuses ». La réplication semi-active combine la plupart des avantages des deux autres formes de réplication : ses sur coûts de communication et de recouvrement sont faibles, comme dans la réplication active, elle ne requiert pas le déterminisme, comme la réplication passive. De plus, en supposant un protocole de diffusion fiable entre meneur et suiveurs d'une part, et en supposant que les suiveurs soient capables de rejeter les choix invalides du meneur d'autre part, alors la réplication semi-active supporte aussi les fautes en valeur. Les différentes propriétés et les hypothèses sur lesquelles s'appuie chacune des stratégies sont reprises sous une forme résumée dans la TABLE 2.1. Les trois formes de réplication ainsi présentées nécessitent la mise en place de mécanismes de détection d'erreur et de configuration, notamment pour relancer une réplique, ou mettre à jour la vision qu'ont du système les différentes répliques (protocole de groupe).

2.5.1.2 Intercepteurs

BENNANI et Fabre [4] présentent une gestion de la plate forme Daisy (Dependable Adaptative Interceptors and Serialization-based system) qui est utilisée par les composants de tolérance aux

fautes. La FIGURE 2.14 montre l'architecture à composants de DAISY, par ailleurs, l'apport des composants utilitaires pour le recouvrement des défaillances des serveurs d'applications.

Lors de l'occurrence d'une défaillance par arrêt d'un serveur d'application, cette anomalie est détectée, selon le serveur, se base sur les intercepteurs . Ces différents composants notifient le gestionnaire de réplication de cette défaillance d'un serveur pour qu'il puisse la pallier. les intercepteurs étant déjà le centre de plusieurs traitements complexes, ils n'ont pas ajouté les traitements relatifs à la mise en service de nouvelle réplique dans le cas de l'arrêt d'un serveur du groupe. Ce choix a pour but d'améliorer les performances de la plate-forme.

Comme le montre la FIGURE 2.15, lorsque le gestionnaire de réplication est notifié de la défaillance d'un serveur du groupe de réplication, pour n'importe quelle mode de réplication, il commence par vérifier si la machine qui héberge le serveur en question est opérationnelle (état). Si la machine fonctionne, le gestionnaire de réplication lance le nouveau serveur : il lance un serveur subsidiaire dans le cas de la réplication active et un serveur réplique dans le cas de la réplication passive. Dans le cas où la machine ne fonctionne pas, le gestionnaire de réplication se met à la recherche d'une autre machine pour y lancer le nouveau serveur. S'il n'en trouve pas une, il n'acquitte pas l'intercepteur demandeur, et selon le nombre de répliques opérationnelles, la plate-forme remplira le contrat de réplication ou délivrera un service de tolérance aux fautes dégradé.

Le service dégradé de tolérance aux fautes concerne le nombre de fautes à tolérer qui est relatif au nombre de serveurs opérationnels ; en effet, dans le cas où le gestionnaire de réplication n'arrive pas à lancer un nouveau serveur, c'est le nombre de fautes tolérées qui est revu à la baisse.

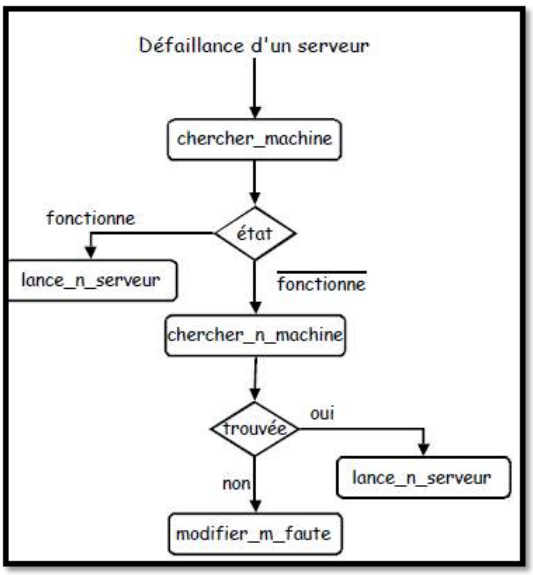


FIGURE 2.14 – Traitement de l'arrêt d'un serveur

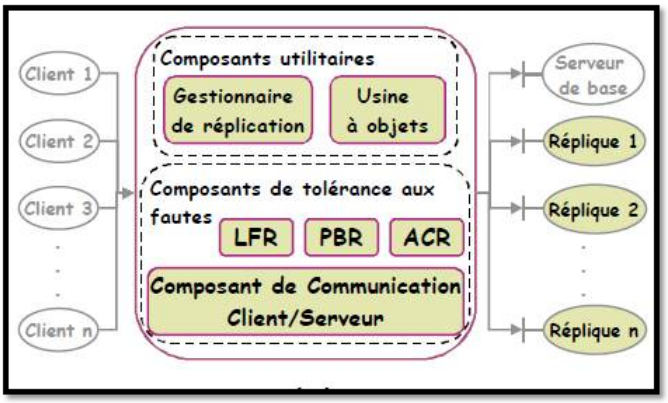


FIGURE 2.15 – Vue Générale de l'architecture

2.6 Conclusion

La sûreté de fonctionnement des services Web est un domaine de recherche actif et important. L'architecture distribuée avec le couplage faible des services Web a apporté des avantages pour le développement d'applications distribuées plus flexibles et hétérogènes.

Cependant, une telle architecture est par nature peu fiable. La recherche sur la sûreté de fonctionnement des applications à base de services Web doit faire face à deux types de défaillances à savoir les défaillances par crash et les défaillances Byzantines. De nombreuses approches ont été développées pour assurer la fiabilité du service Web. Cependant, notre analyse montre que des limites de ces solutions empêchent leur efficacité. De nouvelles solutions sont nécessaires pour

améliorer la fiabilité des applications de service Web du point de vue des clients afin de minimiser les problèmes causés par les défaillances des services. De nouvelles techniques sont également nécessaires pour améliorer l'efficacité de ces solutions en utilisant explicitement des stratégies de recouvrement de services sûrs de fonctionnement.

Nous cherchons dans ce mémoire à déterminer de façon automatique la stratégie de recouvrement basée sur la diversité pour rendre un groupe de diversité plus fiable. Cette détermination doit répondre à plusieurs critères et exigences clients. Le chapitre suivant présente notre solution basée sur la diversité des services Web et détermine la meilleure stratégie de tolérance aux fautes à utiliser dans un groupe de diversité.

Chapitre 3

Détermination de stratégie de tolérance aux fautes : une approche basée sur la diversité

Sommaire

3.1 Introduction

Le chapitre précédent discute les notions fondamentales des Services Web et de sûreté de fonctionnement. Dans ce chapitre, nous proposons une approche dynamique pour la sélection et la détermination d'une stratégie de tolérance aux fautes pour un ensemble de services Web équivalents. Le problème de sélection d'une stratégie de tolérance aux fautes (parallèle avec vote, parallèle sans vote et séquentielle) se réduit à un problème de décision multicritères. En réalité, sous certains critères et conditions, un client peut choisir explicitement la stratégie à utiliser pour recouvrir les fautes observées dans un groupe de service. Dans ce cadre, l'élaboration d'un mécanisme de sélection automatique de la stratégie de tolérance aux fautes (recouvrement) est nécessaire, en prenant compte des données plus complètes sur le type de fautes, l'origine de la faute, ainsi que les paramètres de QoS (temps, coût). La décision en présence de critères multiples est difficile car les critères sont souvent conflictuels. Pour cela, plusieurs méthodologies d'aide multi critères à la décision ont été développées. Dans ce contexte, nous allons nous intéresser sur l'aide à la décision multi critères et plus précisément la méthode Electre I, qu'est orientée de telle sorte à ce que le résultat soit une sélection d'un ensemble de « bonnes » alternatives.

Ce chapitre est organisé comme suit : La première section, présente les différentes définitions fondamentales. La deuxième section donne une vue générale sur notre approche. La troisième section, présente l'utilisation d'une méthode de décision multi critères pour la modélisation. Enfin, la dernière section s'intéresse à notre algorithme de détermination de la stratégie de TF basée sur la diversité des services Web.

3.2 Objectif et motivation

La nature compositionnelle des services Web ainsi que la forte dynamique d'Internet rendent les stratégies de tolérance aux fautes classiques non pratiques dans un environnement du monde réel. En effet, ces dernières ont été traditionnellement appliquées à un ensemble immuable de services Web sémantiquement équivalents et sont agnostiques au contexte [19]. Il devient alors évident que l'efficacité d'un groupe de diversité dépend de la disponibilité des services Web le composant. En effet, un vote majoritaire parmi ces services Web ne pourra pas garantir la disponibilité du service attendu de l'application si la majorité devient simultanément indisponible. Pour les applications temps-réel, tout délai additionnel dans le temps de réponse d'un service Web peut impacter l'efficacité du groupe de diversité à délivrer les résultats en temps voulu. Toutes ces caractéristiques non prévisibles des services Web sont un véritable challenge lors de la détermination de la configuration du groupe appropriée pour assurer son bon fonctionnement. Pour cela, il y a un besoin urgent pour trouver des solutions garantissant le bon fonctionnement d'un groupe de

diversité. Déterminer la configuration d'un groupe de diversité dépend de trois paramètres :

1. Le nombre de services Web à impliquer dans le groupe de diversité (Combien de SW ?) [1]
2. Les services Web à impliquer dans le GD (Quels sont ses services Web ?) [1]
3. La stratégie de TF basée sur la diversité à utiliser pour un GD (Quelle stratégie ?)

Dans notre travail, nous focalisant sur la troisième question et plus précisément, une détermination automatique de la stratégie de TF selon les paramètres du contexte, en termes d'exigences client et des comportements antérieurs des services Web.

3.3 Définitions

Nous commencerons, tout d'abord, par les définitions des différentes notions qu'on utilise dans notre solution.

3.3.1 Définition d'un service concret

Un service web concret correspond à une fonctionnalité possédant une implémentation, et décrit dans un document WSDL. Il est accessible à travers une URI. Nous notons un service web concret par *SW*.

Exemple d'un service web concret : Un exemple de service web concret est le service amazon accessible à travers l'URI : `http://aws.amazon.com/fr/` et possédant le document WSDL : `http://webservices.amazon.com/AWSECommerceService/2005-03-23/US/AWSECommerceService.wsdl`.

3.3.2 Définition d'un service abstrait

Un service abstrait correspond à une fonctionnalité mais sans aucune implémentation directe. Il n'a d'intérêt que s'il correspond à des services concrets. Nous notons un service web abstrait par *SWA*.

Exemple d'un service Web abstrait : Un exemple de service web abstrait est le service de réservation de vol, celui-ci peut être implémenté par un service Web concret.

3.3.3 Définition d'un Groupe de diversité

Un groupe de diversité correspond à un ensemble de services concrets qui implémente un même service abstrait.Ainsi, un groupe de diversité est défini par le couple $<SWA_j,\{sw_1,sw_2,...,sw_n\}>$ où SWA_j et sw_i(i=1,...,n) correspondent respectivement à un service abstrait et des services

concrets. Par exemple < réservation de vol, {SW-Air Algérie, SW-Air France, SW-Britch Airways}> et nous notons un groupe de diversité par GD. La FIGURE 3.1 schématise les trois notions de service concret, service abstrait et groupe diversité.

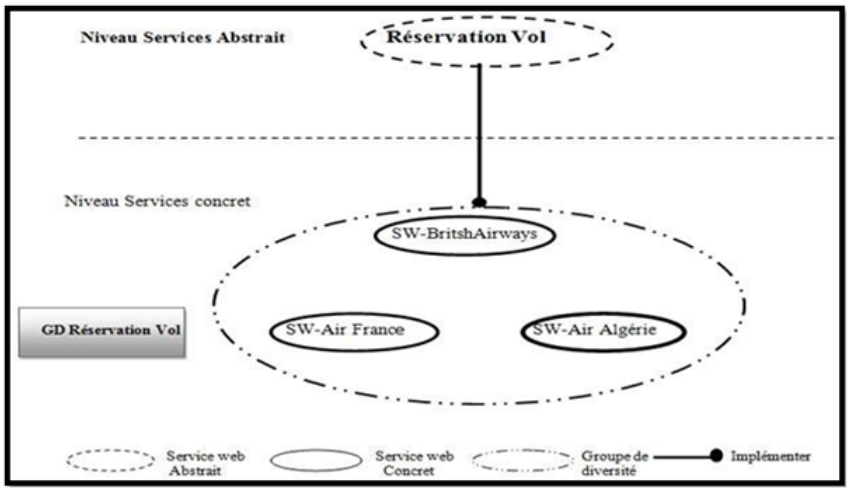


FIGURE 3.1 – Exemple Service Web Abstrait implémenté par des services concrets

3.3.4 Mode de défaillance

a. La Défaillance de type Byzantine

Une défaillance byzantine se produit quand le résultat délivré par un service ne permet plus l'accomplissement de la fonction du système. Ceci se produit en générale quand le résultat délivré ne respecte pas les spécifications attendues. Nous considérons deux types de défaillance byzantine :

Défaillance Byzantine sans spécification : On a aucune spécification sur le résultat, nous utilisons un vote majoritaire pour rendre un résultat au client.

Défaillance Byzantine avec spécification : Nous avons une spécification concernant le résultat à retourner, par exemple, on attend que le prix d'un service Web de vol soit compris entre deux valeurs (intervalle)

Les défaillances byzantines peuvent être, malveillantes, c'est-à-dire produites de façon intentionnelle, ou non malveillantes quand elles sont produites sans aucun objectif de, malice. Nous traitons dans notre approche les défaillances byzantines non malveillantes. Nous Considérons un seul type de résultat :

- **Résultat incorrect :** Le service web délivre des résultats qui ne correspondent pas à ceux attendus par l'utilisateur. Par exemple, un utilisateur peut s'attendre à recevoir un prix dans l'intervalle [60 100], alors que le service web lui renvoie un prix supérieur à 100. Cette situation

correspond à une situation de résultat incorrect.

b. Défaillance de type Crash

Lorsqu'un service crash, il cesse immédiatement et de façon indéfinie de répondre à toute sollicitation ou de générer de nouvelles requêtes (jusqu'à une réparation).

3.4 Overview sur l'approche proposée

Dans cette section, nous présentons un schéma qui formule le problème posé (FIGURE 3.2 :

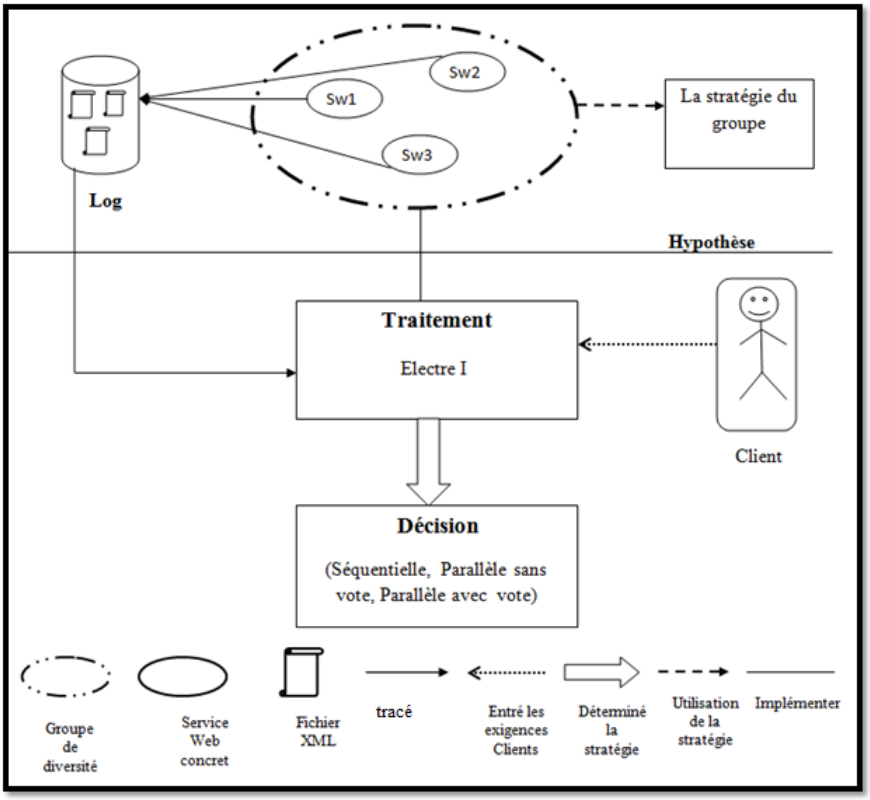


FIGURE 3.2 – Schéma global sur l'approche proposée

Le problème posé se base sur des étapes différentes, nous expliquons chaque étape comme suit :

- **Groupes diversités (GD) des services Web :**

Nous partons d'un ensemble de services Web équivalents regroupé au sein d'un même GD, le nombre de services Web dans le groupe dépend des fautes détectées et tolérées dans le GD [1].

- **Historique (Log) :**

Le fichier Log contient toutes les données nécessaires pour comprendre le comportement d'un service web au sein du groupe de diversité. Les données sont en terme de mode de défaillance des

différents services Web dans les exécutions précédentes, ainsi que le coût moyen d'exécution des services Web et le temps moyen d'invocation de ses services. Nous notons les critères de Log par :

Défaillance de type Byzantine avec spécification : DBA_{His}

Défaillance de type Byzantine sans spécification : DBS_{His}

Défaillance de type Crash : DC_{His}

Le temps moyen d'invocation : $Temps_{His}$

Le coût moyen d'exécution : $Coût_{His}$

- **Exigences clients :**

Elles représentent les besoins fonctionnels en tolérance aux fautes des clients c'est à dire le degré maximal tolérable par les clients, Aussi des besoins non fonctionnels en termes de temps de réponse maximum toléré et de coût. Nous notons les critères client par :

Défaillance de type Byzantine avec spécification Tolérée : DBA_T

Défaillance de type Byzantine sans spécification Tolérée : DBS_T

Défaillance de type Crash Tolérée : DC_T

Le temps moyen d'invocation :$Temps$

Le coût moyen d'exécution : $Coût$

Pour chaque critère, on évalue un poids w_i qui augmente avec l'importance du critère, w_i ne correspond pas à un taux de substitution mais plutôt à un nombre de voix associé au critère i, c.-à.-d. si une alternative a est préférée à b, alors le critère i est en faveur de a et le poids w_i lui est entièrement attribué sans tenir compte de l'intensité de cette préférence, ce qui limite l'impact d'une mauvaise évaluation de ce paramètre sur le résultat final.

- **La stratégie de TF :**

Nous avons trois stratégies de tolérance aux fautes, qui sont : Séquentielle (S), Parallèle avec vote (PAV), Parallèle sans vote (PSV). Le schéma qui représente la stratégie séquentielle : (voir FIGURE 3.3) : Ce mécanisme nécessite f+1 services pour toléré f fautes de type Crash et la FIGURE 3.4 représente la stratégie parallèle : Ce mécanisme nécessite 2f+1 services pour toléré f fautes de type Byzantine.

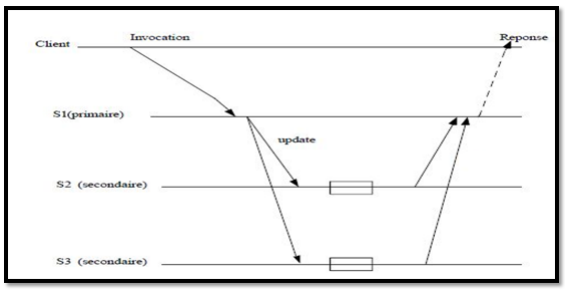


FIGURE 3.3 – La stratégie séquentielle

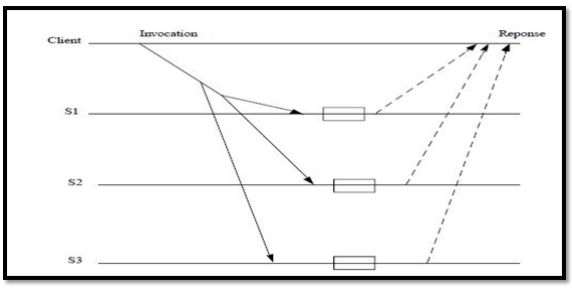


FIGURE 3.4 – La stratégie parallèle

- **Traitement :**

A cette étape, nous modélisant notre problème de détermination automatique de stratégie de TF avec une des méthodes de décision multi critères. Nous appliquerons notre algorithme de détermination de la stratégie de TF afin d'obtenir le meilleur résultat entre les trois stratégies de TF basée sur la diversité.

- **Décision :**

Dans cette dernière étape nous déterminons la bonne stratégie parmi les trois stratégies (séquentielle, parallèle avec vote, Parallèle sans vote) à partir des critères exigés : type de défaillance (byzantine ou crash), temps et le coût d'un service web au sein du groupe de diversité.

3.5 Modélisation

Notre problème de détermination de la stratégie de TF se réduit à un problème décisionnel. Les critères sont conflictuels, donc on cherche à réaliser un compromis entre les différents critères, tout en favorisant les exigences des clients.

Nous présentons dans ce qui suit la méthode Electre I qu'est préconisée pour répondre à une problématique de choix.

3.5.1 ELECTRE I

ELECTRE I (ÉLimination Et Choix Traduisant la REalité) ; est une méthode proposée par (Roy, 1968)[23], elle permet de résoudre les problèmes multi critères de choix. Elle détermine un sous ensemble des actions qui surclasse le reste des actions par la Constitution d'un noyau d'action qui surclassent les autres. Ce noyau n'existe pas nécessairement et n'est pas nécessairement unique.

La définition du problème requiert une compréhension de la situation étudiée, du contexte et des acteurs impliqués dans la prise de décision. L'interaction avec les différents acteurs permet de comprendre le processus de décision, les enjeux, l'objet de la décision et la nature de la décision à prendre. Il s'agit donc de définir la nature du problème posé en le formulant : un problème de choix.

La détermination de l'objet de la décision consiste à identifier l'ensemble des actions ou alternatives sur lesquelles va porter la décision.

a. Actions

Une action est une entité qui représente l'objet de la décision. En aide multicritères à la décision l'ensemble des actions A est construit sous forme d'une liste : A = $\{a1, a2, ...\}$,qui représentent les différents choix.

Il s'agit en effet d'identifier et mesurer les conséquences des actions sur lesquelles va porter la décision.

b. Critères

Un critère doit permettre de mesurer les préférences du décideur vis-à-vis de chaque action, relativement à un point de vue. (ex : âge). Les critères sont définies par une fonction appelée fonction de critères.

c. Fonction de critères

Un critère est une fonction $g : A \rightarrow X$ inclue dans R qui permet relativement à un point de vue donné et pour un acteur identifié, de comparer deux actions qcq a et b :

$$g(a) > g(b) \Longleftrightarrow aSb, a, b \in A$$

Où S : « au moins aussi bon que », nous supposons ici que g est un critère à maximiser.

d. Relation de surclassement

Une relation de surclassement est une **relation binaire** S définie dans A telle que aSb si, étant donné ce que l'on sait des **préférences du décideur** et étant donnée la **qualité des évaluations des actions** et la **nature du problème**, il y a suffisamment d'arguments pour admettre que a est **au moins aussi bonne** que b, sans qu'il y ait de raison importance de **refuser cette affirmation**.

Ainsi, a surclasse b, si :

$$aSb \Longleftrightarrow C(a,b) \geqslant Sc, D(a,b) \leqslant Sd$$

Sachant que : Sc : seuil de concordance (il y'a suffisamment d'arguments en faveur de a), et Sd : seuil de discordance (il n'y a pas assez d'arguments contre le surclassement). Sc est le niveau exigé de concordance pouvant s'interpréter comme un niveau de majorité et Sd niveau de minorité.

e. Concordance

La concordance globale apprécie la contribution de l'ensemble des critères à l'affirmation *a* S *b*.

L'indice de concordance pour deux actions a et b est notée par C (a,b), compris entre 1 et 0, il mesure la pertinence de l'assertion « a surclasse b », comme suit :

$$C(a,b) = \frac{\sum_{\forall i, g_i(a) \geqslant g_i(b)} w_i}{w}$$

avec

$w = \sum_i^n w_i$ Pour chaque critère, on évalue un poids w_i qui augmente avec l'importance du critère.

f. Discordance

L'indice de discordance est notée par D (a,b) représente le fait qu'il existe de critère où l'avantage de b sur a contredit l'affirmation *aSb*, est défini par :

$$D(a,b) = 0$$

si

$$\forall i, g_i(a) \geqslant g_i(b)$$

Sinon

$$D(a,b) = \frac{1}{\delta} max_i[g_i(b) - g_i(a)]$$

avec δ est la différence maximale entre le même critère pour deux actions données.

g. Noyau

La procédure d'exploitation repose sur la recherche d'un noyau dans un graphe orienté G = (A,U) où les sommets U = $\{x_a1,..,x_am\}$ représentent les alternatives et $(x_{ai},x_{aj}) \in A \Longleftrightarrow a_iSa_j$. La relation de surclassement (nette) S peut être représentée par un graphe orienté :

$$G = (A,U)$$

où

$$U = \{(a_i,a_j)2A \times A/a_iSa_j\}$$

on recherche un sous-ensemble N tel que :

$$\forall b \in (A-N), \exists a \in N/aSb$$

$$\forall (a,b) \in N \times N, aSb$$

La recherche de N est équivalente à la recherche du noyau du graphe G représentant S.
Une fois la relation de surclassement est déterminée, elle est exploitée pour répondre au problème.

En Ce qui suit nous présentons la modélisation de notre approche basée sur la méthode Electre I :

3.5.2 Processus d'élaboration de stratégie de TF basée sur la diversité

a. Actions

Dans notre solution, le choix porte sur 3 actions, les actions représentent les stratégies de TF à déterminer. Dans notre modélisation, nous notons
$strage_j$ (j=1..n, n est le nombre de stratégies)

$$\begin{cases} strage_1 & = & \text{Séquentielle} \\ strage_2 & = & \text{Parallèle avec vote} \\ strage_3 & = & \text{Parallèle sans vote} \end{cases}$$

b. Critères

On définit pour chaque critère une fonction appelé fonction de critère $g_i(strage_j)\ \forall\ strage_j$ (i=1 jusqu'à 5 critères) et les 5 critères sont : $\{DBA_T, DBS_T, DC_T, Temps, Coût\}$. On doit calculer la fonction des différents critères pour les différentes stratégies.

c. Fonction de Critères

La fonction de critères définit par l'exemple suivant :

$g_{DBA_T}(strage1)$, $g_{DBS_T}(strage1)$, $g_{DC_T}(strage_1)$, $g_{Temps}(strage_1)$, $g_{Coût}(strage_1)$.

d. Table de performance

Le choix des stratégies repose sur les critères en termes de types de défaillances (byzantine, crash),et les exigences client vis à vis du temps de réponse et le coût, d'une manière intuitive les stratégies peuvent être comme suit : La TABLE 3.1 ci dessus Récapitule les différents cas où

Critères exigés / Stratégies	Byzantines sans sp	Byzantines avec sp	Crash	Temps	Coût
Séquentielle		X	X		XX
Parallèle avec vote	XX		X	X	
Parallèle sans vote	X	X	X	XX	

TABLE 3.1 – Choix de la stratégie de tolérance aux fautes selon les différents critères exigés

l'utilisation d'une stratégie est recommandée. Le symbole XX signifie que l'utilisation d'une telle stratégie en présence de certaines conditions est très recommandée. Par exemple si une application a une forte contrainte du Temps, l'utilisation de la stratégie parallèle sans vote sera le bon choix, comme il ne consomme pas beaucoup de ressources. Le symbole X signifie que l'utilisation d'une telle stratégie en présence de certaines conditions est moins recommandée. Par exemple si une application a une faible contrainte du Temps, l'utilisation de la stratégie parallèle avec vote sera le mauvais choix.

e. Relation de surclassement

La comparaison au sein d'un couple de stratégies se fait en utilisant la notion de surclassement dans notre cas par exemple $strage_1$ est une moins aussi bonne que $strage_2$ donc en dit "$strage_1$ S $strage_2$".

f. Concordance

La concordance globale apprécie la contribution de l'ensemble des critères à l'affirmation : $strage_j$ S $strage_{j+1}$. Où $strage_j \in$ {Seq, PAV, PSV}.

L'indice de concordance globale C ($strage_j$, $strage_{j+1}$) $\in$[0, 1] est tel que :

- C ($strage_j$, $strage_{j+1}$) = 0 lorsqu'aucun critères $gi(strage_j)$ n'est en faveur de $strage_j$ S $strateg_{j+1}$.

- C ($strage_j$, $strage_{j+1}$) = 1 lorsque tous les critères $gi(strage_j)$ sont en faveur de $strage_j$ S $strage_{j+1}$.

- C ($strage_j$, $strage_{j+1}$) $\in$]0, 1[lorsque certains critères $gi(strage_j)$ sont en faveur de $strage_j$ S $strage_{j+1}$.

• Ce qui peut se formuler par :

$strage_1$ et $strage_2$:

$$C(strage_1, strage_2) = \frac{\sum_{\forall i, g_i(strage_1) \geqslant g_i(strage_2)} w_i}{w}$$

avec $w = \sum_{i=1}^{5} w_i$ La concordance signifie que une majorité de critères doit être en accord avec $strage_j$ S $strage_{j+1}$(principe majoritaire).

g. Discordance

On définit, sur chaque critère, un indice de discordance D ($strage_j$, $strage_{j+1}$) $\in$ [0, 1] tel que :

- D ($strage_j$, $strage_{j+1}$) = 0 si $g_i(strage_j)$ ne s'oppose pas à $strage_j$ S $strage_{j+1}$,

- D ($strage_j$, $strage_{j+1}$) = 1 si $g_i(strage_j)$ s'oppose totalement à $strage_j$ S $strage_{j+1}$,

- D ($strage_j$, $strage_{j+1}$) $\in$]0, 1[si $g_i(strage_j)$ s'oppose en partie à $strage_j$ S $strage_{j+1}$,

L'indice de discordance D ($strage_j$, $strage_{j+1}$) est défini par :

D ($strage_j$, $strage_{j+1}$) = 0 si $\forall$i, $g_i(strage_j) \geqslant g_i(strage_{j+1})$

Sinon

$$D(strage_j, strage_{j+1}) = \frac{1}{\delta} max_i[g_i(strage_{j+1}) - g_i(strage_j)]$$

avec δ est la différence maximale entre le même critère pour deux stratégies données.

Par exemple :

$strage_1$ et $strage_2$:

D ($strage_1$, $strage_2$)=0 si $\forall$i, $g_i(strage_1) \geqslant g_i(strage_2)$

Sinon

$$D(strage_1, strage_2) = \frac{1}{\delta} max_i[g_i(strage_2) - g_i(strage_1)]$$

La discordance signifie qu'aucun des critères non concordants ne doit réfuter fortement $strage_j$ S $strage_{j+1}$ (principe de respect des minorités).

h. Choix de seuil

Le choix de seuil dépend des indices de concordance et discordance, où Sc est le niveau exigé de concordance pouvant s'interpréter comme un niveau de majorité et Sd niveau de minorité. Ce choix permet de surclasser les stratégies.

3.6 L'algorithme

L'algorithme détermine la meilleure stratégie de tolérance aux fautes. Il prend comme paramètres : un groupe de diversité GD = $\{sw_i..sw_n\}$, fonction de critères $g_i(strage_j)$, ensembles de fonction de critères noté G, l'ensemble de critères i$\in$ $\{DBA_T$, DBS_T,DC_T , $Temps$, $Coût\}$, ensemble des stratégies $strage_j \in \{strage_1, strage_2, strage_3\}$, les stratégies et les critères forme une matrice appelée "matrice de performance" ; où les lignes représentent les stratégies et les colonnes représentent les critères, les exigences de client vis à vis chaque critères noté w_i, les seuils : seuil de concordance noté Sc, seuil de discordance noté Sd.

Dans la première partie, il s'agit de calculer la matrice de concordance et discordance, ensuite on doit surclasser les indices de concordance et de discordance avec le choix de seuil, ce choix borne les valeurs de concordance et discordance pour les stratégies de tolérance aux fautes basée sur la diversité.

Algorithm 1 Détermination de stratégie de TF basée sur la diversité

ENTRÉES :

GD = $\{sw_i..sw_n\}$

$g_i(strage_j)$: i$\in$ $\{DBA_T, DBS_T, DC_T, Temps, Coût\}$

$strage_j \in \{strage_1, strage_2, strage_3\}$

w_i, Mc_{ij}

Sc, Sd

SORTIE : Stratégie à retourner.

$strage_j \in$ {Seq, PAV, PSV } = acceptée ou rejetée

Pour chaque $strage_j$ **faire**

Pour chaque $g_i(strage_j) \in G$ **faire**

Si $g_i(strage_j) \geqslant g_i(strage_{j+1})$ **alors**

C($strage_j$, $strage_{j+1}$) = $\frac{\sum_i^5 w_i}{w}$ avec

$w = \sum_{i=1}^{5} w_i$

D ($strage_j$, $strage_{j+1}$) = 0

Sinon

D ($strage_j$, $strage_{j+1}$) = $\frac{1}{\delta} max_i[g_i(strage_{j+1}) - g_i(strage_j)]$

Fin Si

Si $C(strage_j, strage_{j+1} \geqslant Sc$ **alors**

Si $C(strage_j, strage_{j+1} \leqslant Sd$ **alors**

Retourner $strage_j$ = acceptée

Sinon

Retourner $strage_j$ = rejetée

Fin Si

Sinon

Retourner $strage_j$ = rejetée

Fin Si

Fin Pour

Fin Pour

3.7 Exemple illustratif sur l'algorithme

Nous illustrons l'algorithme par un exemple, pour mieux comprendre les étapes étudiées.

l'importance de chaque critère représentent un vecteur :

- **L'exigence client :** $w(DBS_T)$=0.25 , $w(DBA_T)$=0.33, $w(DC_T)$=0.5, $w(Temps)$=0.66, $w(Coût)$=0.75

Chaque stratégie est évalué en fonction des critères retenus à l'aide d'une échelle qualitative et des scores. Le tableau de performance est donné dans le tableau suivant :

Critères exigés / Stratégies	DBS_T	DBA_T	DC_T	$Temps$	$Coût$
Seq	F	M	M	F_t	$E_{Coût}$
PAV	E	F	M	M_t	$F_{Coût}$
PSV	M	M	M	E_t	$F_{Coût}$

TABLE 3.2 – Choix de la stratégie de tolérance aux fautes selon les différents critères exigés (**matrice performance**)

F =Faible, M =Moyen, E=Élevé

- (Seq, DBS_T) = F l'utilisation de la stratégie Seq est faiblement recommandée lorsque aucune défaillance de type Byzantine ni tolérée, surtout si on ne possède aucune spécification sur le résultat retourné.

- (PAV, DBS_T) = E l'utilisation de la stratégie PAV est fortement recommandée lorsque aucune défaillance de type Byzantine ni tolérée, surtout si on ne possède aucune spécification sur le résultat retourné : le résultat est délivré par le moyen de vote.

- (PSV, DBS_T) = M l'utilisation de la stratégie PSV est moyennement recommandée lorsque aucune défaillance de type Byzantine ni tolérée, surtout si on ne possède aucune spécification sur le résultat retourné.

- (Seq, DBA_T) = M L'utilisation de la stratégie Seq est moyennement recommandée, si on possède une spécification sur le résultat à retourner.

- (PAV, DBA_T) = F L'utilisation de PAV est faiblement recommandée comme elle consomme beaucoup de ressources dans le cas de DBA_T.

- (Seq, DC_T) = M L'utilisation de Seq est moyennement recommandée dans le cas du crash d'un SW, cela dépend des critères de temps et coût.

- (Seq, $Temps$) = F_t l'utilisation de la stratégie Seq est faiblement recommandée lorsque la contrainte du temps de l'exécution est faible et le Temps entre]0.66 ; 1].

- (PSV, $Temps$) = E_t l'utilisation de la stratégie PSV est fortement recommandée lorsque la contrainte du temps de l'exécution est forte et le Temps entre [0 ; 0.33[.

- (PAV, $Temps$) = M_t l'utilisation de la stratégie PAV est moyennement recommandée, si on a une moyenne contrainte du Temps et le Temps entre]0.33 ; 0.66[.

-(PAV, $Coût$) = $F_{Coût}$ L'utilisation de PAV est fortement recommandée, si on a une contrainte

faible du coût et le coût entre]0.66 ; 1].

-(Seq, *Coût*) = $E_{Coût}$ l'utilisation de la stratégie Seq est fortement recommandée lorsque le coût est entre [0 ; 0.33 [.

Intervalle de Temps :

F_t :] 0.66 ; 1] cette intervalle mesure une contrainte faible du temps, l'utilisation du stratégie Séquentielle sera le mauvais choix.

M_t :] 0.33 ; 0.66 [cette intervalle mesure une contrainte moyenne du temps, l'utilisation de la stratégie Parallèle avec vote est au moins aussi bonne que le premier choix.

E_t : [0 ; 0.33 [cette intervalle mesure une contrainte forte du temps, l'utilisation du stratégie Parallèle sans vote sera le bon choix.

Intervalle de Coût :

$E_{Coût}$: [0 ; 0.33 [cette intervalle mesure une contrainte forte du Coût, l'utilisation du stratégie Séquentielle sera le bon choix.

$F_{Coût}$:] 0.66 ; 1] cette intervalle mesure une contrainte faible du Coût, l'utilisation du stratégie Parallèle avec vote et Parallèle sans vote sera le mauvais choix.

Intervalle de défaillance Byzantine (avec et sans spécification) et Crash :

E :[0 ; 0.33 [

M :] 0.33 ; 0.66 [

F :] 0.66 ; 1]

- **Matrice de concordance :**

La matrice des indices de concordance est donnée par :

	Seq	PAV	PSV
Seq	1	0.59	0.59
PAV	0.60	1	0.86
PSV	0.73	0.63	1

Nous présentons un exemple de calcul de l'indice de concordance :

C(Seq, PAV) = $\frac{0+0.33+0.5+0.66+0}{2.49}$ = 0.59

- **Matrice de discordance :**

La matrice des indices de discordance est donnée par :

L'indice de discordance est calculé pour une valeur de δ = 1 - 0 = 1

D(Seq, PAV) = $\frac{1}{1}$ = 1

	Seq	PAV	PSV
Seq	0	1	1
PAV	0.5	0	0.5
PSV	1	0.5	0

- **Construction du graphe :**

$D_{ij} \leqslant Sc$ et $C_{ij} \geqslant Sc \Rightarrow i \rightarrow j$(i domine j)

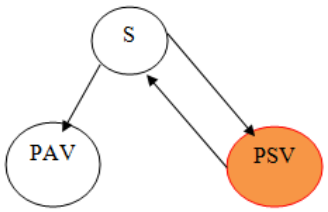
sinon

incomparabilité i et j (stratégies)

L'intérêt de la méthode Electre I est d'isoler un sous ensemble de solutions, dans notre cas identifier les stratégies avec le moins d'impacts.

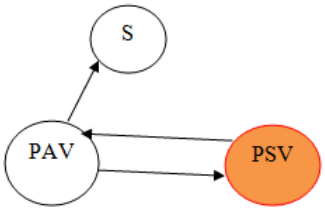
1er approche :

Sc = 0.59 ; Sd = 1



2éme approche :

Sc = 0.5 ; Sd = 0.6



3.8 Implémentation de l'algorithme

3.8.1 Choix de langage de programmation

Le choix de l'environnement ayant été décidé, il nous fallais maintenant choisir un langage de programmation. Nous comptons utiliser le langage Java couplé au XML.

a. Environnement de Développement Intégré (EDI) Java

- Définition

Les IDE sont des programmes qui regroupent un ensemble d'outils pour le développement de logiciels. De façon générale, un IDE contient un éditeur de texte, un compilateur, des outils automatiques de fabrication, et très souvent un débogueur. Il existe des IDE pour de nombreux langages, cependant il est très courant qu'un IDE soit conçu pour un seul langage de programmation.[14]

- Outil

Les besoins maintenant exprimés, il a fallut réaliser une présélection des IDE afin de ne pas se disperser avec un comparatif trop large. L' environnements retenu est donc NetBeans. Cela principalement du fait que c'est l' IDE java connue pour être le plus performante et le plus largement utilisée.

• NetBeans :

Cet IDE a été créé à l'initiative de Sun Microsystems. Il présente toutes les caractéristiques indispensables à un environnement de qualité, que ce soit pour développer en Java, Ruby, C/C++ ou même PHP. [14]

Il permet également de déployer des applications Web, non seulement vers Tomcat et Glassfish qui sont livrés avec le "Pack Web", mais aussi vers JBoss, WebSphere 6.1, WebLogic 9.

Enfin cet IDE possède un débogueur de grande qualité ainsi qu'une interface graphique améliorée.

b. L'API utilisée

- Présentation de JDOM

DOM est l'acronyme de Document Object Model. C'est une spécification du W3C pour proposer une API qui permet de modéliser, de parcourir et de manipuler un document XML.

Le principal rôle de DOM est de fournir une représentation mémoire d'un document XML sous la forme d'un arbre d'objets et d'en permettre la manipulation (parcours, recherche et mise à jour).[15]

Il existe des Outils API (Application Programming Interface) faits à partir de SAX et DOM pour une manipulation aisée des données XML (SAXP, SAXB, JDOM...). Nous avons choisit d'utiliser une de ces solutions pour notre projet et avons choisi JDOM.

JDOM est une API (Application Programming Interface) du langage Java développée indépendamment de Sun Microsystems. Elle permet de manipuler des donnés XML plus simplement qu'avec les API classiques. Son utilisation est pratique pour tout développeur Java et repose sur les API XML de Sun.[15]

- JDOM utilise des collections SAX pour parser les fichiers XML.

Nous avons choisi d'utiliser JDOM pour sa simplicité.

- Structure XML de la base

La structure de fichier respecte les spécifications XML et se limite sur une arborescence à deux niveaux. Le premier niveau de l'arborescence XML représente le groupe de diversité tandis qu'au deuxième niveau, sont regroupés les services avec leurs comportements. La FIGURE 3.5 représente le fichier XML de l'application.

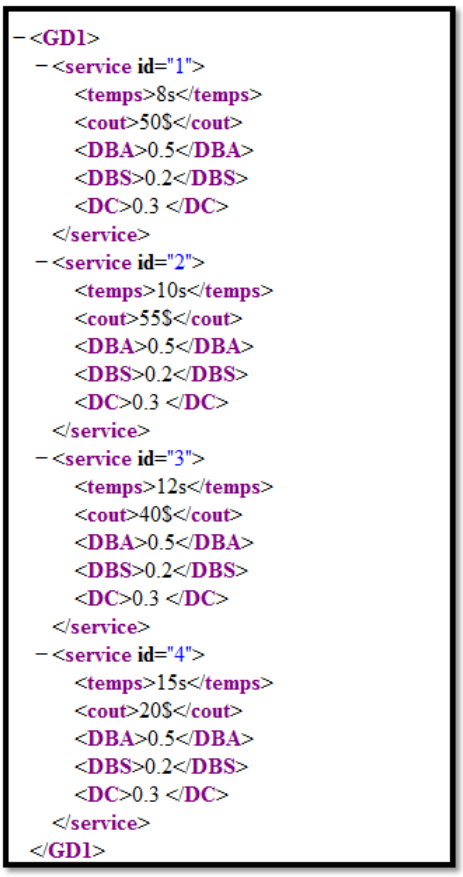


FIGURE 3.5 – Le modèle XML des services web avec leurs comportements

3.8.2 Diagramme Classe

La FIGURE 3.6 montre le diagramme de classe associé à la mise en œuvre de notre travail. Ce diagramme résume les principales classes de notre implémentation.

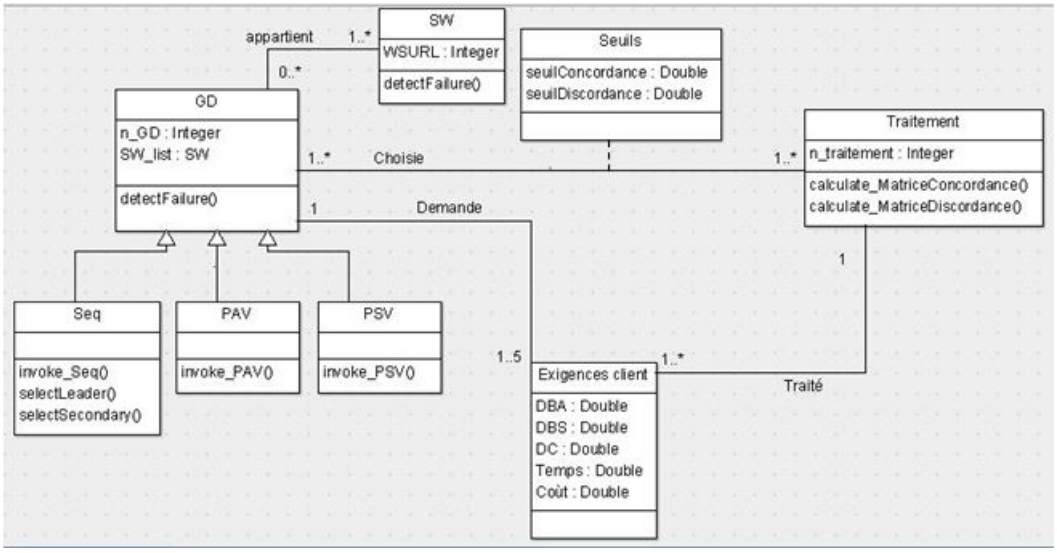


FIGURE 3.6 – Diagramme de classe de la solution

3.8.3 Description de l'application

Cette partie concerne la description de notre application, détermination de la meilleure stratégie de tolérance aux fautes.

Elle est composée d'un fichier XML qui contient les services Web et leurs comportements. Nous avons exécutés notre algorithme avec un exemple.

a. Page d'authentification

Cette forme permet à l'utilisateur de s'introduire à la page qui charge la base. Il devra saisir le mot de passe.

FIGURE 3.7 – Page d'authentification

b. Page qui charge la base de données

Permet de charger le fichier XML pour être traité par la suite.

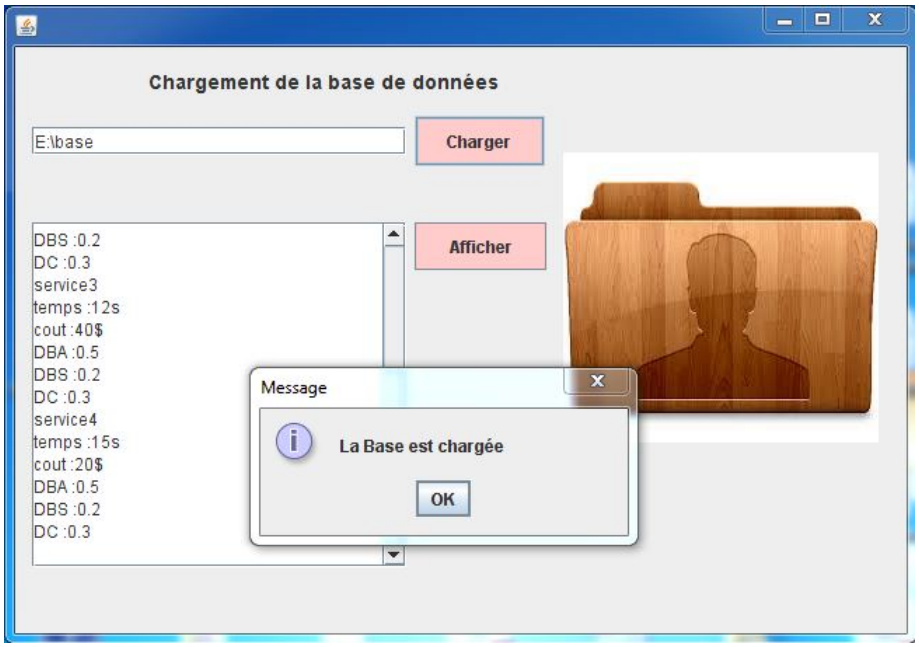


FIGURE 3.8 – Page qui charge la base

c. Page de traitement

Cette forme donne la main au client pour introduire ses exigences vis-à-vis de chaque critères. Elle affiche par la suite les deux matrices calculé par notre algorithme. Nous introduisons aussi les seuils de concordance et discordance.

- Exigences client

Cette partie de la forme traitement permet d'entrer les critères de client pour obtenir les matrices.

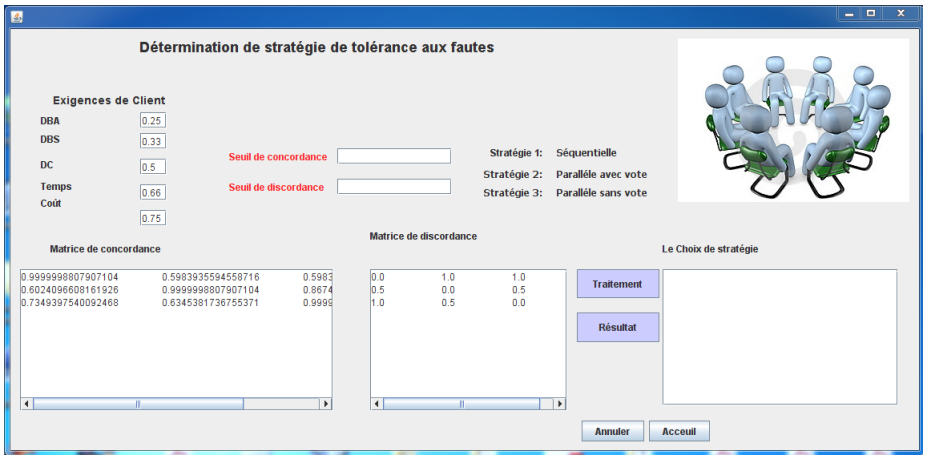


FIGURE 3.9 – Page de traitement (exigences client)

Si l'exigences de client dépassent le 1 , donc on a un message d'erreur.

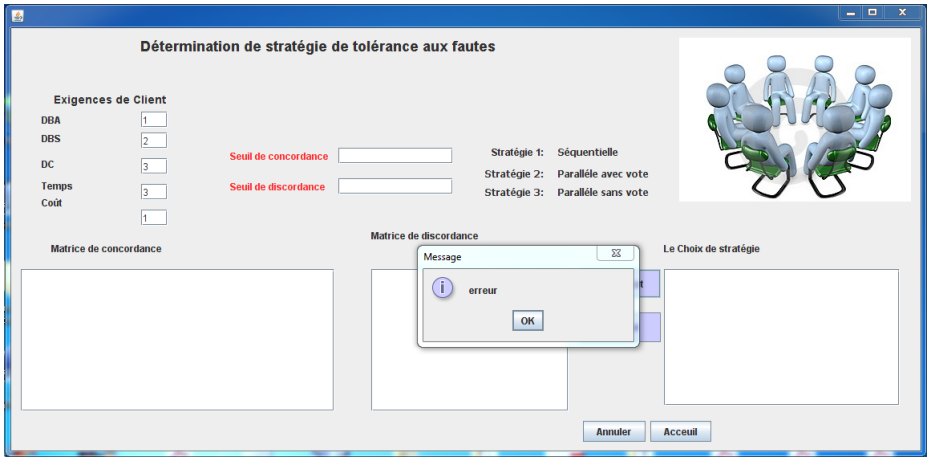


FIGURE 3.10 – Page de traitement (erreur)

- Déterminer le choix de stratégie

Cette partie présente la phase finale de l'algorithme, qui est la détermination d'une stratégie de TF basée sur la diversité et les exigences fonctionnelles de client en terme de taux de fautes acceptées, et en terme d'exigences non fonctionnelles (temps et le coût).

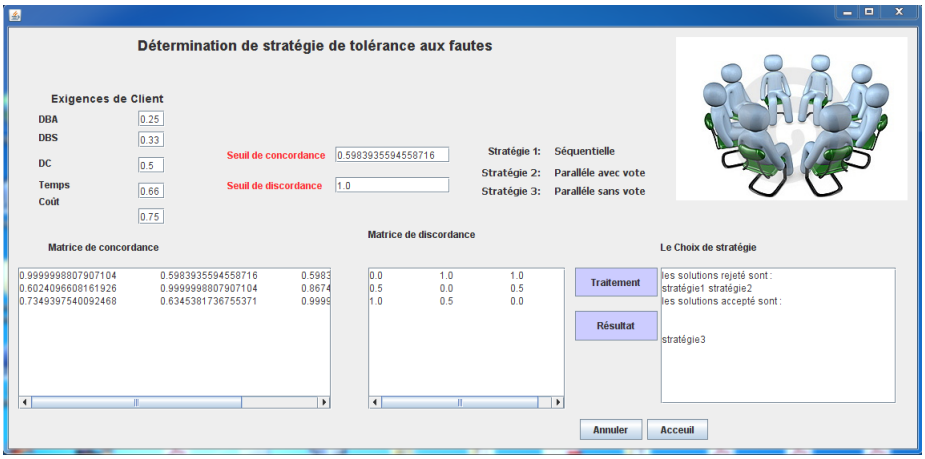


FIGURE 3.11 – Page de traitement (choix de stratégie)

3.9 Conclusion

L'élaboration d'une stratégie de tolérance aux fautes basée sur la diversité des services Web requiert la définition de différents détails concernant le groupe de diversité.

Nous avons proposé dans ce chapitre une approche de détermination automatique de stratégie de TF basée sur la diversité. Le processus de sélection est basé sur les exigences clients vis-à-vis les différents critères fonctionnels et non fonctionnels. Dans le cadre de cette approche, nous avons proposé une modélisation de la problématique par une des méthodes de surclassement qui est ElectreI, et un algorithme de calcul automatique de la stratégie de TF basé sur cette modélisation. Nous validons notre travail par une implémentation de notre algorithme sous IDE (NetBeans v6.8).

Chapitre 4

Conclusion et perspectives

Sommaire

4.1 Conclusion générale

Les Services Web constituent la technologie de base pour le développement d'Architectures Orientées Services. Ces architectures, de plus en plus répandues, permettent de mettre en place des applications critiques telles que celles relatives au contrôle aérien et au domaine militaire. Cependant, ses applications sont sujettes à des défaillances et ses défaillances peuvent avoir des conséquences catastrophiques selon la criticité de cette application. C'est pourquoi le faite de placer une confiance justifiée dans ses applications constitue un véritable challenge, cette propriété refaire à la sûreté de fonctionnement. Afin d'assurer le bon fonctionnement des applications notamment les services Web, nous devons déterminer la configuration de groupe de diversité. Cette configuration dépend de trois paramètres : à savoir le nombre de service Web dans le GD, les services Web à impliquer dans le GD et la stratégie de TF basée sur la diversité. Notre travail, nous focalisant sur la troisième question et plus précisément, une détermination automatique de la stratégie de TF selon les paramètres du contexte, en termes d'exigences client et des comportements antérieurs des services Web.

Modélisation de problème de détermination de stratégies de TF basées sur la diversité par une méthode de décision multi critères : Electre-I.

Nous avons modélisé notre problématique de détermination de stratégie de TF basée sur la diversité par une méthode de surclassement qui est Electre-I. En réalité, les critères de sélection sont conflictuels, c'est pourquoi, nous avons cherché à réaliser un compromis entre les différents critères.

Proposition d'un algorithme de sélection automatique de stratégies de TF. Nous avons proposé un algorithme de sélection automatique de stratégie de TF basée sur la diversité. L'algorithme se base sur la modélisation précédemment expliquée. Il détermine en fonction des paramètres d'entrées (exigences clients vis-à-vis de chaque critère et le comportement des SW's dans les exécutions précédentes) de retourner la meilleure stratégie qui assure un compromis entre les différents critères.

Implémentation. Nous avons implémenté notre algorithme sous l'IDE Netbeans, nous avons montré l'efficacité de cet algorithme qui détermine de façon automatique la stratégie de TF basée sur la diversité.

4.2 Perspectives

Dans l'objectif d'améliorer notre approche nous proposons différentes perspectives :

- Améliorer l'approche en prenant en compte d'autres modes de défaillance, ainsi que d'autres critères qui marque le comportement des services Web impliqué dans le GD dans les exécutions

précédentes. En réalité, comme nous avons déjà expliqué dans le chapitre 3, assurer le bon fonctionnement d'un GD dépend fortement des services Web à impliquer dans le GD, et par la suite même la stratégie de TF à utiliser dépend de quel service à impliquer dans l'exécution parallèle ou séquentielle. En fait l'utilisation des services Web « peu » fiable peut entrainer une augmentation du coût, du temps car elle nécessite plus de services Web pour recouvrir la faute.

- Combiner l'algorithme avec d'autres méthodes d'aide à la décision comme Electre II et Electre III, dans le but d'accroitre son efficacité (temps et optimalité).

- Tester notre approche sur une large base de services web équivalents et la comparer avec d'autres approches de choix.

Bibliographie

[1] H. Abdeldjelil. Une approche adaptative basée sur la diversité pour la gestion des fautes dans les services Web. Etablissement : Université Lyon1 2013.

[2] Algirdas Avizienis,Jean-Claude Laperie,Brian Randell, and Carl-wehr,Basic and taxonomy of dependable and secure computing,*Trans.Dependable Secur.*,2004

[3] Algirdas Avizienis. The n-version approch to fault-tolerant software.*IEEE Trans.Software Eng.*,11(12) :1491-1501. :1985.

[4] Mohamed Taha BENNANI and Jean-Charles FABRE.Tolérance aux Fautes dans les Systèmes Répartis à base d'Intergiciels Réflexifs Standards, pages 122-125,2005.

[5] Stefan Bruning,Stephan Weissleder,and Miroslaw Malek.A fault taxonomy for service-oriented architecture.In *Proceedings of the 10th IEEE High Assurence System Engineering Symposium*,HASE 07,pages 367- 368,Washigton,DC,USA,2007. IEEE Computer Society. : 2007.

[6] K.S. Chan,Judith Bishop,Johan Steyn,Luciano Baresi,and Sam Guinea.service-oriented computing-icsoc 2007 workshops.chapter A Fault Taxonomy for Web Service Compositon,pages 363-375.*Springer- Verlag,Berlin,Heidelberg*,2009.

[7] L. F. Cabrera, G. Copeland, M. Feingold, R. W. Freund, T. Freund, J. Johnson, S.Joyce, C. Kaler, J. Klein, D. Langworthy, M. Little, A. Nadalin, E. Newcomer, D. Orchard, I. Robinson, J. Shewchuk, and T. Storey, "Web Services Coordination (WSCoordination), Version 1.0," August 2005.

[8] David A.Chappell.*Entreprise service bus-theory in practice.*O Reilly,2004.

[9] Du Wan Cheun,Hyun Jun La,and Soo Dong Kim.A taxonomic framework for autonomous service management in service-oriented architecture.*Journal of Zhejiang University -Science C*,13(5) :339- 354,2012.

[10] N.Delgado,A.Q.Gate,and S.Roach.A taxonomy and catalog of runtime software-fault monitring tools.*IEEE Trans.Softw.Eng.,30 (12)*,2004.

[11] Glen Dobson,Stephan Hall,and Sommerville.A container-based approach to fault tolerance in service-oriented architectures,2005.

[12] J. Griffin, A. Associates, C. Hage, and C. H. Associates, "EDI Meets the Internet,"RFC 1865, `http://www.ietf.org/rfc/rfc1865.txt`, January 1996

[13] K. Heather. "Web services conceptual architecture (wsca 1.0) ", may 2001.`http://www-306.ibm.com/software/solutions/webservices/pdf/WSCA.pdf`.

[14] `http://www.developpez.com`

[15] Jean-Michel DOUDOUX, Développons en Java, `http://www.jmdoudoux.fr/java/dej/chap-jdom.htm`, 1999-2014.

[16] J.-C. Laprie, "Dependable Computing : Concepts, limits, challenges," *In 25th IEEE International Symposium on Fault-Tolerant Computing - Special Issue* pp. 42-54, 1995.

[17] J. -C. Laprie, J. Arlat, J. P. Blanquart, A. Costes, Y Crouzet, Y. Deswarte, J.C. Fabre, H. Guillermain, M. Kaâniche, K. Kanoun, C. Mazet, D. Powell, C. Rabéjac et P.Thévenod, "Guide de la sûreté de fonctionnement", 324p., Cépaduès-Editions, Toulouse, France, 1995.

[18] Nik Looker,Malcolm Munor,and Jie Xu.Increasing web service denpendability throught consensus voting.*In COMPSAC (2)*,pages 66- 69,2005.

[19] Christine Louberry,Philippe Roose,and Marc Dalmau. Kalimucho : Contextual deployement for qos management. In DAIS, pages 43- 56, 2011.

[20] G. Muller, M. Banâtre, N. Peyrouse, B. Rochat, “Lessons from FTM : An Experiment in the Design and Implementation of a Low Cost Fault Tolerant System” IEEE Transactions on Reliability, 45 (2), pp.332- 340, juin 1996.

[21] Organization for Advancement of Structured Information Standards(OASIS).*Web Services Business Process Execution Language (WS-BPEL)Version2.0*,April 2007.

bibitemPTDL03M.P.Papazoglou,P.Traverso,S.Dustdar,and F.Leymann.Service-oriented computing.*Communication of the ACM*,46,2003.

[22] Brian Randell.System structure for software fault tolerance.*IEEE Trans.Software Eng* .,221-232,1975.

[23] B. Roy : Méthodologie multicritère d’aide à la décision. Ed. Economica, 1985.

[24] Matjaz B.Juric.*Business Process Execution Language for Web Services BPEL and BPEL4WS* 2nd Edition.Packt Publishing,2006.

[25] Ian Sommerville,Stephen Hall,and Glen Dobson.Dependable service engineering ;a fault-tolerance based approach dependable service engineering :A fault-torance based abstract approach,2005.

[26] SOAP Version 1.2 Part o :Primer,2002.Working Draft.

[27] Srinivasan, L. and Treadwell, J. (2005). An overview of service oriented architecture, web services and grid computing. HP Software Global Business Unit

[28] François TAIANI,Jean-Charles FABRE and Marc Olivier KILLIJIAN.La Réflexivité dans les architectures multi niveaux : application aux systèmes tolérant les fautes.pages 75-77,2004.

[29] OASIS,"Universal Description, Discovery,and Integration (UDDI)- Version 3",UDDI Spec Technical Committee Draft, October 2004.

[30] W3C, "Extensible Markup Language (XML) 1.0 (Third Edition)," W3C Recommendation, `http://www.w3.org/TR/REC-xml/`, 04 February 2004.

[31] W3C, "Web Services Definition Language (WSDL) 1.2," W3C Working Draft, `http://www.w3.org/TR/2003/WD-wsdl12-20030611/`.

[32] W3C-WSA-Group(2004).W3C Web Service Architecture Group.Web

[33] M Wiesmann,F. Pedonne and A .Schipper, A Systematic Classification of Replited Database Protocols based on Atomic Broadcast, In Proceedings of the 3th European Research Seminar on Advances in Distributed Systems(ERSADS99), Madeira Island()Portugal, April 23-28,1999.BROADCAST Esprit WG22455.

[34] M.Wiesmann, F.Pedone, A. Schiper, B. Kemme , and G. Alonso. Database replication techniques : a threee parameter classification . In Proceedings of 19th IEEE Symposium on Reliable Distributed Systems(SRDS2000),Nüenberg , Germany , October 2000. IEEE Computer Society

[35] W3C,"Simple Object Access Protocol (SOAP) 1.2," `http://www.w3.org/TR/soap/`, 2003.