**REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE**

**MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET LA**

**RECHERCHE SCIENTIFIQUE**

**UNIVERSITE LARBI BEN M'HIDI-OUM EL BOUAGHI**

**FACULTE DES SCIENCES EXACTES ET DES SCIENCES DE LA NATURE ET DE LA VIE**

**Département Mathématique et Informatique**

**N°D'ordre :……..**

**Série :…….…..**

***MEMOIRE***

***POUR L'OBTENTION DU DIPLOME DE MAGISTER EN INFORMATIQUE***

***OPTION : Intelligence Artificielle et Imagerie***

***PRESENTE PAR : Fellah Hadjer***

***THEME***

# *CLOUD COMPUTING ET SECURITE :*

*Une architecture organique pour la sûreté de fonctionnement des processus métiers*

**Soutenu le : …….**

***Devant le jury composé de :***

| | | |
|---|---|---|
| **Pr. BENMOHAMED MOHAMED** | **Président** | **Université de Constantine2** |
| **Pr. BATOUCHE MOHAMED** | **Rapporteur** | **Université de Constantine2** |
| **Dr. BOUTEKKOUK FATEH** | **Examinateur** | **Université d'Oum El Bouaghi** |
| **Dr. NINI BRAHIM** | **Examinateur** | **Université d'Oum El Bouaghi** |

**2013/2014**

# Remerciement

C'est avec un grand plaisir que je réserve cette page en signe de gratitude et de profonde reconnaissance à tous ceux qui ont bien voulu apporter l'assistance nécessaire au bon déroulement de ce travail.

Je tiens à remercier en premier lieu mon « Dieu » de m'avoir accorde des Connaissances, de la science et de m'avoir aidé à réaliser ce travail.

Je veux remercier mon encadreur « Batouche Mohamed Chawki », professeur dans le département des sciences exactes à l'université de Constantine 2, qui n'a pas épargné le moindre effort dans l'encadrement de ce mémoire afin de me permettre de défier les entraves rencontrées et de travailler avec volonté et qui a été toujours disponible pour m'orienter à entreprendre les bonnes décisions. J'espère être à la hauteur de son confiance. Qu'il trouve dans ce travail le fruit de ses efforts et l'expression de ma profonde gratitude.

Je tiens à exprimer mon profond respect et mes vifs remerciements envers les membres du jury : Pr. Benmohamed Mohamed pour l'honneur qu'il m'a fait en acceptant de le présider et Dr. Nini Brahim et Dr. Boutekkouk Fateh pour l'honneur qu'ils m'ont fait en acceptant de lire et d'évaluer le manuscrit et en l'accompagnant de nombreuses suggestions pertinentes.

A tous ceux qui de loin ou de prés, ont contribué par leurs conseils, leurs encouragements et leurs amitiés, à l'édification de ce modeste travail, trouvent ici l'expression de me profonde reconnaissance. Je vous souhaite une bonne lecture.

# Dédicace

*À la mémoire de mon père ;*

*À ma chère mère ;*

*À mes sœurs et frères ;*

*À ma nièce et mes neveux ;*

*À toute ma famille ;*

*À tous mes amis ;*

*À tous mes collègues ;*

*Je dédie ce travail*

*Hadjer*

## Résumé

La méthode traditionnelle de création et d'exécution des applications d'entreprise s'est complexifiée et alourdie. Il y a trop d'éléments variables à acheter, installer, configurer et maintenir, logiciels comme matériels. Sans parler de l'infrastructure qui exige une maintenance constante pour pouvoir fonctionner comme il se doit. Ces charges générales constituent des obstacles à la productivité dans un développement d'applications métiers personnalisées. Les solutions de ***Cloud Computing*** sont conçues pour aider les entreprises à résoudre ces problèmes, à réaliser des économies et à simplifier leur structure informatique, en leur permettant de disposer d'applications métiers comme autant de «libre services», à la demande, mutualisés, dématérialisés, contractualisés et évolutifs. Mais, le développement rapide de l'utilisation du ***Cloud*** conduit à la publication de plus de services sur celui-ci. Et en raison de la présence de services complexes, un seul service simple ne peut pas satisfaire les exigences fonctionnelles existantes pour de nombreux cas. Alors, pour compléter un service complexe, il est essentiel d'avoir un ensemble de services simples atomiques coopérés les un avec les autres. Par conséquent, il existe un fort besoin d'incorporer une composition de service dans le ***Cloud,*** qui est facilitée par l'utilisation des ***architectures orientées service.*** Tandis que le nombre de services traités par les applications ***Cloud*** augmente exponentiellement, un défi majeur porte sur la ***disponibilité*** des services. En outre, Plusieurs types d'attaques dangereuses affectant la ***disponibilité*** des services ***Cloud*** qui ne sont pas spécifiques à l'environnement ***Cloud***, mais lancé largement dans les systèmes de ***Cloud*** en raison des caractéristiques des systèmes de nuages. L'objectif de ce travail est de proposer une architecture organique pour la sûreté de fonctionnement des processus métiers. L'architecture comprend une ensemble de modules qui contiennent une série de ***mécanismes de fiabilité*** liés aux deux entités essentiels dans le ***Cloud***: ***Les services Web*** et les ***machines virtuelles,*** et sont en mesure de créer une solution de tolérance aux fautes avec des propriétés désirées. Pour y parvenir, nous nous appuyons sur l'idée qu'une solution de tolérance de panne peut être considérée comme une combinaison d'un ensemble d'activités distinctes coordonnés dans une logique spécifique basé sur les événements. Ce système, basé sur l'approche organique, prend des décisions d'une manière autonome et sans une intervention extérieur, et ce, en utilisant des politiques de haut niveau. Son statut sera constamment vérifié et optimisé. En plus, le système s'adapte automatiquement aux conditions changeantes.

***Mots clés*** : *Informatique dans le nuage, sécurité, disponibilité, SOA et services Web, composition de services, tolérance aux fautes, l'informatique organique*.

## *Abstract*

The traditional method of creating and running business applications become more complex and heavier. There are too many elements to buy, install, configure and maintain software as hardware. Not to mention the infrastructure that requires constant maintenance to function as it should. These general expenses act as barriers to productivity in a custom business application development. Cloud Computing solutions are designed to help companies solve this problems, save money and simplify their IT infrastructure, enabling them to have business applications such as on demand, shared, dematerialized contracted and scalable "free services". But the rapid development of the use of cloud led to the publication of more services on it. And due to the presence of complex services, one simple service can't meet the existing functional requirements for many cases. So, to complete a complex service, it is essential to have a single set of atomic services cooperated with each other. Therefore, there is a strong need to incorporate a service composition in the Cloud, which is facilitated by the use of service-oriented architectures. While the number of services processes by ***Cloud*** applications increases exponentially, a major challenge addresses the ***availability*** of services**.** In addition, several types of dangerous attacks affect the ***availability* of *Cloud*** services that are not specific to the ***Cloud*** environment, but introduced widely in ***Cloud Computing*** systems due to the characteristics of ***Cloud*** systems. The purpose of the work is to propose an organic architecture for the dependability of business processes. Architecture modules contain a series of reliability mechanisms associated with two key entities in the ***Cloud*****:** ***Web services*** and ***virtual machines*****,** and are able to create a fault tolerant solution with the desired properties. To achieve this, we rely on the idea that a solution of fault tolerance can be considered as a combination of a set of distinct activities coordinated in a specific logic based on events. This system based organic approach takes decisions autonomously and without outside intervention using high-level policies, it will constantly check and optimize its status and automatically adapt to changing conditions.


***Keywords:*** *Cloud Computing, security, availability, SOA and Web services, services composition, fault tolerance, organic computing.*

## ملخـــص

الطريقة التقليدية في إنشاء وتشغيل تطبيقات الأعمال أصبح أكثر تعقيدا وأثقل. هناك الكثير من الأشياء التي تحتاج الشراء، التركيب، التكوين والحفاظ على البرمجيات والأجهزة. ناهيك عن البنية التحتية التي تتطلب صيانة مستمرة لتعمل كما يجب. هذه النفقات العامة هي العقبات التي تعترض الإنتاجية في تطوير تطبيقات الأعمال المخصصة. حلول الحوسبة السحابية مصممة لمساعدة الشركات على حل هذه المشاكل، وتوفير المال وتبسيط البنية التحتية لتكنولوجيا المعلومات الخاصة بهم، وتمكينهم من أن تكون لهم تطبيقات الأعمال كـ "خدمات مجانية" عند الطلب، مشتركة، متعاقدة، غير مادية وقابلة للتطوير. ولكن التطور السريع لاستخدام الحوسبة السحابية أدى إلى نشر المزيد من الخدمات. بسبب وجود الخدمات المعقدة، خدمة واحدة بسيطة لا يمكنها تلبية الاحتياجات في كثير من الحالات. لذلك، لإكمال الخدمة المعقدة، فمن الضروري أن يكون هناك مجموعة من الخدمات البسيطة المتعاونة مع بعضها البعض. وبالتالي، هناك حاجة قوية لدمج تكوين الخدمات في الحوسبة السحابية، والتي سهلت من خلال استخدام البنية الخدمية. في حين أن عدد الخدمات التي تتم معالجتها بواسطة التطبيقات السحابية يزيد أضعافا مضاعفة،هناك تحدي يظهر في توافر الخدمات. بالإضافة إلى ذلك، عدة أنواع من الهجمات الخطرة تؤثر على توافر الخدمات السحابية التي ليست خاصة بالبيئة السحابية، ولكن انتشرت على نطاق واسع في الأنظمة السحابية ويرجع ذلك إلى خصائص هذه النظم.

الهدف من هذا العمل هو تقديم بنية عضوية للتشغيل المأمون للعمليات التجارية. تتضمن هذه البنية مجموعة من الوحدات التي تحتوي على سلسلة من آليات الموثوقية المرتبطة باثنين من العناصر الرئيسية في الحوسبة السحابية: خدمات الويب والأجهزة الافتراضية، وهي قادرة على إنشاء حل التسامح مع الخطأ بالخصائص المطلوبة. لتحقيق ذلك، نحن نعتمد على فكرة أن حل التسامح مع الخطأ يمكن اعتباره مزيج من مجموعة من الأنشطة المتميزة منسقة في منطق معينة استنادا إلى الأحداث. هذا النظام، المعتمد على نهج العضوية، يتخذ القرارات بشكل مستقل ودون تدخل خارجي، باستخدام سياسات رفيعة المستوى. بالإضافة إلى ذلك، النظام يتكيف تلقائيا مع الظروف المتغيرة.


***الكلمات الرئيسية***: الحوسبة السحابية ، الأمن، توافر، البنية الخدمية ، خدمات الويب، تكوين الخدمات، التسامح مع الخطأ، ،الحوسبة العضوية.

## Table des matières

## Table des figures

## Liste des tableaux

# Introduction générale

*Si vous avez construit des châteaux dans les nuages, votre travail n'est pas vain, c'est la qu'ils doivent être. A présent, donnez –leurs des fondations.*

*Henry David Thoreau*

# Introduction générale

## 1. Contexte

Face à l'augmentation continuelle des coûts de mise en place et de maintenance des systèmes informatiques, les entreprises externalisent de plus en plus leurs services informatiques en les cédants à des entreprises spécialisées comme les fournisseurs de ***Clouds***. L'intérêt principal de cette stratégie pour les entreprises réside dans le fait qu'elles ne paient que pour les services effectivement consommés. Quant au fournisseur du **Cloud**, son but est de répondre aux besoins des clients en dépensant le minimum de ressources possibles. Une des approches qu'utilise le fournisseur consiste à mutualiser les ressources dont il dispose afin de les partager entre plusieurs entreprises. Pour ce fait, le ***Cloud Computing*** se base sur la technologie de ***Virtualisation*** qui est utilisée pour configurer des ressources informatiques via Internet en suivant le modèle « ***pay-per-use*** ». Dans le ***Cloud***, un seul hôte physique est souvent utilisé comme un ensemble de plusieurs hôtes virtuels par le fournisseur de service, ce qui présente un ensemble inépuisable de ressources de calcul disponibles fourni aux utilisateurs. En conséquence, un nombre croissant d'utilisateurs se tournent vers des services de ***Cloud*** pour la réalisation de leurs applications et leurs processus métiers. Les services fournis par les fournisseurs de services peuvent être des ressources d'infrastructures, de plates-formes ou de logiciels. Chacun de ces services est respectivement appelée ***Infrastructure as a Service*** (***IaaS***), ***Platform as a Service*** (***PaaS***) et ***Software as a Service*** (***SaaS***).

Généralement, pour répondre aux besoins d'un utilisateur ***Cloud***, qui sont devenus de plus en plus complexes, plusieurs services doivent être combinés pour former un seul service, on parle de ***la composition de services.*** Cette dernière est un des enjeux principaux du ***Cloud Computing***. Pour faciliter la composition de services, le ***Cloud Computing*** repose sur un concept qui a influencé les architectures ces derniers temps : l'***architecture orientée service*** (***SOA*** pour ***Service Oriented Architecture***). C'est un moyen approprié pour développer, avec un couplage lâche, des systèmes distribués. Il réutilise les services développés qui peuvent parvenir de différents fournisseurs de services.

Mais, en dépit des progrès rapides des applications de ***Cloud Computing*** au cours des dernières années, les risques de sécurité et de fiabilité constituent toujours un défi important. Plusieurs types d'attaques dangereuses affectent la ***disponibilité***, la ***confidentialité*** et l'***intégrité*** (***CIA*** : ***Confidentiality***, ***Integrity*** and ***Availibility***) des ressources et des services ***Cloud*** qui ne sont pas spécifiques à l'environnement ***Cloud***, mais lancé largement dans les systèmes de ***Cloud Computing*** en raison des caractéristiques des systèmes de nuages.

**2. Motivations et problématique:**

La dimension des risques sur les applications de composition de services dans le ***Cloud Computing*** a également changé, ce qui a un impact sur ***la disponibilité*** et les performances des services ***Cloud***. Aussi, des changements inattendus peuvent apparaitre au niveau des services comme l'arrivé de nouveaux services, l'échec d'exécution, service défectueux ...etc. En outre, les moyens traditionnels d'atteindre la tolérance aux fautes obligent les utilisateurs à avoir une connaissance approfondie des mécanismes sous-jacents, alors que, en raison des couches d'abstraction et le modèle métiers de ***Cloud Computing***, les détails architecturaux du système ne sont pas largement disponibles pour les utilisateurs. Cela implique que les méthodes traditionnelles de la tolérance aux fautes peuvent ne pas être très efficaces dans le ***Cloud Computing***.

Dans ce contexte, les applications nécessitent des capacités de tolérance aux fautes afin qu'elles puissent surmonter les effets de défaillances du système et exécuter correctement leurs fonctions en cas de défaillances. Le problème majeur qui nous intéresse est le fonctionnement sûr de la composition de services ***Cloud*** afin de répondre aux besoins de multiples utilisateurs et adapter cette composition à des changements d'une façon dynamique, transparente et sans intervention humaine.

**3. Contribution et objectifs**

Pour surmonter les limites des approches traditionnelles de tolérance aux fautes, une approche de sûreté de fonctionnement des processus métier basée sur l'***Organic Computing*** a été proposée. Cette dernière est inspirée des systèmes vivants et comporte des propriétés biologiques qui permettent l'adaptation dynamique aux conditions réelles

de son environnement. L'***Organic Computing***, « ou l'informatique Organique », est caractérisée par les ***propriétés d'auto-X*** : *auto-configuration*, *auto-organisation*, *auto-optimisation*, *auto-réparation*, et *auto-protection*. Dans cette architecture nous présentons différentes propositions pour la fiabilité des services Web et les machines virtuelles. La principale contribution est l'utilisation des techniques de tolérance aux fautes, comme « ***la reprise d'erreur*** », « ***le vote*** » ,« ***la Réplication*** » et « ***la migration*** » et ainsi que l'utilisation du langage d'exécution des processus métiers (***WS-BPEL***), ce qui permettrait de proposer une architecture fiable pour les services Web.

Dans ce mémoire, nous visons à fournir au fournisseur de services ***Cloud*** une solution pour composer un ensemble de services afin de répondre à une demande spécifié par un client qu'elle ne peut pas être satisfaite par un seul service. Considérons un scénario où un client ***Cloud*** demande un service de vidéo en ligne du fournisseur. Le service de vidéo en ligne est un service complexe qui se compose de trois services séparés dans le catalogue du fournisseur de service: un service de stockage (pour stocker le contenu multimédia), un service de transcodage (pour convertir les médias dans un format compatible avec le terminal de l'utilisateur) et un ***Service Web*** (pour fournir un portail Web pour l'utilisateur). La solution proposée dans ce mémoire vise, et se ne limite pas, à aider les fournisseurs de services de composer ces services pour fournir le service de vidéo en ligne demandée.

Puisque le ***Cloud Computing*** est un environnement dynamique, qui laisse les services et les machines virtuelles volatiles, pour lequel le processus de composition doit s'adapter aux changements de l'environnement, ce processus est analogue à un système vivant. L'application de l'***Organic Computing*** permet de veiller sur cet environnement et de contrôler ces changements afin de répondre aux exigences des utilisateurs. A côté de l'***Organic Computing***, des techniques de tolérance aux fautes, conforme à la technologie de ***Cloud***, ont été ajoutées pour améliorer la ***disponibilité*** des services et ainsi garantir la continuité de leur composition.

L'objectif principal de l'approche proposée est la mise en œuvre d'une architecture organique, fonctionnant sous l'***Hyperviseur,*** pour la composition sûre des services. Elle

est basée sur le fait que l'assurance de la haute disponibilité des services Web est l'une des principales caractéristiques des services de ***Cloud Computing*** et aussi l'un des principaux problèmes cruciaux et difficiles pour le fournisseur de services ***Cloud***. Cette solution doit assurer l'adaptation de composition aux changements de l'environnement ***Cloud***. Dans notre approche quatre paramètres de base sont pris en compte: le temps de réponse (*response time*), la disponibilité (*availability*) en cours de fonctionnement, l'exactitude des résultats (*correctness of results*), et la fiabilité des VMs (*VMs reliability*) exécutants les compositions.

Plus précisément cette contribution s'organise autour de :

- Une architecture générique pour répondre aux exigences de plus haut niveau en termes de disponibilité et de fiabilité ;
- Une architecture offrant la capacité d'exécuter des compositions de services ***Cloud*** à l'aide des modules. Chaque composant dispose d'un ensemble d'éléments logiciels de sorte qu'il participe à la fourniture, d'une manière transparente et sans l'intervention humaine, d'une composition adaptable aux changements de l'environnement ***Cloud*** ;
- Garantir une haute disponibilité des services composés par la mise en œuvre d'un ensemble de mécanismes de tolérance aux fautes tel que « ***watchdog*** » et « ***roll back*** » ainsi que la ***réplication*** et la ***migration des instances VMs ;***
- Assurer une gestion de composition par une surveillance de l'ensemble de machines virtuelles exécutant cette composition et effectuer une technique de correction de fautes après la détection de cette dernière ;
- Répondre aux exigences de l'utilisateur d'une manière transparente ;
- Garantir la composition de services tout en prenant en considération l'adaptation aux changements de l'environnement : au niveau de services ou bien machines virtuelles ;
- Contrôler l'exécution de la composition en veillant que le système s'auto-répare dans le mode d'échec ou en cas d'attaque.

**4. Structure du mémoire**

Afin de faciliter la lecture du manuscrit et d'en améliorer la compréhension, celui-ci a été construit selon un plan structuré. L'objectif de cette section est de montrer l'organisation des chapitres. Le présent manuscrit est décomposé en cinq chapitres :

Tout d'abord, le premier chapitre présente le contexte dans lequel s'inscrivent les travaux présentés dans ce mémoire : le « ***Cloud Computing*** » et ensuite les facteurs importants dans le développement de ce dernier ainsi que ses fondamentaux tel que les modèles de services, les modèles de déploiement, les avantages et quelques exemples d'utilisation. A la fin de ce chapitre nous décrivons les inconvénients et les défis de cette technologie.

Le deuxième chapitre est consacré à la présentation de notions essentielles de la sécurité ***Cloud***, les attaques et les différents problèmes de sécurité liés à ce paradigme.

Le troisième chapitre présente, en premier lieu, l'architecture orienté services, ses principaux éléments, son modèle de fonctionnement et ses avantages ainsi que la notion de services web et ses standards. Enfin, nous présentons la composition des services web et les différents langages de description de services en se focalisant sur le langage ***BPEL***.

Nous introduirons par la suite le quatrième chapitre par décrire les notions de sûreté de fonctionnement et la tolérance aux fautes, ensuite nous présentons les différents modèles basés sur les techniques de tolérances aux fautes dans le ***Cloud Computing.***

Le cinquième chapitre est dédié à la description de l'architecture proposée. Nous commençons par définir les notions de base de l'***Organic Computing***, puis nous passons à décrire en détails l'approche proposée et ses différents modules ainsi que les techniques de tolérance aux fautes utilisées**.**

# Chapitre I

# Cloud Computing

*« Le commencement de toutes les sciences, c'est l'étonnement de ce que les choses sont ce qu'elles sont »*

*Aristote*

## I. Introduction :

L'avancement rapide des technologies de l'information et de la communication a permis le développement de nouveaux paradigmes informatiques, où les techniques de traitement, de stockage, de communication, de partage et de diffusion de l'information ont radicalement changés. Les individus et les organisations sont de plus en plus recours à des serveurs externes pour le stockage et la diffusion efficace et fiable d'informations.

Le ***Cloud Computing***, ou « informatique dans les nuages », est une nouvelle technologie informatique qui permet le déplacement des traitements et fichiers informatiques de l'ordinateur local vers des serveurs distants. Elle consiste à proposer les services informatiques sous forme de services à la demande, accessibles de n'importe où, n'importe quand et par n'importe qui grâce à une connexion internet.

Basé sur le principe de fournir les ressources informatiques sous forme de services et de facturer leur utilisation en fonction de leur usage (***pay as you go*** en anglais), le ***Cloud Computing*** permet d'effectuer des économies d'échelle grâce à l'externalisation de ces ressources vers des fournisseurs spécialisés. C'est le cas par exemple de Google App Engine, Amazon EC2 ou Microsoft Azure, les offres permettant d'utiliser les infrastructures de calcul et de communication de Google, Amazon ou Microsoft comme des services utilitaires.

Ces offres proposent différents types de service généralement spécifiques à l'usage que les utilisateurs peuvent en faire. Une architecture en couches a été proposée pour catégoriser les différents types de services et les usages correspondants. Cette architecture est appelée ***modèle SPI*** (***Software/Platform/Infrastructure***).

Du fait de fournir un service à des utilisateurs qui paient pour celui-ci, les infrastructures, les plates-formes et les application de ***Cloud Computing*** doivent assurer une disponibilité quasi-totale de leur service tout en réduisant ou limitant les coûts pour les utilisateurs par rapport à un usage « ***Non Cloud*** » des ressources informatiques. Dans ce but, elles utilisent le plus souvent des ressources spécialisées ainsi que des techniques de tolérances aux pannes tout en gérant, dynamiquement et sans intervention humaine, les ressources allouées en fonction des besoins des utilisateurs ou des applications. On

parle d'élasticité des ressources. Le ***Cloud Computing*** ou notamment la mutualisation des ressources permet d'envisager une gestion plus efficace de celle-ci et donc une gestion plus efficace de la consommation énergétique.

Malgré les avantages offerts par le ***Cloud***, celui-ci intègre aussi quelques problèmes. Par exemple, la dynamique de la tarification rend difficile l'estimation du coût pour une entreprise utilisatrice d'offres de ***Cloud***. De plus, les solutions techniques utilisées dans les offres ***Cloud*** sont généralement « ***Vendor Locking*** », c'est-à-dire qu'elles enferment l'utilisateur dans la solution technique choisie rendant difficile la migration d'une solution à une autre.

## II. Historique

Le terme « ***Cloud*** », ou bien « ***nuage*** », a été utilisé historiquement comme une métaphore de l'Internet. Cet usage a été à l'origine dérivé de sa représentation commune dans les diagrammes de réseau comme l'ébauche d'un nuage, utilisé pour représenter le transport des données entre les réseaux fédérateurs porteurs (qui détenait le nuage) à un emplacement de point final sur l'autre côté du nuage. Ce concept remonte dès 1961, lorsque le professeur ***John McCarthy*** a suggéré que la technologie d'ordinateur en temps partagé « time-sharing » pourrait conduire à un avenir où la puissance de calcul et même les applications spécifiques pourraient être vendus comme un service public, [**34, 46**].

Cette idée est devenue très populaire dans les années 1960, mais au milieu des années 1970 l'idée s'évanouit quand il est devenu clair que les technologies liées aux IT de la journée étaient incapables de soutenir un tel modèle informatique futuriste. Cependant, depuis le tournant du millénaire, le concept a été revitalisé. C'est au cours de cette période de relance que le terme ***Cloud Computing*** a commencé à émerger dans les milieux technologiques, [**34**]. Le ***Cloud*** a entraîné un changement perturbateur dans la technologie. Ce changement, a été et, va continuer à révolutionner la façon dont les entreprises acquit et fournit des services IT.

# III. *Cloud Computing*

## III.1. Définition

La définition exacte du ***Cloud Computing*** est encore en évolution. Le ***Cloud Computing*** est un nuage de services et de données. Plus précisément, c'est un paradigme, et à ce titre il est difficile de lui donner une définition exacte et de dire avec certitude s'il s'agit ou non de ***Cloud.***

De nombreuses définitions ont été proposées pour le ***Cloud Computing***, en mettant l'accent sur les différents aspects qui caractérisent le paradigme, [**27, 39, 47**]. Nous considérons la définition de ***Cloud Computing*** proposée par l'Institut national de la norme et de la technologie (NIST), [**80**], car elle illustre les aspects essentiels du ***Cloud*** :

« *Le **Cloud Computing** est un modèle permettant un accès aisé, à la demande et au travers d'un réseau, à un ensemble partagé de ressources informatiques (par exemple des serveurs, des espaces de stockage, des applications) qui peuvent être rapidement mises en service avec un effort minimum de gestion et d'interaction avec le fournisseur de ce service.* »

L'expression « ***Cloud Computing*** », ou « informatique dans les nuages » peut se définir comme une approche visant à disposer d'applications, de puissance de calcul, de moyens de stockage, etc. comme autant de « services ». Ceux-ci seront mutualisés, dématérialisés (donc indépendants de toutes contingences matérielles, logicielles et de communication), contractualisés (en termes de performances, coûts…), évolutifs (en volume, fonction, caractéristiques…) et en libre-service.

Avec le ***Cloud Computing***, Les machines, les applications et les données pourront être disséminées ou centralisées dans un, ou dans différents sites internes, chez des prestataires, dans un data center « datacenters » situé à l'autre bout de la planète ou sur une myriade de serveurs appartenant à un même « nuage ».

La figure 1.1 présente une description générale du ***Cloud Computing*** selon **NIST** (**N**ational **I**nstitute of **S**tandards and **T**echnology [**80**]:

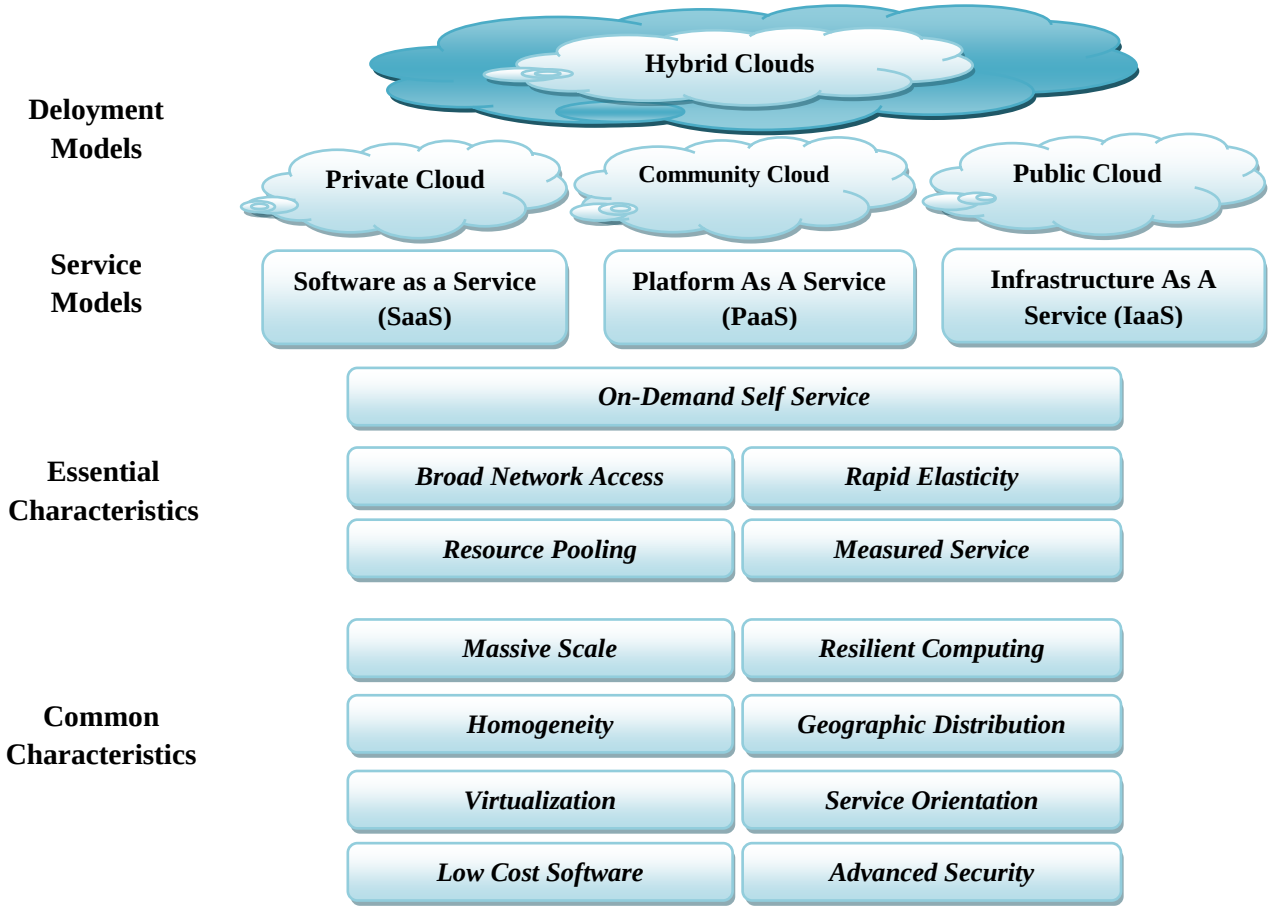



**Figure 1.1 : Description du *Cloud Computing* selon NIST [80]**

### III.2. Facteurs Importants Dans Le Développement De *Cloud Computing*

Le ***Cloud*** est développé à partir des technologies et des approches commerciales qui ont émergé au fil des années, telles que les logiciels des normes d'interopérabilité, les technologies de virtualisation, la communication à haut débit et Web 2.0 qui sont contribué à l'émergence du ***Cloud***. La figure 1.2 montre l'émergence de ***Cloud Computing,*** [70]:

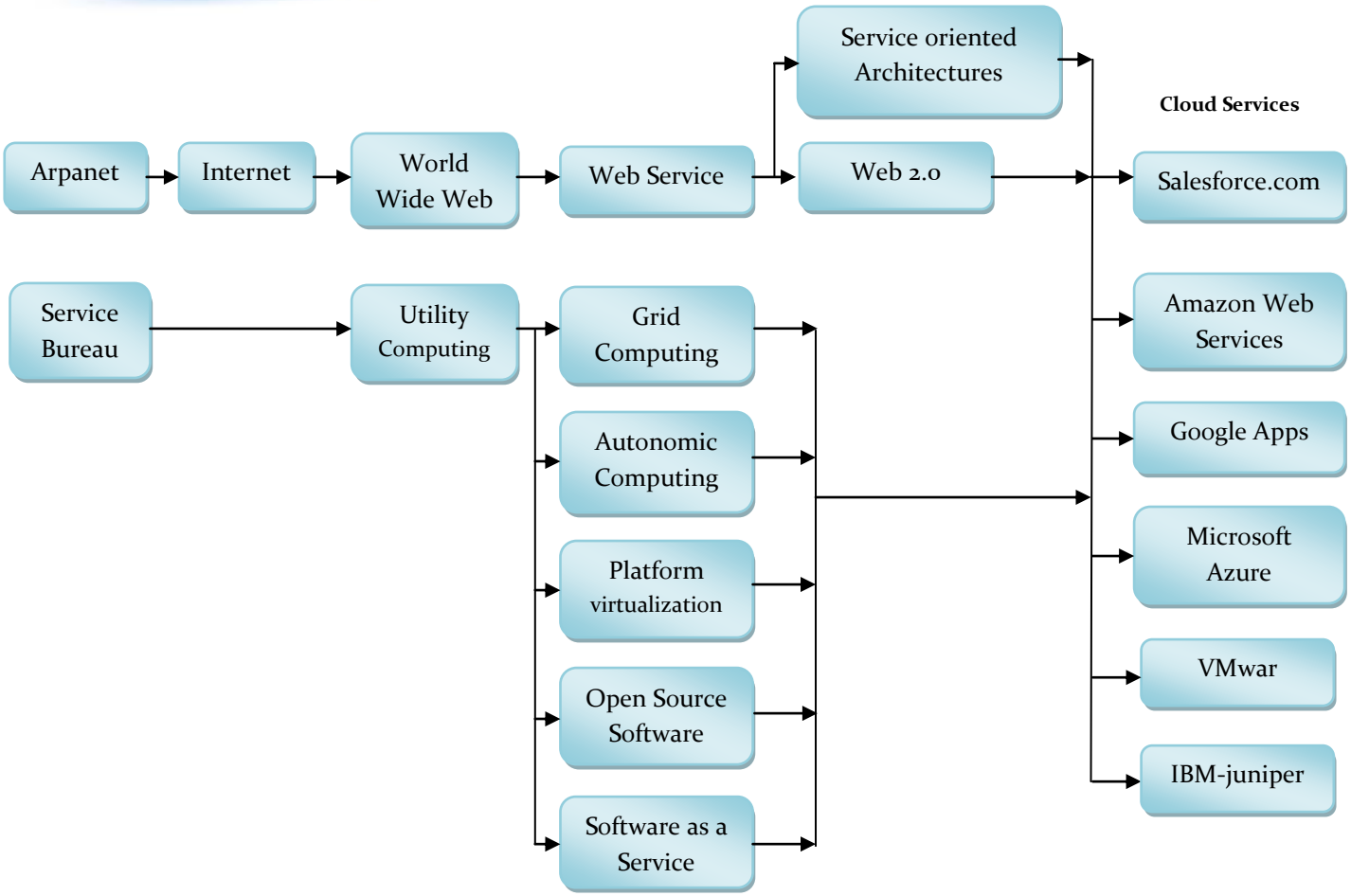



**Figure 1.2 : Origines de l'informatique dans les nuages [70]**

Les principaux éléments sont résumés dans les points suivants :

- **Informatique utilitaire « *Computer Utility* » :** Utility Computing peut être définie comme la mise à disposition (packaging) de ressources informatique, telle que le calcul et le stockage, comme un service mesuré en cas de besoin. L'objectif est d'utiliser efficacement les services tout en réduisant les coûts associés, [**70**]. Ce modèle a l'avantage d'un coût initial faible ou nul d'acquérir des ressources informatiques. Ce reconditionnement (*repackaging*) des services informatiques est devenu le fondement du passage à la informatique «*à la demande*», logiciel en tant que service « **SaaS** » et le ***Cloud Computing*** qui ont, plus loin, propagé l'idée de l'informatique, application et réseau en tant que service.

- **Informatique en grille « *Grid Computing* » :** Contrairement aux réseaux Traditionnels qui mettent l'accent sur la communication entre les appareils, l'informatique en grille exploite les cycles de traitement inutilisés de tous les ordinateurs dans un réseau pour résoudre des problèmes trop intenses pour toute machine autonome. Dans le ***Grid Computing,*** [70], les serveurs, le stockage et les réseaux sont combinés pour former des nœuds informatiques puissants qui peuvent être dynamiquement provisionnés selon les besoins.
- **Informatique autonome « *Autonomic Computing* » :** Se réfère aux caractéristiques d'auto-gestion ***(self-managing)*** des ressources informatiques distribuées, et l'adaptation aux changements imprévisibles, tout en cachant la complexité intrinsèque aux opérateurs et utilisateurs. ***L'AC,*** [70], est le fonctionnement d'un système informatique sans contrôle externe. L'objectif de l'informatique autonome, c'est d'avoir l'ordinateur exécuté des fonctions critiques et complexes, sans aucune intervention majeure par un utilisateur.
- ***Platform Virtualisation :*** C'est le partitionnement logique des ressources informatiques physiques dans des environnements d'exécution multiples, y compris les serveurs, les applications et systèmes d'exploitation. La ***virtualisation*** est basée sur le concept d'une machine virtuelle s'exécutant sur une plate-forme informatique physique. La ***Virtualisation*** est commandée par un moniteur de machine virtuelle (VMM), connu sous le nom d'un hyperviseur. [70].
- **Logiciel en tant que service « *SaaS : Software as a Service* » :** C'est une distribution de logiciel et modèle de déploiement dans lequel les applications sont fournies aux clients sous forme de service. Les applications peuvent s'exécuter sur les systèmes informatiques des utilisateurs ou les serveurs Web du fournisseur (ex : le gestionnaire de relation client « **CRM** : **C**lient **R**elation **M**anagement ») [70].
- **Architectures Orientées Services « *SOA* » :**

  C'est un ensemble de services qui communiquent les uns avec les autres, dont les interfaces sont connues et décrites, dont les fonctions sont faiblement couplés (le type d'interface n'est pas lié à la mise en œuvre), et dont l'utilisation peut être constituée par plusieurs organisations [70].

Les solutions ***Cloud*** reposent sur les technologies précédentes. Trois caractéristiques clés du ***Cloud*** le différencient des solutions traditionnelles :

- Services à la place de produits technologiques avec mise à jour en continu et automatiquement;
- Self-service et paiement à l'usage (en fonction de ce que l'on consomme);
- Mutualisation et allocation dynamique de capacité (adaptation élastique aux pics de charge).

### III.3. Les caractéristiques

La définition de **NIST, [80]**, décrit cinq caractéristiques essentielles du ***Cloud*** :

1) ***Libre service à la demande :*** Les utilisateurs des services de ***Cloud*** attendent un accès à la demande et, presque, instantané aux ressources. Pour soutenir cette attente, le ***Cloud*** doit permettre un accès libre-service « Self-Service » afin que les clients puissent demander, personnaliser, payer et utiliser les services sans l'intervention d'opérateurs humains, **[61]** ;
2) ***Un accès ubiquitaire au réseau***: les services de ***Cloud*** sont accessibles via le réseau, généralement Internet, en utilisant des mécanismes et des protocoles standards. Cet accès au ***Cloud*** est depuis n'importe où, **[70]** ;
3) ***La mutualisation des ressources informatiques (Multi-tenancy)***: les ressources utilisées pour fournir le service en nuage sont réalisés à l'aide d'une infrastructure homogène qui est partagé entre tous les utilisateurs du service. La mutualisation des ressources entre les utilisateurs du ***Cloud*** et l'utilisation simultanée de ces ressources communes sont des caractéristiques fondamentales du ***Cloud***. Un même client du ***Cloud*** va donc utiliser les mêmes accès réseaux, les mêmes machines ou les mêmes systèmes de stockage que d'autres clients, le tout étant rendu transparent via des technologies de **virtualisation**, **[21] ;**
4) ***Mesure de la qualité de services*** : c'est d'évaluer et garantir un niveau de performance et de disponibilité adapté aux besoins spécifiques des clients. Le ***Cloud*** élimine l'engagement initial par les utilisateurs en permettant de demander et d'utiliser seulement la quantité nécessaire, **[21].** Les services doivent être évalués sur une base à court terme (par exemple : à l'heure), permettant aux utilisateurs de

libérer (et ne pas payer pour) les ressources dès qu'elles ne sont plus nécessaires. Pour ces raisons, le ***Cloud*** doit implémenter des fonctionnalités pour permettre des échanges efficaces de service tels que la tarification, la compatibilité et la facturation, [**79**] ;

5) ***Élasticité rapide***: Le ***Cloud*** donne l'illusion de ressources informatiques (presque) illimitées et disponibles à la demande, c'est pourquoi les utilisateurs attendent que le ***Cloud*** fournisse rapidement des ressources dans n'importe quelle quantité à n'importe quel moment. Le besoin d'élasticité dans le ***Cloud*** était à l'origine de nouvelle solution de gestion des infrastructures. Cette élasticité va au-delà d'une simple allocation flexible des ressources lorsqu'un client a besoin de serveurs et d'espace de stockage supplémentaires. Les ressources peuvent être mises à l'échelle de haut en bas rapidement et de manière élastique c.-à-d. les ressources supplémentaires peuvent être provisionnés, éventuellement automatiquement, lorsqu'une charge d'application augmente et, libéré lorsque la charge diminue. Mais la dynamique introduit par l'élasticité (par exemple, la répartition rapide et la désallocation de ressources, etc) augmente d'une part la complexité de l'analyse des incidents et de réponse, et d'autre part, il peut fournir des moyens de contenir les attaques, [**35, 61**].

## IV. Les fondements technologiques de Cloud Computing

Avant de parler des technologies de base du ***Cloud***, nous allons définir les composants fonctionnels de ce dernier, [**71**]:

- **Le prestataire de service Cloud** (***CSP* pour *Cloud Service Provider***) : Il s'agit d'une entité qui gère le serveur de stockage ***Cloud*** (***CSS* pour *Cloud Storage Server***), l'espace de stockage pour préserver les données des clients et la puissance de calcul élevée ;
- **Le Client/propriétaire** (***Cloud Client*/*Owner***) : Il s'agit d'une entité, qui a d'une grande quantité de fichiers de données à stocker dans le ***Cloud*** et s'appuie sur ce dernier pour la gestion des données et du calcul, il peut être soit un consommateur individuel ou une organisation ;

- **L'utilisateur (*Cloud User*)**: Il s'agit d'une unité, qui est inscrit sur le propriétaire et utilise les données de celui-ci stockées sur le ***Cloud***. L'utilisateur peut être un propriétaire lui-même.
- **Courtier (*CB ou Cloud Broker*)** : En général, deux types de Brokers dans le ***Cloud*** peuvent être distinguées. Tout d'abord, il ya des Brokers qui se concentrent sur la négociation des relations entre les consommateurs et les fournisseurs sans posséder ou gérer l'infrastructure ***Cloud***. Ils fournissent, par exemple, des services de conseil aux consommateurs de ***Cloud*** pour déplacer leurs ressources informatiques dans un Cloud approprié. Deuxièmement, il ya des Brokers qui ajoutent des services supplémentaires sur le dessus de l'infrastructure / la plate-forme / le logiciel d'un prestataire de ***Cloud*** afin d'améliorer et sécuriser l'environnement ***Cloud*** pour les consommateurs. Par exemple, un Broker peut apporter au consommateur un service de gestion d'identité et d'accès au dessus de service de base offert par le fournisseur ***Cloud.*** À titre d'exemple, le Broker peut développer des API afin de rendre les services ***Cloud*** interopérables et portable, [73];

Les technologies et les infrastructures de base nécessaires pour construire un ***Cloud***, indépendamment du son type sont:

1) **Les centres de données « Data centers » :** Un ***Cloud*** a besoin de serveurs sur un réseau, et ils ont besoin d'une maison (station). Cette maison physique et tout le matériel y faire un centre de données. Le centre de données est un site hébergeant l'ensemble des systèmes nécessaires au fonctionnement des applications informatiques, [74]. Il est toujours constitué de trois composants élémentaires:
   - L'infrastructure, c'est à dire l'espace et les équipements nécessaires au support des opérations du data centre. Cela comprend les transformateurs électriques, les alimentations, les générateurs, les armoires de climatisation, les systèmes de distribution électrique, etc.
   - Les équipements informatiques comprenant les serveurs, le stockage, le câblage ainsi que les outils de gestion des systèmes et des équipements réseaux.

- Les espaces d'exploitation, c'est-à-dire le personnel d'exploitation qui pilote, entretient et répare les systèmes lorsque cela est nécessaire ;

Le ***Cloud*** contient des centres de données avec 10000 ou plus de serveurs sur site, toutes consacrée à l'exécution des applications qui sont construites avec des composants d'infrastructure cohérente (tels que grilles, matériel, OS, réseau et ainsi de suite), [**36**].

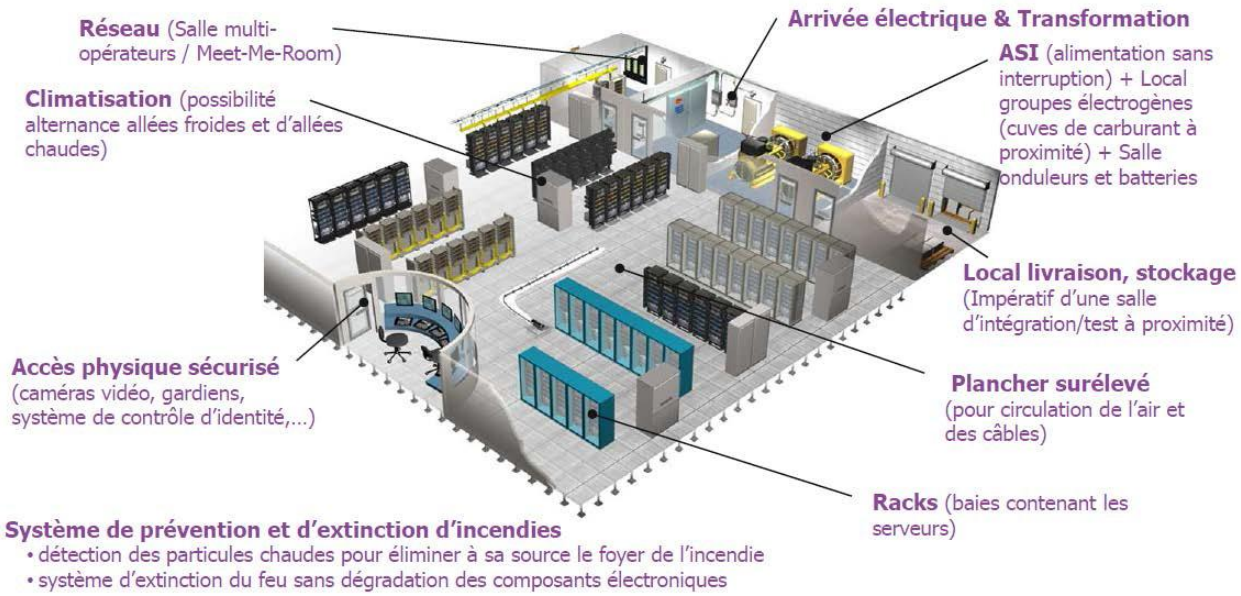



**Figure 1.3 : Une salle d'hébergement de serveurs ou data center [17]**

**2)** La Virtualisation :

La virtualisation est une méthode d'exécuter plusieurs systèmes d'exploitation virtuels indépendants sur un seul ordinateur physique. La création et la gestion des machines virtuelles a souvent été appelée virtualisation de plate-forme (*platform virtualisation*). La Virtualisation de plate-forme est exécutée sur un ordinateur donné (plate-forme matérielle) de logiciel appelé un programme de contrôle. Le programme de contrôle crée un environnement simulé, un ordinateur virtuel, ce qui permet d'utiliser un logiciel hébergé spécifique à l'environnement virtuel, parfois appelé logiciel invité. (*Guest Software*), [**34**]. La Virtualisation peut être d'une grande utilité pour les systèmes de *Cloud* comme il peut améliorer la mutualisation des ressources et permettre l'approvisionnement des ressources rapides et élastique. Ces avantages font de réseaux flexibles, agiles menant à d'importantes réductions de coûts. Dans les

applications de ***Cloud*** typiques, des serveurs, des dispositifs de stockage et réseaux peuvent tous être virtualisés, **[64, 70]**

Certains des principaux avantages de la virtualisation, qui sont indigènes au ***Cloud***, sont les suivants:

- Une facturation basée sur l'utilisation (la tarification des services) et non pas la capacité matérielle fixe ;
- Le déploiement rapide de serveurs supplémentaires ;
- La promotion des économies d'échelle ;
- La séparation de la clientèle des d'emplacements de serveurs physiques ;
- L'utilisation repose sur Service-Level Agreements (SLA) ;
- La tolérance aux pannes ;
- La mobilité des applications entre les serveurs et les centres de données.

Une **VM** est un conteneur de logiciels totalement isolé, capable d'exécuter ses propres systèmes d'exploitation et applications, à l'instar d'un ordinateur physique. Une machine virtuelle se comporte exactement comme un ordinateur physique. Elle contient un processeur, une mémoire RAM, un disque dur et une carte d'interface réseau virtuels (autrement dit, basés sur des logiciels) qui lui sont propres, **[92]**.

En générale, la **virtualisation** est réalisée par l'utilisation d'un « **Hyperviseur** ». L'**Hyperviseur** est un logiciel qui permet à plusieurs images virtuelles de partager une seule machine physique et d'attribuer et séparer logiquement les ressources physiques. L'**Hyperviseur** permet au matériel informatique d'exécuter plusieurs systèmes d'exploitation invités en même temps. Chacun des systèmes d'exploitation invités est isolé et protégé contre tous les autres fonctionnant sur la même machine physique et n'est pas affecté par des problèmes ou de l'instabilité qui se produisent sur d'autres machines virtuelles.

Plusieurs **Hyperviseurs** puissants incluant **KVM** (**K**ernel-based **V**irtual **M**achine), **Xen** et **QEMU** sont open source. **VMWare** est actuellement le leader du marché dans le domaine de la **virtualisation** et plusieurs de ses produits sont basées sur l'open source.

La figure **1.4** montre une vue de haut niveau d'un **Hyperviseur** où les ressources de la machine hôte sont partagés entre un nombre d'invités, dont chacun peut exécuter des applications et chacun a un accès direct aux ressources physiques sous-jacents.

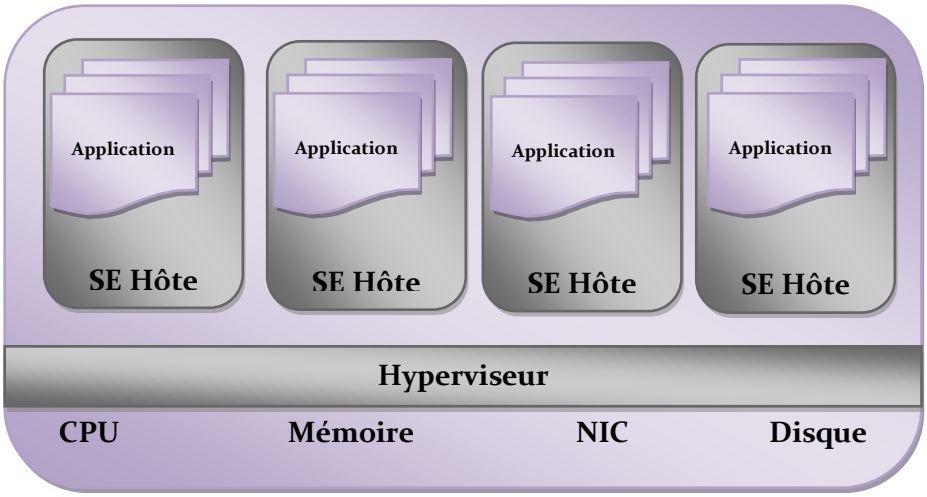



**Figure 1.4 : Hyperviseur**

### 3) Les API d'accès

Un ***Cloud*** a besoin d'une API d'accès. Les utilisateurs de ***Cloud*** ont besoin d'un moyen pour accéder aux nuages, fournir de nouveaux serveurs virtuels, obtenir des données dans et hors de stockage, démarrer et arrêter les applications sur les serveurs et déclasser les serveurs qui ne sont plus nécessaires. Tout cela doit être possible à distance, parce que les utilisateurs de ***Cloud*** jamais mis les pieds à l'intérieur du centre de données.

## V. Les Différentes Couches du *Cloud Computing*

Partant des capacités d'abstraction et du paradigme des « services », le ***Cloud*** peut être représenté en trois composants principaux, (figure 1.5), dont il est indifféremment l'une, les deux ou les trois combinées :

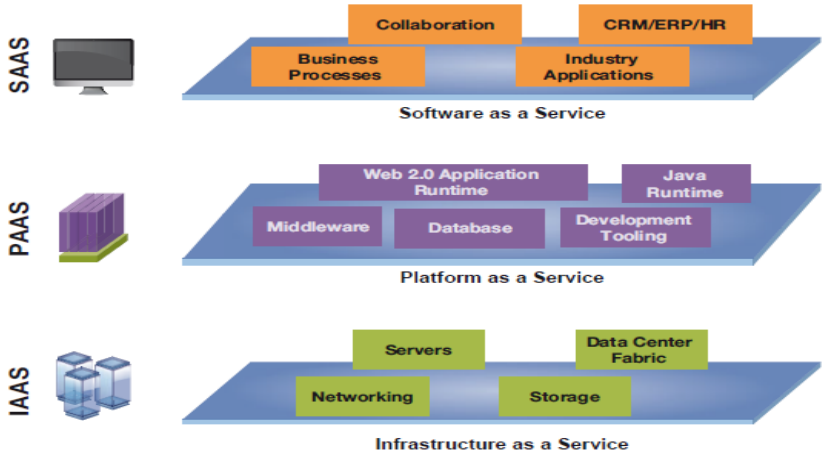



**Figure 1.5 : la pile de *Cloud Computing***

### V.1. Infrastructure en tant qu'un Service (*IaaS* pour Infrastructure as a Service)

L'infrastructure fournit des capacités de calcul et de stockage ainsi qu'une connectivité réseau. Les serveurs, les systèmes de stockage, les commutateurs, les routeurs et autres équipements, sont mis à disposition pour gérer une charge de travail demandée par les applications, [**41**].

L'***IaaS*** permet de disposer une infrastructure à la demande, pouvant héberger et exécuter des applications, des services ou encore stocker des données. Concrètement, cela se caractérise par une infrastructure physique souvent mise à disposition par un fournisseur de services. La solution de ***virtualisation*** permet la création des «***centres de données virtuels*** ».

Grâce aux solutions de ***virtualisation*** existantes, il est possible de créer très facilement des machines virtuelles connectées sur des réseaux, aussi virtuels, et qui seront exécutés sur les ***Hyperviseurs*** des machines physiques. Cette ***virtualisation*** donne une grande flexibilité parce qu'elle permet d'abstraire la couche matérielle sur laquelle les applications qui vont pouvoir être déployées et redéployées sans être liées à un serveur spécifique. La ***virtualisation*** répond de manière dynamique là où les serveurs physiques fournissent un ensemble de ressources allouées selon les besoins, et où la relation entre les applications et les ressources de calcul, de stockage et de réseau, pourront s'adapter de manière automatique pour répondre à la charge de travail et aux exigences demandées.

Les systèmes ***IaaS*** comprennent généralement une partie ou l'ensemble des fonctions suivantes, [**41**]:

- Le choix des machines virtuelles prêtes à l'emploi avec des systèmes d'exploitation préinstallés, y compris de nombreuses versions de Windows, Linux et Solaris;
- Un choix d'Appliances virtuelles machines virtuelles avec des ensembles spécifiques de logiciels préinstallés;
- Possibilité de stocker des copies des données particulières dans différents endroits à travers le monde pour faire de téléchargements des données aussi vite que possible;
- Outils logiciels pour aider à traiter les grandes quantités et d'effectuer des calculs complexes à l'aide de grands réseaux de serveurs virtuels fonctionnant en parallèle sur le même problème;
- Capacité d'augmenter ou diminuer manuellement les ressources informatiques qui sont assignées à l'utilisateur à l'aide d'un navigateur Web en fonction de ses besoins;

**Exemple d'IaaS :**

Dans cette section, nous nous concentrons sur les ***Cloud*** IaaS, allant de nuages commerciaux populaires aux projets émergents de recherche open-source, car ils offrent les services de base sur lesquels toutes les autres couches sont construites, ainsi que le plus haut degré de flexibilité et de contrôle :

- **Amazon Elastic Compute Cloud :** Amazon EC2, [**5,6**], fournit une grande infrastructure informatique, où les utilisateurs peuvent louer des environnements virtualisés. Il offre l'illusion d'une capacité de calcul infinie, ce qui permet aux utilisateurs de redimensionner dynamiquement les ressources louées, tout en utilisant un modèle d'affaires « pay-per-use ». C.-à.-d., un utilisateur crée une image de la machine virtuelle, appelée Amazon Machine Image (**AMI**), contenant toutes les applications nécessaires, [**4**];

  Amazon offre un large ensemble d'outils pour télécharger l'image dans le système et lancer plusieurs instances. Une instance Amazon **EC2** est une machine

virtuelle démarrée à partir d'un AMI spécifique, en s'appuyant sur le moteur de virtualisation **Xen**. **EC2** fournit des environnements flexibles qui peuvent être adaptées à une multitude d'applications, allant des services Internet aux applications scientifiques. Même si l'architecture et la mise en œuvre des services Amazon est propriétaire, l'interaction avec le client se fait via les APIs publiques qui sont devenus très populaires et ont été adoptées par un grand nombre de projets open-source ;

- **Nimbus :** Nimbus, **[52]**, est une application open-source ***IaaS*** conçu pour les besoins informatiques de la communauté scientifique. Son interface est compatible avec celle d'Amazon **EC2**, pour permettre aux utilisateurs de passer d'une offre ***Cloud*** à une autre sans modifier leurs outils d'accès. Le projet **Nimbus** fournit ainsi un ensemble d'outils supplémentaires destinés aux applications scientifiques **[4]**.Ainsi, il prévoit des mécanismes pour créer des clusters virtuels configurables, des outils d'interopérabilité pour explorer le fédération de ***Cloud*** et des composants de gestion de machine virtuelle efficaces pour soutenir le déploiement simultané de centaines de machines virtuelles, **[37]** ;
- **OpenNebula :** **[55]**, fournit des services d'***IaaS*** entièrement open source conçus pour répondre aux besoins des entreprises (requirments of business) dans de nombreux secteurs, telles que l'hébergement web, télécommunications, administration en ligne (e-Government). Il consiste en un ensemble d'outils de ***virtualisation*** pour la gestion des centres de données locaux, mais aussi pour l'interconnexion de plusieurs environnements de ***Cloud,*** **[4, 60]** ;

  Les plus importants principes de conception sur lesquelles s'appuie le projet d'**OpenNebula** comprennent une architecture modulaire et extensible, évolutive pour les infrastructures à grande échelle, l'interopérabilité avec les offres existantes de ***Cloud***, implémentation open-source. En outre, **OpenNebula** vise à fournir des interfaces standardisées pour la gestion des **VMs** et des données, telles que l'API de l'OCCI **[53],** Open Grid Forum **[54]** ou le standard « de facto » d'Amazon EC2.
- **Eucalyptus :** Le projet Eucalyptus, **[20],** vise à répondre à plusieurs exigences importantes de ***Cloud*** de type ***IaaS*** : extensibilité, évolutivité, haute disponibilité,

reconfiguration élastique des **VMs**. Il est entièrement compatible avec l'interface Amazon **EC2** et il comprend des outils robustes pour la création de ***Clouds*** hybrides reposant sur **EC2**. Eucalyptus est l'une des solutions ***IaaS*** les plus largement utilisés pour les centres de données privés. Pour répondre aux besoins des environnements d'entreprise, le ***Cloud*** **Eucalyptus** est équipé des services sécurisés pour la gestion des utilisateurs et des ressources, ainsi que de nombreux outils pour contrôler les cycles de vie des **VMs**, **[4, 14]** ;

- **OpenStack :** c'est une boite à outils (toolkit) open source, **[56],** conçu pour configurer et gérer des grands réseaux des **VMs**. Il peut être intégré avec une variété de configurations matérielles et ***Hyperviseur***, stockage et solutions réseau. Il fournit des services fiables et évolutifs adaptés à une large classe de centres de données. En outre, il définit des API spécifiques pour permettre aux utilisateurs de créer, lancer et d'interagir avec leurs instances facilement et en toute sécurité, **[4].**

### V.2. Plateforme en tant qu'un Service (PaaS pour Platform as a Service)

***PaaS,*** **[81],** est la plate-forme d'exécution, de déploiement et de développement des applications. ***PaaS*** met à disposition des environnements prêts à l'emploi, fonctionnels et performants et offre aux consommateurs un environnement stable en ligne où ils peuvent créer, tester et déployer des applications Web en utilisant des outils de développement de logiciels. Il ya moins de travail impliquée dans la création d'une application utilisant ***PaaS*** que dans l'approche traditionnelle, **[41].** Les systèmes ***PaaS*** comprennent généralement une partie ou l'ensemble des fonctions suivantes:

- Environnement de développement basé sur un navigateur pour créer des BDs et éditer le code des applications soit directement, soit à travers des outils visuels;
- Sécurité, contrôle d'accès et interfaces de services Web;
- Intégration facile avec d'autres applications sur la même plate-forme;
- Outils pour se connecter à des applications en dehors de la plate-forme;
- Outils pour la conception de formulaires web, la définition des règles d'affaires et la création de workflows.

Certaines solutions ***PaaS*** permettent aux non-développeurs de créer des applications Web en utilisant des outils visuels plutôt qu'un langage de programmation, et certains donnent le meilleur des deux (outils visuels et langage de programmation) pour utiliser les outils visuels pour créer des applications et le langage de programmation pour étendre les fonctionnalités si nécessaire, [**41**].

Grâce aux ***PaaS***, le déploiement d'applications dans différents environnements est très facile (test, pré-production et production sans se soucier de l'infrastructure et de la plate-forme dans lesquelles vont s'exécuter l'application ou le stockage de données).

**Exemples PaaS**

- **Amazon Web Services (AWS)** : permet aux développeurs d'héberger, de déployer et gérer des applications dans le nuage.
- **Google AppEngine** : est la réponse de Google à la tendance actuelle de ***Cloud Computing*** dans l'industrie. Il permet aux organisations de créer et d'héberger leurs applications Web. Le vendeur promet une administration centralisée, de disponibilité de 99,9%, et de la **sécurité.**
- **Microsoft Azure**: Microsoft offre des services de plate-forme **PaaS** à l'aide de Windows Azure. Ces services de middleware de nuage comprennent Service Bus, contrôle d'accès, la mise en cache, l'intégration et application Composite. Le vendeur assure la compatibilité avec tous les langages de programmation et les cadres de développement, y compris les langages de programmation .NET, Java, Ruby et PHP.

### V.3. Logiciel en tant qu'un Service (SaaS pour Software as a Service)

La dernière couche est celle applicative mettant à disposition des applications complètes fournies à la demande. On y trouvera différents types d'application allant du **CRM** (***C**ustomer **R**elationship **M**anagement),* à la gestion des ressources humaines, outils collaboratifs, messagerie, **BI** (***B**usiness **I**ntelligence*) et d'autres applications métiers.

*Il n'y a donc aucun prérequis sur le poste client si ce n'est d'avoir un accès réseau au* ***Cloud*** *(en général Internet). Le déploiement, la maintenance, la supervision du bon*

*fonctionnement de l'application et la sauvegarde des données, sont alors de la responsabilité du fournisseur de services,* **[89].**

Sur cette couche, des acteurs tels que **Salesforce.com**, proposent des applications à la demande de type **CRM** et des outils de collaboration. Cela permet à un client de bénéficier d'une application de manière instantanée et à la demande sans aucun frais en immobilisation (serveur ou licence), aucun déploiement ou maintenance à assurer. Il suffit de créer un compte et de profiter immédiatement d'un service de collaboration ou d'un **CRM** prêt à l'emploi. L'exemple le plus populaire et familière du ***SaaS*** est « e-mail » dans un navigateur web, mais les applications ***SaaS*** sont de plus en plus sophistiqués.

Les Capacités ***SaaS*****, [41],** fournis en ligne comprennent des outils pour:

- La gestion financière, les stocks et la commerce électronique;
- La collaboration entre les employés et les clients sur des projets;
- Créer des organigrammes, des schémas, des plans et autres dessins techniques;
- Les relations clientèles (**CRM**);
- L'édition, le stockage et le partage de documents, présentations, feuilles de calcul, blogs, pages Web et des vidéos;
- La gestion de projet;
- Web-mail, agenda, messagerie instantanée, la vidéoconférence et les réseaux sociaux.

Les avantages des services en ligne de type **SaaS, [41],** sont :

- Souplesse et facilité de mise en œuvre
- Fin du déploiement de solution monolithique et lourde
- Mise à jour du côté éditeur
- Tarification plus réelle à la consommation
- Migration des données vers les services en ligne

**Exemples d'utilisations :**

1) **Les services de stockages en ligne :** L'avantage principal du stockage dans le ***Cloud*** (ou stockage en ligne) c'est de pouvoir disposer des documents a partir de n'importe qu'elle appareil connecté à Internet.

| Nom | Adresse du service | Caractéristiques de l'offre gratuite |
|---|---|---|
| **Dropbox** | https://www.dropbox.com/ | 2 go + synchronisation<br>Possibilité de partage |
| **HubiC** | http://www.ovhtelecom.fr/hubiC/ | 25 go |
| **ICloud** | https://www.apple.com/fr/icloud/features/ | 5 go<br>obligation d'avoir un iphone ou un ipad avec IOS5 |
| **Skydrive** | https://skydrive.live.com/ | 25 go<br>La taille maximale des fichiers est de 100 Mo. |
| **Minus** | http://minus.com/ | 50 go (cliquer-déposer, pas de synchro mais url perso et possibilité de partager des données) |
| **Opendrive** | https://www.opendrive.com/ | 5 go +synchro (Windows, Apple, Android, Iphone, Ipad) |
| **cx** | https://www.cx.com/ | 10 go + synchro ( Windows, Apple, Android, Iphone, Ipad) |

**Tableau 1.1: Quelques services de stockage en Cloud**

**2) Les applications en ligne**

De même que le stockage en ligne, le **Cloud** offre la possibilité d'utiliser des programmes distants directement sur internet via un navigateur web.

Les avantages des applications en ligne :

- Pas besoin de télécharger ou d'installer ces programmes, utilisation directe ;
- Ils fonctionnent quelque soit votre système d'exploitation (windows, Mac, Linux,...) ;
- Ces programmes sont accessibles depuis n'importe quel ordinateur connecté à internet ;
- La mis à jour de ces programmes n'est pas nécessaire ;
- La majorité de ces programmes sont gratuit, nécessitant éventuellement une inscription.

En outre de ces avantages. Les applications en ligne dans le Cloud présentent des inconvénients :

- Sans connexion internet les programmes en ligne ne sont pas accessibles ;
- La vitesse de la connexion internet détermine la vitesse d'exécution du programme ;
- Il faut être attentif à la sécurité des données que les programmes peuvent garder.

Le tableau suivant présente une liste non exhaustive d'applications en ligne:

| Type d'application en ligne | Le nom d'application | Le site web |
|---|---|---|
| **Suite bureautique** (traitement de texte, tableur, présentations, etc…) | **Google Document** | http://docs.google.com |
| | **gOFFICE**: | http://goffice.com/ |
| | **Zoho:** | http://www.zoho.com/ |
| | **iNetWorld**: | www.inetword.com/ |
| **Création de PDF** | **PDF Online:** | http://www.pdfonline.com |
| **Convertisseurs** | **Zamzar** | http://www.zamzar.com/ |
| | **MediaConverter** | http://www.mediaconverter.org/ |
| **Bureaux virtuels** | **JoliCloud** | http://www.jolicloud.com/ |
| | **eyeOS;** | http://www.eyeos.com/ |
| **Calendriers** | **Kiko** | http://kiko.com/ |
| | **Google Agenda** | https://www.google.com/calendar/render?hl=fr&pli=1 |
| | **30 Boxes** | http://30boxes.com/welcome.php |
| | **Remenber the milk** | http://www.rememberthemilk.com/ |
| **Retouche d'images** | **Pixlr** | http://pixlr.com/ |
| | **Pixenate** | http://pixenate.com/ |
| | **Pixer** | http://pixer.us/ |
| | **Onlinephototool** | http://www.onlinephototool.com/ |
| **Mixage vidéo** | **Eyespot** | http://www.eyespot.com/ |
| | **You tube editor** | http://www.youtube.com/editor |
| | **Flixtime** | http://flixtime.com/ |
| **Messagerie instantanée** | **Meebo** | http://www.meebo.com/ |
| | **Koolim** | http://www.koolim.com |
| **Mixage Audio** | **Audiotool** | http://www.audiotool.com/ |
| **généalogie** | **Geni** | http://www.geni.com/ |
| **calculatrice** | **Calcoolate** | http://www.calcoolate.com/ |
| **récupération de données** | **e-ROL** | http://www.e-rol.com/ |
| **création de diagrams** | **Gliffy** | http://www.gliffy.com/index-g.php |
| **Creation de diaporama** | **Animoto** | http://animoto.com/ |

**Tableau 1.2: Quelques applications en ligne**

## VI. Modèles de déploiement de Cloud

Il existe différents types de nuages :

1) **Cloud public « Public Cloud » :**

Un ***Cloud*** public (figure 1.6) peut être consulté par tout abonné disposant d'une connexion Internet et un accès à l'espace nuage et est géré par une organisation. L'organisation peut être une entreprise (comme Google), un département académique ou gouvernemental. Le fournisseur de ***Cloud*** possède et gère l'infrastructure ***Cloud***. Un nuage public ne signifie pas que les données d'un utilisateur est publiquement visible car les fournisseurs de ***Cloud*** public offrent généralement des mécanismes de contrôle d'accès pour les utilisateurs, **[64].**

En général, le ***Cloud*** est exploité et géré dans un centre de données appartenant à un fournisseur de services qui héberge plusieurs clients et utilise le provisionnement dynamique. La mise en œuvre d'une plate-forme évolutive de services et la licence « ***pay-as-you-go*** » sont également des éléments attrayants de ***Cloud*** public. La Mise en œuvre du ***Cloud*** public peut être d'une grande aide à éliminer la charge paralysante de la maintenance des infrastructures sur les organisations informatiques. , [70]:

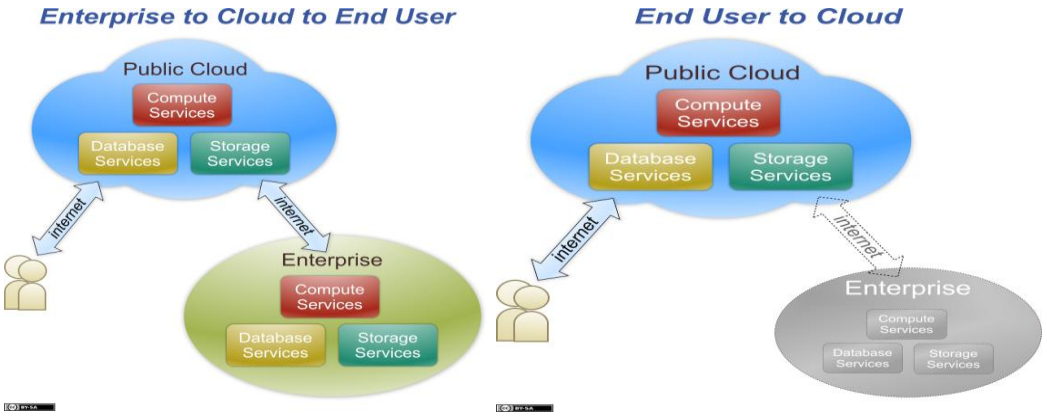



**Figure 1.6 : Cloud Public**

2) **Cloud privé « Private Cloud »**

Un ***Cloud*** privé (figure 1.7) est établi pour un groupe ou une organisation spécifique et limite l'accès à tout ce groupe. Beaucoup de grandes organisations préfèrent, ou sont légalement obligés, de garder leurs serveurs, des logiciels et des données au sein de leurs

propres centres de données et les ***Clouds*** privés leur permettre d'atteindre certains de l'efficacité de ***Cloud*** tout en assumant la responsabilité de la sécurité de leurs propres données. Contrairement au modèle « pay-as-you-go » de ***Cloud*** public, cependant, les ***Clouds*** privés exigent d'importants coûts de développement initiaux, les coûts des centres de données, l'entretien courant, du matériel, des logiciels et de l'expertise interne. Les ressources sont la propriété de l'entreprise, qui les gère et les partage. L'entreprise bénéficie d'économies d'échelle et de coûts avantageux (même s'ils ne sont pas comparables à ceux d'un ***Cloud*** public). Elle peut également profiter d'une transparence et d'un contrôle plus développés.

Le Déploiement de ***Cloud*** privé semble être dirigé principalement par les grandes organisations et agences gouvernementales, plutôt que les petites entreprises et les utilisateurs finaux. La différence de coûts de démarrage est une des principales raisons pour cela, **[70].**

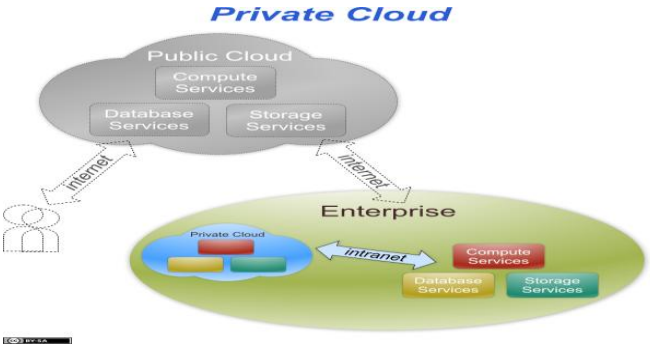



**Figure 1.7 : *Cloud* Privé**

### 3) Cloud Communautaire « Community Cloud»

C'est un modèle de déploiement qui est en cours de mise en œuvre rapide. Sur le plan conceptuel, il réside quelque part entre un ***Cloud*** privé et un ***Cloud*** public. Un ***Cloud*** communautaire est partagé entre deux ou plusieurs organisations qui ont des exigences similaires de nuages, telles que les objectifs d'affaires non concurrentielles, ou un besoin de mutualiser les moyens de sécurité de haut niveau. Le ***Cloud*** est détenue et gérée par un ou plusieurs des collaborateurs dans la communauté. Un exemple de ceci est **OpenCirrus** formé par HP, Intel, Yahoo, et autres, [**61, 64**].

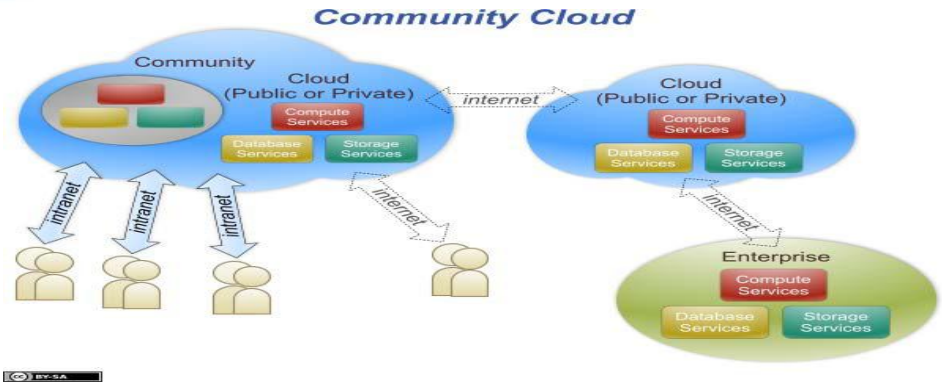



**Figure 1.8 : Cloud Communautaire**

**4) Cloud hybride « Hybrid Cloud »**

Un ***Cloud*** hybride est essentiellement une combinaison d'au moins deux nuages, où les nuages sont un mélange de ***Clouds*** public, privé ou communautaire qui demeurent des entités uniques, mais sont liés entre eux par des technologies normalisées ou propriétaires qui permet la portabilité des applications.

Un exemple de déploiement de ***Cloud*** hybride peut consister en une organisation qui déploie des applications de logiciels non critiques dans le ***Cloud*** public, tout en gardant les applications critiques ou sensibles dans un ***Cloud*** privé. Les ***Clouds*** hybrides combinent les deux modèles de ***Cloud*** public et privé, et ils peuvent être particulièrement efficaces lorsque les deux types de nuages sont situés dans le même établissement, **[70].**

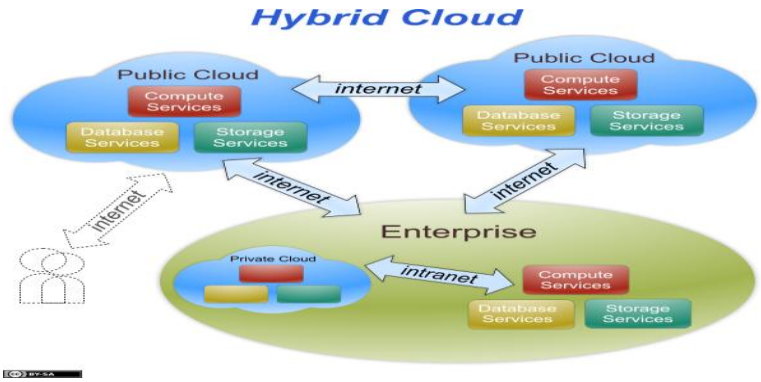



**Figure 1.9 : *Cloud* Hybride**

## VII. Les avantages de Cloud Computing:

Parmi de nombreux avantages de ***Cloud*** on cite les suivants :

### VII.1. Avantages financiers :

Les avantages financiers du ***Cloud*** sont, généralement pour les ***Clouds*** publics, où les ressources informatiques sont acquises comme un service d'utilité à la demande auprès de fournisseurs externes.

1. **Pay-as-You-Go :** Les coûts de l'utilisation de ***Cloud*** varient entre les trois principaux modèles de services: ***SaaS***, ***IaaS*** et ***PaaS***, mais le principe est le même. Les coûts ***SaaS*** dépendent de nombre d'utilisateur, les coûts ***PaaS*** augmentent en proportion avec l'utilisation et la taille des applications développées, et les coûts ***IaaS*** couvrent l'utilisation des serveurs et du stockage ;
2. **Les charges opérationnelles :** Traditionnellement, Construire une infrastructure informatique implique des coûts initiaux importants sur le matériel et le logiciel, entraînée par la planification à long terme, basée sur des prévisions de croissance de l'entreprise et les tendances du marché, tandis qu'un système de ***Cloud*** évolue et, si nécessaire, se rétrécit. Avec le ***Cloud***, les matériels et les licences logicielles ne sont pas laissés inutilisés lorsque l'entreprise réduit sa taille ;
3. **Réduire les coûts de gestion :** La gestion des infrastructures informatiques ou le déploiement des applications logicielles d'entreprise pour les postes de travail des employés posent une surcharge d'administration, qui est un facteur important dans le coût total de possession. Le ***Cloud*** permet de réduire cette surcharge en déchargeant les problèmes d'installation, de gestion et de maintenance du matériel (à travers ***IaaS***), des systèmes d'exploitation de serveur (à travers ***PaaS***), et de déploiement de l'application (à travers ***SaaS***).

### VII.2. Les avantages technologiques

Les ***Clouds*** publics permettent un accès à la demande à un ensemble de ressources informatiques rapidement évolutives de n'importe où.

1. **Évolutivité rapide à la demande :** Deux des cinq caractéristiques essentielles du ***Cloud*** sont : le libre-service à la demande et l'élasticité rapide, qui accélèrent tout à voir avec l'approvisionnement. Le ***Cloud*** peut fournir rapidement de nouveaux

employés (temporaire ou permanente) avec des comptes utilisateur pour les applications ***SaaS***, et ils peuvent utiliser n'importe quel ordinateur personnel pour y accéder. Le ***PaaS*** permet de développer de nouvelles applications logicielles commerciales basées sur le Web sans se soucier des serveurs, les pare-feux, la sécurité ou les systèmes d'exploitation. L'***IaaS*** est utilisé pour obtenir un accès temporaire à une puissance de calcul apparemment illimitée et de stockage de données au besoin ;

2. **Accès n'importe où :** Les services de ***Cloud*** sont basé sur le Web et ne dépend pas de l'ordinateur utilisé. Les documents et applications hébergés dans le ***Cloud*** sont accédés de n'importe où ;
3. **Aucun engagement à long terme :** Les ressources informatiques sont immédiatement disponibles et ils peuvent être utilisés aussi longtemps que nécessaire et se retira ensuite parce qu'ils sont acquis sur un mois en mois, voire de minute en minute ;
4. **Ressources informatiques pratiquement illimitées** : Ressources telles que la puissance de calcul et espace de stockage de données sont disponibles à la demande en fonction des besoins, ce qui permet un haut degré de flexibilité et d'évolutivité à répondre aux besoins changeants de l'entreprise. On oublie trop souvent la notion de saturation des machines et des processeurs ;
5. **L'épreuve du futur :** Avec ***SaaS*** et ***PaaS***, l'obtention de la dernière version du logiciel « la mise à jour » est automatique. Il n'y a aucun frais pour la mise à niveau vers la prochaine version des applications préférées ou de la plate-forme de développement et il est dans l'intérêt de fournisseur d'assurer que leurs systèmes améliorent et restent compétitifs. En outre, puisque les technologies (***SaaS*** et ***PaaS***) sont basées sur le Web, ils utilisent des protocoles de transfert d'information standard, qui facilitent les connexions avec d'autres logiciels sur le Web. Ces avantages ne s'appliquent pas à l'***IaaS*** où toutes les applications logicielles d'entreprise sont gérées par les développeurs, mais les panneaux de contrôle du système et l'infrastructure sous-jacente seront tenus à jour.

**VII.3. Avantages environnementaux :**

1. **Partage des ressources :** Quelques arguments, pour que le ***Cloud*** soit une solution informatique économe en énergie, sont les suivants:
   - Les clients partagent un ensemble de ressources informatiques;
   - Les fournisseurs utilisent des centres de données plus grande, plus moderne et plus éconergétique (économe en énergie)
   - L'utilisation accrue des serveurs en raison de la **virtualisation** des serveurs.
2. **Réduction des déplacements :** Naturellement, le ***Cloud Computing*** signifie que les utilisateurs de ***Cloud*** n'ont plus à se rendre à un bureau pour faire le travail, ni les administrateurs système doivent rendre aux centres de données pour installer de nouveaux serveurs.
3. **Collaboration de groupe plus facile** : pour de nombreux utilisateurs, c'est un avantage important du ***Cloud*** lorsque plusieurs utilisateurs peuvent facilement collaborer sur des documents et projets.

## VIII. Les inconvénients de Cloud Computing :

La popularité du ***Cloud Computing*** indique que de nombreuses entreprises ont déjà changé et beaucoup sont prêtes à déménager leurs ressources informatiques vers le ***Cloud***. La réduction des coûts est la principale raison de l'adoption élevée du ***Cloud***. Aussi, la deuxième raison est que les entreprises obtiennent les ressources immédiatement pour leur service informatique dans les ***Clouds***. Lorsque les entreprises externalisent leurs données, ils perdent complètement le contrôle des données et simplement comptent sur le fournisseur de ***Cloud*** pour sécuriser leurs données. Les fournisseurs de ***Cloud*** doivent avoir un fort moyen pour protéger leurs données contre tout accès non autorisé. Le ***Cloud*** est un concept très nouveau et il a encore un long chemin à parcourir pour se mettre en place en termes de confiance et de fiabilité. En outre, il ya d'autres facteurs qui peuvent aussi limiter l'utilisation des services de ***Cloud***:

1) **Vitesse** : La vitesse d'accès aux données pourrait diminuer parce qu'à chaque fois que l'utilisateur envoie une demande d'accéder aux données, toutes les données sont extraites du ***Cloud*** et renvoyé à nouveau. Il est très différent du système informatique traditionnel, où les données sont accessibles localement et

immédiatement. En outre, les données sont cryptées donc pour extraire le résultat souhaité, la procédure de décryptage de l'ensemble des données pourrait aussi affecter la vitesse globale ;

2) **Connectivité** : Le système ***Cloud*** dépend totalement de la connexion internet. Si la connexion internet échoue, l'accès aux services de nuage échoue aussi. Une connexion internet « Mauvaise » conduite à des performances pauvres de service nuage. En outre, il est limité aux régions où l'accès internet est disponible ;
3) **Fonctionnalités peuvent être limitées:** Cette situation est destinée à changer, mais aujourd'hui de nombreuses applications basées sur le Web ne sont pas complètes comme leurs applications de bureau. Par exemple, vous pouvez faire beaucoup plus avec Microsoft PowerPoint qu'avec Google Présentation sur le Web ;
4) **Accès Physique** : L'accès physique d'une seule personne mal intentionnée qui possède une excellente connaissance de l'implémentation physique du ***Cloud*** et de ses points névralgiques peut suffire à mettre le ***Cloud*** hors service, provoquant une rupture dans la continuité du service et empêchant tout accès externe au ***Cloud***. Les conséquences d'une telle intrusion peuvent être désastreuses :
   - Isolement complet ou partiel du service dans le cas de coupure des liaisons d'accès ;
   - Perte des données en production mais aussi des données sauvegardées, sans aucune possibilité de récupération si celles-ci sont détruites ou détériorées.
5) Une catastrophe d'origine humaine ou naturelle peut avoir des impacts radicaux sur le fonctionnement du ***Cloud Computing*** et amplifier une panne totale ou partielle du service. La perte totale de l'infrastructure du ***Cloud Computing*** pourrait entrainer une interruption de service d'une durée indéterminée et une perte de données irrémédiable sans possibilité de remise en service de l'infrastructure.

## IX. Les défis de Cloud Computing

Si le ***Cloud Computing*** apporte de nombreux bénéfices «rêvés» par les entreprises, quelques freins peuvent ralentir son adoption en masse. La sécurité, est le point faible le plus souvent cité par les entreprises. Dans le cas du ***Cloud*** Public, la problématique se justifie : héberger les applications et les données dans un centre de données fragilise les

politiques de sécurité de l'entreprise. Dans un ***Cloud*** Public, les données sont mutualisées dans le centre de données du fournisseur. Une crainte qui n'a pas lieu d'être dans le cadre du ***Cloud* Privé**, qu'il soit interne ou externe : l'infrastructure étant dédiée et les accès sécurisés, l'isolation est garantie et les politiques de sécurité de l'entreprise sont respectées. De plus, l'entreprise sait où sont ses données.

D'autres problèmes spécifiques au ***Cloud*** restent posés, notamment :

1) **Disponibilité :** L'architecture de ***Cloud*** doit garantir un accès au service en très haute disponibilité avec des performances optimales. Une seule défaillance d'un équipement matériel peut engendrer une dégradation ou une coupure du service voire une perte de données ;
2) **Auto-guérison** : en cas d'échec d'application ou de stockage de données, il y aura toujours une sauvegarde en cours d'exécution sans retards importants, rendant le changement des ressources transparent pour l'utilisateur ;
3) **Gestion des données** : la distribution, le partitionnement, la sécurité et la synchronisation des données ;
4) **Sécurité** : Les utilisateurs utilisent les services de ***Cloud*** et stockent leurs données dans l'infrastructure du fournisseur, le problème de sécurité le plus important est la confidentialité des données de l'utilisateur et de la protection des renseignements personnels. Les utilisateurs veulent savoir où sont stockées les informations, et qui est dans le contrôle de ces renseignements en plus de propriétaires. Ils veulent aussi être garantis que les informations critiques ne sont pas consultées et utilisées illégalement, même par les fournisseurs de ***Cloud*** ;
5) **Emplacement des ressources** : les utilisateurs finaux utilisent les services offerts par les fournisseurs de ***Cloud*** sans savoir exactement où les ressources de ces services se trouvent, éventuellement dans d'autres domaines législatifs. Cela pose un problème potentiel en cas de différends, qui est parfois indépendant de la volonté des fournisseurs de ***Cloud***. Les données enregistrées auprès des fournisseurs de services ***Cloud*** ne sont pas seulement affectées par les politiques du fournisseur, mais également par la législation du pays où résident les fournisseurs, **[12]**. Lorsque vous utilisez ces services, les utilisateurs devront accepter les « conditions

d'utilisation » qui octroient les fournisseurs le droit de divulguer des informations de l'utilisateur en conformité avec les lois et les demandes de mise en application de loi

6) ***Multi-tenancy*** : ce problème constitue un défi pour protéger les données utilisateur contre tout accès non autorisé d'autres utilisateurs exécutant des processus sur les mêmes serveurs physiques. En fait, Ce n'est pas un nouveau problème en tenant compte de la préoccupation actuelle des services d'hébergement Web. Cependant, avec la généralisation du ***Cloud Computing*** et avec le fait que les utilisateurs stockent des données plus importantes dans le nuage, Ce problème doit être réexaminé sérieusement, **[12]** ;
7) **Authentification et confiance des informations acquises**: comme les données critiques sont situées dans l'infrastructure de fournisseur de ***Cloud***, les données peuvent être modifiées sans le consentement du propriétaire. Les données modifiées peuvent ensuite être récupérées et traitées par le propriétaire pour prendre des décisions cruciales. L'authenticité des données dans ce cas est très importante et doit donc être garantie, **[12]** ;
8) **Surveillance du système et les journaux** (logs): comme plusieurs applications stratégiques sont migrées vers le ***Cloud***, les clients peuvent demander que les fournisseurs de ***Cloud*** assurent plus de surveillance. Les résultats de la surveillance et les journaux peuvent contenir des informations sensibles et sont traditionnellement utilisés en interne par les fournisseurs mais ne pas tous les fournisseurs de ***Cloud*** sont disposés à le faire. Il faudra beaucoup de négociation entre les fournisseurs de ***Cloud*** et les clients à venir avec une surveillance appropriée et les informations du journal dans le cadre d'un contrat de service, **[12] ;**
9) **Les normes de *Cloud***: les normes sont nécessaires entre les différentes organisations de développement standards pour réaliser l'interopérabilité entre les nuages et d'accroître leur stabilité et leur sécurité. Par exemple, les services de stockage actuel fournit par un fournisseur ***Cloud*** peuvent être incompatibles avec ceux d'autres fournisseurs. Afin de conserver leurs clients, les fournisseurs de ***Cloud*** introduit, ce qu'on appelle, « **sticky services**» qui créent des difficultés pour les utilisateurs s'ils veulent migrer d'un fournisseur à l'autre, **[12]**. Pour accroître

l'interopérabilité des nuages et la libre circulation des données entre les nuages, des normes « ***interCloud*** » sont nécessaires dans les domaines suivants :

- L'architecture du réseau ;
- Format des données ;
- Qualité de service ;
- La fourniture de ressources ;
- La Sécurité, gestion d'identité et la confidentialité.

## X. Conclusion

Le premier chapitre vis à donner une description globale de la technologie de **Cloud Computing**. Ainsi son développement remarquable ces dernières années qui suscite de plus en plus l'intérêt des différents utilisateurs de l'internet et de l'informatique. C'est un nouveau modèle économique qui, en effet, promet un changement dans le mode d'investissement et d'exploitation des ressources. Avec le ***Cloud,*** les organisations, les institutions et les entreprises n'ont plus besoin d'investir lourdement dans des ressources informatiques, nécessairement limitées, et nécessitant une gestion interne lourde et couteuse. Aujourd'hui elles ont le choix de migrer vers un modèle ***Cloud Computing*** où elles peuvent acheter ou louer des ressources en ligne. Ce modèle leur épargne les coûts de gestion interne, puisque les ressources informatiques sont administrées au niveau du fournisseur du ***Cloud Computing***.

Avec tous les avantages du paradigme de ***Cloud*** et son potentiel pour diminuer les coûts et réduire le temps nécessaire pour lancer de nouvelles initiatives, les ressources virtualisées, les serveurs géographiquement dispersés et la co-implantation de traitement et de stockage posent des défis et opportunités, en même temps, pour les fournisseurs et utilisateurs de ***Cloud***. Mais la sécurité du ***Cloud*** sera toujours une préoccupation majeure en raison de la nature ouverte et publique de ***Cloud***. En effet, l'intégrité, la confidentialité, la disponibilité sont encore des problèmes ouverts qui appellent des solutions efficaces et efficientes.

# Chapitre II

# Sécurité Cloud

*« Tout problème authentique est justiciable d'une technique, et toute technique consiste à résoudre des problèmes d'un type déterminé »*

*Gabriel Marcel*

## I. Introduction

La ***sécurité*** est un enjeu majeur des technologies numériques modernes. L'infrastructure de télécommunication, les réseaux sans fils (bluetooth, WiFi, WiMax), l'Internet, les systèmes d'informations, les routeurs, les systèmes d'exploitation et les applications informatiques, toutes ces entités présentent des vulnérabilités : faille de sécurité, défaut de conception ou de configuration. Ces systèmes tombent en panne, subissent des erreurs d'utilisation et sont attaqués de l'extérieur ou de l'intérieur. Une approche globale de la sécurité des systèmes est essentielle pour protéger la vie privée et réduire les vulnérabilités des grands systèmes informatiques. Cette sécurité se conçoit pour l'ensemble des processus relatifs à ces données, qu'il s'agisse de leur création, leur utilisation, leur sauvegarde, leur archivage ou leur destruction et concerne leur confidentialité, intégrité, authenticité et disponibilité.

Aujourd'hui, de nombreuses organisations transfèrent leurs services informatiques vers le ***Cloud Computing***, ce qui rend leur traitement informatique disponible beaucoup plus pour les utilisateurs. En outre, le ***Cloud Computing*** est une technologie attrayante et économique car il offre des options d'accessibilité et de fiabilité pour les utilisateurs et les ventes évolutives pour les fournisseurs. En dépit d'être attrayant, le ***Cloud*** apporte aussi de nouvelles menaces et défis concernant la sécurité et la fiabilité. Par conséquence, la sécurité et la protection des données sensibles ont été identifiés comme les plus grands obstacles de l'utilisation de ***Cloud Computing***.

Le ***Cloud*** peut assurer la sécurité des données de l'utilisateur à l'aide des politiques de sécurité au sein de sa propre périphérie ou périmètre. Étant donné que le concept de ***Cloud*** nécessite le « ***Polling*** » des ressources avec d'autres propriétaires ***Clouds*** [71], par conséquent, les données importantes du client sont non seulement disponibles aux ***Clouds*** mais aussi aux tiers de ***Clouds***. La sécurité est donc un élément majeur dans l'infrastructure informatique de tout ***Cloud***, parce qu'il est essentiel de s'assurer que seulement les accès autorisé sont permis et le comportement sécuritaire est prévu.

## II. Définition de la sécurité informatique :

La sécurité informatique c'est l'ensemble des moyens mis en œuvre pour réduire les vulnérabilités d'un système contre les menaces accidentelles ou intentionnelles, [**38**]. Ces systèmes informatiques et leurs réseaux d'interconnexion sont également la proie de vandales, des égoïstes malveillants, des terroristes et un éventail d'individus. Outre ces attaques intentionnelles sur les systèmes informatiques, il existe d'innombrables façons dont les erreurs involontaires peuvent endommager ou détruire la capacité d'un système à s'acquitter de ses fonctions prévues.

La sécurité d'un réseau peut être définit comme un niveau de garantie que l'ensemble des machines du réseau fonctionnent de façon optimale et que les utilisateurs des machines possèdent uniquement les droits qui leur ont été octroyés.

## III. Service Level Agreement

Le ***Cloud Computing*** suit le modèle « **pay-as-you-go** » qui fournit des possibilités pour faire avancer les objectifs organisationnels de l'augmentation de l'efficacité de la prestation des services, l'allocation dynamique des ressources, et la diminution des coûts informatiques. Pour les organisations qui exploitent des systèmes critiques, tels que les systèmes militaires, le déplacement vers le ***Cloud*** exige des considérations de sécurité spéciales. Ces organisations doivent se conformer à un ensemble de contrôles de sécurité « *Security Controls* » qui mettent en œuvre des politiques de sécurité, régissent le fonctionnement du système, et protègent les données et les fonctionnalités sensibles. Il faut faire attention, lors du déplacement des ressources vers le ***Cloud***, afin d'assurer que tous les fournisseurs de services de ***Cloud Computing*** (***CSP*** pour *Cloud Service Provider* en anglais) ont pris des mesures de sécurité adéquates qui sont conformes aux contrôles de sécurité définies [**45**]. L'objectif de gestion des SLA est d'établir un cadre de gestion des SLA évolutif et automatique qui peut s'adapter aux changements environnementaux dynamiques en considération de multiples paramètres de qualité de service, **[85].**

Les organisations doivent s'appuyer sur les **SLA**s pour évaluer les offres de sécurité du fournisseur de ***Cloud***, car les implémentations de service sont des boites noires « *blackboxed* en anglais » et sont développés en externe par le fournisseur de service. Les

**SLAs** contiennent un ensemble de conditions descriptives de service (**SDT** pour *Service Description Terms* en anglais) décrivant les fonctionnalités qu'offrent les services et peuvent inclure des fonctionnalités de sécurité [**45**].

Le **SLA** est un concept qui a été introduit dès les années 80 pour gérer la qualité de service (**QoS** pour *Quality of Service* en anglais) dans le domaine des télécommunications. Le **SLA**, que l'on pourrait traduire en français par accord de niveau de service ou contrat de niveau de service, est donc un contrat dans lequel on formalise la **QoS** prescrite entre un prestataire et un client [**92**].

Un **SLA**, comme tout contrat, contient un ensemble d'éléments (obligatoires et optionnels) à savoir la période de validité, les parties impliquées, les services, les garanties en terme d'objectifs, les pénalités et la suspension ou la résiliation du contrat et les sanctions [**92**]. Il s'agit clairement d'un contrat juridique extrêmement important entre un prestataire de services de ***Cloud*** et le consommateur qui doit avoir les qualités suivantes, [**40**] :

1. Identifier et définir les besoins du client ;
2. Fournir un Framework pour le client afin de mieux comprendre les installations disponibles ;
3. Simplifier les questions complexes ;
4. Réduire les zones de conflit ;
5. Faciliter les solutions en cas de litige ;
6. Éliminer des attentes irréalistes.

## IV. Notions de sécurité :

Pour mieux comprendre la sécurité informatique, différents termes doivent être décrits:

1) **Menace (*Threat*) :** Une menace est un danger qui existe dans l'environnement d'un système indépendamment de celui-ci : accident, erreur, malveillance. Une malveillance est l'action d'individus et/ou d'organisations qui exploitent des vulnérabilités dans les systèmes. Une malveillance peut être :

- passive : elle ne modifie pas l'information et porte essentiellement sur la confidentialité;
- active : elle modifie le contenu de l'information ou le comportement des systèmes de traitement.

2) **Vulnérabilité (*Vulnerability*) :** Une vulnérabilité est une faiblesse du système qui le rend sensible à une menace:
   - bogues dans les logiciels,
   - mauvaises configurations,
   - erreurs humaines,
   - services permis et non utilisés,
   - virus ou chevaux de Troie,
   - saturation de la liaison d'accès à l'Internet...

3) **Risque (*Risk*) :** Le risque est la probabilité qu'une menace particulière puisse exploiter une vulnérabilité donnée du système. Traiter le risque, c'est prendre en compte les menaces et les vulnérabilités.

Trois principes complémentaires qui prennent en charge l'assurance de l'information sont la confidentialité, l'intégrité et la disponibilité, [44]. Outre de ces principes, l'authentification et la non-répudiation sont des concepts importants dans le domaine de la sécurité. Ces concepts de la sécurité sont résumés dans ce qui suit :

1) **Confidentialité :** L'accès par les utilisateurs aux informations et documents doit être limité à ceux qui leur sont propres, et ceux qui sont publics ou partagés. La nécessité de garder le secret des informations provient de l'utilisation des ordinateurs dans des domaines sensibles tels que le gouvernement et l'industrie. En particulier, il est interdit de prendre connaissance d'informations détenues par d'autres utilisateurs, quand bien même ceux-ci ne les auraient pas explicitement protégées.

   Quand il s'agit de la confidentialité des données stockées dans un ***Cloud*** public, il existe deux problèmes potentiels. Tout d'abord, *quel contrôle d'accès existe pour protéger les données*? Le contrôle d'accès consiste à la fois l'authentification et l'autorisation. Les fournisseurs ***Cloud*** utilisent généralement des mécanismes d'authentification faibles (par exemple, le nom d'utilisateur mot de passe). Ensuite,

est-ce que les données d'un client sont chiffrées lorsqu'elles sont stockées dans le ***Cloud***? Et si oui, avec quel algorithme de chiffrement, et avec quelle clé? Cela dépend, et plus précisément, du fournisseur utilisé. Par exemple, ***MozyEnterprise EMC*** crypte les données des clients. Cependant, ***AWS S3*** ne crypte pas. Les clients sont en mesure de chiffrer leurs données eux-mêmes avant de les télécharger (Upload), [**79**].

Si un fournisseur crypte les données du client, le prochain examen concerne quel algorithme de cryptage il utilise. D'un point de vue cryptographique, Seuls les algorithmes qui ont été publiquement contrôlés par un organisme de normalisation formelle (par exemple **NIST**), ou au moins de façon informelle par la communauté cryptographique doivent être utilisés. N'importe quel algorithme qui est exclusif doit absolument être évité, [**79**].

2) **Intégrité :** L'intégrité se réfère à la fiabilité des données ou des ressources, et il est habituellement exprimé en termes de prévention des changements inadéquats ou non autorisés. L'intégrité comprend l'intégrité des données (le contenu de l'information) et l'intégrité de l'origine (la source des données, souvent appelée authentification), [**86**]. L'utilisateur s'engage à ne pas apporter volontairement de perturbations au bon fonctionnement des systèmes informatiques et des réseaux (internes ou externes), que ce soit par des manipulations anormales du matériel, ou par l'introduction de logiciels parasites connus sous le nom générique de virus, chevaux de Troie, bombes logiques. Bien que l'utilisation de HTTPS (au lieu de HTTP) ait atténué le risque d'intégrité, les utilisateurs qui n'utilisent pas le protocole HTTPS rencontrent à un risque accru que leurs données auraient modifié en transit sans leur connaissance.

3) **Disponibilité** : La disponibilité réfère à la capacité d'utiliser une information ou une ressource désirée alors elle s'agit de garantir qu'une ressource sera accessible au moment précis où quelqu'un souhaitera s'en servir. L'aspect de la disponibilité qui est pertinente pour la sécurité, c'est que quelqu'un peut délibérément prendre des dispositions pour refuser l'accès à des données ou à un service en le rendant indisponible.

L'objectif de disponibilité pour les systèmes de ***Cloud*** (y compris ces applications et infrastructures) est de s'assurer que ses utilisateurs puissent les utiliser à tout

moment, à n'importe quel endroit. Le système de ***Cloud*** permet à ses utilisateurs d'accéder au système (par exemple, applications, services) à partir de n'importe où. De nombreux fournisseurs de systèmes ***Cloud*** fournissent des infrastructures ***Cloud*** et des plates-formes basées sur des machines virtuelles.

La redondance est la principale stratégie utilisée pour améliorer la disponibilité du système de ***Cloud***. Les grands fournisseurs des systèmes de ***Cloud*** (par exemple, Amazon, Google) offrent une redondance géographique dans leurs systèmes de ***Cloud***, permettant une haute disponibilité à un fournisseur unique. Les zones de disponibilité sont des emplacements distincts qui sont conçus pour être isolés des échecs dans d'autres zones de disponibilité et de fournir une connectivité réseaux peu couteuse et à faible latence aux autres zones de disponibilité dans la même région. L'utilisation des instances dans les zones de disponibilité séparées, peut protéger les applications de la défaillance d'un seul emplacement. Google possède plus d'un million de machines qui sont distribués dans 36 centres de données à travers le monde.

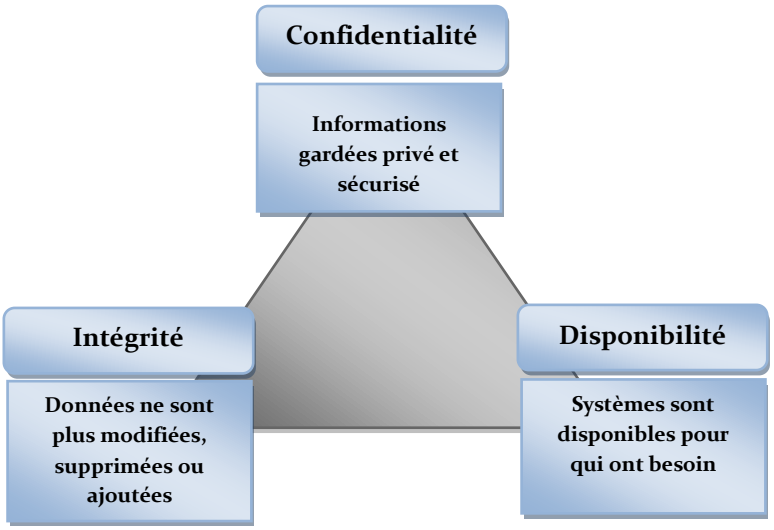



**Figure 2.1 : Représentation des caractéristiques principales de la sécurité : CIA**

4) **Authentification** : L'authentification est la procédure qui consiste à vérifier l'identité d'une entité (personne, ordinateur…), afin d'autoriser l'accès de cette entité à des ressources (systèmes, réseaux, applications…). L'authentification permet donc de valider l'authenticité de l'entité en question. Généralement, l'authentification est

précédée d'une identification qui permet à cette entité de se faire reconnaître du système par un élément dont on l'a doté.

***S'identifier c'est communiquer une identité préalablement enregistrée, s'authentifier c'est apporter la preuve de cette identité.***

5) **Non-répudiation :** la non-répudiation signifie la possibilité de vérifier que l'émetteur et le destinataire sont bien les parties qui disent avoir respectivement envoyé ou reçu le message. Autrement dit, la non-répudiation de l'origine prouve que les données ont été envoyées, et la non-répudiation de l'arrivée prouve qu'elles ont été reçues.

Un client de ***Cloud*** doit délibérer l'état de chaque modèle de sécurité avant de s'engager à un en particulier, afin de procéder à une évaluation stratégique et être au courant des questions d'actualité. Plus précisément, une évaluation doit être réalisée en termes d'exigences de sécurité [76].

| Les exigences de sécurité | Modèles de déploiement | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | *Cloud* Public | | | *Cloud* Privé | | | *Cloud* Hybride | | |
| | IaaS | Paas | SaaS | IaaS | Paas | SaaS | IaaS | Paas | SaaS |
| Identification et authentification | ✓ | * | ✓ | ✓ | * | ✓ | * | * | ✓ |
| Autorisation | ✓ | ✓ | ✓ | * | * | ✓ | * | * | ✓ |
| Confidentialité | * | * | ✓ | * | ✓ | ✓ | * | * | ✓ |
| Disponibilité | ✓ | ✓ | * | ✓ | ✓ | ✓ | * | * | * |
| Intégrité | ✓ | * | ✓ | * | ✓ | ✓ | ✓ | ✓ | ✓ |
| Non-répudiation | * | * | ✓ | * | * | ✓ | * | * | * |

**Tableau 2.1 : les modèles de services par rapport aux modèles de déploiement publics, privés et hybrides qui concernent les exigences de sécurité. (✓) : Signifie une exigence obligatoire, (*) : signifie une exigence optionnelle, [76]**

A partir du tableau 2.1, les conditions d'autorisation des trois modèles de services sont obligatoires dans les ***Cloud***s publics, afin d'empêcher l'accès non autorisé. Le modèle hybride est moins exigeant en termes de sécurité que les deux modèles public et privé. Parmi les trois modèles de déploiement, les propriétés d'intégrité sont bien désirées, soulignant (pointing out) l'intérêt à vérifier l'exactitude des données et si elles ont été altéré ou endommagé. En outre, le SaaS est un modèle de service avec plus d'exigences de sécurité de tous les modèles de déploiement sous-jacents.

## V. Différents types d'attaque dans le Cloud Computing

Le piratage s'est développé de façon exponentielle depuis l'avènement d'Internet. Dans les actes de piratage, on peut distinguer deux catégories, d'abord les attaques qui consistent à gêner un ou des utilisateurs, par exemple faire planter un ordinateur ou surcharger un site de connexion, ensuite les intrusions qui visent à pénétrer sur un ordinateur, un réseau protégé. Cependant, cette distinction n'est pas si claire, en effet de nombreuses attaques visent à recueillir des informations pour permettre une intrusion.

Les informations ou les systèmes d'informations peuvent subir des dommages de plusieurs façons : certains intentionnels (malveillants), et d'autres par accident. Ces événements seront appelés des « ***attaques*** ».

Il existe quatre catégories principales d'attaque, [75] :

1) ***Attaque d'accès*** : Une attaque d'accès est une tentative d'accès à l'information par une personne non autorisée. Ce type d'attaque concerne la confidentialité de l'information, par exemple : Sniffing, chevaux de Troie, porte dérobée, ingénierie sociale et craquage de mot de passe….
2) ***Attaque de modification*** : Une attaque de type « modification » consiste, pour un attaquant, à tenter de modifier des informations. Ce type d'attaque est dirigé contre l'intégrité de l'information, par exemple : virus, vers et chevaux de Troie,
3) ***Attaque de saturation :*** Les attaques par saturation sont des attaques informatiques qui consistent à envoyer des milliers de messages depuis des dizaines d'ordinateurs, dans le but de submerger les serveurs d'une société, de paralyser pendant plusieurs heures son site Web et d'en bloquer ainsi l'accès aux

internautes. Cette technique de piratage assez simple à réaliser est jugée comme de la pure malveillance. Elle ne fait que bloquer l'accès aux sites, sans en altérer le contenu. Il existe différente attaque par saturation : Flooding, le TCP-SYN Flooding, ...

4) ***Attaque de répudiation*** : La répudiation est une attaque contre la responsabilité. Autrement dit, la répudiation consiste à tenter de donner de fausses informations ou de nier qu'un événement ou une transaction se soient réellement passé. Par exemple l'IP Spoofing...

Les attaques peuvent être réalisées grâce à des moyens techniques ou par ingénierie sociale. Il ya plusieurs types d'attaques dangereuses affectant la disponibilité, la confidentialité et l'intégrité des ressources et des services ***Cloud*** qui ne sont pas spécifiques à l'environnement ***Cloud***, mais lancé largement dans les systèmes de ***Cloud Computing*** en raison des caractéristiques des systèmes de nuages. Ces menaces sont énumérées ci-dessous:

**V.1 Sniffing :**

L'écoute du réseau ou le ***Sniffing*** est une technique qui consiste à analyser le trafic réseau. Ce type d'attaque est utilisé par les pirates informatiques pour obtenir des mots de passe. Grâce à un logiciel appelé ***renifleur*** de paquets (***Sniffer***), qui permettre d'intercepter toutes les paquets qui circulent sur un réseau même ceux qui ne sont pas destinés (aux pirates). Par exemple, lors d'une connexion le mot de passe de l'utilisateur va transiter en clair sur le réseau. Il est aussi possible de savoir à tout moment quelles pages web regardent les personnes connectées au réseau, les mails en envoi ou réception. Cette technologie n'est pas forcement illégale car elle permet aussi de détecter des failles sur un système. **[75, 84]**

Si les parties de la communication n'utilisaient pas des techniques de cryptage pour la sécurité des données les attaquants peuvent capturer les données pendant la transmission comme un tiers (third party). C'est pour ça il faut utiliser le cryptage des données comme contre mesure du ***Sniffing***.

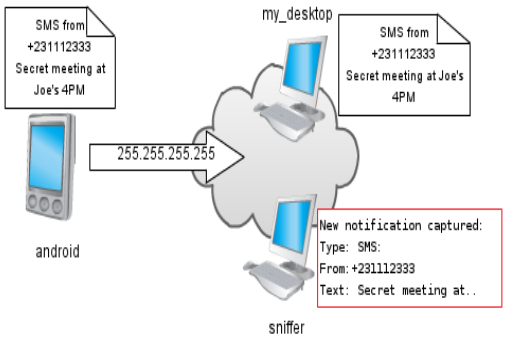



**Figure 2.2 : un Sniffing simple de réseau qui révélé les SMS**

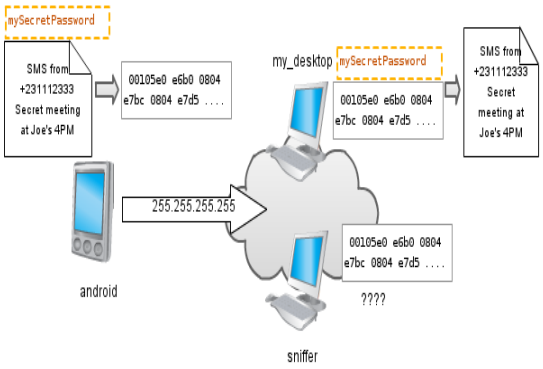



**Figure 2.3 : Cryptage des SMS contre le Sniffing**

### V.2 Balayage de ports

Le Balayage de ports (***Ports Scanning***) fournit la liste des ports ouverts, des ports fermés et les ports filtrés. Grâce au balayage de ports, les attaquants peuvent trouver des ports ouverts et attaquent les services qui s'exécutent sur ces ports. Les détails liés au réseau tels que l'adresse IP, l'adresse MAC, le routeur, la passerelle de filtrage, les règles de pare-feu, etc peuvent être connus par cette attaque. Il existe diverses techniques de balayage de ports qui sont le balayage TCP, UDP, SYN, FIN, ACK, etc.

Dans le scénario ***Cloud***, l'attaquant peut attaquer les services offerts par le balayage de ports (en découvrant les ports ouverts sur lesquels ces services sont fournis). Il peut y avoir certains problèmes en ce qui concerne le balayage des ports qui pourraient être utilisées par un attaquant comme le Port 80 (HHTP) qui est toujours ouvert et est utilisé pour fournir les services web à l'utilisateur. Autres ports, tels que le port 21(FTP), ne sont pas ouverts tout le temps, il s'ouvrira en cas de besoin donc les ports devrait être assuré par le cryptage jusqu'à ce que le logiciel du serveur est correctement configuré. La

Contre-mesure pour cette attaque est l'utilisation d'un pare-feu pour sécuriser les données.

### V.3. *Flooding* (Attaque d'inondation)

Dans cette attaque, l'attaquant tente d'inonder la victime en envoyant un grand nombre de paquets de l'hôte innocent (zombie) dans le réseau. Les paquets peuvent être de type TCP, UDP, ICMP ou un mélange d'entre eux. Ce type d'attaque peut être possible en raison de connexions réseau illégitimes, [2].

En cas de ***Cloud***, les demandes pour les machines virtuelles sont accessibles par tout le monde via Internet, ce qui peut provoquer des attaque ***DoS*** (ou ***DDoS***) par l'intermédiaire de zombies. L'attaque d'inondation affecte la disponibilité du service à l'utilisateur autorisé. En attaquant un seul serveur, fournissant un service donné, l'attaquant peut provoquer une perte de disponibilité du service prévu. Une telle attaque est dite attaque directe de type ***DoS***. Si les ressources matérielles du serveur sont complètement épuisées par le traitement des demandes d'inondation, les autres instances de service sur la même machine ne sont plus en mesure d'accomplir leur tâche prévue. Ce type d'attaque est appelé attaque indirecte de type ***DoS***. Le ***flooding*** peut considérablement rend le ***Cloud*** n'est pas en mesure d'établir la distinction entre l'usage normal et anormal.

Le ***Cloud*** est plus vulnérable aux attaques ***DoS***, parce qu'elle est partagée par de nombreux utilisateurs, ce qui rend les attaques ***DoS*** beaucoup plus dommageable. Lorsque le système d'exploitation de ***Cloud Computing*** remarque une charge de travail forte sur le service inondé, il va commencer à fournir plus de puissance de calcul (plus de machines virtuelles, plus des instances de service) pour faire face à la charge de travail supplémentaire. En ce sens, le système de ***Cloud*** tente de travailler contre l'attaquant (en fournissant plus de puissance de calcul), mais en fait, dans une certaine mesure, l'attaquant lui permet de faire plus de dégâts possibles sur la disponibilité d'un service, à partir d'un seul point d'entrée. Ainsi, l'attaquant n'a pas besoin d'inonder tous les serveurs qui fournissent un certain service cible, mais seulement peut inonder une seule adresse ***Cloud*** afin de réaliser une perte complète de la disponibilité sur le service prévu.

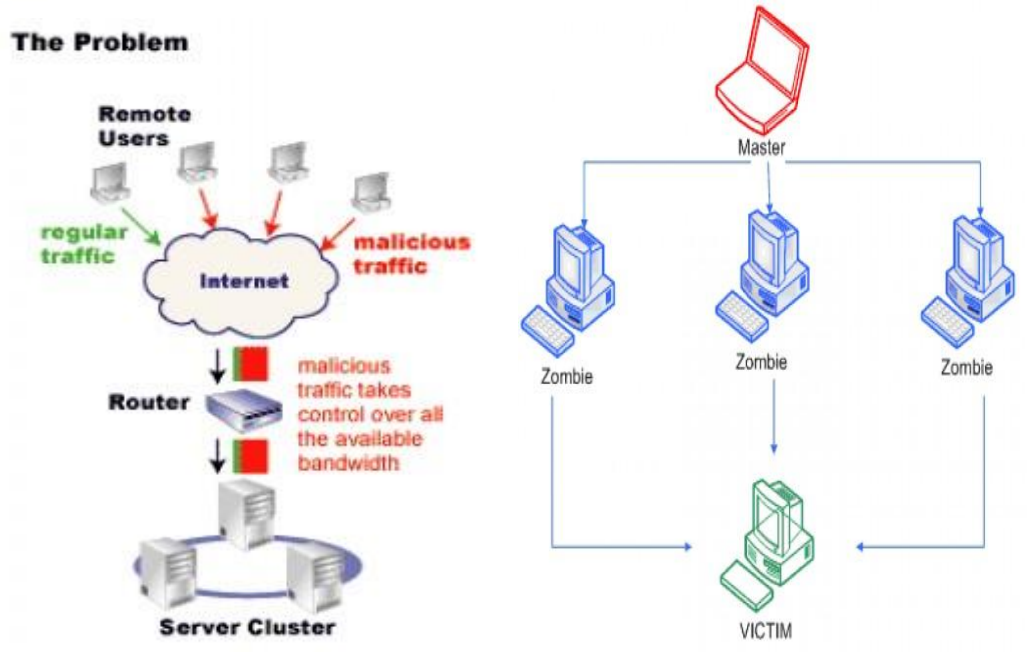



**Figure 2.4 : Attaque de flooding**

### V.4 Attaques par les canaux des portes dérobées

Il s'agit d'une attaque passive qui permet aux pirates d'accéder à distance au nœud infecté afin de compromettre la confidentialité des utilisateurs.

Lorsqu'un un pirate arrive à accéder à un serveur à l'aide d'une des techniques présentées dans cette partie, il souhaiterait y retourner sans avoir à tout recommencer. Pour cela, il laisse donc des portes dérobées (backdoor) qui lui permettra de reprendre facilement le contrôle du système, [75]. Il existe différents types de portes dérobées :

- Création d'un nouveau compte administrateur avec un mot de passe choisi par le pirate ;
- Création de compte ftp ;
- Modification des règles du pare-feu pour qu'il accepte des connections externes.

Dans tous les cas, l'administrateur perte le contrôle total du système. L'attaquant peut alors récupérer les données qu'il souhaite, voler des mots de passe ou même détruire des données. À l'aide des canaux de porte dérobée, l'attaquant peut contrôler les ressources de la victime et peut le rendre aussi zombie pour tenter une attaque ***DDoS***. Il peut également être utilisé à divulguer les données confidentielles de la victime. Pour cette raison, le système compromis face à des difficultés dans l'accomplissement de ses tâches

régulières. Dans l'environnement de ***Cloud Computing***, l'attaquant peut accéder et contrôler les ressources de l'utilisateur de nuage à travers les canaux des portes dérobées et rendre le **VM** un Zombie pour initier une attaque ***DoS /DDoS***.

Dans le ***Cloud***, Avoir des machines virtuelles permettrait indirectement toute personne disposant d'un accès pour les fichiers de disque hôte de la machine virtuelle pour prendre un cliché ou de la copie illégale de l'ensemble du système. Cela peut conduire à l'espionnage et le piratage des produits légitimes.

Le Pare-feu (dans le nuage) pourrait être la solution commune pour prévenir certaines des attaques répertoriées ci-dessus. Pour prévenir les attaques sur VM / hyperviseur, des **IDS** (***Intrusion Detection Systems)*** basés sur les anomalies peuvent être utilisées. Pour les attaques ***flooding*** et les portes dérobées les **IDS** basés signature ou basés anomalie peuvent être utilisés.

### *V.5* L'homme du milieu

Lorsqu'un pirate, prenant le contrôle d'un équipement du réseau, se place au milieu d'une communication il peut écouter ou modifier celle-ci. On parle alors de « l'homme du milieu » *(man in the middle)*, [2].Dans le ***Cloud Computing***, la mauvaise configuration du protocole SSL (Secure Socket Layer) qui est un protocole couramment utilisé pour la gestion de la sécurité d'une transmission de messages sur Internet va créer un problème de sécurité connu sous le nom "***Man in the Middle attaque*** ". S'il ya un problème avec SSL, il donne une chance au pirate de lancer une attaque sur les données des deux parties et dans un environnement comme le ***Cloud Computing*** peut créer des catastrophes.

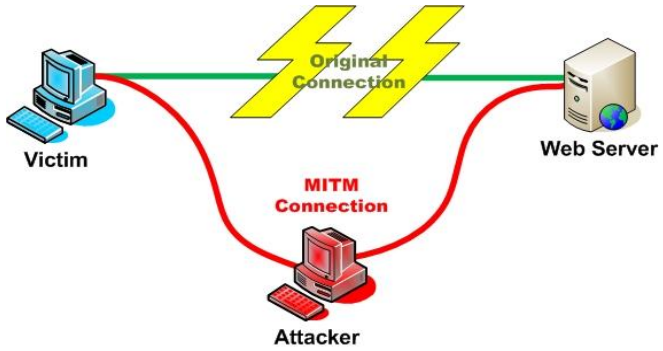



**Figure 2.5 : attaque de l'homme au milieu**

Les points sensibles, [38], permettant cette technique sont :

- **DHCP** : Ce protocole n'est pas sécurisé et un pirate peut fournir à une victime des paramètres réseau qu'il contrôle. Solution : IP fixe.
- **ARP** : Si le pirate est dans le même sous réseau que la victime et le serveur (même si commutateur), il peut envoyer régulièrement des paquets ARP signalant un changement d'adresse MAC aux deux extrémités. Solution : ARP statique.
- **ICMP** : Un routeur peut émettre un ICMP-redirect pour signaler un raccourci, le pirate peut alors demander de passer par lui. Solution : refuser ICMP-redirect ou seulement vers des routeurs identifiés.
- **RIP** : Le pirate envoie une table de routage à un routeur indiquant un chemin à moindre coût et passant par un routeur dont il a le contrôle. Solution : nouvelle version de RIP qui intègre une identification des routeurs de confiance.
- **DNS** : Par « ID spoofing » un pirate peut répondre le premier à la requête de la victime et par « cache poisoning » il corrompt le cache d'un serveur DNS. Solution : proxy dans un réseau différent des clients, désactivation de la récursivité, vidage du cache DNS régulier.
- **Proxy HTTP** : Par définition un proxy est en situation d'homme du milieu. Une confiance dans son administrateur est nécessaire de même qu'un contrôle du proxy lors de son départ !
- **Virus** : un virus, éventuellement spécifique à la victime et donc indétectable, peut écrire dans le fichier « hosts »... Solution : bloquer les .vbs et .exe

### V.6 IP Spoofing

C'est une des techniques de piratage très connus dans lequel l'intrus envoie des messages à un ordinateur, ce qui indique que le message vient d'un système de confiance. Dans le processus de l'usurpation d'adresse IP, le pirate a d'abord détermine l'adresse IP d'un système sécurisé et modifie les en-têtes de paquets pour apparaître comme si elles sont originaires d'un système de confiance. Il existe des variantes car le ***Spoofing*** peut se faire aussi sur des adresses e-mail, des serveurs DNS ou NFS, [70] ;

**V.7 Les attaques sur la machine virtuelle (VM) ou hyperviseur**

La Virtualisation est une technologie logicielle largement utilisée dans le ***Cloud Computing***, qui emploie le matériel et/ou le logiciel de simulation afin d'exécuter plusieurs systèmes d'exploitation et les applications sur une architecture matérielle partagée. L'environnement produit par cette simulation est connu sous le nom « machine virtuelle » (VM).

La Virtualisation/multi-tenancy est la base de l'architecture de ***Cloud Computing***. Il existe principalement trois types de **Virtualisation** utilisés: ***virtualisation de niveau OS*** (système d'exploitation), ***Virtualisation basée applications*** et ***Virtualisation basée Hyperviseur***. Dans le niveau OS, plusieurs systèmes d'exploitation invités sont en cours d'exécution sur un système d'exploitation hébergement qui dispose de visibilité et de contrôle sur chaque système d'exploitation invité. Dans ces types de configuration, un attaquant peut obtenir contrôle sur l'ensemble des systèmes d'exploitation invités en compromettant le système d'exploitation hôte. Au niveau d'application, la Virtualisation est activée sur la couche supérieure de l'OS hôte. Dans ce type de configuration, chaque machine virtuelle dispose de son système d'exploitation invité et les applications liées. La Virtualisation basée application souffre également de la même vulnérabilité comme celle basée sur OS. Hyperviseur ou VMM (Virtual Machine Monitor) est comme un code incorporé au système d'exploitation hôte. Ce code peut contenir des erreurs natives et est disponible au moment du démarrage du système d'exploitation hôte pour contrôler plusieurs systèmes d'exploitation invités.

En compromettant la couche inférieure hyperviseur, l'attaquant peut prendre le contrôle de machines virtuelles installées. Par exemple, BluePill , SubVir et DKSM sont quelques attaques bien connues sur la couche virtuelle. Grâce à ces attaques, les pirates peuvent être en mesure de compromettre l'hyperviseur installé, pour gagner le contrôle de l'hôte. De nouvelles vulnérabilités, telles que la vulnérabilité zero-day, sont trouvées dans les machines virtuelles (VM) qui attirent un attaquant d'obtenir un accès à l'hyperviseur ou d'autres machines virtuelles installées. Les exploits de Zero-day sont utilisés par des attaquants avant que le développeur du logiciel cible connaît la vulnérabilité. Une vulnérabilité zero-day a été exploitée dans l'application de

virtualisation Hyper VM qui a abouti à la destruction de nombreux sites web basé sur le serveur virtuel.

### V.8 Les attaques par injection de logiciel malveillant (Malware Injection Attack) :

Une première tentative d'attaque vise à injecter une implémentation du service malveillant ou une machine virtuelle dans le système de ***Cloud***. Ce type d'attaque pourrait servir à un usage particulier que l'adversaire est intéressé, allant de l'écoute via des modifications de données subtiles aux changements ou blocages de toutes les fonctionnalités. Cette attaque nécessite que l'adversaire crée son propre module d'implémentation du service malveillant (SaaS ou PaaS) ou instance de machine virtuelle (IaaS), et l'ajouter au système de ***Cloud***. Ensuite, l'adversaire doit tromper le système ***Cloud*** de sorte qu'il traite la nouvelle instance d'implémentation du service comme l'une des instances valides pour le service particulier attaqué par l'adversaire. [**2, 30**]

## VI. Les Top menaces dans Cloud Computing par CSA

Le CSA a publié la version 1 du document « Menaces de ***Cloud Computing***» qui identifie les tops menaces de ***Cloud Computing***. Ces menaces sont décrites dans ce qui suit [**13**]:

- Utilisation néfaste et abus de ***Cloud Computing*** « *Abuse and Nefarious Use of* ***Cloud Computing***"
- Interfaces de programmation d'applications non sécurisé « *Insecure Application Programming Interfaces* »
- Initiés malveillants « *Malicious Insiders* »
- Vulnérabilités de la technologie partagée « *Shared Technology Vulnerabilities* »
- Perte/fuites de données « *Data Loss/Leakage* »
- Détournement de Compte, de Service & de trafic « *Account, Service & Traffic Hijacking* »
- Profil de risque inconnu « *Unknown Risk Profile* »

### VI.1. l'Utilisation néfaste et abus de *Cloud Computing*

Les fournisseurs ***IaaS*** offrent à leurs clients l'illusion de calcul, réseau et de capacité de stockage illimitée souvent associée à un processus d'enregistrement où n'importe qui avec

une carte de crédit valide peut s'inscrire et commencer immédiatement l'utilisation des services de ***Cloud Computing***. Certains fournisseurs offrent même des périodes d'essai gratuites et limitées. En abusant de l'anonymat derrière ces enregistrement et modèles d'utilisation, les spammeurs, les auteurs de codes malveillants, et autres criminels ont pu mener leurs activités en toute impunité. Les fournisseurs de PaaS ont traditionnellement le plus souffert de ce genre d'attaques, mais des observations récentes montrent que les pirates ont commencé à cibler les fournisseurs IaaS, **[11 ,13]**.

### VI.2. Interfaces de programmation d'applications non sécurisé

Les fournisseurs de ***Cloud Computing*** exposent un ensemble d'interfaces de logiciels ou des API que les clients utilisent pour gérer et interagir avec des services de ***Cloud Computing***. L'approvisionnement, la gestion, l'orchestration et le suivi sont tous effectués dans ces interfaces. La sécurité et la disponibilité des services de nuages dépendent de la sécurité de ces API de base. De l'authentification et le contrôle d'accès au chiffrement et le suivi des activités, ces interfaces doivent être conçues pour protéger contre les tentatives accidentelles et malveillantes. En outre, des organisations et des tiers se fondent souvent sur ces interfaces pour offrir des services à valeur à leurs clients. Cela introduit la complexité de la nouvelle API, elle augmente également le risque, **[11 ,13].**

Étant donné que les ***Cloud***s suivent une approche SOA, le problème de l'intégrité des données se amplifié par rapport à d'autres systèmes distribués. En outre, les services Web s'appuient généralement sur XML, SOAP et REST, et les API. La plupart des fournisseurs offrent leurs API sans le support des transactions, ce qui complique davantage la gestion de l'intégrité des données entre plusieurs applications SaaS. En outre, le modèle PaaS fournit des plates-formes et des frameworks pour développer des applications de nuage. Au cours du cycle de développement d'applications, les programmeurs doivent déployer des mesures de sécurité rigoureuses en se concentrant sur l'authentification, le contrôle d'accès et le cryptage. Néanmoins, des API faibles (peut permettre d'accueillir des chevaux de Troie) avec les appels systèmes insécurité peuvent comporter des points d'entrée aux intrus, **[11 ,13].**

Bien que la plupart des fournisseurs assurent que la sécurité est bien intégrée dans leurs modèles de services, il est essentiel pour les consommateurs de ces services de

comprendre les conséquences de sécurité liées à l'utilisation, la gestion, l'orchestration et la surveillance des services de ***Cloud Computing***. Le recours à un ensemble d'interfaces et de faibles API expose les organisations à une variété de problèmes de sécurité liés à la confidentialité, l'intégrité et la disponibilité, [**11 ,13**].

### VI.3. Initiés malveillants

L'une des préoccupations primaires de sécurité dans le ***Cloud Computing*** est que le client perd le contrôle direct sur les données sensibles et confidentielles et Cela nécessite plus d'attention par le fournisseur. Les personnes qui travaillent dans une entreprise prestataire de ***Cloud*** sont le point le plus crucial de la sécurité. La plupart du temps, les attaquants sont des initiés qui savent tout sur le système. Le risque d'un initié malveillant est l'un des menaces de sécurité les plus dangereuses. Le ***Cloud*** Security Alliance résume que ce genre de situation clairement crée une opportunité attrayante pour un ennemi allant du pirate amateur, au crime organisé, à l'espionnage industriel. Le niveau d'accès accordé pourrait permettre un tel ennemi de recueillir des données confidentielles ou de prendre le contrôle complet sur les services en **nuage** avec un peu (ou aucun) de risque de détection. Il peut conduire à des situations catastrophiques comme l'incidence financière, les dégâts de la marque, et les pertes énormes de productivité, etc, **[11 ,13].**

### VI.4. Vulnérabilités de la technologie partagée

Fournisseurs IaaS offrent leurs services d'une manière évolutive par partage des infrastructures. Souvent, les composants sous-jacents qui composent cette infrastructure (par exemple, les caches CPU, GPU, etc) n'ont pas été conçus pour offrir des propriétés d'isolation fortes pour une architecture multi-locataire (Multi-tenancy). Pour remédier cette lacune, un hyperviseur de virtualisation gère l'accès entre les systèmes d'exploitation invités et les ressources de calcul physiques. même les hyperviseurs ont montré des failles qui ont permis à des systèmes d'exploitation invités à avoir des niveaux inappropriés de contrôle ou d'influence sur la plate-forme sous-jacente. Une stratégie de défense en profondeur est recommandée, et devrait inclure la sécurité et la surveillance de calcul, de stockage, et des réseaux afin d'assurer que les clients individuels n'affectent pas les opérations des autres locataires fonctionnant sur le même fournisseur de ***Cloud*** ou que

les clients n'ont pas un accès à des données réelles ou résiduelles de toute autre locataire, **[11 ,13].**

### VI.5. Perte/fuites de données

La perte ou la fuite de données peut avoir un impact dévastateur sur une entreprise. Au-delà de l'endommagement de sa marque et sa réputation, une perte pourrait avoir un impact important sur la confiance et le moral des employés, partenaires et client. Pire encore, selon les données perdues, il pourrait y avoir des conséquences juridiques, **[11 ,13].**

### VI.6. Détournement de Compte, de Service et de trafic

Le détournement de compte ou de service n'est pas nouveau. Les méthodes d'attaque traditionnelles sont toujours utilisées pour obtenir des résultats dans le ***Cloud Computing***. Les informations d'identification et les mots de passe sont souvent réutilisés, qui amplifie les effets de ces attaques. Les solutions de ***Cloud*** ont ajouté une nouvelle menace. Si un attaquant obtient l'accès à vos informations d'identification, il peut espionner vos activités et opérations, manipuler des données, retourner des informations falsifiées, et rediriger vos clients à des sites illégitimes. Votre compte ou instance de service peut devenir une nouvelle base pour l'attaquant. De là, il peut exploiter la puissance de votre réputation pour lancer des attaques ultérieures

Le détournement de compte et de service, généralement avec des informations d'identification volées, reste une menace supérieure. Avec les informations d'identification volées, les attaquants peuvent souvent accéder à des zones critiques de services déployés de ***Cloud Computing***, leur permettant de compromettre la confidentialité, l'intégrité et la disponibilité de ces services. Les organisations doivent être conscients de ces techniques ainsi que la défense commune en profondeur les stratégies de protection pour contenir les dégâts (et de possibles litiges) résultant d'une violation, **[11 ,13]**.

### VI.7. Profil de risque inconnu

Les versions des logiciels, les mises à jour de codes, les profils de vulnérabilité, les tentatives d'intrusion et la conception de la sécurité, sont tous des facteurs importants pour l'estimation de la sécurité de votre entreprise. Lors de l'adoption d'un service de ***Cloud Computing***, les caractéristiques et les fonctionnalités peuvent être bien annoncés,

mais qu'en est-il les détails ou le respect des procédures internes de sécurité? Comment vos données et les journaux associés sont stockés et qui a accès à eux? Quelles informations le vendeur divulguer dans le cas d'un incident de sécurité? Souvent, ces questions ne sont pas clairement répondues ou sont négligés, laissant les clients avec un profil de risque inconnu qui peut inclure des menaces sérieuses, **[11 ,13].**

| **Les Modèle de service / Les Menaces** | **IaaS** | **PaaS** | **SaaS** |
|---|---|---|---|
| Utilisation néfaste et abus de ***Cloud Computing*** | ✓ | ✓ | x |
| Interfaces de programmation d'applications non sécurisé | ✓ | ✓ | ✓ |
| Initiés malveillants | ✓ | ✓ | ✓ |
| Vulnérabilités de la technologie partagée | ✓ | x | x |
| Perte/fuites de données | ✓ | ✓ | ✓ |
| Détournement de Compte, de Service & de trafic | ✓ | ✓ | ✓ |
| Profil de risque inconnu | ✓ | ✓ | ✓ |

**Tableau 2.2 : Les principales menaces de *Cloud Computing*, comme décrit dans la norme CSA, ainsi que les modèles de services qu'elles affectent. (✓) : Signifie que la menace affecte le modèle sous-jacent. (x) : Signifie le contraire**

## VII. Gestion d'identités et des accès :

Dans une organisation typique où les applications sont déployées dans le périmètre de l'organisation, la "zone de confiance" est principalement statique et est surveillé et contrôlé par le service informatique. Dans ce modèle traditionnel, la limite de confiance englobe le réseau, les systèmes et les applications hébergées dans un ***Datacenter*** privé géré par le service informatique. Et l'accès au réseau, aux systèmes et aux applications est sécurisé via des contrôles de sécurité de réseau, y compris les réseaux privés virtuels (**VPN**), des systèmes de détection d'intrusion (**IDS**), les systèmes de prévention des intrusions (**IPS**) et l'authentification multifactorielle, [**79**].

Avec l'adoption des services en ***Cloud***, la limite de confiance de l'organisation deviendra dynamique et se déplace hors du contrôle de celui-ci (sous le contrôle de l'organisation). Avec le ***Cloud Computing***, les limites de réseau, de système et d'application d'une organisation seront étendra dans le domaine de fournisseur de service. Pour compenser la perte de contrôle du réseau et pour renforcer l'assurance des risques, les organisations seront obligés de compter sur d'autres contrôles de logiciels de niveau supérieur, tels que la sécurité des applications et des contrôles d'accès des utilisateurs. En particulier, les entreprises doivent faire attention à l'architecture et le traitement de la Fédération d'identité, car ils peuvent renforcer les contrôles et la confiance entre les organisations et les fournisseurs de services de ***Cloud***, [79].

Le ***Cloud*** nécessite de fortes capacités de gestion des identités, de facturer correctement et avec précision le client grâce à un approvisionnement correct. Ainsi, contrairement aux IDM traditionnel, la gestion des utilisateurs et des services n'est pas suffisante. Lorsqu'un déploiement ou un service ou une machine (VM) est hors service, l'IDM doit être informée afin que les futur accès à celle-ci soient révoqués. L'IDM dans le **nuage**, devrait idéalement garder ses détails jusqu'à ce qu'il devienne actif. Pendant ce temps l'accès à ses données stockées pertinentes doit être surveillé et accordée par le niveau d'accès défini pour ce mode comme mentionné dans la SLA. L'IDM traditionnel n'est pas directement prête pour le ***Cloud Computing*** en raison des caractéristiques de ***Cloud***. Le ***Cloud*** d'aujourd'hui exige une gouvernance dynamique des problèmes typiques d'IDM, comme l'approvisionnement/de-provisionnement, la synchronisation, les droits, la gestion du cycle de vie, etc. Alors Les clients devraient avoir le droit de savoir comment les sociétés prestataires prennent soin de ce problème, **[79]**. Alors, La gestion des identités permet d'éviter les failles de sécurité et joue un rôle important pour aider votre entreprise à atteindre les règles de conformité en matière de sécurité, **[36]**.

Comprendre et documenter clairement les exigences spécifiques de l'utilisateur est impératif dans la conception d'une solution visant à assurer ces nécessités. La vérification des identités, dont beaucoup partagent des exigences communes en matière de sécurité fondamentales, et de déterminer les besoins spécifiques de protection des données et de la sécurité de l'information, peut être un des éléments les plus complexes de la

conception. Cet environnement distribué multiutilisateurs propose des défis uniques de sécurité, dépendant du niveau auquel l'utilisateur actionne, l'application, virtuel ou physique (tableau 2.3) :

| Niveau | Niveau de service | Utilisateurs | Les exigences de sécurité | menaces |
|---|---|---|---|---|
| **Application** | **SaaS** | Client final : une personne ou une organisation qui s'abonne à un service offert par un fournisseur de ***Cloud*** et est responsable de son utilisation | • Protection des données dans un environnement multi-tenant<br>• Protection des données à partir de l'exposition (restes)<br>• Contrôle d'accès<br>• Protection de la Communication<br>• Sécurité des logiciels<br>• Disponibilité du service | • Interception<br>• Modification des données au repos et en transit<br>• interruption de données (suppression)<br>• violation de la Confidentialité<br>• Usurpation d'identité<br>• détournement de Session<br>• Exposition dans le réseau |
| **Virtuel** | **PaaS, IaaS** | Développeur : une personne ou une organisation qui déploie le logiciel sur une infrastructure de ***Cloud*** | • Contrôle d'accès<br>• La sécurité des applications<br>• La sécurité des données, (données en transit, les données au repos, la rémanence)<br>• la sécurité du contrôle de gestion<br>• la protection de ***Cloud*** virtuel<br>• Sécurité des communications | • Défauts de programmation<br>• Modification de logiciel<br>• Interruption logicielle (suppression)<br>• usurpation d'identité<br>• Détournement de session<br>• Analyse de flux de trafic<br>• Exposition dans le réseau<br>• Dégradation<br>• inondations des Connexions<br>• DDOS<br>• perturbation des communications |
| **Physique** | **Datacenter** | Propriétaire: une personne ou une organisation qui possède l'infrastructure sur laquelle les nuages sont déployés | • utilisation légale et non abusive du ***Cloud Computing***<br>• La sécurité du matériel<br>• la fiabilité du matériel<br>• Protection du réseau<br>• Protection des ressources dans le réseau | • Les attaques de réseau<br>• inondation de connexion<br>• DDOS<br>• interruption de matériel<br>• le vol de matériel<br>• modification du matériel<br>• Détournement de l'infrastructure<br>• Les catastrophes naturelles |

**Tableau 2.3: défis de sécurité dans le *Cloud,* [16]**

## VIII. Conclusion

Le ***Cloud Computing*** est une cible idéale pour de nombreuses applications car elle fournit les besoins de stockage et de calcul pour les utilisateurs avec un coût relativement faible. Malgré les progrès rapides des applications de ***Cloud Computing*** au cours des dernières années, les risques de sécurité constituent toujours un défi important. C'est parce qu'une fois les utilisateurs de ***Cloud*** hébergent leurs données à un nuage, le contrôle d'accès à ces données devient la responsabilité du fournisseur de ***Cloud***.

Dans ce chapitre, nous avons présenté les notions essentielles de la sécurité informatique dans les systèmes distribués et en général dans le ***Cloud Computing***. En suite nous avons passé à la description des différentes attaques dans l'environnement de ***Cloud*** ainsi les tops menaces définis par la CSA. Cependant, le ***Cloud Computing*** n'est pas nécessairement plus ou moins sécurisé que les environnements actuels, Comme pour toute nouvelle technologie, il crée de nouveaux risques et de nouvelles possibilités.

Les applications ou les clients de ***Cloud Computing*** nécessitent des fonctionnalités qui ne peuvent être fournis par un seul fournisseur de services, ou qui ne peuvent pas être répondus par un service atomique. Ce qui nécessite un ensemble des services répondants aux besoins d'une seule requête. L'architecture orientée services (SOA) permet de composer plusieurs services afin de répondre à ces besoins.

# Chapitre IIII

# SOA & Service Web

*« Chacun sait que dans le train de l'entreprise,*

*le vendeur est la locomotive. »*

*Winston Churchill*

## I. Introduction

La méthode traditionnelle de création et d'exécution des applications d'entreprise s'est complexifiée et alourdie. Il y a trop d'éléments variables à acheter, installer, configurer et maintenir, logiciels comme matériels. Sans parler de l'infrastructure qui exige une maintenance constante pour pouvoir fonctionner comme il se doit. Ces charges générales constituent des obstacles à la productivité dans un développement d'applications métiers personnalisées.

Les solutions de ***Cloud Computing*** sont conçues pour aider les entreprises à résoudre ce problèmes, à réaliser des économies et à simplifier leur structure informatique, en leur permettant de disposer d'applications métiers comme autant de «libre services», à la demande, mutualisés, dématérialisés, contractualisés et évolutifs. Ces applications prêtes à l'emploi ne nécessitent pas de maintenance ; les mises à jour, le déploiement, le stockage et la sauvegarde sont du ressort du fournisseur de service. Tout en leur offrant la flexibilité et l'agilité dont elles ont besoin pour prospérer.

Le ***Cloud Computing*** offre un autre emplacement pour les processus métier à l'aide de l'infrastructure externalisée et les processus métier réutilisables qui sont accessibles à la demande. Le ***Cloud Computing***, dans de nombreux cas, fournissent une meilleure vitesse de développement, un accès aux ressources presque illimité, et beaucoup plus d'avantages par rapport aux approches traditionnelles d'entreprises. Le déplacement avec succès dans le ***Cloud Computing*** nécessite une architecture qui supporte les nouvelles fonctionnalités de ***Cloud Computing***. L'**architecture orienté service** peut directement améliorer la qualité de la conception et la performance des solutions basées sur le ***Cloud***, en particulier ceux avec des services partagés. La ***SOA*** est une approche technique qui peut aller loin dans la sécurité, la fiabilité et la réutilisabilité des ***services***. Le principe d'orientation vers le service signifie que la logique nécessaire pour résoudre un grand problème peut être mieux construit, réalisé et géré, si elle est décomposée en une collection de petits morceaux (services atomiques), dont chacun répond à une préoccupation ou une partie spécifique du problème. La ***SOA*** encourage les unités individuelles de la logique d'exister de manière autonome mais pas isolés les uns des autres.

Mais, le développement rapide de l'utilisation du ***Cloud Computing*** conduit à la publication de plus de services sur ***Cloud Computing***. Et en raison de la présence de services complexes, un seul service simple ne peut pas satisfaire les exigences fonctionnelles existantes pour de nombreux cas dans le monde réel. Pour compléter un service complexe, il est essentiel d'avoir un ensemble de services simples atomiques coopérés les un avec les autres. Par conséquent, il existe un fort besoin d'incorporer une composition de service dans le ***Cloud Computing***. En raison de grand nombre de services simples fournis par de nombreux fournisseurs de services dans le ***Cloud***, la composition de service dans le ***Cloud*** est considérée comme un problème NP-difficile.

## II. Architecture Orienté Service

### 1) Définition :

Une architecture orientée services (***SOA*** pour **S**ervices **O**riented **A**rchitecture) est une architecture logicielle s'appuyant sur un ensemble de services simples. L'objectif d'une architecture orientée services est donc de décomposer une fonctionnalité en un ensemble de fonctions basiques, appelées ***services***, fournies par des composants et de décrire finement le schéma d'interaction entre ces services. L'idée sous-jacente est de cesser de construire la vie de l'entreprise autour d'applications pour faire en sorte de construire une architecture logicielle globale décomposées en services correspondant aux processus métiers de l'entreprise, **[15].**

Les ***SOA***s sont considérées comme une infrastructure pertinente prend en charge pour la description, la découverte, la composition et l'invocation des **services Web**. La ***SOA*** promeut la réutilisation de composants logiciels au niveau macro (en comparaison avec la programmation orientée objet qui promeut la réutilisation au niveau micro (classes, objets).

### 2) Acteurs de la *SOA*

Il y a trois types d'acteurs dans l'architecture des services Web qui sont respectivement le ***Fournisseur* de services**, l'***Annuaire* des services** et le ***Client***.

Un ***annuaire de services*** est une base de données spécialisée permettant de stocker des informations de manière hiérarchique et offrant des mécanismes simples

de navigation ou de requêtes avec filtres pour permettre la recherche de services par des clients selon des critères variés. L'annuaire fournit aussi des renseignements sur les fournisseurs de services comme leur domaine d'activité et leur localisation,**[90].**

Depuis 2000, le standard ***UDDI*** a été proposé pour permettre l'établissement d'un format d'***annuaire des services Web***. L'***UDDI*** propose une formalisation complète des informations nécessaires au bon fonctionnement de l'architecture des **services Web**. Il recueille les inscriptions des fournisseurs de **services Web** et permet à tout d'effectuer des recherches pour y trouver les services particuliers dont il a besoin.

Dans la figure 3.1, le ***Fournisseur*** est l'application fournissant des services pour le compte d'un ***Client***. Il enregistre auprès de l'***Annuaire*** les services qu'il détient (***étape 1***). Le ***Client*** est l'application qui réclame un service fourni par un ***Fournisseur*** de services. Le ***Client*** cherche dans l'***Annuaire*** le service dont il a besoin (***étape 2***). S'il le trouve (***étape 3***), l'***annuaire*** donne les références du ***fournisseur*** de ce service afin que les procédures de demande et de réponse (***étape 4 et 5***) puissent s'effectuer entre le ***Client*** et le ***Fournisseur***.

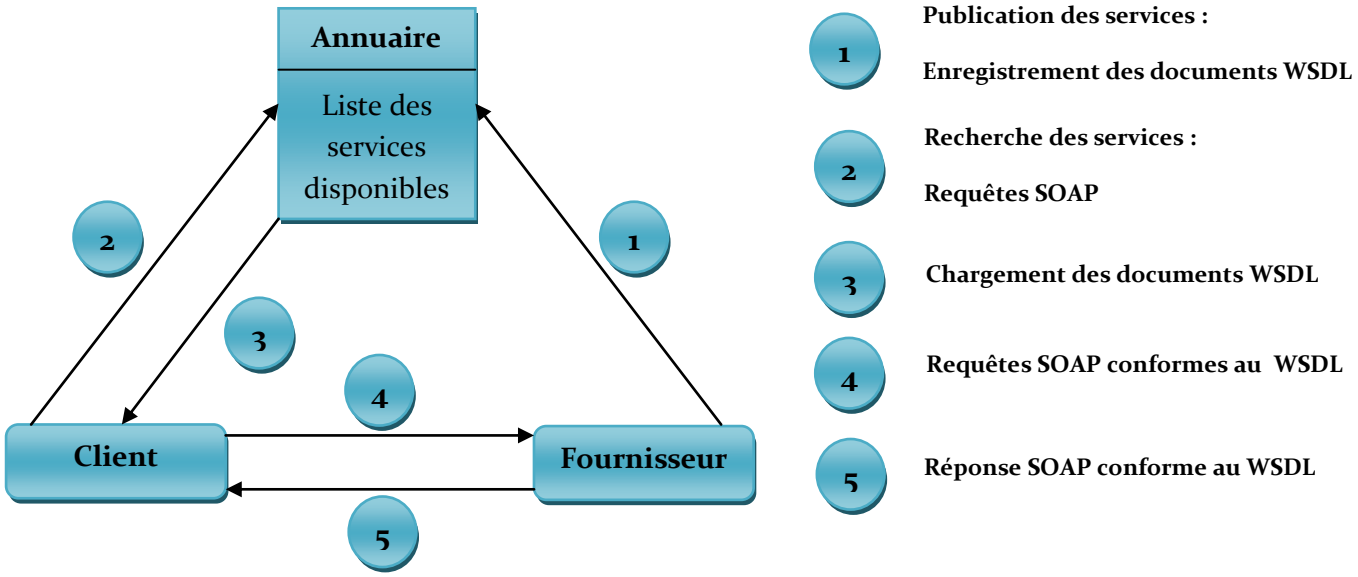



**Figure 3.1: architecture d'un Service Web**

### 3) Éléments d'une architecture orientée services

La ***SOA*** présente une approche pour construire des systèmes distribués qui fournissent leurs fonctionnalités sous la forme des services. La pile architecturale (figure

3.2) est divisée en 2 parties. La partie droite aborde les aspects fonctionnels, alors que, la deuxième s'intéresse aux aspects non-fonctionnels (qualité de service).

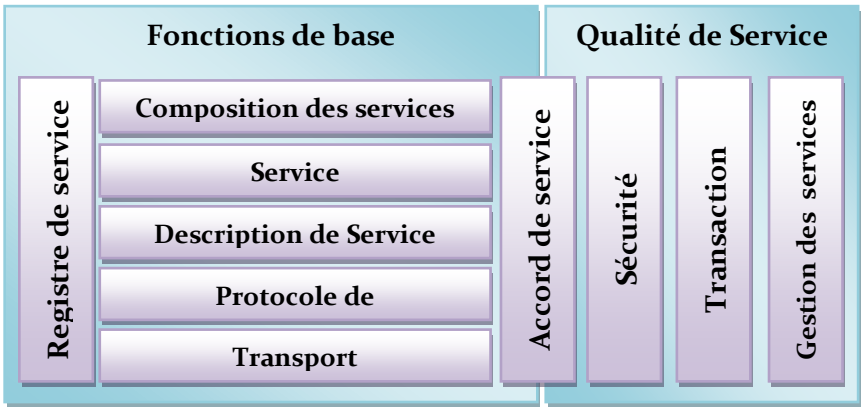



**Figure 3.2 : Éléments d'une architecture orientée services**

La partie des aspects fonctionnels inclut :

- ***Protocole de communication et transport :*** Il assure la communication (le transport des requêtes et réponses) entre le client et le fournisseur ;
- ***La description du service :*** Elle est utilisée pour décrire les capacités fonctionnelles et les éléments non-fonctionnels du service, ainsi que la manière dont il doit être invoqué ;
- ***Le registre des services :*** Il contient les descriptions des services mis à disposition par les fournisseurs. Il permet aussi aux clients de rechercher les services satisfaisants ses besoins ;
- ***La composition de services.*** Cet élément offre les mécanismes nécessaires pour assembler des services et créer des processus métiers.

La partie des aspects non-fonctionnels inclut :

- ***L'accord de service*** : On trouve cet élément dans les 2 parties. Il représente le contrat de service, qui représente un ensemble de conditions ou de règles d'utilisation de service par le client. Dans ce contrat, les aspects fonctionnels (les fonctionnalités que doit offrir le service) et les aspects non-fonctionnels (le temps de réponse, le coût, la fiabilité…) sont spécifiés ;
- ***La sécurité :*** C'est l'ensemble des règles qui pourraient être appliquées à l'identification, à l'autorisation, et au contrôle d'accès des clients de service ;

- ***La gestion des services :*** L'ensemble d'attributs qui pourraient être appliqués à la gestion des services fournis ou utilisés ;
- ***La transaction :*** Elle assure la cohérence des données, surtout dans le cas d'une collaboration entre plusieurs services.

### 4) Avantages d'une architecture orientée service

Une ***SOA***, avec sa nature faiblement couplée, permet d'obtenir tous les avantages d'une architecture client-serveur, **[62]**, et notamment :

- ***Une modularité*** : permettant de remplacer facilement un composant (service) par un autre ;
- ***Une réutilisabilité possible des composants*** : Ceci est particulièrement puissant pour créer des nouveaux processus métiers rapides et fiables.
- ***Recomposition*** : La capacité de modifier les processus métiers existants ou d'autres applications basées sur l'agrégation des services.
- ***De meilleures possibilités d'évolution*** : il suffit de faire évoluer un service ou d'ajouter un nouveau service ;
- Une plus grande tolérance aux pannes ;
- Une maintenance facilitée.

## III. Services Web

### 1) Définition:

Un **Service Web** (***SW***), (voir figure 3.3), est un logiciel accessible par Internet grâce au protocole ***SOAP*** (***S**ervice **O**riented **A**rchitecture **P**rotocol*), développé au dessus du protocole ***HTTP*** (***H**yper**T**ext **T**ransfer **P**rotocol*), offrant une interface décrite dans un langage de description d'interface appelé **WSDL** (***W**eb **S**ervice **D**escription **L**anguage*). ***SOAP*** est un standard de communication qui fixe le format des messages échangés. La représentation des messages est donnée en ***XML***. Les ***SWs*** sont faits pour permettre la réalisation rapide et efficace des systèmes d'information distribués sur des réseaux en intégrant des applications existantes et nouvelles. Une architecture à base de ***SW*** vise à permettre aux applications de communiquer facilement via Internet quelque soit la plate-forme sous-jacente, c'est-à-dire le système d'exploitation et le langage de programmation.

Pour ce faire, les ***SW***s s'appuient sur un ensemble de protocoles standardisant les modes de communication entre des composants applicatifs. Les standards utilisés dans l'architecture des ***SWs*** sont tous basés sur ***XML*** (*eXtensible Markup Language*), [**31, 82**].

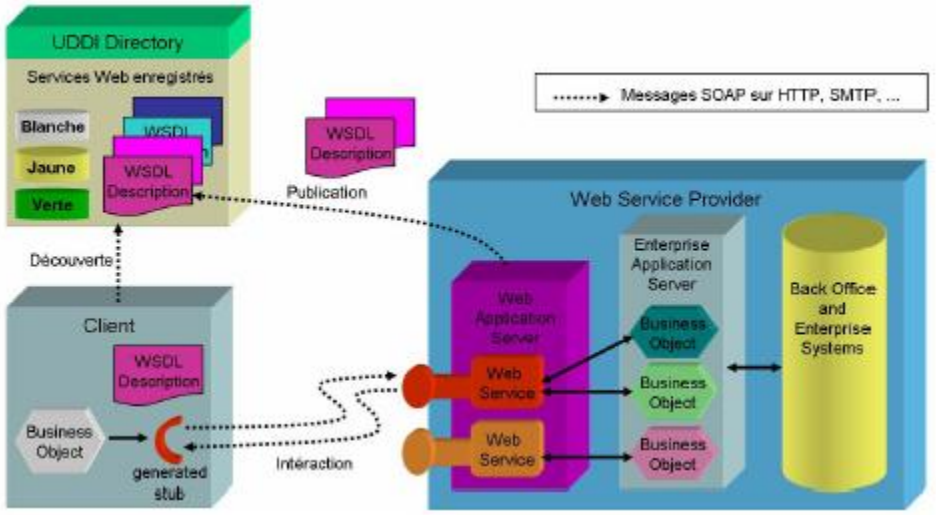



**Figure 3.3 : les protocoles de base des services Web**

**2) Apports des services Web**

Les ***SWs*** sont la technologie utilisée pour supporter le développement des systèmes d'information distribués. Aujourd'hui, ils semblent être la solution la plus adaptée pour assurer l'interopérabilité sur l'Internet. En fait, les acteurs du monde informatique désiraient supporter l'interopérabilité de façon uniforme et extensible. Les ***SWs*** sont basés sur des technologies standardisées, ce qui réduit l'hétérogénéité et fournit un support pour l'intégration d'applications. De plus, les ***SWs*** sont le support pour de nouveaux paradigmes, comme le traitement et l'architecture orientée service, **[67].** L'utilisation des ***SWs*** engendre plusieurs avantages dont on peut citer :

- ***L'interopérabilité*** : C'est la capacité des ***SWs*** d'interagir avec d'autres composantes logicielles via des éléments ***XML*** et utilisant des protocoles de l'Internet ;
- ***La simplicité*** : Les ***SWs*** réduisent la complexité des branchements entre les participants. Cela se fait en ne créant la fonctionnalité qu'une seule fois plutôt

qu'en obligeant tous les fournisseurs à reproduire la même fonctionnalité à chacun des clients selon le protocole de communication supporté ;

- ***Une composante logicielle légèrement couplée*** : L'architecture modulaire des ***SWs***, combinée au faible couplage des interfaces associées, permet l'utilisation et la réutilisation de services qui peuvent facilement être recombinés à différents autres applications ;
- ***L'hétérogénéité*** : Les ***SWs*** permettent d'ignorer l'hétérogénéité entre les différentes applications. En effet, ils décrivent comment transmettre un message (standardisé) entre deux applications, sans imposer comment construire ce message ;
- ***Auto-descriptivité*** : Les ***SWs*** ont la particularité d'être auto-descriptifs, c.à.d. capables de fournir des informations permettant de comprendre comment les manipuler. La capacité des services à se décrire par eux-mêmes permet d'envisager l'automatisation de l'intégration de services.

**3) Standards des SWs :**

L'infrastructure des ***SWs*** s'est concrétisée autour de trois spécifications considérées comme des standards, à savoir ***SOAP***, **UDDI** et **WSDL** :

- **Transport :** ***SOAP*** (**S**imple **O**bject **A**ccess **P**rotocol appelé aussi **S**ervice **O**riented **A**ccess **P**rotocol) est un mécanisme (protocole) qui permet d'échanger des messages entre les applications, principalement en utilisant le protocole ***HTTP***. Du fait qu'il est basé sur ***XML***, il permet l'échange de données structurées indépendamment des langages de programmation ou des systèmes d'exploitation. ***SOAP*** permet l'échange d'informations dans un environnement décentralisé et distribué, comme Internet, indépendamment du contenu du message **[67].** Il peut être employé dans tous les styles de communication : synchrones ou asynchrones, point à point ou multi-point. ***SOAP*** a été développé à l'origine pour traverser via ***HTTP*** les pare-feux qui bloquaient les appels ***RPC*** (***R****emote* ***P****rocedure* ***C****all*) de **DCOM** ou **IIOP**, **[31]**.

- ***Découverte: UDDI*** (*Universal Description, Discovery* ***and*** *Integration*) est une norme d'annuaire de ***SWs*** appelée via le protocole ***SOAP***. Pour publier un nouveau **Service Web**, il faut générer un document appelé « Business Registryé, il sera enregistré sur un « ***UDDI*** Business Registry Node », **[67]**. Le Business Registry comprend 3 parties :
  - ✓ **Pages blanches** : noms, adresses, contacts, identifiants des entreprises enregistrées ;
  - ✓ **Pages jaunes** : informations permettant de classer les entreprises, notamment l'activité, la localisation, etc ;
  - ✓ **Pages vertes** : informations techniques sur les services proposés.

Le protocole d'utilisation de l'***UDDI*** contient trois fonctions de base :

- ✓ « ***publish*** » pour enregistrer un nouveau service ;
- ✓ « ***find*** » pour interroger l'annuaire ;
- ✓ « ***bind*** » pour effectuer la connexion entre l'application cliente et le service.

Comme pour la certification, il est possible de constituer des annuaires ***UDDI*** privés, dont l'usage sera limité à l'intérieur de l'entreprise.

- ***Description*** **: WSDL, la documentation des Web services :** est un langage de description de ***SWs***. Il définit un format de document qui permet de décrire chaque service ***SOAP*** sous la forme d'une documentation en ligne. Ce document décrit les messages échangés, habituellement sous la forme d'un schéma **XML**, ainsi que les détails des interfaces publiques de ce service, **[67].**

Les services peuvent s'invoquer par le protocole ***SOAP*** dès qu'ils sont rendus disponibles. Pour rendre un service disponible, il faut :

- ✓ Fournir l'exécutable du fichier décrivant le service, par exemple en Java ;
- ✓ Fournir l'interface ***WSDL*** du service ;
- ✓ Déployer le service auprès **d'AXIS** par exemple.

**4) Qualité de service (QoS) :**

Le I/O, pre-/post-conditions de services sont utilisés pour décrire leur comportement. Ils expriment leurs propriétés fonctionnelles. Au cours de composition de ***SWs***, elles sont nécessaires pour envisager la conception d'un service composite répondant à l'objectif de l'utilisateur. Outre les propriétés fonctionnelles du service composite, l'utilisateur peut être intéressé aussi dans certaines propriétés non-fonctionnelles qui déterminent la qualité des services (**QoS**), [57].

En présence de plusieurs services Web avec des fonctionnalités identiques, la **QoS** fournit des caractéristiques non fonctionnelles pour la sélection de ***SWs*** optimal. Par conséquence, les informations **QoS** sont utilisées pour évaluer le degré qu'un **SW** répond aux exigences de qualité spécifiées dans une demande de service. Ces informations peuvent être de la performance d'un **SW** (en termes de temps de réponse, temps de latence ...), la disponibilité, l'accessibilité, la sécurité, etc. La **QoS** a des impacts importants de l'utilisateur sur les attentes et l'expérience de l'utilisation d'un **SW, [94]**. Les propriétés de qualité de service les plus représentatifs sont indiqués ci-après :

- ***Disponibilité***: le pourcentage de temps qu'un ***SW*** fonctionne pendant un certain intervalle de temps ;
- ***Prix*** : les frais que l'utilisateur de services doit payer pour invoquer un **SW** ;
- ***Popularité***: le nombre d'appels reçus d'un ***SW*** pendant un certain intervalle de temps ;
- ***Données de taille***: la taille de la réponse de l'invocation de ***SW*** ;
- ***taux de Succès***: la probabilité que la demande est correctement répondue dans le délai maximum prévu ;
- ***Temps de réponse***: la durée de temps entre l'utilisateur du service et l'envoi d'une demande de réception d'une réponse ;
- ***La réussite globale de taux***: la valeur moyenne du taux de réussite de l'invocation de tous les utilisateurs de services ;

- **Temps global de réponse**: la valeur moyenne des temps de réponse de tous les utilisateurs de services ;

## IV. Composition des services Web

Généralement, un **SW** unique ne répond pas aux besoins des utilisateurs qui sont de plus en plus complexes. Pour fournir une solution à une tâche complexe, des ***SWs*** peuvent être combiné pour former un seul service; on parle alors de **composition de services Web**.

### IV.1. Définition :

La composition de services est un des enjeux principaux du ***SOA***. En effet, les caractéristiques essentielles du ***SOA*** telles que le couplage faible entre les services, l'indépendance par rapport aux aspects technologiques et la mise à l'échelle favorisent la composition des ***SWs*, [18]**. Un ***SW*** est dit ***composé*** ou ***composite*** lorsque son exécution implique des interactions avec d'autres **SW** afin de faire appel à leurs fonctionnalités. La composition de **SW** spécifie quels services ont besoin d'être invoqués, dans quel ordre et comment gérer les conditions d'exception.

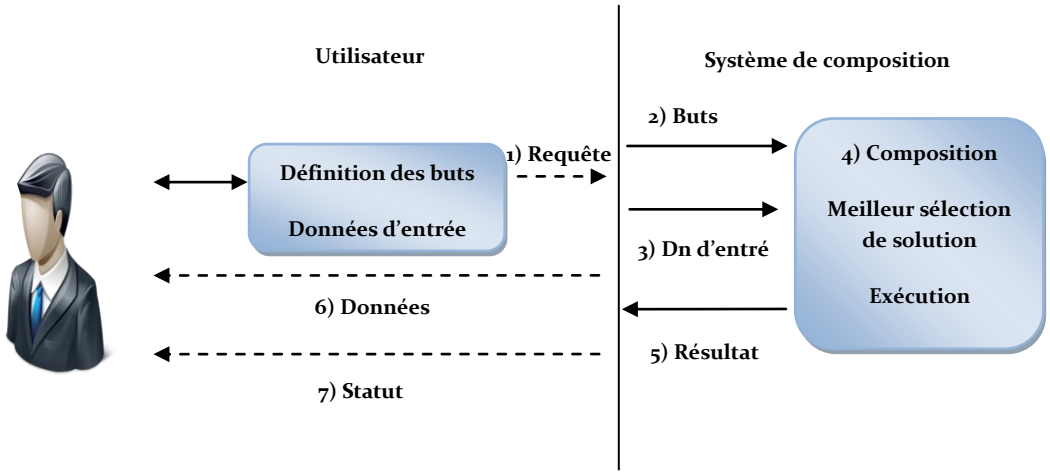



**Figure 3.4: processus de composition de services**

### IV.2. Services Métiers

Pour réussir avec la composition de service, nous devons d'abord comprendre qu'est ce qu'un service d'affaires ou service métier (***business service***). Nous avons déjà vu que la notion de service fait partie intégrante de la conception d'architectures orientées services. Plus importante encore que les caractéristiques techniques des services est la façon dont

nous concevons le service et la granularité des opérations qu'un service expose par le biais de l'interface de l'entreprise. Les services d'affaires dans une entreprise sont les activités effectuées afin de répondre à une demande, qu'elle soit interne ou externe. Les services d'affaires sont une réaction aux demandes des clients et toujours effectuent une action qui conduit à un résultat de l'entreprise. Les activités qui n'ont pas été demandées par les clients ne sont pas considérées comme des services d'affaires, **[87].**

Les Services d'affaires peuvent être de complexité différente, **[87]**:

- ✓ Services discrets sont des services qui exposent les opérations discrètes. Les opérations discrètes sont celles qui terminent dans une seule interaction. Ils sont exécutés en tans qu'ensemble. Les services distincts peuvent être synchrones ou asynchrones. Exemples de services discrets : affichage des messages, obtenir des informations (comme une facture, bon de commande, etc) ;
- ✓ Les services composites ou les services orchestrés sont des services qui exposent les opérations qui nécessitent plus d'une interaction à compléter.

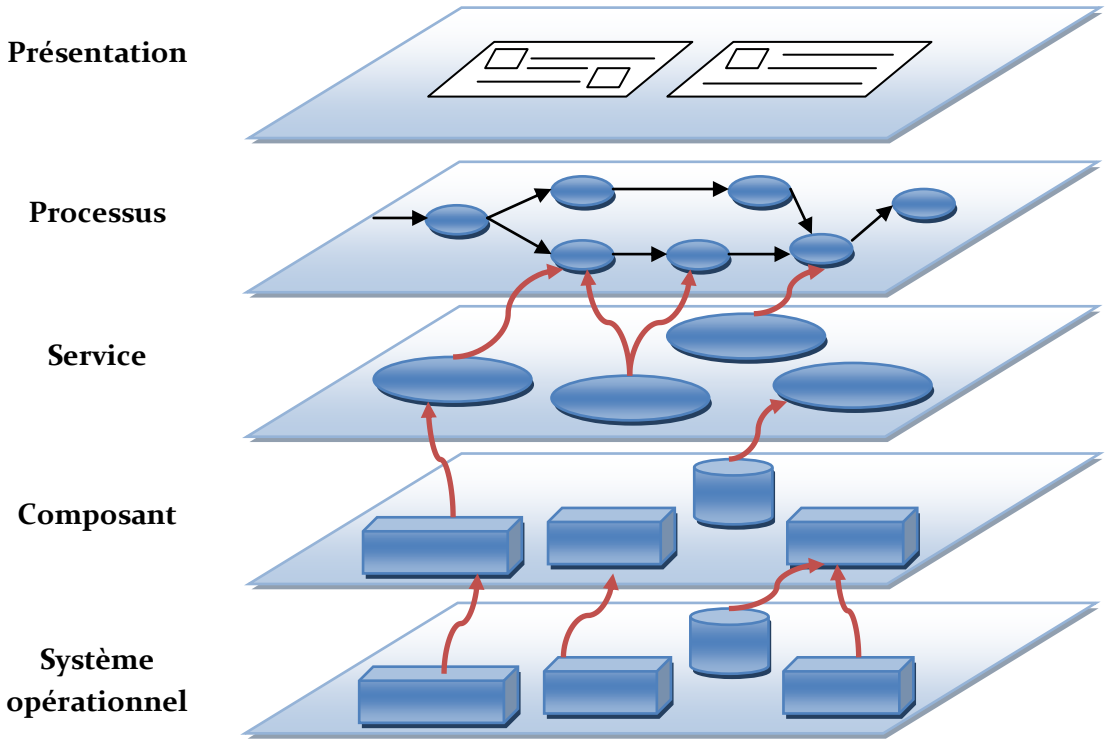



**Figure 3.5 : Position des processus métier dans l'architecture de la SOA**

La figure 3.5 Présente la SOA comme une architecture multicouche dont chacune utilise la couche précédente :

- **Couche système opérationnel** : cette couche est composée des applications personnalisées existantes, autrement dit les systèmes existants qui incluent les applications existantes des systèmes anciennes orienté objet ;
- **Couche service compostant** : cette couche contient des composants logiciels, chacun d'eux fournit la mise en œuvre ou la réalisation d'un service ou l'exploitation d'un service d'où le nom de service composant. les services interrogent les composantes de l'entreprise, qui a ensuite traiter la requête en utilisant les ressources disponibles dans les systèmes opérationnels. ;
- **Couche service** : cette couche se compose de tous les services qui sont définis au sein de SOA. La couche service peut être considérée comme contenant des descriptions de service et un conteneur pour l'implémentation des services ;
- **Couche processus** : les entreprises sont réalisées à partir des processus d'affaire ou une collection de ces processus, ces derniers contiennent des services d'orchestration et des compositions et la possibilité d'insérer l'intervention humaine ;
- **Couche présentation** : la couche des consommateurs fournit les capacités requises pour fournir des fonctions d'IT et de données aux utilisateurs. Elle peut être une interface pour l'application à une application de communication, et d'autres services et des canaux ;

Les Services d'affaires sont la pièce maîtresse d'une composition de services dans les processus d'affaires. Cette composition peut se faire de deux manières différentes: orchestration ou de chorégraphie :

1) **Orchestration** : Une orchestration décrit un ensemble d'actions en communication et d'actions internes dans lesquelles un service donné peut ou doit s'engager (afin de remplir ses fonctions) ainsi que les dépendances entre ces actions. Les ***SWs*** n'ont pas de connaissance (et n'ont pas besoin de l'avoir) d´être mêlées dans une composition et d'être partie d'un processus métier. Seulement le coordinateur de l'orchestration a

besoin de cette connaissance. En d'autres termes, elle propose une vision centralisée décrivant la manière par laquelle les services peuvent agir entre eux.

La figure 3.6 montre le workflow dans l'orchestration des SWs. Un coordinateur prend le control de tous les services web impliqués et coordonne l'exécution des différentes opérations des SWs qui participent dans le processus.

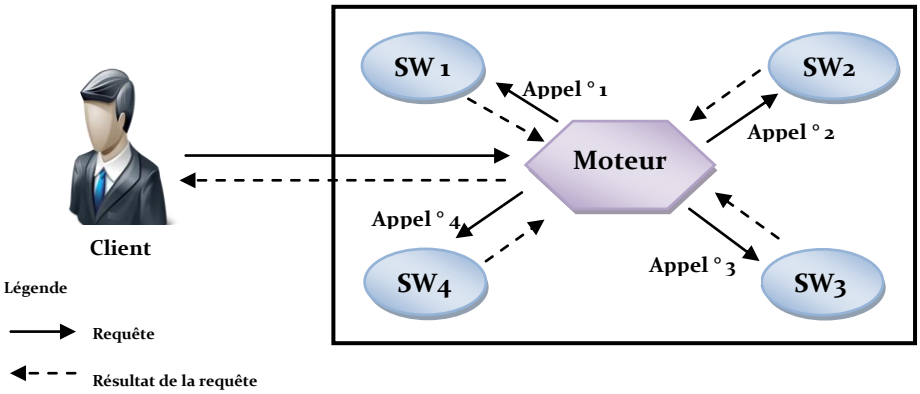



**Figure 3.6 : vue générale de l'orchestration**

2) **Chorégraphie** : Une chorégraphie décrit, d'une part un ensemble d'interactions qui peuvent ou doivent avoir lieu entre un ensemble de services (représentés de façon abstraite par des rôles), et d'autre part les dépendances entre ces interactions. Contrairement à l'orchestration, la chorégraphie n'a pas un coordinateur central. Chaque **SW** mêlée dans la chorégraphie connaît exactement quand ses opérations doivent être exécutées et avec qui l'interaction doit avoir lieu. La chorégraphie est un effort de collaboration dans lequel chaque participant du processus décrit l'itération qui l'appartient. Elle trace la séquence des messages qui peut impliquer plusieurs ***SWs***.

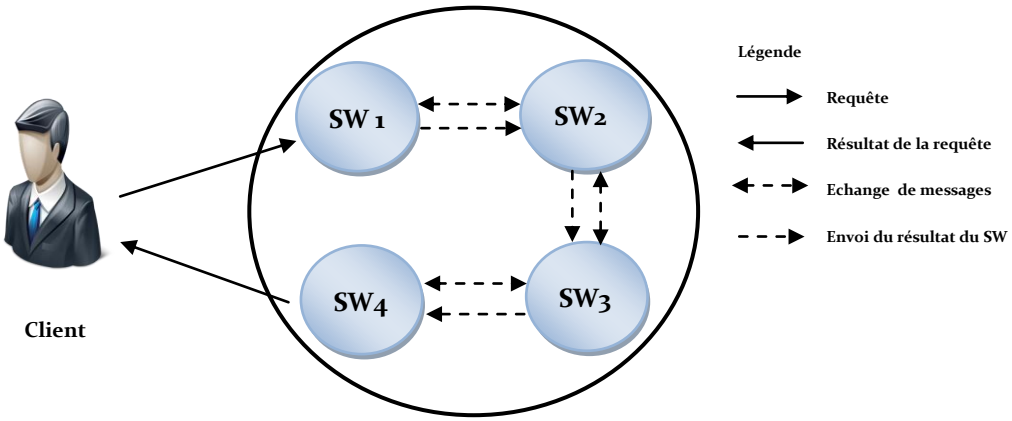



**Figure 3.7 : vue générale de la chorégraphie**

En comparaison avec la chorégraphie, l'orchestration est évidemment plus efficace et plus flexible quand il s'agit de composer des ***SWs*** pour exécuter des processus d'affaires. En fait, elle présente les avantages suivants par rapport à la chorégraphie:

**1.** La coordination des composantes d'un processus est gérée de manière centralisée par un coordonnateur spécifique ;

**2.** Des scénarios alternatifs peuvent être mis en place en cas de défauts.

**IV.3. Types de composition :** La composition peut être classifiée dans trois catégories, [51]:

- **Composition manuelle :** La composition manuelle des ***SWs*** suppose que l'utilisateur génère la composition à la main via un éditeur de texte et sans l'aide d'outils dédiés ;
- **Composition semi-automatique :** Les techniques de composition semi-automatiques sont un pas en avant en comparaison avec la composition manuelle, dans le sens qu'ils font des suggestions sémantiques pour aider à la sélection des ***SWs*** dans le processus de composition ;
- **Composition automatique :** La composition totalement automatisée prend en charge tout le processus de composition et le réalise automatiquement, sans qu'aucune intervention de l'utilisateur ne soit requise.

La composition manuelle est, parmi les trois, la plus adaptable aux besoins de l'utilisateur, car il va tout définir à son goût depuis le début. Mais, malheureusement, elle nécessite des connaissances de bas niveau de programmation. Cependant, il faut maintenir le plus grand niveau de flexibilité possible afin que l'utilisateur puisse définir son propre processus. C'est pour cette raison que la composition automatique a aussi été exclue car l'utilisateur n'intervienne pas dans la définition du processus de composition. La composition semi-automatique étant à mi-chemin entre les deux antérieures. L'utilisateur maintiendra certain contrôle sur le processus envisagé mais il n'aura pas besoin de connaissances de programmation et il pourra définir son propre processus en utilisant des outils graphiques pour modéliser et concevoir des ***SWs*** composites.

### IV.4. La composition de services dans le Cloud Computing :

L'augmentation du nombre de services disponibles dans le Cloud entraîne une augmentation du nombre de services de fonctions similaires de différents serveurs. Ces services similaires sont situés dans des endroits différents et ont des valeurs distinctes en termes de paramètres de qualité de service. Pour cette raison, la composition de service applique des techniques appropriées pour sélectionner un service atomique entre les différents services similaires pour permettre d'atteindre une QoS plus élevée selon les exigences de l'utilisateur final, [3].

En raison de changements intrinsèques dans l'environnement de Cloud, les services disponibles et les exigences de l'utilisateur final, la composition de service devrait être conçue dynamiquement, avec des capacités automatisée. Par conséquent, la sélection appropriées et optimales des services simples, pour être combinés ensemble pour fournir des services complexes, est l'un des problèmes les plus importants dans la composition de services car le service composite obtenu doit répondre à la fois aux exigences fonctionnelles et de QoS de l'utilisateur. La nature dynamique des environnements de Cloud implique des changements qui exposent le ***Cloud Computing*** à différents défis dans la composition de services. Les défis plus remarquables sont les suivantes, [3] :

- ***Contracter dynamiquement les fournisseurs de services*** : La politique de tarification de la plupart des fournisseurs de service est déterminée par les frais de service basés sur l'offre et la demande. Ainsi, les mécanismes de mise à jour du tableau des ressources disponibles doivent être prévus ;
- ***Ressources de Cloud incomplètes*** : La sélection optimale de service dépend de la disponibilité des informations complètes et actualisées sur les services. Les changements dans les caractéristiques des services pourraient entraîner la perte de certaines données ;
- ***Dépendance/conflit Interservices :*** La dépendance ou conflits qui existent entre deux ou plusieurs services conduit à un problème de composition de service complexe. Dans les scénarios réels, rencontrant la dépendance et les

conflits entre les services est assez fréquent et doit être considéré dans la Composition du Service

- ***Sécurité***. La conception des règles, des politiques et des instructions de sécurité sont parmi les responsabilités de base des fournisseurs de services de ***Cloud Computing***. Toutefois, un Framework auto-administré pour la fourniture des services, dans laquelle les préoccupations et les politiques de sécurité des vendeurs sont observés, doit être fournie

**IV.5. Langages de descriptions des services Web :**

- **XLANG**

Créé par Microsoft, le ***XLANG*** est une extension de ***WSDL***. Il fournit en même temps un modèle pour une orchestration des services et des contrats de collaboration entre celles-ci. ***XLANG*** a été conçus avec une base explicite de théorie de calcul. Les actions sont les constituants de base d'une définition de processus de ***XLANG***. Les quatre types d'opérations de ***WSDL*** (requête/réponse, sollicitation de la réponse, le sens unique, et la notification) peuvent être employés comme actions de ***XLANG***. ***XLANG*** ajoute deux autres genres d'action: arrêts (date-limite et durée) et exceptions ; [**1, 15**]

- **Web Service Flow Language (WSFL)**

***WSFL*** est un langage basé sur ***XML*** pour la description des ***SWs*** composite, [**51**]. ***WSFL*** considère deux types de compositions de ***SWs***:

- ✓ Le premier type indique le modèle approprié d'utilisation d'une collection de services web, de telle manière que la composition résultante décrive comment réaliser le but particulier d'un business. Typiquement, le résultat est une description d'un processus métier ;
- ✓ Le deuxième type indique le modèle d'interaction d'une collection de services web; dans ce cas, le résultat est une description des interactions globales associées.

- **Web Service Choreography Interface (WSCI)**

***WSCI*** est un langage reposant sur ***XML***. Il propose de se focaliser sur la représentation des ***SWs*** en tant qu'interfaces décrivant le flux de messages échangés (la chorégraphie de messages). Il propose ainsi de décrire le comportement externe observable du service. Pour cela, ***WSCI*** propose d'exprimer les dépendances logiques et temporelles entre les

messages échangés à l'aide de contrôles de séquences, corrélation, gestion de fautes et transactions. On remarque que le ***WSDL*** et ses définitions abstraites sont réutilisées afin de pouvoir également décrire par la suite les modalités de concrétisation des éléments manipulés pour modéliser un service, [1];

- **Web Service Conversation Language (WSCL)**

***WSCL*** propose de décrire à l'aide de documents ***XML*** les ***SWs*** en mettant l'accent sur les *conversations* de ceux-ci. En outre, les messages à échanger sont pris en compte. ***WSCL*** a été pensé pour s'employer conjointement avec ***WSDL***. Les définitions ***WSDL*** peuvent être manipulées par ***WSCL*** pour décrire les opérations possibles ainsi que leur chorégraphie. En retour, ***WSDL*** fournit les concrétisations vers des définitions de messages et des détails techniques pour les éléments manipulés par ***WSDL,*** [1];

- ***BPEL4WS***

***BPEL4WS*** est largement utilisé dans le cadre de la mise en œuvre d'une ***SOA***. Il se caractérise par rapport aux autres langages par sa gestion des exceptions (fautes et événements), l'exécution parallèle des activités et la synchronisation des flots, son mécanisme de compensation, et sa gestion de la corrélation des messages.

## V. *BPEL4WS* :

### V.1. Définition :

Initialement connu sous le nom de ***BPEL4WS*** (**B**usiness **P**rocess **E**xecution **L**anguage for **W**eb **S**ervice), renommé par la suite ***WS-BPEL*** (**W**eb **S**ervice-***BPEL***), cette spécification est plus connu sous le nom de ***BPEL***. Il s'agit d'un langage d'exécution des processus métiers qui permet la composition d'un ensemble de **SWs**, et spécifie les règles de dialogue entre eux. Il définit un protocole d'interaction d'un seul **SW** tout en spécifiant l'ordre d'invocation de ses opérations. Le fichier ***BPEL*** agit donc sur des éléments comme la transformation de données, l'envoi de messages ou l'appel d'une opération. ***BPEL*** est apparu en 2002, et a été standardisé par **OASIS** en 2007, et est supporté par de nombreux éditeurs de logiciel comme ***Adobe***, ***BEA Systems***, ***HP***, ***IBM, Oracle, JBoss, Sun, Tibco, Webmethods, Microsoft.***

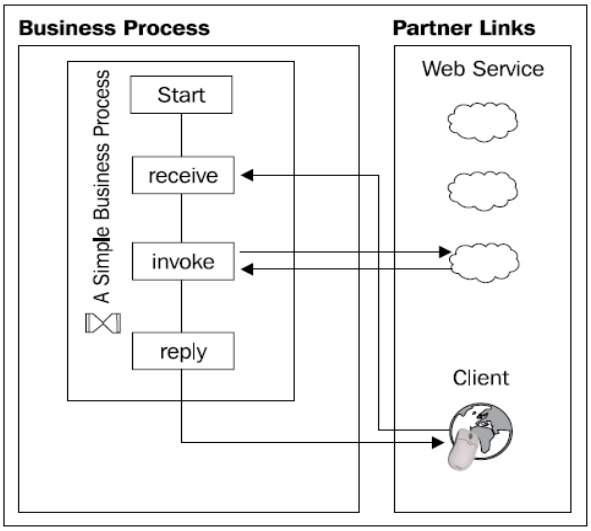



**Figure 3.8 : Un processus métier par *BPEL***

Il supporte deux types différents de processus :

- ✓ **Les processus exécutables** qui nous permettent de spécifier les détails du processus métier. Ils peuvent être exécutés au moyen d'un moteur d'orchestration. Dans la majorité des cas, ***BPEL*** est utilisé dans ce type de processus ;
- ✓ **Les processus abstraits** qui nous permettent de spécifier l'échange de messages entre partenaires du processus. Ils ne sont pas exécutables.

Avec la composition, nous composons des services dans les processus plus larges (ou services composites). Elle se réfère à la représentation de la logique d'état de transition de haut niveau d'un système. Les langages de programmation traditionnels n'ont pas été conçus à ces fins. Utiliser ces langages pour la composition des processus d'affaires est trop complexe, car les développeurs doivent faire face à trop de problèmes de bas niveau, au lieu de se concentrer sur l'orchestration. Par conséquent, ces tentatives aboutissent généralement à des solutions rigides, en particulier parce qu'il n'y a pas de séparation claire entre le flux de processus et la logique métier, qui ne devrait pas être étroitement couplé, **[87].**

La composition de services dans les processus a aussi quelques autres exigences spécifiques. L'une d'elles est le support pour plusieurs instances de processus simultanés. Un autre est le soutien aux processus qui s'exécutent pour une plus longue période de temps (jours, semaines, voire des mois). Pour ces processus de

longue durée, il est particulièrement important de fournir une sémantique transactionnelle, qui se fait souvent à l'aide des opérations de compensation. Enfin, il ya la nécessité d'une corrélation, la gestion des événements, des invocations d'opérations asynchrones, rappels, et plusieurs autres choses. Dans tous les thèmes mentionnés, ***BPEL*** est supérieur aux langages de programmation traditionnels et est donc le choix de facto pour le développement des processus d'affaires dans la ***SOA***.

### V.2. Caractéristiques du langage *BPEL* :

Ci-dessous sont les caractéristiques les plus importantes que ***BPEL*** fournit:

- Décrire la logique des processus d'affaires à travers la composition de services ;
- Composer de grands processus métiers à partir des petits procédés et services ;
- Invoquant des opérations de service en séquence ou en parallèle ;
- Compensation sélective des activités réalisées en cas de défaillance ;
- Maintien des multiples activités transactionnelles de longue durée, qui sont aussi interruptible ;
- Reprise des activités interrompues ou qui ont échoué à réduire le travail à refaire ;
- Gérer des exceptions, en particulier, des fautes et des événements ;
- Gérer la corrélation des messages, pour le cas des communications asynchrones.

***BPEL*** propose également des constructions, telles que des boucles, des branches, des variables, etc. Ces constructions sont très similaires à celles des langages de programmation traditionnels et permettent de définir les processus d'affaires de manière algorithmique. Il est moins complexe que les langages de programmation traditionnels.

### *V.3.* La structure d'un processus métier « Business process »

Le rôle de ***BPEL,*** [29], est de définir un nouveau ***SW*** en composant un ensemble de services existants. Voici ci-dessous le corps d'un processus ***BPEL*** dans la figure 3.9 :

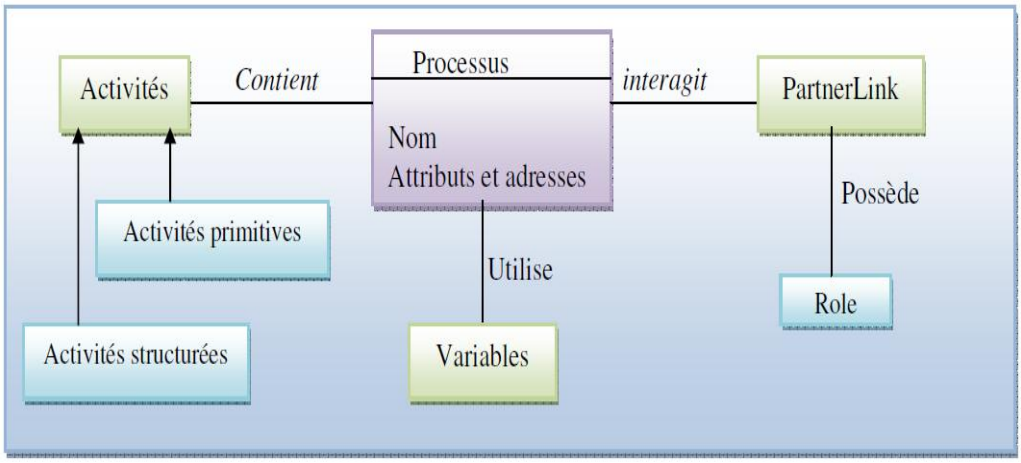



**Figure 3.9: schéma représentatif d'un processus *BPEL***

Pour construire un processus ***BPEL***, les éléments suivants sont requis:

- Les partenaires métiers « ***Business partners*** » qui vont interagir avec le processus ;
- Les informations sur le type d'échange de données entre le processus et les partenaires d'affaires ;
- Le flux de travail qui définit l'ordre d'exécution de processus (réception, appeler les **SWs**, et répondre à des partenaires commerciaux) ;
- le Processus ***BPEL*** a besoin d'un fichier ***WSDL*** afin de créer une définition exécutable de ***BPEL***. Le fichier ***WSDL*** se compose de l'espace de noms « *namespace* », les types des partenaires de liens « *partner link types* », les opérations et les messages qui sont nécessaires pour définir les activités de processus ;
- Un ***BPEL*** doit avoir des espaces de noms qui pointent vers les emplacements de schéma ***WSDL*** associés et d'autres ressources telles que des feuilles de style ***XSL*** et les fichiers ***XML*** utilisés dans les fonctions personnalisées du catalogue de ressources.

Voici la description des différents éléments dans un processus métier ***BPEL*** :

a) **Processus « process » :** Il est l'élément racine de la définition de processus ***BPEL***. Il possède un attribut de nom et il est utilisé pour spécifier la définition des espaces de noms liés ;

b) **Liens des partenaires « *Partner links elements* » :** Dans un processus ***BPEL***, Ces éléments définissent l'interaction des services participants au processus. Ils décrivent que ce sont les processus et les rôles des services à cette étape dans le flux, et ils définissent le type de données qui peuvent être traitées par les parties dans ces rôles ;

c) **Variables :** Un processus ***BPEL*** permet de déclarer des variables afin de recevoir, manipuler et envoyer des données. Par exemple, le processus reçoit un ordre de réservation d'un client et lui attribue le message à une variable d'entrée ***XML***, et cette variable peut être copié sur une autre opération. Les variables sont définies soit dans les types de messages ***WSDL***, ou des types de schémas ***XML***, ou des éléments de schéma ***XML*** ;

d) **Gestionnaires** « ***Handlers*** »**: le** Processus ***BPEL*** a trois types de gestionnaires, [**87**], qui sont :

1. **Gestionnaires de fautes :** sont invoqués lors d'une faute pendant le temps d'exécution d'un processus ***BPEL*** **.** La ***SOA*** est basée sur le concept de couplage faible. La communication entre les ***SWs*** se fait via les connexions Internet qui peuvent ou ne peuvent pas être très fiable. Les ***SWs*** peuvent également soulever des défauts dus à des erreurs logiques et les erreurs d'exécution découlant de défauts de l'infrastructure. Par conséquent, les processus opérationnels ***BPEL*** devront gérer les erreurs de manière appropriée. Les fautes dans ***BPEL*** peuvent survenir dans diverses situations. Une faute peut se produire lorsque:
   - Un processus ***BPEL*** appelle (invokes) une opération de SW synchrone. L'opération peut retourner un message d'erreur ***WSDL***, qui se traduit par une faute ***BPEL*** ;
   - Un processus ***BPEL*** peut signaler (lancer) explicitement une faute ;
   - Une faute peut être lancée automatiquement, quand un certain critère d'exécution n'a pas été respecté ;
   - Le serveur de processus ***BPEL*** peut rencontrer une condition d'erreur lors de l'exécution ou dans les communications réseau, etc.

Lorsqu'une erreur se produite dans un processus métier, cela signifie que le processus ne peut pas se terminer correctement. Le processus peut se terminer correctement que si la faute est gérée au sein du processus ***BPEL*** (à l'intérieur d'un champ d'application « ***scope*** »). Un processus métier peut gérer une faute par un ou plusieurs gestionnaires de faute. Au sein d'un gestionnaire de fautes,

les processus métier définissent des activités personnalisées qui sont utilisées pour récupérer la faute et le travail partiel (infructueux) de l'activité dans laquelle l'erreur s'est produite.

2. **Gestionnaires d'événements :** Un processus ***BPEL*** doit réagir à certains événements. Ces gestionnaires permettent un processus de réagir aux événements extérieurs. Ceux-ci peuvent être des événements de message, qui sont liées à l'invocation des opérations sur le processus ***BPEL*** par les clients, ou des événements d'alarme, qui peut se produire à un certain moment ou représenter une durée. Dans la plupart des processus métier, nous aurons besoin de réagir à deux types d'événements:
   - **Événements du message**: Ces événements sont déclenchés par des messages entrants par l'appel d'opération sur les types de ports (***porttypes***);
   - **Événements d'alarme**: qui sont reliés au temps et sont déclenchés après une certaine durée ou à un moment précis.

   Les gestionnaires d'événements permettent à un processus ***BPEL*** de réagir aux événements qui se produisent pendant l'exécution de processus commercial. En d'autres termes, nous ne voulons pas le processus métier d'attendre pour l'événement (et ne font rien d'autre mais attendre). Au lieu de cela, le processus doit exécuter et encore écouter les événements et les gérer chaque fois qu'ils se produisent. En cas d'événements correspondants, les gestionnaires d'événements sont appelés en même temps que le processus métier. L'usage typique des gestionnaires d'événements est de gérer un message d'annulation par le client.

3. **Gestionnaire de compensation :** Compensation, ou annulations des étapes dans le processus métier qui ont déjà réalisé avec succès, est l'un des concepts les plus importants dans les processus métier. L'objectif de la compensation est de renverser les effets des activités précédentes qui ont été menées dans le cadre d'un processus métier qui est abandonné. La compensation est liée à la nature de la plupart des processus métiers, qui ont un temps d'exécution long. Les processus métiers sont souvent sensibles en termes de réussite parce que les données qu'ils manipulent est sensible.

La compensation diffère de la gestion des fautes. Dans la gestion des fautes, un processus métier tente de récupérer une activité qui ne pourrait pas terminer normalement en raison d'une situation exceptionnelle s'est produite. D'autre part, L'objectif de la compensation est d'inverser les effets d'une activité précédente ou un ensemble d'activités qui ont été menées avec succès dans le cadre d'un processus métier qui est abandonnée.

Les Gestionnaires de compensation peuvent être appelées uniquement après que l'activité qui doit être compensée a terminé normalement. Si nous essayons de compenser une activité qui s'est terminée anormalement, rien ne se passera car une activité <empty>sera appelée.

**e) Scopes**

***Scopes*** fournissent un moyen de diviser un processus métier complexe en des pièces hiérarchiquement organisées. Les ***Scopes*** fournissent des contextes de comportement pour les activités. Ils nous permettent de définir différents gestionnaires de fautes pour différentes activités (ou ensembles d'activités). En plus de gestionnaires de fautes, les ***scopes*** sont aussi un moyen de déclarer des variables qui sont visibles uniquement dans le cadre. Les ***Scopes*** nous permettent également de définir des ensembles locaux de corrélation, les gestionnaires de compensation et des gestionnaires d'événements.

Chaque **Scope** a une activité principale. Ceci est similaire à la structure globale du processus où nous avons dit qu'un processus ***BPEL*** a également une activité primaire. L'activité primaire, souvent un *<sequence>* ou *<flow>*, définit le comportement d'un champ pour l'exécution normale. Des gestionnaires de fautes et d'autres gestionnaires définissent le comportement des scénarios d'exécution anormale.

### V.4. Liens de composition

Nous allons étudier les liens de composition du point de vue du flux de contrôle. Nous présentons ainsi dans cette section ce formalisme pour générer automatiquement le code ***BPEL*** au format ***XML***. Pour chaque lien, nous présentons une définition, une représentation graphique et l'expression en ***BPEL*** correspondante.

**a) Séquence :** Dans une séquence, un **SW** est disponible une fois que les services précédents aient terminés leur exécution.


**Figure 3.10 : Schématisation d'une séquence**

En ***BPEL***, l'activité appelée « *sequence* » permet de définir qu'un ensemble d'activités ***BPEL*** sera exécuté séquentiellement.

***b) Parallélisme :*** Le parallélisme est un lien de contrôle se divisant en plusieurs liens s'exécutant en parallèle.

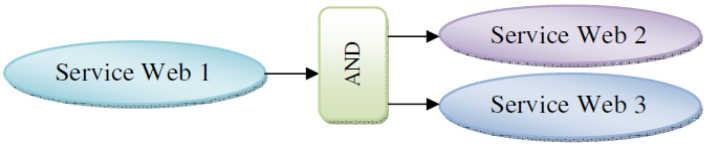



**Figure 3.11 : Schématisation d'un branchement parallèle.**

Pour la représentation de ce lien, ***BPEL*** contient un type d'activité nommé « *flow* ». Ce type d'activité permet de spécifier les activités qui seront exécutées de façon concurrente. L'extrait de code suivant illustre un exemple de ce lien en formalisme ***BPEL*** :

```
<flow>
        <invoke name="invokeWS2"
                        ...>
        </invoke>
        <invoke name="invokeWS3"
                        ...>
        </invoke>
</flow>
```

**L'extrait de code : le parallélisme en *BPEL***

***c) Synchronisation :*** La synchronisation est un point dans le procédé où plusieurs flux de contrôle convergent et fusionnent dans un seul flux de contrôle synchronisant tous ces liens.

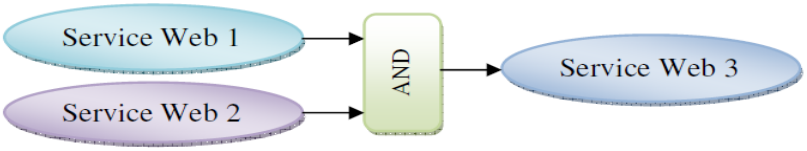



**Figure 3.12: Schématisation de la synchronisation.**

***BPEL*** réalise ce branchement avec le même type d'activité présenté précédemment « *flow* ». Cette dernière permet de fusionner les liens pour continuer l'exécution dans un seul flux de procédé.

***d) Choix exclusif :*** Le choix exclusif est un point dans le procédé où un chemin est choisi entre plusieurs. Ce choix est fait à l'aide d'une décision prise au moment de l'exécution. Cette décision est basée sur une information ou une donnée du procédé. Contrairement au Parallélisme, un seul lien dans le flux de contrôle est activé.

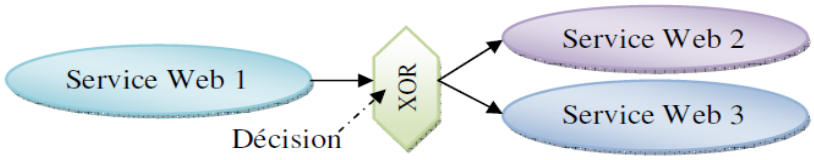



**Figure 3.13 : Schématisation du branchement conditionnel.**

***BPEL*** utilise le type d'activité « *switch* » pour permettre de réaliser le choix exclusif. Chaque lien inclus dans ce choix exclusif est contenu dans un élément « *case* ». Au moment de l'exécution du « *switch* », chaque case contient la séquence d'activités qui sera exécutée si la condition liée au « *case* » est vraie. Le « *switch* » utilise également un autre élément : le « *otherwise* ». Cet élément est optionnel, il représente un lien qui sera choisi par défaut si aucun « *case* » ne contient une condition vraie. Du fait que cet élément est optionnel, s'il n'existe pas explicitement dans la définition du procédé, le moteur générera un lien « *otherwise* » avec un type d'activité « *empty* ».

***e) Fusion simple :*** La fusion simple est un point dans le procédé dans lequel une ou plusieurs branches du flux de contrôle se joignent sans nécessité de synchronisation.

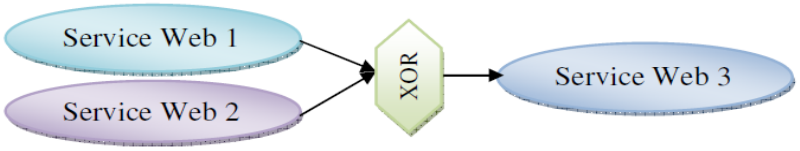



**Figure 3.14 : Schématisation de la fusion simple.**

Au moment d'arriver à la terminaison de l'exécution du ***SW1*** ou du ***SW2*** ou des deux, le procédé continue avec l'exécution du ***SW3***. Ce lien est représenté dans ***BPEL*** par la fin d'une activité de type « *switch* ». Pour compléter son exécution, ce type

d'activité attend la terminaison de la branche activée (selon l'élément « *case* » sélectionné) parmi les branches qui y sont connectées.

***f) Choix multiple :*** Le choix multiple est un point dans le procédé où plusieurs branches sont sélectionnées et exécutées en parallèle. Ce choix est fait à l'aide d'une décision prise au moment de l'exécution. Cette décision est basée sur une information ou une donnée du procédé.

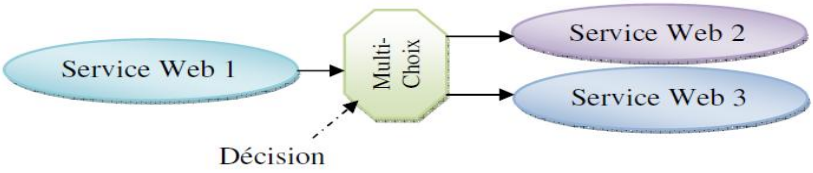



**Figure 3.15 : Schématisation du choix multiple.**

Après la terminaison de l'exécution du SW 1, plusieurs services seront exécutés en parallèle. Pour représenter ce lien, ***BPEL*** utilise le type d'activité « *flow* », l'activité « *empty* » et le concept du « *link* ».

***g) Choix différé :*** Ce lien représente un point dans le procédé dans lequel une branche est choisie parmi plusieurs. Ce choix est basé sur une information ou une donnée qui n'est pas nécessairement disponibles au moment où ce point du procédé est atteint. La sélection de la branche est alors retardée jusqu'à ce que certains événements arrivent pour donner au procédé l'information requise.

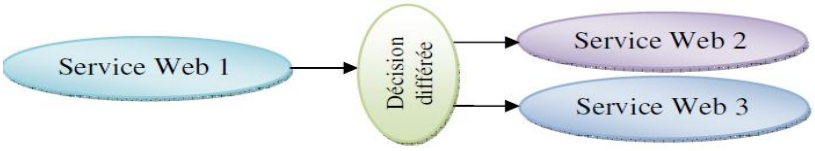



**Figure 3.16: Schématisation du choix différé**

A la fin de l'exécution du SW1, le procédé doit attendre une information ou une donnée qui va lui permettre de choisir entre les deux services 2 et 3.

En ***BPEL***, ce lien est représenté principalement par le type d'activité « *pick* ». Ce type d'activité attend la réception d'un des messages et continue l'exécution en suivant la branche correspondante au message qui a été reçu. Un exemple de la codification ***BPEL*** pour ce lien est présenté dans l'extrait de code suivant :

```
<receive name="receiveWS1"
                    ...
</receive>
<pick>
        <onMessage activation2">
                <invoke name="invokeWS2"
                        ...>
                </invoke>
        </onMessage>

        <onMessage activation3">
                <invoke name="invokeWS3"
                        ...>
                </invoke>
        </onMessage>
</pick>
```

**Extrait de code : Choix différé en *BPEL*.**

### V.5. Les avantages de *BPEL*

L'utilisation de ***BPEL*** offre de nombreux avantages intéressants tels que:

- ✓ ***BPEL*** basé sur ***XML***, il est donc portable sur plusieurs plateformes et fournisseurs ;
- ✓ Soutenir une communication asynchrone fiable, la gestion des terminaux et la persistance de processus ;
- ✓ Offrir la possibilité de gérer des transactions atomiques ainsi que des transactions métiers de longue durée ;
- ✓ Être rapidement prototype, développé et modifié même par des personnes ayant une expérience limitée de développement ;
- ✓ Permettre de modéliser la collaboration des processus métiers à travers <partnerLink> ;
- ✓ Permettre de modéliser le contrôle de l'exécution des processus métiers (grâce à l'utilisation d'un bloc autonome et de langage de transition structuré qui prend en charge la représentation des graphes orientés) ;
- ✓ Représentant des relations de rôle des participants (par <partnerLinkType>).

## VI. SOA et Cloud Computing

Le ***Cloud Computing*** prend toutes les ressources informatiques dans l'internet en tant que services aux utilisateurs, tels que le matériel, l'infrastructure, la plate-forme, le logiciel, le stockage et ainsi de suite. Alors que la ***SOA*** est un modèle de conception de logiciel. ***SOA*** peut intégrer des services bien définis pour une nouvelle solution, **[90].**

La relation entre la ***SOA*** et ***Cloud Computing*** est résumé dans les 4 point essentiels suivants :

1) ***SOA* peut aider à construire l'environnement de Cloud**

   Intégrer et combiner des services de ***Cloud Computing*** par la technologie ***SOA*** permet de former une solution réutilisable pour résoudre les problèmes spécifiques des processus métiers. La solution permet de simplifier l'intégration et la combinaison de problèmes de services ***Cloud*** en la réutilisant et la partageant. Le téléphone mobile intelligent basée sur ***SOA*** peut soutenir les services ***Cloud*** mobiles;

2) ***SOA* et Cloud Computing sont complémentaires**

   Il y a quelques différences entre la ***SOA*** et le ***Cloud Computing***, mais ils sont en fait complémentaires. D'une part, ***SaaS*** fournit des composants de service pour ***SOA***. D'autre part, ***SOA*** peut aider à réaliser rapidement les ***SaaS***. Chacun fournit des fonctionnalités et capacités uniques, et les deux travaillent ensemble pour fournir une solution d'agilité d'entreprise ;

3) **Cloud Computing offre une réalisation pour la *SOA***

   La ***SOA*** est un modèle d'architecture, alors que le ***Cloud Computing*** est un exemple d'architecture, ou une option architecturale. La ***SOA*** est plus globale et stratégique, ce qui signifie qu'il traite l'entreprise complète alors que le **Cloud Computing** est plus tactique et est un moyen de résoudre un problème. Ils sont liés, et il est difficile de faire l'un sans l'autre si vous cherchez à résoudre des problèmes au niveau de l'entreprise ;

4) ***SOA* et Cloud Computing devraient fusionner**

   Le ***Cloud Computing*** et la ***SOA*** devront être fusionnés parce qu'ils marchent mieux ensemble. Le ***Cloud*** et la ***SOA*** sont sur la prestation de service à l'entreprise avec une agilité augmentée, la rapidité et l'efficacité des coûts, cela peut mener à une plus grande innovation et améliore le rendement sur l'investissement.

Il existe deux principales directions dans lesquelles l'influence mutuelle de l'évolution des infrastructures de ***Cloud*** et les architectures orientées services peuvent se traduire par des avantages pour la maturité et la facilité d'utilisation des deux technologies. Tout

d'abord, le déploiement et l'exploitation de service peuvent bénéficier d'une infrastructure de ***Cloud*** capable de fournir, de manière transparente, des ressources de stockage et de calcul élastique et à la demande. D'autre part, les services d'infrastructure de ***Cloud*** sont essentiellement orientée services et donc apte à tire profit des avantages des services, comme la messagerie, la sécurité, la comptabilité, la composition et donc de simplifier leur intégration dans les processus d'affaires, **[43].**

## VII. Conclusion:

Durant la première partie de ce chapitre, nous avons fourni un vue globale de l'architecture orientée service, en décrivant ces éléments, ces différents acteurs, et ces avantages. En suite, nous avons passé à la description des services web et la composition de ces derniers. Pour la composition des services, différents langages de description des Services ont été décrire en concentrant sur le langage d'exécution des processus métier BPEL.

***SOA*** est une architecture dans laquelle toutes les fonctions sont définies comme des services indépendants avec des interfaces invocables qui peuvent être appelées dans des séquences définies pour former des processus d'affaires. ***SOA*** peut fournir un ensemble de solutions pour les compositions de services dans un environnement de ***Cloud Computing***. Mais, Les services de ***Cloud Computing*** sont différents des services traditionnels dans la ***SOA***, qui apporte de nouveaux défis à la technologie ***SOA***. En raison des caractéristiques hétérogènes, autonomes et dynamiques de services de ***Cloud***, diverses incertitudes peuvent affecter l'exactitude, la disponibilité et la fiabilité de la composition de services dans les environnements de ***Cloud Computing***. La composition des Services Web au niveau de ***Cloud Computing*** a des exigences élevées de sûreté de fonctionnement qui font appel à des mécanismes de tolérance aux fautes dédiés.

# Chapitre IV

# Tolérance aux fautes

*« Personne n'ignore par expérience que le danger inconnu est mille fois plus saisissant et plus terrible que le péril visible et matérialisé. »*

*Citations de Alexandre Dumas père*

# I. Introduction

Le ***Cloud Computing*** est un nouveau paradigme pour fournir les ressources matérielles et logicielles sous forme des services à la demande. La complexité croissante et la dynamique induite par le ***Cloud Computing*** posent de nouveaux défis et opportunités pour offrir une fiabilité et résilience de service. Par conséquence, La disponibilité et la confidentialité sont de sérieux défis pour les applications hébergées sur l'infrastructure ***Cloud***.

La ***tolérance aux fautes***, la fiabilité et la résilience dans le ***Cloud*** sont d'une importance primordiale pour assurer un fonctionnement continu et des résultats corrects, même en présence d'une quantité maximale des composants défectueux. La ***tolérance aux fautes*** (parfois nommée « ***tolérance aux pannes*** », en anglais « ***fault tolerance*** ») est la capacité d'un système à fonctionner malgré une défaillance d'une de ses composantes. Idéalement, dans le cas d'une panne, les ressources fautives devront pouvoir être « extractibles à chaud » (en anglais « hot swappable »), c'est-à-dire pouvoir être extraits puis, par exemple, remplacés sans interruption de service.

# II. La sûreté de fonctionnement:

La sûreté de fonctionnement (***Dependability*** en anglais) s'est largement développée, depuis une quarantaine d'année, dans de nombreux domaines industriels pour lesquels il est indispensable, pour des raisons sécuritaires (espace, avionique, centrale nucléaire, etc.) et/ou économiques (industrie chimique, pétrolière, etc.), pour assurer le fonctionnement le plus sûr et le plus optimal possible du système contrôlé même à la présence des évènements perturbateurs.

## II. 1. Définition

La sûreté de fonctionnement est la propriété qui permet aux utilisateurs d'un système de placer une confiance justifiée dans le service qu'il leur délivre, **[8].** La sûreté de fonctionnement cherche donc à éviter les défaillances, ou au moins à prévenir les plus catastrophiques. Ainsi, on préférera faire arriver les voyageurs d'un train en retard (ce qui est en soi une « défaillance » du système), plutôt que de leur faire risquer une collision mortelle (qui est aussi une défaillance, mais catastrophique), [**23, 50**].

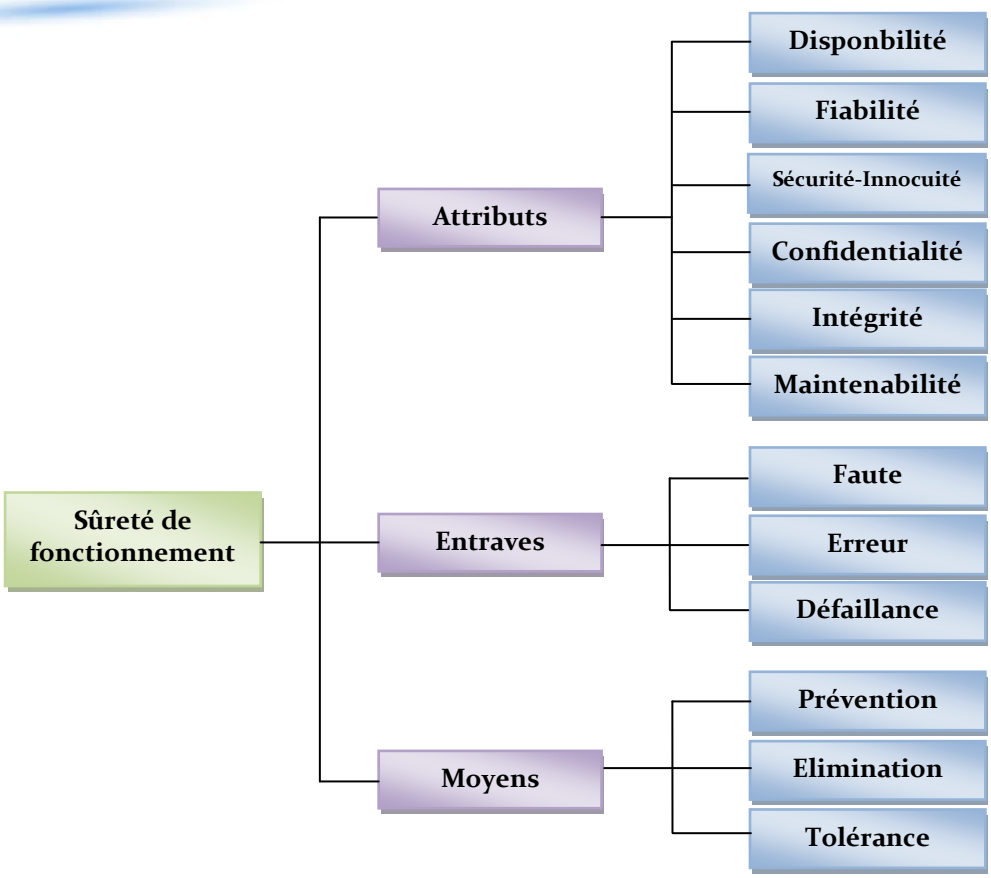



**Figure 4.1 : l'arbre de sûreté de fonctionnement**

La sécurité n'a pas été caractérisée comme un seul attribut de la sûreté de fonctionnement. Ceci est en accord avec les définitions habituelles de sécurité, qui la considèrent comme une notion composite, à savoir, « la combinaison de la confidentialité, la prévention de la divulgation non autorisée de l'information, l'intégrité, la prévention de la modification non autorisée ou de la suppression de l'information et de la disponibilité, la prévention de la retenue non autorisée des informations », [7].

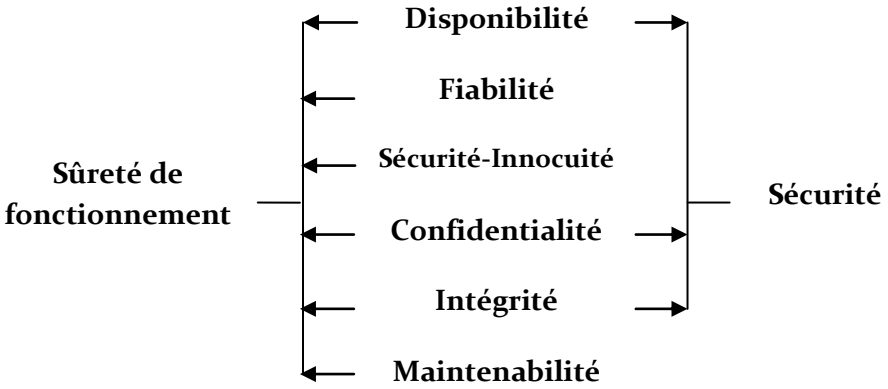



**Figure 4.2 : La relation entre la sécurité et la sûreté de fonctionnement, [7]**

## II.2. Les attributs de la sûreté de fonctionnement

Les attributs de la sureté de fonctionnement correspondent aux propriétés que doit vérifier un système. Ces attributs permettent d'évaluer la qualité de service fournie par le système [91].Les six attributs les plus habituellement considérés, [58], sont :

1. **Fiabilité** (*Reliability*) : *La fiabilité **R (t)** d'un système à l'instant **t** est la probabilité pour que le système fonctionne sans défaillance dans l'intervalle **[0, t]**, étant donné que le système fonctionne correctement au temps 0,* [19].

   La fiabilité est capacité d'un système à accomplir une tâche dans des conditions défavorables, pendant une durée déterminée et dans une plage spécifiée. Cela concerne donc la continuité du service assuré par ce système, [8] ;

2. **Disponibilité** (*Availability*) : *La disponibilité **A (t)** d'un système à l'instant **t** est la probabilité pour que le système fonctionne correctement, à l'instant **t,*** [19].

   Relativement peu de systèmes sont conçus pour fonctionner en continu sans interruption et sans maintenance de toute nature. Dans de nombreux cas, nous nous intéressons non seulement à la probabilité de défaillance, mais aussi au nombre d'échecs et, en particulier, dans le temps requis pour effectuer des réparations. Pour ces applications, l'attribut qui nous tenons à maximiser est la fraction du temps que le système est dans l'état de fonctionnement, exprimé par la disponibilité, [19]. Donc, la disponibilité représente le temps de fonctionnement efficace et sans problème du système, [8];

3. **Sécurité ou sécurité-innocuité** (*safety*) : absence de conséquences catastrophiques engendrées par les fautes sur les personnes, l'environnement ou les autres équipements en interaction avec le système, [8].

   La sécurité peut être considérée comme une extension de la fiabilité, à savoir la fiabilité par rapport à des défaillances qui peuvent créer des risques pour la sécurité. Du point de vue de la fiabilité, toutes les défaillances sont égales. Pour des raisons de sécurité, les échecs sont divisés en des échecs sécurisés (fail-safe) et des échecs dangereux (fail-dangerous). A titre d'exemple, considérons un système d'alarme. L'alarme peut soit ne pas fonctionner correctement, même si un danger existe, ou il peut donner une fausse alarme lorsque aucun danger est présent. L'ancien est classé

comme un échec fail-dangereux. Ce dernier est considéré comme un un fail-safe. Plus formellement, la sécurité est définie comme suit, [8].

*La Sécurité* ***S (t)*** *d'un système à l'instant* ***t*** *est la probabilité que le système soit exerce sa fonction correctement soit se désiste de son fonctionnement dans le cas d'un échec sécurisé dans l'intervalle* ***[o, T],*** *étant donné que le système fonctionnait correctement au moment* ***o,*** **[19].**

4. **Maintenabilité** (*maintainability*) : correspond à l'aptitude du système aux réparations et aux évolutions, [91];
5. **Intégrité** (*integrity*) : absence d'altérations inappropriées de l'information [91] ;
6. **Confidentialité** (*confidentiality*) : correspond à l'absence de divulgation non autorisée de l'information, [91].

L'importance de chaque attribut peut différer selon l'application et les besoins auxquels le système informatique est destiné. Pour les applications parallèles de longue durée, les principaux attributs seront la fiabilité et la maintenabilité, [91].

### II.3. Les entraves de la sûreté de fonctionnement :

Les entraves à la sûreté de fonctionnement se décomposent en 3 classes, [8]: les fautes (*fault*), les erreurs (*error*), et les défaillances (*failure*).

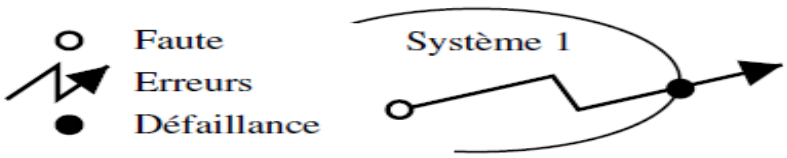



**Figure 4.3 : Faute, erreurs et défaillance, [8]**

1. **Une faute :** est une déviation d'au moins un élément du système ou d'une de ses propriétés caractéristiques. Une faute est la cause adjugée ou supposée d'une erreur : faute active, [8]. Une faute active est soit une faute interne qui était préalablement dormante et qui a été activée par le processus de traitement, soit une faute externe qui a profité d'une vulnérabilité. Une faute interne peut évoluer cycliquement entre états dormant et actif;

   Les fautes peuvent être classées selon différents critères. Chacune de ces classes de fautes nécessite l'introduction de moyens palliatifs appropriés. Par exemple, les

fautes d'incompétence peuvent être évitées par une formation adéquate des utilisateurs, mais aussi une conception ergonomique et des vérifications de la cohérence des ordres des utilisateurs, [**78**].

| Critère de classement | Type de fautes | description |
|---|---|---|
| **Phase de création ou d'occurrence** | **Fautes de développement** | Crées durant :<br>a) Le développement du système, y compris généralement des procédures de conduite ou de maintenance<br>b) La maintenance en vie opérationnelle |
| | **Fautes opérationnelle** | Surviennent durant les périodes de délivrance du service au cours de la vi opérationnelle |
| **Frontières du système** | **Fautes internes** | Localisées à l'intérieur des frontières du système |
| | **Fautes externes** | Localisées à l'extérieur des frontières du système et propageant des erreurs à l'intérieur du système par interaction ou interférence |
| **Cause phénoméno-logique** | **Fautes naturelles** | Dues à des phénomènes naturels, sans intervention humaine directe |
| | **Fautes dues à l'homme** | Résultent d'actions humaines |
| **dimension** | **Fautes matérielles** | Ont leur source, ou se manifestent, dans le matériel |
| | **Fautes logicielles** | Affectent le logiciel, programme ou données |
| **Intention** | **Fautes malveillantes ou malveillances** | Introduites par un humain, avec l'objectif de causer des dommages au système |
| | **Fautes non malveillantes** | Introduites sans objectif malveillant |
| **Capacité** | **Fautes accidentelles** | Crées par inadvertance |
| | **Fautes délibérées** | Résultent d'une décision |
| | **Fautes d'incompétence** | Résultent d'un manque de compétence professionnelle ou de l'inadéquation de l'organisation du développement du système |
| **Persistance** | **Fautes permanentes** | Dans la présence est continue |
| | **Fautes temporaires** | Dont la présence est temporellement bornée |

**Tableau 4.1 : Classes de fautes élémentaires**

2. Une **erreur** est la partie de l'état du système qui est susceptible d'entrainer une défaillance. Une erreur peut disparaître avant d'être détectée, [**91**].
3. Enfin, **une défaillance** est une déviation du service rendu par le système par rapport à son service nominal. Dans un système compositionnel, les fautes considérées sont les défaillances des composants. [**91**] Ces défaillances peuvent être classées en quatre modes :

- ✓ ***La défaillance par crash*** : le composant cesse toute interaction avec les autres composants du système. La notion d'interaction dépend du modèle de système considéré (appel de procédure, envoi de message, écriture dans une mémoire partagée) ;
- ✓ ***La défaillance temporelle*** : le composant interagit avec les autres composants du système en dehors des fenêtres temporelles attendues. Cela concerne aussi bien des interactions qui ont lieu trop tard (échéance manquée) ou trop tôt. La défaillance par crash est une défaillance temporelle, dans le sens où toutes les interactions seront réalisées trop tard (jamais en l'occurrence).
- ✓ ***La défaillance en valeur*** : le composant interagit avec les autres composants du système en utilisant des valeurs incorrectes.
- ✓ ***La défaillance byzantine*** : le composant défaille de manière arbitraire. Ce mode de défaillance regroupe la défaillance par crash, la défaillance temporelle, la défaillance en valeur, et toute combinaison de ces modes de défaillance (valeur erronée trop tard par exemple).

Une faute activée produit une erreur, qui peut se propager dans un composant ou d'un composant à un autre jusqu'à provoquer une défaillance. La défaillance d'un composant cause une faute permanente ou temporaire dans le système qui le contient, tandis que la défaillance d'un système cause une faute permanente ou temporaire pour les systèmes avec lesquels il interagit. Ce processus de propagation est représenté sur la Figure 4.4, [9].

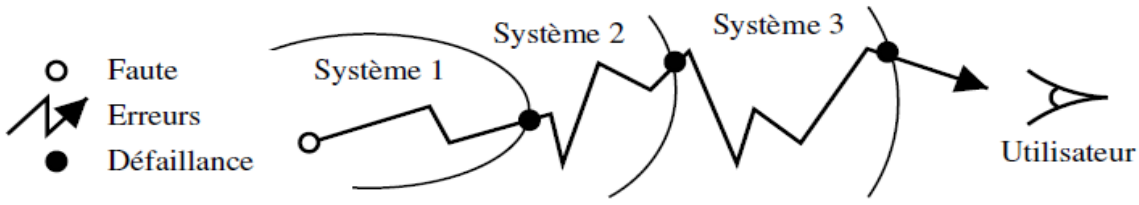



**Figure 4.4 : Récursivité des fautes, erreurs et défaillances, [9]**

## II.4. Les moyens de la sûreté de fonctionnement :

Les moyens d'assurer la sureté de fonctionnement sont définis comme les méthodes utilisées pour assurer cette propriété par combattre les fautes, les erreurs et

les défaillances, si possible en évitant qu'elles se produisent (***prévention***, c'est le premier moyen de la sûreté de fonctionnement), ou en les éliminant (***élimination***, c'est le second moyen de la sûreté de fonctionnement). La prévention et l'élimination des pannes sont des approches qui se basent sur des systèmes de vérification formelle des applications et qui déterminent et minimisent la probabilité de faute, et sur des systèmes matériels qui sont supposés sûrs. Ces deux approches ne sont cependant pas suffisants du fait de l'usure inévitable des systèmes (pour leur partie matérielle), des environnements souvent agressifs dans lesquels ils évoluent, et, particulièrement pour les logiciels, de l'impossibilité de concevoir des systèmes totalement exempts de fautes.

On peut empêcher une erreur de se propager jusqu'à l'utilisateur, on améliore la sûreté de fonctionnement d'un système tout en tolérant ses fautes : c'est le troisième moyen de la sûreté de fonctionnement, ***la tolérance aux fautes***. On peut aussi estimer la présence, le taux futur, et les possibles conséquences des fautes (***Prévision des fautes***).

## III. La tolérance aux fautes

### III.1. Définition :

La capacité d'un système à fonctionner, malgré une défaillance d'un de ses composants, est appelée ***tolérance aux pannes*** (« tolérance aux fautes », en anglais « fault tolerance »).

La tolérance aux fautes est la qualité qui permet d'assurer la délivrance d'un service correct en dépit de fautes affectant les différentes ressources du système. Pour assurer le fonctionnement il faut donc être capable de détecter les fautes et de diagnostiquer précisément leurs origines pour permettre ensuite de délivrer, malgré les fautes, au mieux le service [**8**] ; La tolérance aux fautes est nécessaire car il est pratiquement impossible de construire un système parfait. Le problème fondamental est que, comme la complexité d'un système augmente, sa fiabilité diminue considérablement, sauf des mesures compensatoires sont prises**.**

La tolérance aux pannes vise à atteindre la robustesse et la fiabilité de tout système. Les politiques et les techniques de tolérance de panne peuvent être classés en 2 types: *proactives* et *réactives*. La politique de tolérance aux pannes proactive est pour parer au risque de pannes, d'erreurs et d'échec par les prévoir et remplacer, de manière proactive, le composant suspect c'est à dire détecter le problème avant qu'il vienne réellement. Les politiques de tolérance aux pannes réactives réduisent l'effort d'échecs lorsqu'une panne survient effectivement. Ceux-ci peuvent être classés en deux sous-techniques : traitement d'erreur et traitement des défauts. Le traitement des erreurs vise à éliminer les erreurs de l'état de calcul. Le Traitement des défauts vise à éviter les défauts d'être réactivé.

### III.2. Principe de la tolérance aux fautes

La tolérance aux fautes est mise en œuvre par : la détection d'erreurs qui provoque la levée d'un signal ou d'un message d'erreur à l'intérieur du système, et rétablissement du système, déclenché par le signal ou message, substitue un état jugé exempt d'erreurs et de fautes à l'état erroné détecté,**[78].**

- **Détection d'erreurs :** il permet de détecter la présence d'erreurs dans le système après activation d'une faute, [**78**]. La détection d'erreur peut être réalisée lors d'une suspension de service. On dit alors qu'elle est préemptive. À l'opposé, on dit qu'elle est concomitante lorsqu'elle est réalisée lors de l'exécution normale du service, [**91**]. Les formes les plus utilisées sont les suivantes :
  - ✓ ***La méthode de duplication et comparaison*** utilise au moins deux unités redondantes, qui doivent être indépendantes vis-à-vis des fautes que l'on souhaite tolérer : typiquement, redondance des composants matériels pour les fautes physiques, et diversification pour des fautes de conception.
  - ✓ ***Le contrôle temporel par "chien de garde : watchdog "*** contrôle les temps de réponse ou l'avancée de l'exécution. Il est typiquement utilisé pour détecter la défaillance d'un périphérique en vérifiant que son temps de réponse ne dépasse pas une valeur-seuil, ou pour surveiller périodiquement l'activité d'une unité centrale.
  - ✓ ***Le contrôle de vraisemblance*** ou de données structurées cherche à détecter des erreurs en valeur aberrantes pour le système. Il peut être mis en œuvre soit

par du matériel pour détecter par exemple des adresses mémoires erronées, soit par du logiciel pour vérifier la conformité des entrées, des sorties ou des variables internes du système par rapport à des invariants.

- **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, [9]. Parmi les techniques de traitement d'erreur, on distingue :
  - ✓ ***le reprise d'erreur « Rollback »:*** qui consiste à sauvegarder périodiquement l'état du système pour le ramener dans un état antérieur à la détection d'erreur. Par exemple, on redémarre son logiciel de traitement de texte qui s'est arrêté inopinément et on ouvre le dernier enregistrement de son document avant la défaillance du logiciel. Les principaux inconvénients de la reprise sont la taille des sauvegardes, la difficulté d'effectuer des sauvegardes cohérentes, et le sur-coût temporel nécessaire à leur établissement ;
  - ✓ ***le rétablissement par 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 en l'arrêtant ;
  - ✓ ***la compensation d'erreurs*** : 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 faute responsable de l'état erroné du système en termes de localisation (quels composants sont à l'origine de l'erreur) et de type (transitoire ou permanente), etc.
- ✓ ***l'isolation :*** consiste à exclure la participation du composant erroné de la délivrance du service, par moyen physique ou logiciel. En d'autres termes, la faute devient dormante.

- ✓ ***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 pour tenir compte de la reconfiguration effectuée. Cette mise à jour peut aller jusqu'au redémarrage du système (reset).

Les mécanismes de reconfiguration et de réinitialisation permettent la mise en place de modes dégradés du fonctionnel. Un mode dégradé est un mode dans lequel le système fournit un service partiel à la fois en termes de fonctionnalités (nombre de services diminué), mais aussi en termes de propriétés (implémentation plus simple, moins propice à contenir des fautes, mais fournissant un service de moins bonne qualité), **[78].**

### III.3. Les mécanismes de tolérance aux fautes

Les mécanismes de tolérance aux fautes, **[31],** peuvent être intégrés à différents niveaux dans les systèmes. Ces systèmes peuvent être très différents et les mécanismes de tolérance aux fautes diffèrent également selon les couches où ils sont intégrés ainsi que les techniques d'intégration utilisées. Ces différentes approches peuvent être classées selon la couche du système dans laquelle sont intégrés les mécanismes de tolérance aux fautes :

- ➢ **Des approches de bas niveau :** Les approches système, dans lesquelles les mécanismes sont implémentés dans les couches situées au plus bas niveau. Ces couches accèdent à certaines informations qui ne sont accessibles qu'à l'intérieur du système.
- ➢ **Des approches de niveau intermédiaire (intergiciel) :**
  - ✓ Les librairies, elles se situent entre le système et l'application. C'est au programmeur de les lier à son application. Elles utilisent les services du système pour fournir des briques de base que l'application peut utiliser pour implémenter ses propres mécanismes.
  - ✓ Celles par interception, qui utilisent des mécanismes du système d'exploitation pour intercepter les événements.

- ✓ les approches par intégration, intègrent les mécanismes de tolérance aux fautes au gestionnaire de messages dans la communication entre les systèmes.

➢ **Des approches au niveau applicatif :**

- ✓ les services, qui implémentent certains mécanismes de tolérance aux fautes au-dessus de l'intergiciel, c'est le programmeur qui doit les appeler.
- ✓ et les systèmes réflexifs ou par héritage, qui sont au niveau du langage. Ils usent des mécanismes du langage qui sont la réflexivité et héritage pour relier mécanismes et application.

### III.4. Gestion de la redondance:

Le choix d'un modèle de faute et d'un modèle temporel permet de déterminer le nombre de répliques nécessaires pour fournir un service correct en présence de fautes en fonction du niveau désiré de tolérance aux fautes. La tolérance aux fautes dans un système distribué est assurée par la réplication. L'idée d'intégrer la redondance afin d'améliorer la fiabilité d'un système a été lancé par ***John von Neumann*** dans les années 1950 dans son ouvrage "logique probabiliste et la synthèse d'organismes fiables à partir de composants fiables", **[19].**

Avec plusieurs répliques, une entité répliquée continue à fournir un service à un client même si une ou plusieurs répliques sont défaillantes. Dans un système informatique, la redondance peut être utilisée au niveau du stockage des données, des ressources de calcul, des liens de communication entre client et serveur, ou encore au niveau des composants de l'application elle même. La principale difficulté de cette approche est de conserver une cohérence forte entre les copies, **[50].**

Dans les systèmes distribués, trois modes principaux de réplication sont envisageables: la réplication active, semi-active ou passive. Le choix entre ces différents mécanismes se fait selon les hypothèses de fautes et le domaine d'application, [**91**].

- **Réplication passive (*primary-backup replication*)** : On distingue la copie primaire et les copies secondaires. La copie primaire est la seule qui reçoit les requêtes et qui effectue toutes les opérations. Pour assurer la cohérence, la copie primaire diffuse son

nouvel état aux copies secondaires après chaque modification. Cet état sert de point de reprise en cas de défaillance.

- **Réplication active :** désigne les stratégies dans lesquelles toutes les copies jouent un rôle identique. Toutes les copies reçoivent la même séquence ordonnée de requêtes, qui sont toutes traitées dans le même ordre. Cette stratégie évite d'utiliser des points de reprise couteux. En revanche, elle nécessite un mécanisme de diffusion atomique et requiert que l'exécution des requêtes soit déterministe pour garantir la cohérence.
- **Réplication semi-active :** est une amélioration de la réplication active. À la différence de la réplication active, les copies secondaires attendent une notification de la copie primaire avant de traiter la requête. Cette notification comporte les informations nécessaires qui permettent de résoudre le problème de l'indéterminisme du traitement des requêtes. Au cas où le leader défaille par arrêt, une réplique parmi l'ensemble des secondaires prend la relève. Cette approche est caractérisée par la rapidité du recouvrement et par la non nécessité de la mise à jour de l'état puisque toutes les requêtes sont exécutées parallèlement. En revanche, le déterminisme de toutes les répliques est nécessaire puisque chacune d'entre elles gère la mise à jour de son état interne, **[91].**

## IV. Techniques de tolérance aux pannes existantes dans le *Cloud*

Dans un environnement typique ***Cloud Computing***, les défauts qui apparaissent comme des échecs aux utilisateurs finaux, peuvent être classés en deux types, comme les autres systèmes distribués, [**65**] :

- ***fautes d'accident*** (Crash faults) : qui conduit les composants du système à arrêter complètement le fonctionnement ou restent inactifs pendant les pannes (par exemple, une panne de courant, panne du disque dur);
- ***fautes byzantines*** : qui conduit les composants du système à se comporter de façon arbitraire ou malicieusement durant la panne, provoquant le système à se comporter de façon imprévisible et incorrects.

Il existe différents défauts, **[77],** qui peuvent survenir dans le ***Cloud Computing***. Différentes techniques de tolérance aux pannes sont actuellement répandues dans les nuages qui peuvent être soit au niveau de la tâche ou au niveau de workflow :

- **Check pointing/Restart :** Le système est surveillé en permanence à l'exécution pour valider, vérifier et s'assurer que les spécifications du système correctes sont respectées, **[65].** Lorsqu'une tâche échoue, elle est permise d'être redémarrée à partir de l'état récemment vérifié plutôt que depuis le début. Cette technique est efficace pour les applications qui prennent beaucoup de temps d'exécution ;
- **Job migration :** Des fois, il se trouve que, pour une raison quelconque, un emploi ne peut pas être complètement exécuté sur une machine particulière. Au moment de l'échec d'une tâche, la tâche peut être migrée vers une autre machine ;
- **Réplication :** Les composants critiques du système sont dupliqués en utilisant un matériel, logiciel ou ressources supplémentaires du réseau de telle manière qu'une copie des composants critiques est disponible même après une panne se produite ;
- **Retry** (Réessayer) **:** Dans ce cas, une tâche est implémentée plusieurs fois. C'est la technique la plus simple de réessayer la tâche qui a échoué sur la même ressource ;
- **Task resubmission « Re-soumission des tâches » :** C'est la technique la plus utilisée dans les systèmes de flux de travail (workflow) scientifiques actuels. Chaque fois qu'une tâche échouée est détectée, elle est soumise à nouveau soit à la même ou à une autre ressource pour l'exécuter ;
- **Rescue Workflow :** Cette technique permet le flux de travail de continuer même si la tâche échoue jusqu'à ce qu'il devient impossible d'avancer sans restauration de la tâche qui a échouée ;
- **Preemptive Migration :** Migration préventive repose sur un mécanisme de contrôle de boucle de rétroaction où l'application est constamment surveillée et analysée ;
- **Masquage:** Après l'emploi de la reprise d'erreur, le nouvel état a besoin d'être identifié comme un état transformé. Si ce processus appliqué systématiquement, même en l'absence d'erreur efficace fournissent le masquage d'erreur d'utilisateur ;
- **Reconfiguration:** Dans cette procédure, nous éliminons le composant défectueux du système ;
- **Self-Healing:** Une grosse tâche peut être divisée en plusieurs parties. Cette multiplication est faite pour de meilleures performances. Lorsque plusieurs instances

d'une application sont en cours d'exécution sur différentes machines virtuelles, il gère automatiquement l'échec des instances d'application.

## V. Les modèles basées sur les techniques de tolérance aux fautes

### V.1. AFTRC

« **AFTRC : Adaptive Fault Tolerance in Real Time Cloud Computing** », **[72],** est un modèle de tolérance aux pannes pour le **Cloud Computing** temps réel basé sur le fait qu'un système en temps réel peut profiter de la puissance de calcul et l'environnement virtualisé évolutif du **Cloud Computing** pour mieux mettre en œuvre des applications en temps réel. Dans ce modèle proposé, le système tolère la panne de façon proactive et rend la diction sur la base de la fiabilité des nœuds de traitement. Le mécanisme de base pour atteindre la tolérance de panne est la réplication ou de la redondance. Cette réplication est effectuée sous forme de variantes de logiciels fonctionnant sur de multiples machines virtuelles. Une machine virtuelle est sélectionnée pour le calcul sur la base de la fiabilité, et peut être retiré si elle ne fonctionne pas bien pour des applications en temps réel.

### V.2. LLFT

« **LLFT : Low Latency Fault Tolerance** », **[88],** est un modèle proposé qui contient un middleware de tolérance aux faute de faible latence pour fournir la tolérance aux faute pour les applications distribuées déployées dans l'environnement de **Cloud Computing** comme un service offert par les propriétaires du nuage. Ce modèle est basé sur le fait de garantir que l'application exécutée sur le **Cloud** est sans interruption dans le service fourni à l'utilisateur. Ce middleware reproduit l'application à l'aide de réplication semi-actif ou d'un processus de réplication semi passifs afin de protéger l'application contre divers types de failles.

Le middleware LLFT réplique les processus des applications, en utilisant l'approche de réplication chef / suiveur (the leader/follower replication approch). Les répliques d'un processus constituent un groupe de processus, et les groupes de processus qui fournissent un service pour les utilisateurs constituent un groupe de service. Les nouvelles contributions du middleware LLFT comprennent le protocole de messagerie avec faible

latence (Low Latency Messaging Protocol), le protocole Leader-Determined Membership (the Leader-Determined Membership Protocol) et (the Virtual Determinizer Framework).

### V.3. FTWS

« **FTWS : Fault Tolerance Workflow Scheduling** », **[32],** est un modèle proposé, qui contient un algorithme d'ordonnancement des flux de travail (workflow) tolérant aux fautes pour fournir une tolérance aux fautes à l'aide de la réplication et de la re-soumission des tâches en fonction des priorités des tâches d'un matriciel de l'heuristique (a heuristic metric) qui est calculée en trouvant le compromis entre le facteur de réplication et le facteur de soumission car la réplication seule peut conduire à un gaspillage des ressources et la re-soumission seule peut augmenter le makespan (temps d'exécution total :heuristique). Les tâches sont priorisés en fonction de la criticité de la tâche qui est calculé en fonction des paramètres tels que les degrés (out degree), délais le plus ancien (earliest deadline) et l'impact élevé de re-soumission.

### V.4. FTM

« **FTM** », **[66],** est un modèle proposé pour surmonter la limitation des méthodologies existantes du service à la demande. Pour atteindre la fiabilité et la résilience, ils proposent une perspective novatrice sur la création et la gestion de la tolérance aux pannes. Par cette méthodologie particulière, l'utilisateur peut spécifier et appliquer le niveau désiré de la tolérance aux fautes sans avoir besoin de connaissances sur sa mise en œuvre. L'architecture FTM peut être considérée comme un assemblage de plusieurs composants de services Web, chacun ayant une fonctionnalité spécifique.

Ce modèle est basé sur l'insertion d'une couche de service dédié entre l'infrastructure informatique (**Computing** infrastracture) et les applications qui peuvent offrir un soutien (support) de la tolérance aux pannes à chaque demande individuellement tout en faisant abstraction de la complexité de l'infrastructure sous-jacente.

FTM a été développé pour fonctionner au sommet de l'hyperviseur, visitant tous les nœuds et traversant toutes les couches pour tolérer les défaillances de manière transparente. FTM peut être vu comme un assemblage d'un ensemble de composants services web, chacun ayant une fonctionnalité spécifique. FTM assure une tolérance aux

fautes en répliquant les applications des usagers de manière a ce qu'une copie redondante de l'application soit disponible après qu'une défaillance survient.

### V.5. Candy

« Candy : **Component-based Availability Modeling Framework for Cloud Service Management Using SysML** », **[24],** est un cadre (framework) de modélisation de disponibilité à base de composants, qui construit un modèle global de disponibilité semi-automatique à partir de la spécification du système décrire par un langage de modélisation des systèmes (SysML). Ce modèle est basé sur le fait que l'assurance de la haute disponibilité des services **Cloud** est l'une des principales caractéristiques des services de **Cloud Computing** et aussi l'un des principaux problèmes cruciales et difficiles pour le fournisseur de services **Cloud**.

Les principales contributions de ce modèle peuvent être résumées comme suit. i) SRN sont constitués à des fins d'analyse de la disponibilité des systèmes informatiques complexes. ii) Comportement dynamique des opérations de maintenance du système sont conçus avec des diagrammes d'activité et incorporé dans le modèle de disponibilité. Dans Candy, les diagrammes d'activité sont traduis en SRN et synchronisés avec les SRN connexes pour composer le modèle globale de disponibilité. La méthode de synchronisation proposée n'est pas abordée dans toute la littérature précédente. iii) Notre étude de cas est le premier à construire des modèles analytiques à partir des modèles de spécification de système pour évaluer la disponibilité d'un système d'applications Web hébergées sur une infrastructure de services de **Cloud**.

### V.6. FT-Cloud

« **FT-Cloud**», **[95],** est un framework basé sur les composants de classement et Son architecture est pour construire des applications **Cloud.** FT-**Cloud** emploie la structure d'invocation du composant et de la fréquence pour identifier le composant. Il ya un algorithme pour déterminer automatiquement la tolérance de panne majestueux. Le modèle FT-**Cloud** a été proposé sur la base de l'idée suivante : *en tolérant les défauts d'une petite partie des composants de nuages les plus importants, la fiabilité des applications de* ***Cloud*** *peut être grandement améliorée.* Ce modèle, FT-**Cloud** identifie les composants les plus importants et suggère automatiquement des stratégies optimales de

tolérance aux pannes des composants significatifs. FT-**Cloud** peut être utilisé par les concepteurs d'applications de **Cloud Computing** pour concevoir des applications **Cloud** plus fiables et robustes de manière efficace.

### V.7. Magi-Cube

« **Magi-Cube** », [59], une architecture de stockage de haute fiabilité et faible redondance pour le **Cloud Computing**. La construction du système est sur HDFS et l'utiliser comme un système de stockage de fichiers en lecture / écriture et la gestion des métadonnées. Ils ont aussi construit un script de fichiers et une composante de réparation pour travailler de manière indépendante dans le fond (back ground). Ce modèle repose sur le fait que la haute fiabilité, la performance et le faible coût (l'espace) sont les 3 composantes contradictoires du système de stockage. Pour fournir ces installations à un modèle particulier « Magi-cube » est proposé. Pour réduire la surcharge de l'espace de stockage de fichiers, **Magi-cube** ne conserve qu'un exemplaire pour chaque fichier **HDFS**, et pour atteindre une grande fiabilité, **Magi-cube** utilise un algorithme de codage spécial pour la tolérance aux pannes.

### V.8. Architecture de Behl

L'architecture de Behl, [33], consiste une solution pratique pour l'exécution de processus métiers critiques à base de services web, et particulièrement au sein du **Cloud**, de manière tolérante aux fautes, hautement disponible et hautement configurable. Cette architecture permet d'atteindre cela en utilisant la réplication active des processus métiers ainsi que les services web dans une architecture combinée, en réutilisant les systèmes standards existants et les services de coordination. En fournissant un outil automatisé de transformation, la réplication est réalisée de manière transparente pour les systèmes existants et les flots de traitements (workflow). Les mesures effectuées ont montré que l'architecture proposée permet d'atteindre un temps de réponse meilleur que les systèmes existants et que l'intégration d'un service de coordination impose seulement des coûts modérés tout en simplifiant l'implémentation et conduisant à une solution dynamique adaptable.

Le tableau suivant présent une comparaison entre les différents modèles basés sur la protection contre le type de défaut, et la procédure :

| Modèle | Protection contre le type de défaut | La procédure appliquée pour tolérer la panne |
|---|---|---|
| **AFTRC** | fiabilité | 1. Supprimer nœud en fonction de leur fiabilité<br>2. Reprise en arrière avec l'aide de points de contrôle |
| **LLFT** | Crash-cost, trimming fault | Réplication |
| **FTWS** | Date limite de flux de travail | Réplication et Re-soumission des travaux |
| **FTM** | Fiabilité, disponibilité et service à la demande | Réplication des applications des utilisateurs et en cas d'échec de la réplique utilisation d'un algorithme comme Protocol basé Gossip |
| **CANDY** | Disponibilité | 1. Elle assemble les composants du modèle généré à partir d'IBD et STM selon la notation d'allocation.<br>2. Puis l'activité SNR est synchronisée au système SRN en identifiant la relation entre l'action dans l'activité SNR et la transition de l'Etat dans le système SRN. |
| **VEGA-WARDEN** | Facilité d'utilisation, sécurité, mise à l'échelle | Deux couches d'authentification et de solution technique standard pour l'application. |
| **FT-CLOUD** | Fiabilité, crash et faute de valeur | 1. Le Composant important est déterminé en fonction du classement.<br>2. La Technique pi optimale est déterminée. |
| **MAGI-CUBE** | Performance, fiabilité, coût de stockage faible | 1. la Source de fichier est encodée dans le grand écart pour enregistrer en tant que cluster.<br>2. la Procédure de récupération de fichier est déclenchée quand le fichier d'origine est perdu. |
| **Behl** | Disponibilité, temps de réponse faible | 1. Réplication active des moteurs BPEL<br>2. ***Proxies*** générique<br>3. Service de coordination pour la configuration simplifiée et mise en œuvre<br>4. Apache Zookeeper pour élection de leader, la détection de crash, la configuration dynamique et l'ordre des requêtes |

**Tableau 4.2 : Comparaison entre les différents modèles dans le Cloud basés sur la tolérance aux fautes**

## VI. Mesures de tolérance aux pannes dans le *Cloud Computing*

Les techniques de tolérance aux pannes existantes dans le ***Cloud Computing*** tiennent compte de divers paramètres :

- **Tolérance aux pannes proactive** : la politique de tolérance aux pannes réactive est pour parer au risque de pannes, erreurs et échec en les prévoir et, de manière proactive, remplacer le composant suspecté c'est-à-dire détecter le problème avant il fait venir.
- **Tolérance aux pannes réactive :** Les politiques de tolérance de pannes réactive réduisent l'effort d'échecs lorsqu'une défaillance survient effectivement. Cette technique offre la robustesse à un système.
- **Adaptative** : La procédure faite automatiquement selon la situation**.**
- **Performance:** sert à vérifier l'efficacité du système. Il doit être améliorée à un coût raisonnable par exemple réduire le temps de réponse tout en gardant des délais acceptables.
- **Temps de réponse:** est la quantité de temps nécessaire d'un algorithme particulier pour répondre. Ce paramètre doit être réduit au minimum.
- **Évolutivité:** C'est la capacité d'un algorithme d'exécuter la tolérance aux pannes pour un système avec un nombre fini de nœuds. Cette mesure devrait être améliorée.
- **Débit:** cela sert à calculer le nombre de tâches dont l'exécution est terminée. Il doit être élevé pour améliorer les performances du système.
- **Fiabilité:** Cet aspect vise à donner résultat correct ou acceptable dans un environnement de temps borné.
- **Disponibilité:** La probabilité qu'un élément fonctionnera de manière satisfaisante à un moment donné avec le temps utilisé dans des conditions déterminées.
- **Facilité d'utilisation**
- « **Overhead associated** »: détermine le montant des frais généraux impliqués en mettant en œuvre un algorithme de tolérance aux pannes. Il est composé de surcharge due au mouvement des tâches, inter-processeurs et de la communication inter-processus. Cela devrait être réduit au minimum pour que la technique de tolérance aux pannes puisse travailler efficacement

- **Rentabilité:** Ici, le coût est défini uniquement comme un coût des moniteurs.

## VII. Conclusion

Dans ce chapitre, nous avons, dans un premier temps, fait un tour d'horizon des notions relatives à la sureté de fonctionnement. Puis dans un second temps nous nous sommes focalisés sur les techniques et les mécanismes de tolérance aux fautes. Nous avons ensuite ciblé notre étude à la tolérance aux fautes dans le ***Cloud*** et décrit quelques modèles qui sont réalisés pour tolérer les applications ***Cloud***.

Les principaux avantages de la mise en œuvre de tolérance aux fautes dans le ***Cloud Computing*** comprennent la récupération de l'échec, le coût faible et l'amélioration des performances. Lorsque plusieurs instances d'une application sont en cours d'exécution sur plusieurs machines virtuelles et l'un des serveurs tombe en panne, il est nécessaire de mettre en œuvre des techniques de tolérance aux fautes autonome peut gérer ces types de fautes de manière transparente par rapport à l'utilisateur.

# Chapitre V

# Approche proposée

*«Echouer, c'est tout simplement une occasion de commencer à nouveau, cette fois de façon plus intelligente. »*

*Henry Ford*

## I. Introduction

L'informatisation de notre environnement, et en particulier le ***Cloud Computing***, ouvre des domaines d'application intéressantes mais en même temps elle porte des problèmes de sa contrôlabilité, sa fiabilité et ses performances. Afin d'atteindre ces objectifs, les systèmes devront réagir de façon autonome et sans intervention externe. Des chercheurs dans le domaine informatique tournent leur orientation vers des solutions intelligentes qui sont inspirées (ne signifie pas nécessairement de copier) de la nature, ce qu'on appelle « ***les systèmes bio-inspirés*** ». L'apparition des approches bio-inspirées a très bien contribué dans la mise en œuvre des systèmes complexes.

Beaucoup de systèmes actuels sont conçus et mis en œuvre d'une manière statique. Le ***Cloud Computing*** est un environnement changeant qui apporte la nécessité d'importants changements dans ses systèmes. Ces systèmes sont tolérants aux fautes en introduisant une redondance dans de nombreux composants qui, à son tour, augmente souvent les besoins d'espace et de coût. Une nouvelle approche bio-inspirée, pour traiter ces points, est de concevoir des ***systèmes informatiques organiques***. Il s'agit notamment des propriétés ***d'auto-x***. Ces propriétés sont utilisées pour permettre aux systèmes, eux-mêmes ou leur mode de fonctionnement, de changer si nécessaire.

Pour surmonter les limites des approches traditionnelles de tolérance aux fautes utilisées pour la composition de services ***Cloud***, une approche de sûreté de fonctionnement basée sur l'***Organic Computing*** est proposée. Ce paradigme comporte des propriétés biologiques qui permettent l'adaptation dynamique aux conditions réelles de l'environnement ***Cloud***. Dans cette architecture nous présentons différentes propositions pour la fiabilité des services Web et les machines virtuelles.

## II. Organic Computing

Le calcul algorithmique classique implique une division du travail entre l'homme et la machine. L'infrastructure créative réside principalement dans le domaine humain (zone supérieure) et la machine (zone inférieure) suit aveuglément des commandes. Les deux domaines sont reliés par communication détaillée afin que le programmeur puisse inspecter, comprendre et contrôler le processus dans la machine en détail. En

***informatique organique***, les seules tâches humaines tiennent à la définition des objectifs. Comme la machine est organisée de manière autonome, la communication détaillée entre le programmeur et la machine est limitée à l'algorithme fondamental, qui réalise l'organisation du système, **[10, 69].**

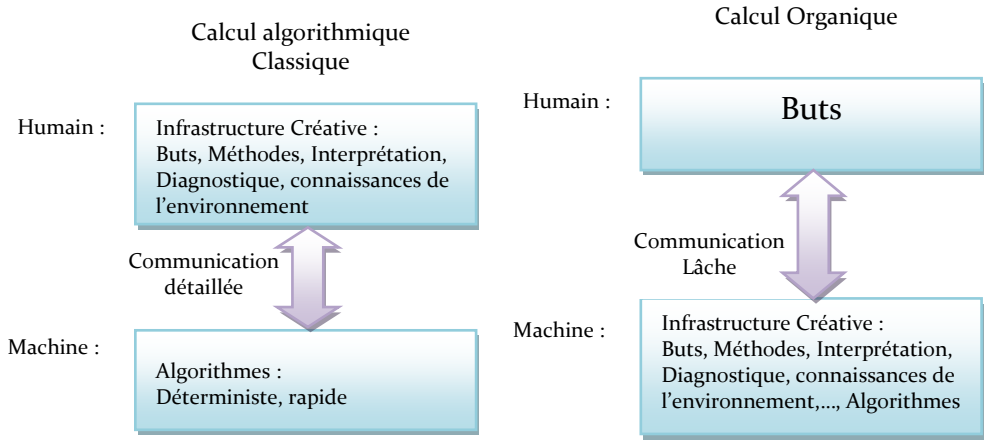



**Figure 5.1 : calcul algorithmique classique et organique**

## II.1. Définition

Un système est appelé « ***organique*** » si tous ces composants et sous-composants sont coordonnés d'une manière utile et réfléchie. Les structures organiques sont des processus hiérarchiquement imbriqués les uns aux autres, structurés de façon à répondre à des changements inattendus des environnements dans les quels ils évoluent à travers des réactions « ***orientés but*** ».

L'***OC*** définit un ordinateur organique (**Organic Computer**) comme un système auto-organisé qui peut s'adapter à un contexte changeant dynamiquement et permet d'obtenir les propriétés ***self-x***. Mais en dépit d'être auto organisé, une caractéristique essentielle des systèmes organiques sera leur capacité à réagir judicieusement aux exigences – en particulier humains – externes et permettre des mesures de contrôle qui pourraient être nécessaires pour maintenir leur comportement au cours des changements environnementaux, **[83].**

L'***Organic Computing*** tiens à profiter de l'une des caractéristiques principales des systèmes biologiques, ils s'adaptent et changent : évoluent, développent et grandissent, et ils les font sans direction ou contrôle externe. L'un des principaux défis pour les systèmes d'***OC*** est de créer des systèmes avec les propriétés biologique, et aussi le faire

d'une manière qui permet de surveiller en permanence, de gérer, et encore plus, de développer ces systèmes, lorsqu'ils sont en opération, **[48].**

**II.2. Les propriétés d'auto X :**

L'***informatique organique*** permet d'avoir que les systèmes soient dynamiques et automatiques et utilisent des mécanismes offrants des degrés différents d'automatisme. Ces mécanismes sont appelés les ***propriétés d'auto-X***, **[42].**

Un système organique doit encapsuler un ensemble de propriétés d'auto-X, y compris les quatre principales propriétés définies par l'***Autonomic Computing*** d'***IBM***, **[10]**: l'auto-configuration (***self-configure***), l'auto-protection (***self-protect***), auto-optimisation (***self-optimize***), l'auto-réparation (***self-healing***).

- **Auto-configuration** : elle couvre le besoin de rendre la configuration et la reconfiguration du système et de ses entités plus dynamique et autonome. Elle est nécessaire quand un nouvel élément de système vient de s'installer dans l'environnement du système. Ce nouveau composant doit se rendre compte des configurations existantes pour adapter son comportement selon les règles ou les politiques en vigueur tandis que le reste des composants du système doit tenir compte de la présence de ce nouvel élément. Donc l'auto-configuration est la capacité de système de s'ajuster dynamiquement pour atteindre le but désiré sans intervention humaine, **[26] ;**
- **Auto-réparation**: la capacité du système de percevoir qu'il ne fonctionne pas correctement. il doit découvrir, diagnostiquer, réparer et remettre les défauts du système quand ils se produisent, et sans intervention humaine. l'auto-réparation comporte des mécanismes permettant de prédire et prévenir les défauts du système qui se produisent en surveillant les paramètres vitaux de système, **[26] ;**
- **Auto-optimisation**: les systèmes doivent continuellement chercher à améliorer leurs performances et opérations. L'auto-optimisation concerne donc l'automatisation de la gestion des performances des systèmes. Ces performances doivent de façon continue être évaluées et optimisées, **[26];**
- **Auto-protection**: L'auto-protection décrit la capacité d'un système d'anticiper et de détecter les attaques malveillantes externes et se protéger en ces d'attaques. Cela

signifie que le système doit être conscient des menaces potentielles et d'être capable de prendre des mesures pour éviter complètement au moins les effets causés par les attaques extérieures, [**26**].

### II.3. architecture Contrôleur /Observateur :

L'architecture générique Observateur/ Contrôleur, (voir figure 5.2), a été proposée pour surmonter la complexité de la technique de conception des systèmes, de sorte que nous laissons un degré de liberté considérable pour leur structure de leur comportement et en leur conférant certaines caractéristiques « **Organic** » qui permettent d'apprendre et de s'adapter dynamiquement aux changements de l'environnement.

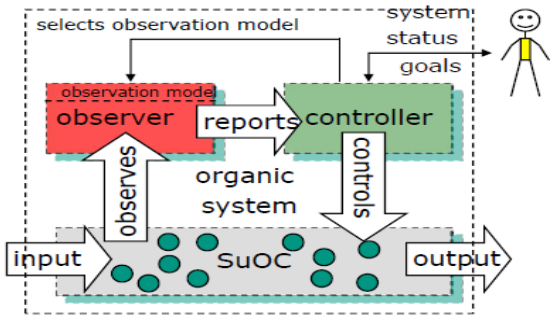



**Figure 5.2 : Architecture générique contrôleur/observateur**

L'observateur/contrôleur a un ensemble de capteurs et d'actionneurs pour mesurer des variables de système et pour commander ce système. L'observateur, le contrôleur et le système sous observation et contrôle (***SuOC***) forme ce qui appelé ***"un système organique".*** Généralement, nous supposons que le système sous l'observation et le contrôle (***SuOC***) se compose d'une grande collection d'objets actifs interactifs possédants certains attributs communs. L'observateur rassemble des données du système (propriétés de niveau micro et macro) et calcule quelques indicateurs "paramètres de situation". Le contrôleur compare les paramètres de situation avec le but définit par l'utilisateur, et décide si l'interposition est exigée, donc quelle action serait la plus appropriée. Pour comparer avec les connaissances attendues des données historiques et actuelles, des règles et des croyances sont supposées, à savoir que ce n'est pas nécessairement une tâche triviale. C'est la tâche de l'observateur de mesurer, de quantifier et de prédire le comportement émergent avec des métriques de base. L'observateur recueille et agrège les

informations sur le ***SuOC***. Les valeurs agrégées (indicateurs de système) sont signalées au contrôleur qui prend les mesures appropriées pour influencer le ***SuOC***, **[83].**

Le contrôleur peut affecte le ***SuOC*** en influençant :

- les règles locales de décision des éléments de ***SuOC***.
- La structure du système comprenant par exemple la communication entre éléments de ***SuOC*** ou le nombre d'éléments.
- L'environnement, qui influencera indirectement sur le système en changeant les données observées par les éléments de ***SuOC*** par leurs capteurs locaux.

**II.4 fonctionnement observateur /contrôleur :**

**1) L'observateur :**

Le procédé d'observation comprend les étapes suivantes : surveillance, prétraitement, analyse de données, prévision et agrégation. Un modèle d'observation adapte l'observateur en choisissant les attributs observables du système, des détecteurs d'analyse, et des méthodes appropriées de prévision, **[83].**

- **Le moniteur** : il prélève les attributs du ***SuOC*** selon un taux d'échantillonnage donné. Les données observées de système se composent de différentes données au niveau des éléments simples, et de quelques attributs globaux de système. Toutes les données mesurées sont stockées dans un dossier de notation pour chaque boucle d'observateur/contrôleur. Ces données stockées peuvent être employées chez le pré-dicteur ou pour le calcul du temps, espace et modèles dans l'analyseur de données.
- **Le prétraitement** : dans cette étape quelques attributs dérivés peuvent être calculés des données brutes. Le prétraitement des données brutes comprend également une sélection des données pertinentes qui sont nécessaires pour calculer les paramètres agrégés de l'ensemble du système. La donnée pré-traitée est transmis à l'analyseur de données et au composant de prédiction.
- **L'analyse de données** : L'analyseur de données applique un ensemble de détecteurs au vecteur de données prétraité. Ceci pourrait être un calcul de regroupement, ou des paramètres d'apparition ou quelques autres méthodes

mathématiques et statiques. A la fin de cette étape une description de l'état actuel du système est fournie ;

- **La prévision** : le prédicteur traite les données venant du pré-processeur et de l'analyseur de données. Il essaye de prévoir un état futur du système. Il peut employer ses propres données combinées avec des méthodes de prévision prises par exemple de l'analyse technique ;
- **L'agrégation** : les résultats de l'analyseur de donnée, du prédicteur, et probablement quelques données non-traitées venant du préprocesseur sont remis à l'assembleur. Ces prétendus paramètres de situation sont transmis au contrôleur.

**2) Le contrôleur :**

Le contrôleur doit guider et contrôler le processus d'auto-organisation basés sur les données reçues de l'observateur. Il s'y mêlera seulement si nécessaire. Il est composé d'un ensemble de module : le sélecteur d'action, Le module d'adaptation et un modèle additionnel de simulation, **[83].**

- **Le sélecteur d'action** : est un module de décision. Il permet de choisir l'action qui est la plus appropriée pour la situation actuelle. Il doit prendre cette décision rapidement pour réagir en temps réel.
- **Le module d'adaptation** : met en application quelques possibilités d'étude (learning) et de planification. L'étude emploie une évaluation des actions mis en application pour améliorer le mapping de la simulation aux actions. L'évaluation est exécutée et basée sur une base de données avec des données d'histoire, laissant attribuer des changements du système aux actions spécifiques ;
- **Un modèle additionnel de simulation :** il est relié au module d'adaptation, qui lui permet de rapprocher les conséquences des actions spécifiques dans les environnements spécifiques simulés avant de les examiner sur le vrai système. Ce module peut également être employé pour concevoir ou projeter des actions complètement nouvelles, par exemple en optimisant des paramètres d'action pour une situation spécifique. L'adaptation vise un comportement du système qui soit plus adéquat avec les buts du système formulés par l'utilisateur courant.

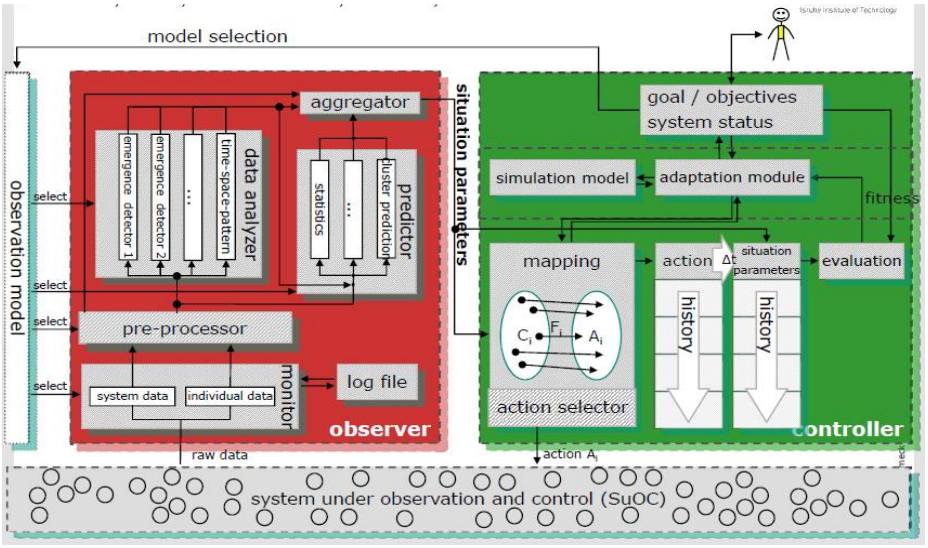



**Figure 5.3: architecture générique détaillée d'observateur/contrôleur**

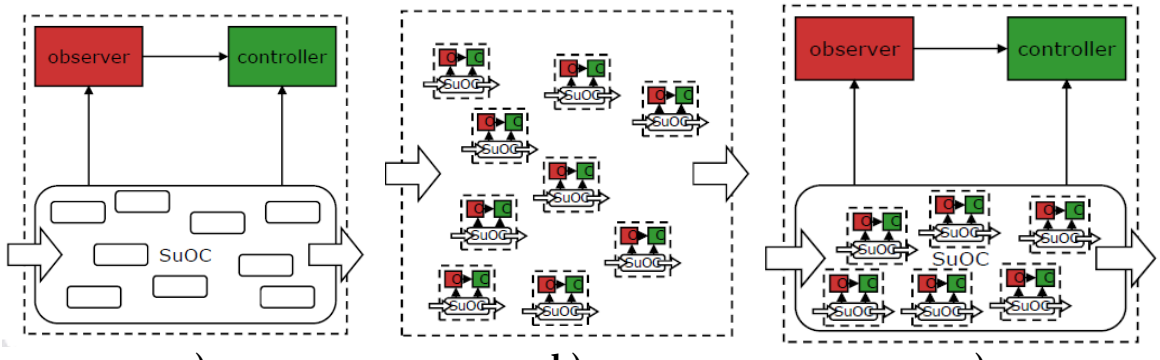
## II.5. Types d'architecture O/C:

L'architecture générique doit être adaptée aux besoins du client à différents scénarios en adaptant les divers composants de l'observateur (modèle y compris d'observation) et du contrôleur, **[25].** L'architecture présentée dans Figure 5.3 peut être réalisé des différentes manières (voir la figure 5.4) :

a) **Architectures centralisée** : un O/C pour l'ensemble du système ;

b) **Architecture distribuée** : un O/C sur chaque élément de système ;

c) **Architecture multi-niveau** : un O/C sur chaque élément de système aussi bien qu'un pour le système technique de totalité.




a) b) c)

**Figure 5.4 : les différentes topologies de l'architecture O/C**

## III. L'architecture proposée

Notre approche est basée sur le travail réalisé dans [28]. L'approche est une architecture de composition des services tolérante aux fautes dans le ***Cloud Computing*** qui a été développée pour fonctionner sous l'***Hyperviseur*** pour garantir que l'application exécutée sur le **Cloud** est ***sans interruption*** dans le service fourni à l'utilisateur. Elle est basée sur le fait que l'assurance de la haute disponibilité des services web est l'une des principales caractéristiques des services de ***Cloud Computing*** et aussi l'un des principaux problèmes cruciaux et difficiles pour le fournisseur de services ***Cloud***. Par conséquent, l'architecture peut être considérée comme un assemblage de plusieurs composants (modules), chacun ayant une fonctionnalité spécifique. Cet assemblage est basé sur l'architecture d'***Observateur/Contrôleur*** de l'***Organic Computing*** afin de tolérer les défaillances de manière transparente. L'architecture d'observateur/contrôleur est responsable de la ***gestion*** et de la (re-) ***configuration*** des composants du système d'une ***manière intelligente***. Cela se fait habituellement par l'observation en continu du système et la prise des mesures de contrôle en cas de besoin.

Notre approche traite quelques problèmes liés aux deux entités essentielles dans le ***Cloud Computing*** qui sont : ***Les services Web*** et les ***machines virtuelles.*** En outre, quatre paramètres de base sont pris en compte: la disponibilité (*availability*) en cours de fonctionnement, temps de réponse (*response time*), l'exactitude des résultats (*correctness of results*), et la fiabilité des VMs (*VMs reliability*) exécutants les compositions.

La tolérance aux fautes est une capacité importante pour les applications ***Cloud***, car elle assure la composition dynamique de services et améliore la fiabilité d'applications ***Cloud***. La détection de fautes est la première étape de la tolérance aux fautes. L'approche proposée se concentre sur cet aspect, et met en avant des mécanismes de détection de fautes pour détecter les services qui ne satisfont pas les exigences de performance lors de l'exécution ainsi que les machines virtuelles qui vont diminuer la fiabilité du système.

Les modules de l'architecture contiennent une série de ***mécanismes de fiabilité*** et sont en mesure de créer une solution de tolérance aux fautes avec des propriétés désirées. Pour y parvenir, nous nous appuyons sur l'idée qu'une solution de tolérance de panne

peut être considérée comme une combinaison d'un ensemble d'activités distinctes coordonnés dans une logique spécifique basé sur les événements. Un module individuel peut être réutilisé sur plusieurs instances de service en réduisant les coûts des ressources.

### III.1. Les différents modules de l'architecture

La figure 5.5 représente l'architecture détaillée de l'approche proposée montrant les différents composants. Dans ce qui suit, nous donnons une brève description de chacun de ces composants :

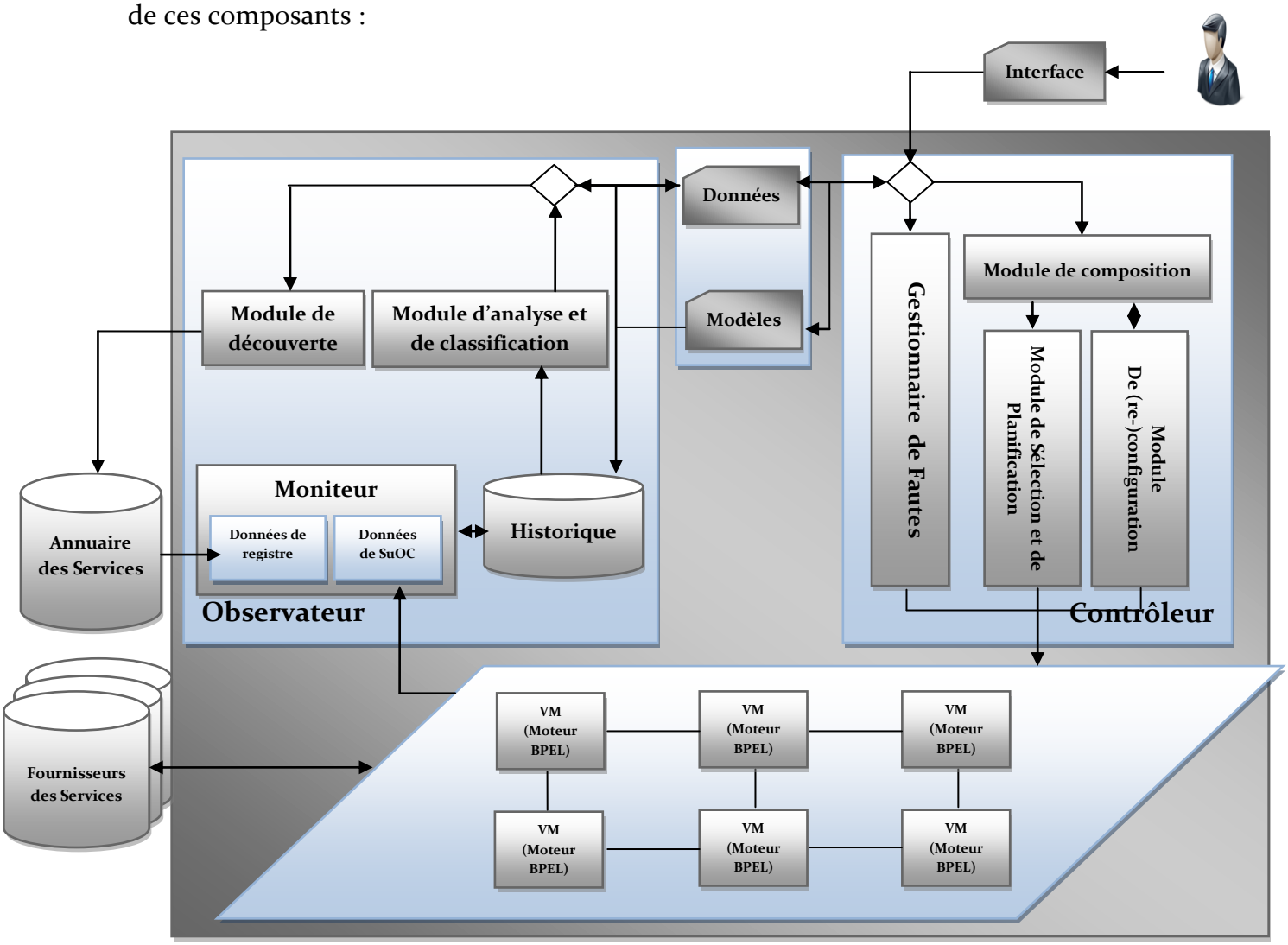



**Figure 5.5 : Schéma illustratif de l'architecture proposée**

1) **Interface client:**

Ce module est considéré comme la première fenêtre du système vers le monde des utilisateurs, il représente la partie visible de l'architecture. Ce module est responsable de la communication entre le système et les différents utilisateurs. Il fournit des outils

à l'utilisateur pour formuler des requêtes, et visualiser les résultats de notre système. L'utilisateur possède à son niveau une interface conviviale, simple et adaptable, qui permet à l'utilisateur de formuler sa requête selon ses besoins pour interroger les processus métiers, et ses préférences en termes de qualité de service. Le module « Interface » assure les fonctionnalités suivantes :

- Identification de l'utilisateur ;
- Transmission des requêtes de l'utilisateur au système pour les traiter après avoir affecter un identifiant unique à chaque requête pour garantir la fiabilité du système;
- Récupération les choix des utilisateurs ;
- Présentation des résultats de notre système d'une manière adaptée aux besoins de l'utilisateur ;

La requête est envoyée au module de composition pour chercher le plan abstrait qui répondent aux besoins d'utilisateur.

**2) Système sous observation et contrôle :**

L'exécution des applications **Cloud** nécessite un ensemble des machines virtuelles : « l'ensemble de ces machines constituent le ***SuOC*** ». Ces machines virtuelles (VMs) exécutent les applications de composition de services (processus métiers). En général, les machines virtuelles sont des répliques les unes des autres, mais elles peuvent être différentes. En outre, comme toute machine physique, la machine virtuelle dans le ***SuOC*** dispose d'un ensemble de ressources (nombre de processeurs, quantité de RAM, le stockage…).

L'un des principes de l'approche proposée est qu'elle tolère les fautes sur la base de la fiabilité de chaque nœud informatique, c'est à dire la machine virtuelle. Une VM est sélectionnée pour le calcul sur la base de sa fiabilité, et peut être retirée si elle ne fonctionne pas bien pour des applications de composition similaires. Par exemple les machines qui ne disposent pas d'assez de ressources pour exécuter des compositions de grande taille seront éliminées lors de l'exécution des prochaines compositions de même type afin de ne pas diminuer la fiabilité du système entier.

### 3) Observateur :

L'objectif de l'observateur consiste à effectuer une agrégation des informations disponibles sur les machines virtuelles et l'exécution des services pour donner une description globale (appelée paramètres de situation) de l'état et la dynamique du système sous-jacent. L'observateur est composé d'un ensemble de modules :

#### 3.1 Moniteur

Le ***SuOC*** est considéré comme un ensemble des machines virtuelles. Le moniteur surveille ses VMs et les services exécutés sur celles-ci. Le moniteur est, à son tour, constitué de :

- **Données de registre :** Les services ont une disponibilité dynamique. Ils peuvent être publiés, modifiés ou retirés du registre à tout moment au niveau du changement de leur fournisseur. Il est important au système d'être au courant des changements dans les services, pour pouvoir par exemple incorporer un nouveau service qui devient disponible, ou bien pour arrêter l'utilisation d'un service qui devient indisponible. Pour faciliter cette tâche le registre va informer le moniteur qui doit les prendre en considération pour que le système s'adapte au nouvel état;
- **Données de *SuOC*** : il contient les données provenant des VMs concernant l'exécution des services par exemple le résultat de l'exécution, l'état d'avancement de chaque VM exécutant la composition et la liste des VMs libres pour permettre au contrôleur de sélectionner celles qui vont exécuter la composition.

  Pour atteindre une allocation efficace et proactive de ressources, et dans le but d'éviter l'approvisionnement durant les défaillances, l'état de fonctionnement des VMs dans le **Cloud** doit être continuellement contrôlé. Le moniteur réalise cette fonctionnalité en sauvegardant les détails concernant le fonctionnement des machines dans ses sous modules**. C**e dernier est responsable de la surveillance des VMs et la détection de fautes. Pour cela, un mécanisme de détection de fautes est utilisé: ***Le contrôle temporel par "chien de garde : watchdog "***. Ce mécanisme contrôle les temps de réponse ou l'avancée de l'exécution de chaque VMs. Il est typiquement utilisé pour détecter la défaillance d'un périphérique en vérifiant que son temps de réponse ne dépasse pas une valeur-seuil, ou pour surveiller

périodiquement l'activité d'un nœud. L'algorithme proposé pour surveiller les nœuds est décrit comme suit (côté Machine virtuelle):

```
(1) Pour chaque VM exécutant la composition i faire
(2)   Cpt ← Q
(3)   Envoyer au moniteur un message contient le temps de départ d'exécution de i
(4)   Démarrer le temporisateur Cpt
(5)   Attendre jusqu'à Cpt=0
(6)   Si Cpt= 0 alors
        Envoyer au moniteur un message qui indique l'activité du VM
        Cpt ← Q
      Fin si
(6)   Retour à (4)
(7) Fin pour
```

Au lieu que le moniteur interroge périodiquement les machines (les 3 répliques), chacun de ces dernières envoie un message au moniteur indiquant le démarrage d'exécution « le temps de départ » lors la réception du plan de composition (***i***) à exécuter, car les machines peuvent ne pas recevoir le plan de composition en même temps à cause de la bande passante et que les machines peuvent ne pas être dans le même emplacement physique. Puis pendant un intervalle régulier ***Q*** (*QUANTUM Q*), elles envoient des messages indiquant la continuité d'exécution.

Si le moniteur ne reçoit pas un message dans un temps Q (Expiration de Temporisateur), il attend pendant une autre période de temps c.-à-d. 2Q car une mauvaise réception du message peut être due à un problème de réseau (figure 5.6). Dans le Cas où le moniteur ne reçoit pas le message dans un temps de 2Q, il déclare que la machine est en panne.

L'algorithme suivant résume les étapes de surveillance du moniteur (côté moniteur):

```
(1)  N = 0
(2)  Cpt(k) ← Q
(3)  Pour k= 1 à 3  faire
(4)      Recevoir le temps de départ d'exécution de la composition i
(5)      Démarrer le temporisateur Cpt (k)
(6)      Attendre jusqu'à Cpt(k) =0
(7)      si Cpt(k) = 0 faire
(8)          Si aucun message n'est reçu faire
(9)              Si N=0 faire
                     N=N+1
                     Cpt (k) ← Q
                     Retour à (5)
                 Fin si
(10)             Si N =1 faire
                     Machine VM est en panne
                     Aller à (13)
                 Fin si
             Fin si
(11)     Cpt(k) ← Q
(12)     Retour à (5)
(13) Fin Pour
```

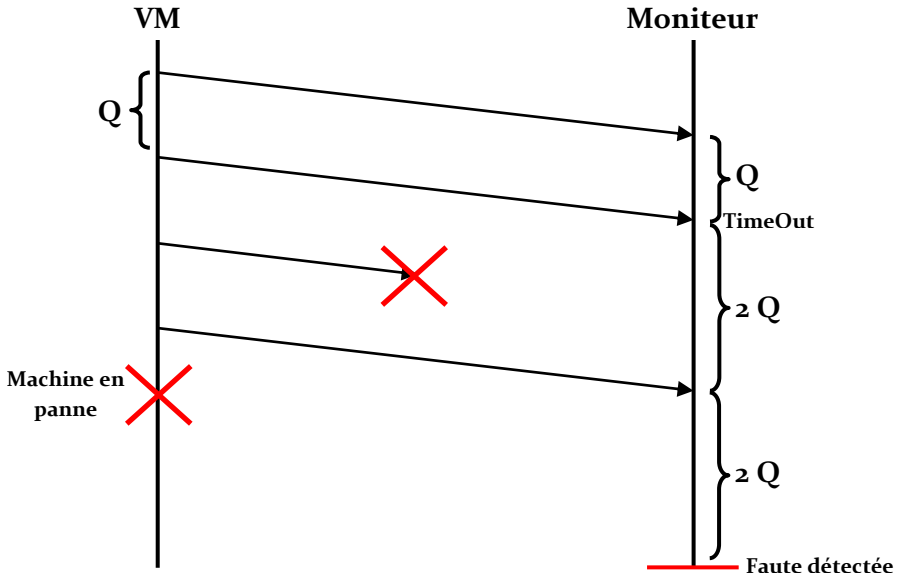



**Figure 5.6 : le chien de garde « Watchdog »**

L'approche proposée consiste à tolérer le système contre l'écrasement des VMs. Pour cela, la technique du ***reprise d'erreur*** « ***Rollback*** » est utilisée. Cette technique consiste à sauvegarder périodiquement (chaque Q' tel que où Q'>Q) l'état de chaque machine. Ces données sont utilisées ultérieurement par l'analyseur dans le cas de panne d'une VM c.-à-d. à la détection d'erreur ;

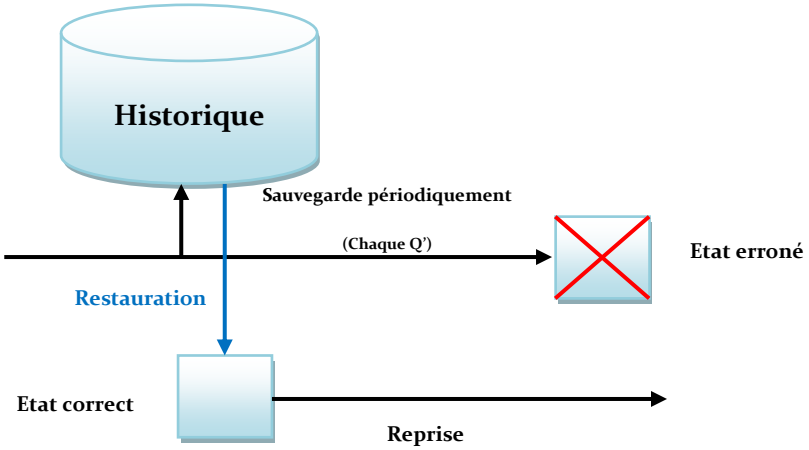



**Figure 5.7 : La reprise d'erreurs**

Chaque machine est caractérisée par VM (*Nom, Id, Etat, DerEtat, TDep, Cpt*) où :

***Nom*** : le nom de la machine VM1, VM2,..., VMn ;

***Id*** : est l'identificateur de la machine dans le réseau ;

***DerEtat*** : est le dernier état sauvegardé de la machine qui comprend le résultat et la dernière instruction exécutée. Cet état est mis à jour continuellement;

***TDep :*** est le temps de départ de l'exécution;

***Cpt*** : est un compteur (temporisateur) initialisé à Q pour permettre de détecter les défaillances des machines lors de son expiration (où Cpt=o).

Et pour ne pas surcharger ce module, les données sauvegardées seront supprimées à la fin de l'exécution de la requête. L'approche proposée est qualifié comme étant auto-conscient (self-awareness) puisque elle est consciente des différentes ressources et de ces états pendant l'exécution des compositions.

**3 .2 Historique :**

Afin de gérer les fautes de manière transparente par rapport à l'utilisateur, les détails des VMs et leur fonctionnement au sein du ***Cloud Computing*** sont stockées dans un fichier journal (log file) ainsi que les plans de toutes les requêtes traitées (les plans de chaque composition). Les données des VMs enregistrées sont utilisées pour réduire le profil de défaillance du système et développer des modèles tolérants aux fautes. Tandis que les plans des compositions sont utilisés pour réduire le temps de réponse c.-à-d. au lieu de replanifier la composition, le module de planification et de sélection cherche dans l'historique s'il y a un plan qui répond à cette requête sinon il effectue la planification dès le début. Pour éviter la surcharge de sauvegarde, les requêtes qui sont rarement utilisables seront supprimées.

**3.3 Le module d'analyse et de classification :**

Il correspond à la phase d'exploitation des valeurs obtenues par le moniteur permettant d'assurer le bon fonctionnement du système et d'analyser et détecter les nouvelles situations. Cette détection produit, dans le cas échéant, des alarmes qui vont déclencher le contrôleur.

Ce module examine les différents changements de l'environnement, ou analyse les différentes notifications qui arrivent au système depuis le moniteur pour vérifier l'influence de la notification sur ce système et prendre la décision si cet évènement doit être envoyé au contrôleur pour que ce dernier le prenne en charge. En outre, ce

module classe les fautes en deux catégories : celles des VMs et celles des Services pour permettre au gestionnaire de fautes de choisir la technique convenable. Les tâches principales de ce module sont :

- Vote majoritaire (Voting) : Après la réception des résultats d'une requête (3 résultats) l'analyseur doit voter pour avoir le bon résultat et le renvoyer au client ;
- Classification des fautes : selon la source de faute (VM/SW) ;
- Production des alarmes qui vont déclencher le contrôleur.

### 3.4 Module de découverte :

Le principe de la découverte est simple : identifier les services qui peuvent répondre à la requête. Pour se faire, il faut identifier un degré de similitude entre les différents concepts qui décrivent le service requis (la requête des clients ***Cloud***) et celles des services offerts (les services publiés par les fournisseurs ***Cloud***). Par exemple pour WSDL, tous les éléments principaux des services doivent être pris en considération dans cette procédure, à savoir : *interface*, *operation*, *input* et *output*. Un tel mécanisme d'évaluation pour la découverte est appelé : *Matching* (Appariement). Dans la littérature, et comme mentionné dans **[93]**, il existe trois catégories d'approches de découverte de services Web: les approches logiques, les approches non-logiques et les approches hybrides. Les approches de découverte qualifiées de non-logiques sont basées sur le calcul du degré de similarité textuelle à partir de graphes structurés construits à cet effet ou encore le calcul de distance (le chemin) entre concepts. Tandis que les approches de découverte logique sont basées essentiellement sur des approches déductives, les approches qui combinent les mécanismes logiques et non-logiques sont qualifiées d'hybrides.

Le module de découverte, selon le plan de composition abstrait, cherche les services dans l'annuaire qui répondent aux besoins des utilisateurs. Le résultat de cette recherche est une liste des SWs participants à la composition.

### 4) Le contrôleur

Le rôle principale du contrôleur est de guider la composition et pour cela il doit générer (ou sélectionner) une réaction adéquate aux indicateurs de la situation présentée envoyée par l'observateur. Il est composé de plusieurs modules :

#### 4.1 Le module de composition :

La requête de l'utilisateur est envoyée à ce module afin d'identifier la composition abstraite appropriée. La composition des processus métiers est faite au préalable « composition statique » par le processus ***BPEL*** qui interagit avec les services partenaires par le biais des interfaces ***WSDL***. La composition abstraite a pour but d'analyser la requête afin de faciliter la découverte.

**Pourquoi *BPEL*? :** Dans [29] une petite analyse des différents langages de composition pour choisir celui qui convenait le plus pour une composition fiable. Les principales raisons de choisir ***BPEL*** sont :

- l'union et l'extension de deux langages XLANG et WSFL ont fait dériver le langage ***BPEL***.
- WSCI permet d'organiser la chorégraphie des messages de services web. Il doit travailler avec un autre langage complémentaire (tel que BPML) qui permettra de décrire les processus métier. Outil complémentaire, BPML permet de décrire les processus métiers génériques en amont de l'enchaînement de messages WSCI. Dans ce cas il faudrait utiliser deux langages pour la conception d'un processus métier. Il est beaucoup plus simple d'utiliser un seul langage qui permet la conception de tout le processus.
- WSCL et WS-CDL sont des langages de chorégraphie. Ils décrivent les interactions entre couples de services web. Par conséquent, WSCI et BPML ne nous conviennent pas.
- Alors que ***BPEL*** a été conçu pour orchestrer des applications publiées sous forme de services web dans un but métier particulier. Nous avons donc opté pour le langage ***BPEL***

### 4.2 Le module de sélection et planification:

Le nombre de services découverts et corresponds à une requête dans un schéma de composition d'un utilisateur peut être relativement important. Alors le module de sélection et planification vient pour surmonter ce problème par proposer les meilleurs services en prenant en considération la demande de l'utilisateur en termes de qualité de service. Un plan de composition est constitué de différents services candidats ayant des niveaux élevés de QoS. Pour chaque service, dans le plan final à exécuter, un autre service similaire est sauvegardé afin de minimiser le temps de réponse en cas où un service ne répond pas et doit être substitué. Ce module assure l'auto-optimisation des plans puisque ces derniers sont toujours changeables afin d'obtenir le meilleur plan avec une bonne qualité de service et dont le but de satisfaire les exigences de l'utilisateur. Le plan est envoyé au module de configuration afin de choisir les machines qui vont l'exécuter.

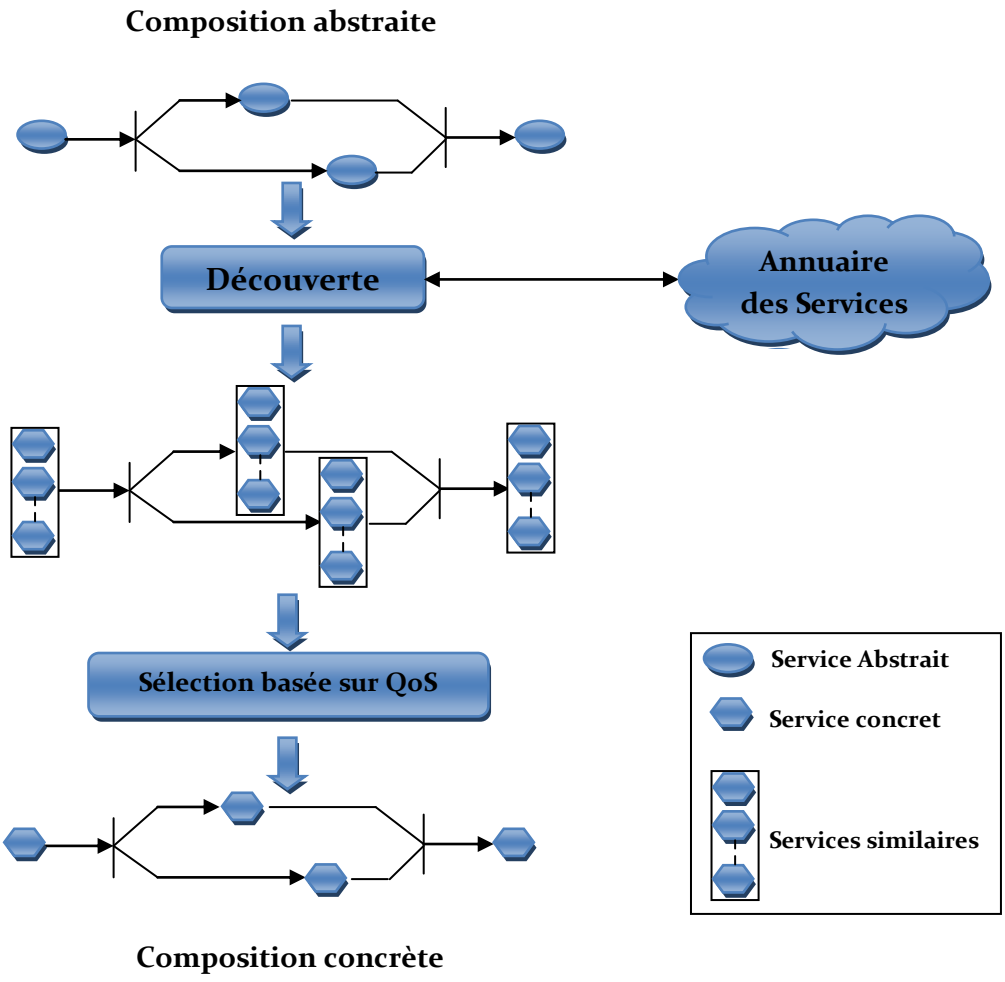



**Figure 5.8: Composition, découverte et planification**

### 4.3 Le gestionnaire de Fautes

Notre approche garantit une disponibilité élevée par la réplication, le recouvrement et le remplacement des VMs défaillantes de manière transparente alors que l'exécution de la composition des SWs continue sans interruption dans une autre instance réplica. Une collection d'algorithmes, qui masquent l'apparition de défaillances et préviennent les fautes à changer en erreurs, est incluse dans ce composant.

Le Moniteur devrait idéalement détecter la faute immédiatement après son occurrence et envoyer une notification à l'analyseur pour qu'il invoque le gestionnaire de fautes. Quand une faute est détectée, le sous-module de Données du ***SuOC*** est également informé pour qu'il mette à jour l'état des ressources du **Cloud** et ensuite envoie une notification à l'historique pour enregistrer la faute et la VM qui engendre cette faute**.**

- **Tolérer les VMs** : Les deux mécanismes principaux utilisés dans l'approche proposée sont la réplication et la Migration des tâches. La réplication des VMs est effectuée dans la phase d'ordonnancement, où la migration est effectuée dans la phase d'exécution.

  - **La réplication :**

    L'approche proposée assure une tolérance aux fautes en répliquant (réplication active) les machines virtuelles en entier exécutant la composition de manière à garantir l'exactitude des résultats après le vote. La réplication active est un protocole général pour la gestion de la réplication qui n'a pas de contrôle centralisé, [**49**]. Pour améliorer les performances de la technique de réplication, la technique de TMR (*Triple Module Replication*) est utilisée, [**63**]**.** Un système TMR est composé de trois unités (les VMs répliquées) fonctionnant en parallèle, chaque VM calcule le résultat indépendamment et l'envoie au moniteur qui, à son tour après la réception des résultats de toutes les répliques, envoie ces résultats au module d'analyse. L'analyseur décidera le résultat final sur la base de l'occurrence maximale de même résultat (*Voting*) et le résultat décidé est renvoyée au client.

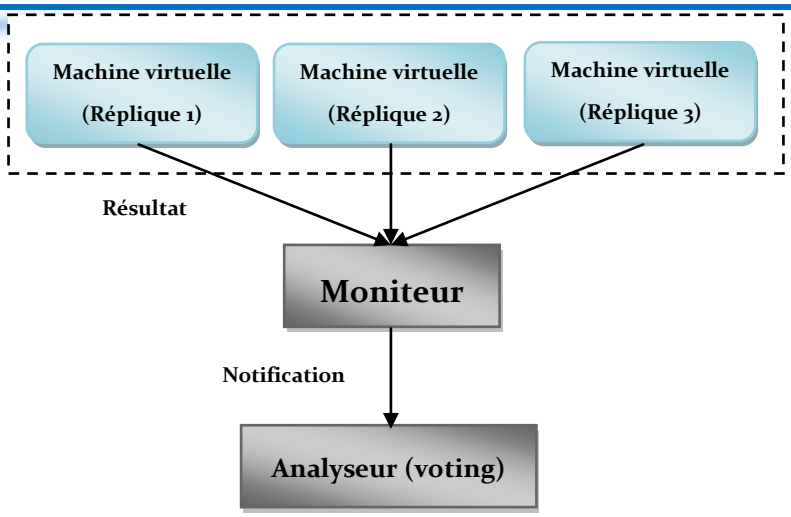



**Figure 5.9: la réplication des VMs en utilisant le TMR**

❖ **La migration des tâches:**

Quand le moniteur détecte une faute au niveau d'une VM, par l'utilisation de la technique de « ***Watchdog*** », il va informer le gestionnaire par l'envoie d'une notification de la machine virtuelle engendrant la faute et son dernier état antérieur à la détection de la faute par la technique de « **Rollback** » :« VM (*Nom, Id, Etat, DerEtat*) » ainsi que la liste des VMs libres et les ressources disposées par chacune. Lors la réception de la notification de panne, le gestionnaire transfère (la migration des tâches) la composition à une autre VM disponible qui va l'exécuter depuis cet état plutôt que depuis le début pour gagner du temps.

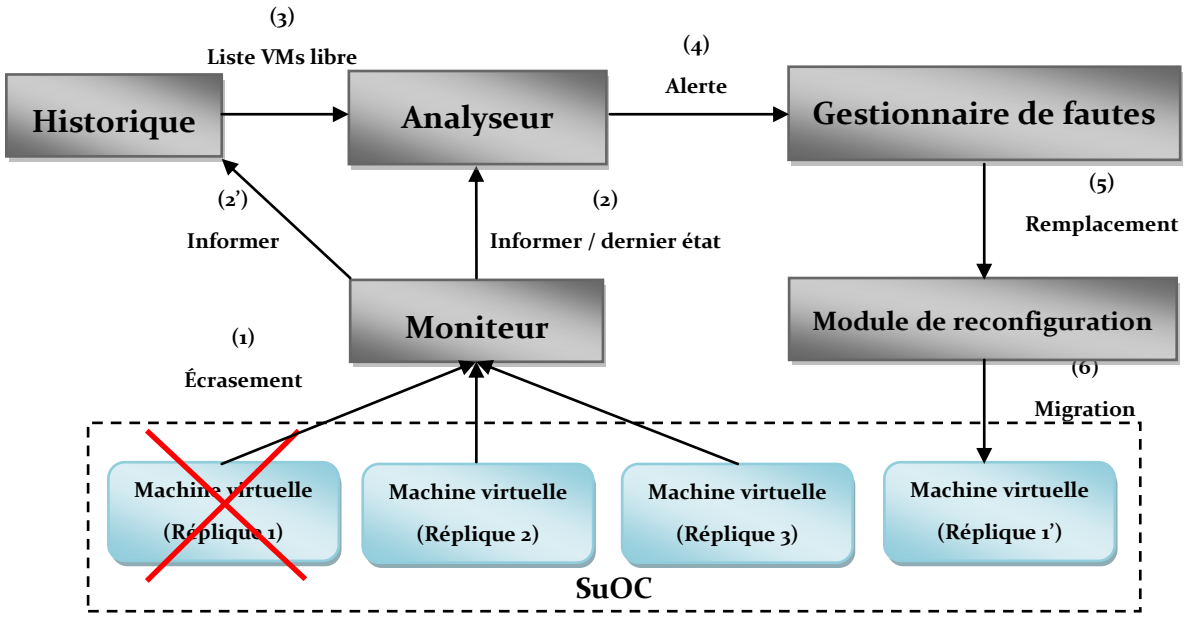



**Figure 5.10 : la migration des tâches**

- **Tolérer les SWs :** Dans l'approche proposée, nous utilisons le langage d'exécution des processus métier ***BPEL*** qui fournit une politique de correction pour permettre aux services de réagir aux évènements inattendus qui mènerait aux fautes. D'ailleurs, dans le cas d'échec, quelques actions devraient être lancées afin de récupérer l'échec tout en répondant aux exigences du système. L'objectif principal est de fournir des mécanismes de réparation afin de garantir le déroulement du processus d'exécution**.**

**L**es principales stratégies, de gestion des exceptions, utilisées dans notre approche sont :

- **Réessayer l'invocation de SW (RetryUntil):** Cette stratégie répète l'exécution d'un service (réessayer l'invocation de service) jusqu'à ce qu'il termine avec succès ou le nombre de tentative d'invocation est terminé. La motivation de cette stratégie est que certains fautes sont transitoires (par exemple, un service Web est inaccessible en raison de problèmes de réseau). La solution est de suspendre l'exécution du processus de composition et de réessayer l'invocation du service indisponible jusqu'à ce qu'il sera disponible. Pour diminuer les ***overheads*** de la communication, un intervalle est défini pour spécifier le temps d'attente entre les tentatives. Cette stratégie est utilisée dans le cas où une seule réplique subit le problème de service échoué. Dans le cas contraire, où toutes les répliques ne peuvent pas atteindre le même service, la deuxième stratégie est appliquée ;
- **Substitution des services web (Alternate) :** c'est une situation plus compliquée. Il y a besoin de la substitution d'un service web par un autre dans les cas suivants :
  1. Pendant l'exécution, le service web échoue ou ne peut pas être atteint ;
  2. Il y a un nouveau service web qui fourni un service mieux, en ce qui concerne les paramètres de qualité de service ;
  3. Une nouvelle version d'un service web choisi, offert par le même fournisseur est disponible.

  Pour effectuer la substitution, dans le cas où un ou plusieurs services sont considérés définitivement indisponibles et pour compléter le processus d'exécution, il faut qu'il existe une compatibilité entre le service défectueux et le nouveau service au niveau de leurs interfaces fonctionnelles (document

WSDL) et leur qualité de service fournie. Pour cela, le moniteur informe le contrôleur, plus précisément le module de sélection, qui dispose la liste des services (découverte antérieurement lors la réception de la requête). Ce module sélectionne parmi les services candidats le service ayant les mêmes fonctionnalités que le service échoué.

Le gestionnaire de fautes assure l'auto-réparation lors de l'exécution d'une composition c.-à-d. que le système peut percevoir qu'il ne fonctionne pas convenablement et d'ajuster, si nécessaire, afin de se restituer à l'état normal, sans l'intervention humaine.

**4.4 Le module de (re-) configuration :**

Le plan de composition et la liste des VMs libres sont envoyés au module de configuration. Ce module sélectionne les machines qui vont exécuter la composition puis transfère le plan de la composition à celles-ci qui sont des moteurs ***BPEL*** (***BPEL*** engine replicas).

La reconfiguration est une entité essentielle dans l'architecture d'O/C, ce module permet au système de survivre dans un environnement dynamique qui présente, d'une part, les changements des propriétés des services web et, d'autre part, les changements des propriétés des VMs ainsi dans le cas de présence d'une faute. Le module de (re-) configuration assure la continuité d'exécution de la composition d'une manière fiable.

Donc notre système est qualifié comme étant auto-adaptatif dont l'adaptation est réalisée de façon interne et qu'elle est déclenché par des changements de l'état de l'application et pas par l'intervention d'utilisateur.

**III.2. Déroulement des actions :**

Afin de montrer le déroulement des actions, dès l'envoie de la requête jusqu'à l'exécution de la composition, et les relations entre les différentes modules, un schéma illustratif est représenté dans la figure 5.11 :

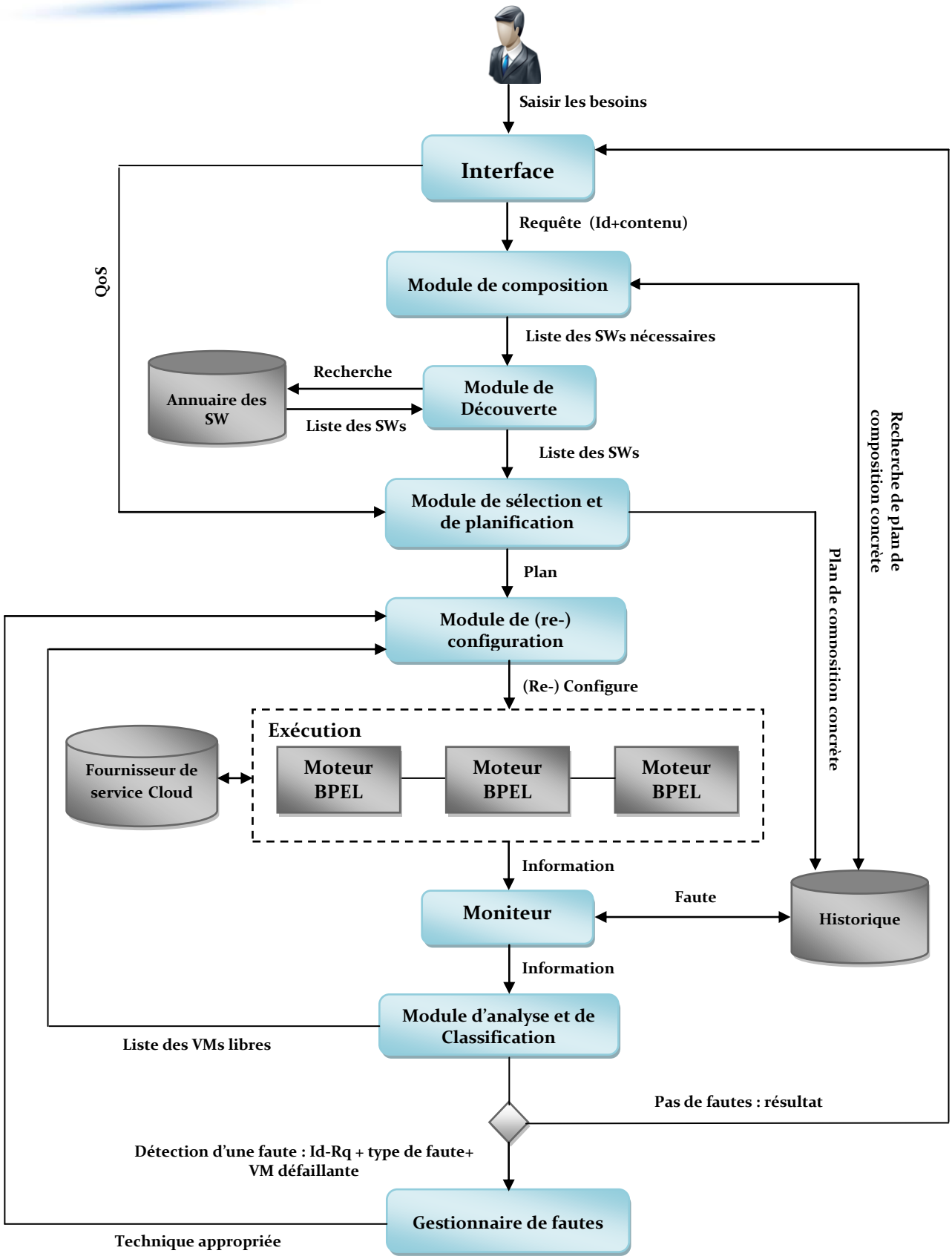



**Figure 5.11: le déroulement des actions entre les différents modules**

Quand le client envoie une requête qui nécessite une composition de SWs, un identifiant unique est affecté à cette requête et est destiné au module de composition pour trouver la meilleur plan abstrait convenable afin de faciliter la découverte et de minimiser le nombre des services découvris. Ce plan abstrait est envoyé au moteur de découverte dans lequel il va chercher la liste des services qui répondent à cette demande à partir du registre chez différents fournisseurs de services. Le registre peut fournir une grande collection de services qui ont les mêmes fonctionnalités mais avec des qualités de services différentes. Cela nécessite une sélection des services qui seront des candidats dans le plan de composition concret, une liste des SWs candidats est élaborée. Cette liste est envoyée au module de sélection et de planification qui élabore le plan de composition qui sera prêt pour l'exécution. Ce plan sera archivé dans l'historique pour une éventuelle réutilisation. Ce plan est envoyé au module de (re-)configuration qui, après la récupération de la liste des VMs libres par le module d'analyse et classification, sélecte les VMs qui vont exécutés le plan et le transmis à celle-ci.

Si l'exécution est bien effectuée, les résultats des différentes VMs seront envoyés à l'analyseur pour décider le résultat final et le renvoyer au client. Sinon, si le moniteur détecte une faute lors la surveillance d'exécution, il informe le module d'analyse et de classification pour identifier le type de faute et qui va, à son tour, informer le contrôleur, plus précisément le gestionnaire de fautes pour qu'il répare immédiatement la faute. Une fois cette réparation est effectuée, une reconfiguration de plan (au niveau de VM ou SW) doit être faite pour ré-exécuter le plan de composition. Dans ce qui suit un diagramme qui montre l'enchainement détaillé des actions :

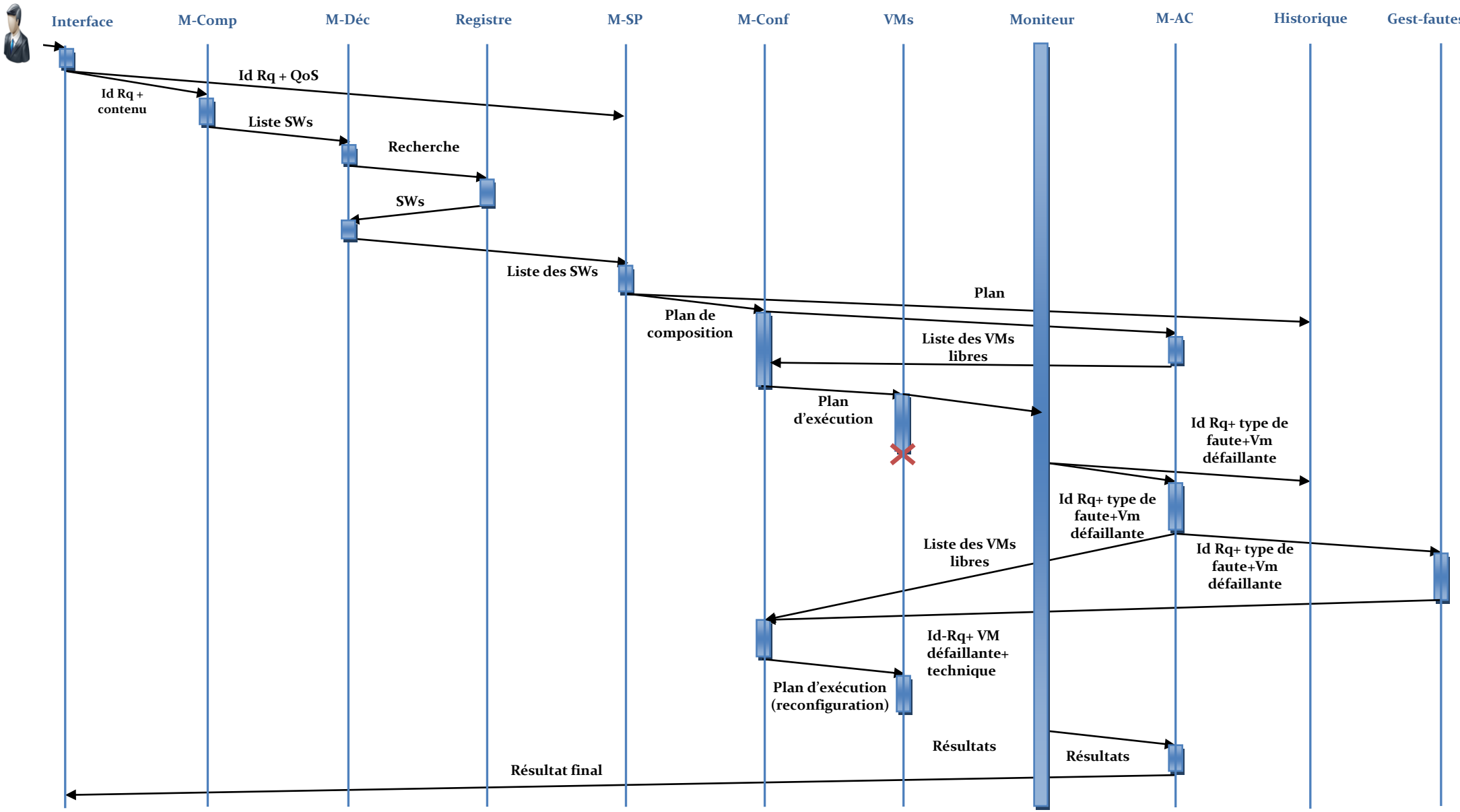



**Figure 5.12 : un diagramme qui montre le déroulement détaillé des actions entre les modules de l'architecture proposée**

## IV. Conclusion

Le ***Cloud Computing*** a été transformé en un modèle important pour la réalisation des applications et des processus métiers à grande échelle. Etant donnée, la situation actuelle de nombreuses vulnérabilités dans les logiciels de production et des dizaines d'attaques malveillantes exploitant ces vulnérabilités, combinée à la complexité croissante des logiciels ainsi que des systèmes, il est peu probable que les ***Clouds*** ne seront pas une cible majeure pour les attaques et les intrusions malveillantes. L'architecture organique permet la sûreté de fonctionnement des processus métiers au niveau de ***Cloud Computing*** assure une disponibilité élevée d'exécution des plans de composition grâce à un ensemble des mécanismes de tolérance aux fautes utilisé. Ces mécanismes suivent une séquence bien définie « détection, reprise et traitement d'erreur » en se basant essentiellement sur l'architecture d'observateur /contrôleur. L'observateur surveille continuellement les changements de l'environnement Cloud afin de détecter immédiatement l'écrasement des VMs ainsi que les services qui empêche la continuité de l'exécution de la composition. Après avoir reçu les informations sur l'ensemble du SuOC, le contrôleur prend des décisions et configure, de manière transparente et sans aucune intervention externe, le système selon les changements.

# Conclusion générale

*« Celui qui argumente et ne conclut pas*
*est comparable à celui qui sème et ne récolte pas. »*
*Michaël Aguilar*

## Conclusion générale

Le ***Cloud Computing*** est devenu l'un des grands paradigmes de l'informatique et propose de fournir les ressources informatiques sous forme de services accessibles au travers de l'Internet. Ces services sont généralement organisés selon trois types ou niveaux. On parle de modèle ***SPI*** pour "***Software***, ***Platform*** et ***Infrastructure***" en anglais. Mais, la recherche de l'interopérabilité est une problématique principale des systèmes distribués et en particulier le ***Cloud Compuitng***. Ce domaine de recherche est favorisé par l'adoption de l'architecture orientée services comme modèle de développement, et particulièrement, par les services web qui combinent les avantages de ce modèle aux langages et technologies développés pour l'internet. ***SOA*** simplifiera la construction, la gestion et la maintenance de systèmes de ***Cloud***, parce que les technologies utilisées pour construire ces systèmes sont souvent normalisées comme ***BPEL,*** qui est un langage d'orchestration qui fournit une infrastructure riche de services web. En outre, ***BPEL*** permet que la création de schémas abstraits et exécutables peut être définie comme un processus métier et s'exécuter dans n'importe quel moteur conforme. La composition de services web, notamment, est considérée comme un point fort qui permet de répondre à des requêtes complexes en combinant les fonctionnalités de plusieurs services au sein d'une même composition.

L'émergence de l'informatique dans les nuages met en avant de nombreux défis qui pourraient limiter l'adoption du paradigme ***Cloud***. Tandis que le nombre de services traités par les applications ***Cloud*** augmente exponentiellement, un défi majeur porte sur la composition et la disponibilité des services.

Après une étude sur les différentes solutions existantes, nous avons découvert que la tolérance aux fautes est la technique la plus importante pour l'exécution fiable de services Web dans le ***Cloud***. Par conséquent, en combinant les techniques de tolérance aux fautes avec WS-BPEL, nous avons proposé une architecture basée sur le paradigme d'***Organic Computing***, pour la composition fiable de services ***Cloud***. Cette architecture est décomposée en un ensemble de composants qui ont des fonctionnalités différentes et qui collaborent entre eux pour répondre à une fonctionnalité globale. D'une part,

l'observateur surveille, d'une manière continue, l'état du système, et d'autre part, le contrôleur effectue des actions selon l'état du système révélé par l'observateur. En conséquence, l'approche garantit les propriétés d'auto-X suivantes :

- L'***auto-optimisation*** puisqu'elle dispose d'un module de sélection qui choisit les meilleurs services qui seront des candidats dans le plan de composition;
- L'***auto-configuration*** par un module de composition qui fournit des plans abstraits de composition selon la requête de client;
- L'***auto-adaptation*** assurée par un module de reconfiguration qui adapte le système aux changements de l'environnement.

Par conséquent, Le système basé sur l'approche ***Organique*** prend des décisions d'une manière autonome et sans une intervention extérieur, en utilisant des politiques de haut niveau. Son état sera constamment vérifié et optimisé. En plus, le système s'adapte automatiquement aux conditions changeantes.

Cette approche se concentre sur la gestion des processus de composition basé sur les propriétés auto-X dans le ***Cloud Computing***. Comme travail futur, nous envisageons de :

- Implémenter l'architecture proposée afin de s'aasurer sa fiabilité ;
- Rajouter d'autres fonctionnalités auto-X comme, par exemple, l'auto-protection et l'auto-guérison qui permettraient au système de se protéger contre les attaques intrusives et de réparer automatiquement les dégâts éventuels ;
- Ajouter d'autres critères de sélection des services Cloud comme les contrats SLAs ;

## *BIBLIOGRAPHIE*

**[1]**: Abdaladhem Albre Shne, Patrik Fuhrer, Jacques Pasquier. Web Services Orchestration and Composition Case Study of Web services Composition. September 2009;

**[2]**: Ajey Singh, Dr. Maneesh Shrivastava. Overview of Attacks on Cloud Computing. *International Journal of Engineering and Innovative Technology (IJEIT)*, Vol1, pages: 321-323, April 2012;

**[3]**: Amin Jula, Elankovan Sundararajan, Zalinda Othman. Cloud computing service composition: A systematic literature review. *Expert Systems with Applications, pages 3809–3824. 2014*. Elsevier*;*

**[4]**: Alexandra Carpen Amarie. BlobSeer as a data-storage facility for Clouds: self-adaptation, integration, evaluation. Université Européenne de Bretagne. Thèse pour l'obtention du grade de Doctorat. Octobre 2011;

**[5]** : Amazon Web Service: http://aws.amazon.com/fr/ ; consultez le: 21/03/2013;

**[6]**: Amazon : http://aws.amazon.com/fr/ec2/; Consulté le : 04/06/2013 ;

**[7]**: Algirdas Avizienis, Jean Claude Laprie, Brian Randell, Carl Landwehr. Basic Concepts and Taxonomy of Dependable and Secure Computing. *IEEE Transactions on Dependable and Secure Computing, vol. 1, pages 11-33. January-March 2004.*IEEE Computer Society;

**[8]**: Bastien Durand. Proposition d'une architecture de contrôle adaptative pour la tolérance aux fautes. *Université de MONTPELLIER*. Thèse pour l'obtention du grade de Doctorat. Juin 2011;

**[9]**: Benjamin Lussier. Tolérance aux fautes dans les systèmes autonomes. *Institut National Polytechnique de TOULOUSE*. Thèse pour l'obtention du grade de Doctorat. Avril 2007 ;

**[10]**: Bing Xie, Juergen Branke, S. Masoud Sadjadi, Daqing Zhang, Xingshi Zhou. Autonomic and Trusted Computing. *Proceeding of 7th International Conference, ATC Xi'an, China*. October 2010. Springer;

**[11]:** Chiraj Modi, Dhiren Patel, Bhavesh Borisaniya, Avi Patel, Muttukrishnan Rajarajan. A survey on security issues and solutions at different layers of Cloud computing. *Springer Science+Business Media New York. 2012*;

**[12]**: Chunming Rong, Son T. Nguyen , Martin Gilje Jaatun. Beyond lightning: A survey on security challenges in cloud computing. *Computers and Electrical Engineering, pages 47-54. 2013*. Elsevier;

**[13]:** Cloud Security Alliance. «Top Threats to Cloud Computing V1.0».March 2010;

**[14]:** Daniel Nurmi, Rich Wolski, Chris Grzegorczyk, Graziano Obertelli, Sunil Soman, Lamia Youseff, Dimitrii Zagorodnov. The Eucalyptus Open-Source Cloud-Computing System. *9th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID), USA, pages 124 – 131. 2009*. IEEE Computer Society;

**[15]:** Définition de la SOA: http://www.commentcamarche.net/contents/1241-soa-architecture-orientee-service. Consulter le : 12/03/2013;

**[16]**: Dimitrios Zissis, Dimitrios Lekkas. «Addressing cloud computing security issues». *Future Generation Computer Systems, pages 583-592* . 2012. Elsevier;

**[17] :** Direction régionale des entreprises, de la concurrence, de la Consommation du travail et de l'emploi. Le Cloud Computing : une nouvelle filière fortement structurante. Septembre 2012 ;

**[18]:** Douglas K. Barry. David Dick. "Web Services, Service-Oriented Architectures, and Cloud Computing". 2013. Elsevier;

**[19]**: Elena Dubrova. Fault-Tolerant Design. *SPRINGER* 2013;

**[20]**: Eucalyptus Project. http://open.eucaplytus.com; Consulté le : 04/06/2013 ;

**[21]** : Fei Teng. « Management des données et ordonnancement des tâches sur architectures distribuées ». École Centrale Paris et Manufactures. Thèse pour l'obtention du grade de Docteur, octobre 2011 ;

**[22]**: Ferda Tartanoglu, Valerie Issarny, Alexander Romanovsky, Nicole Levy. Coordinated Forward Error Recovery for CompositeWeb Services. 2003 ;

**[23]**: François Taïani, Marc-Olivier Killijian, Jean-Charles Fabre. «Intergiciels pour la tolérance aux fautes : Etat de l'art et défis». Synthèse. *Computing Department, Lancaster University Grande Bretagne.* 2006 ;

**[24]**: Fumio Machida, Ermeson Andrade, Dong Seong Kim and Kishor S.Trivedi. Candy: Component-based Availability Modeling Framework for Cloud Service Management Using SysML. *30th International Symposium on Reliable Distributed Systems, pages 209-218.* 2011. IEEE Computer Society;

**[25]:** Hartmut Schmeck, Christian Müller-Schloer, Emre Çakar, Moez Mnif, Urban Richter. Adaptivity and Self-organisation in Organic Computing Systems. *ACM Transactions on Autonomous and Adaptive Systems* (TAAS), *Vol.* 5, pages 5-37. 2011;

**[26]**: Hella Seebach, Frank Ortmeier, Wolfgang Reif. Design and Construction of Organic Computing Systems. *IEEE Congress on Evolutionary Computation, pages 4215-4221.* 2007. IEEE Computer Society;

**[27]**: Ian Foster, Yong Zhao, Ioan Raicu, Shiyong Lu. Cloud Computing and Grid Computing 360-Degree Compared. T*he International Workshop of Grid Computing Environments (GCE), USA, pages 1-10.* November 2008. IEEE Computer Society;

**[28]**: Ichrak Mazzouni, Hibat-Allah Zettal. Une architecture organique pour la composition dynamique des services web . *Université de Constantine 2*. Mémoire pour l'obtention du grade de Master. Juin 2012;

**[29]**: Imen Khalfaoui, Kenza Haliche. «Conception et réalisation d'un système pour la composition de services web, application au e_learning». Mémoire présenté en vue de l'obtention du diplôme d'ingénieur d'état en Informatique. 2007-2008;

[30]: Isabel Del C, M.Vega, M. Vargas-Lombardo. Emerging Threats, Risk and Attacks in Distributed Systems: Cloud Computing. *Innovations and Advances in Computer, Information, Systems Sciences, and Engineering, pages 37-51. 2013.* Springer;

[31] : Jacqueline Konaté. Le traitement des erreurs dans l'exécution des services web composées. *Université Joseph Fourier*. Juin 2006;

[32]: Jayadivya S K, Jaya Nirmala S, Mary Saira Bhanu S. Fault tolerant workflow scheduling based on replication and resubmission of tasks in Cloud Computing. *International Journal on Computer Science and Engineering (IJCSE), Vol. 4, pages 996-1006,* Juin 2012;

[33]: Johannes Behl, Rüdiger Kapitza, Matthias Schunter. Providing Fault-tolerant Execution of Web-service–based Workflows within Clouds. *CloudCP: 2nd International Workshop on Cloud Computing Platforms, Switzerland, 2012.* ACM Digital Library;

[34]: John W. Rittinghouse, James F. Ransome. Cloud Computing Implementation, Management, and Security. CRC Press 2010;

[35]: Jothy Rosenberg, Arthur Mateos. The Cloud at Your Service. 2011;

[36]: Judith Hurwitz, Robin Bloor, Marcia Kaufman, Fern Halper. Cloud computing for dummies. WILEY 2010;

[37]: Kate Keahey, Tim Freeman. Science Clouds: Early Experiences in Cloud Computing for Scientific Applications. *The Conference on Cloud Computing and Its Applications (CCA)*, USA. 2008;

[38]: Lescop Yves. «La sécurité informatique.1. Principes de la sécurité ». 2007 ;

[39]: Luis M. Vaquero, Luis Rodero Merino, Juan Caceres, Mail Lindner. «A break in the clouds: towards a Cloud definition». *SIGCOMM Computer Communication Review, Vol 39, pages 50-55.* Janvier 2009. ACM Digital Library;

[40]: Madhan Kumar Srinivasan, K. Sarukesi, Paul Rodrigues, Sai Manoj, P. Revathy. State-of-the-art Cloud Computing Security Taxonomies-A classification of security challenges in the present cloud computing environment. *International Conference on Advances in Computing, Communications and Informatics, CHENNAI, India*, pages:470-476. *August 2012*.ICM Digital Library;

[41]: Mark Ian Williams. A Quick Start Guide to Cloud Computing: Moving your business into the cloud. 2010;

[42]: Markus Huebscher, Julie McCann. A survey of Autonomic Computing-Degrees, Models, and Applications. 2008. ACM Digital Library;

[43]: Mary Grammatikou, Constantinos Marinos, Yuri. Demchenko, Diego Lopez, Krzysztof Dombek, Jordi Jofre. GEMBus as a Service Oriented Platform for Cloud-Based Composable Services. *The Third IEEE International Conference on Coud Computing Technology and Science, Pages 666 – 671.* 2011. IEEE Computer Society;

[44]: Matt Bishop. Introduction to Computer Security. 2004;

**[45]**: Matthew Hale, Rose Gamble. Risk Propagation of Security SLAs in the Cloud. *First International workshop on Management and Security technologies for Cloud Computing, pages 730-735. 2012.* IEEE Computer Society**;**

**[46] :** Maurice Audin. «Etat de l'art du Cloud Computing et adaptation au Logiciel Libre». 2009;

**[47]**: Micheal MILLER. Cloud Computing: Web-Based Applications That Change the Way You Work and Collaborate Online. 2009;

**[48]**: Mieso K. Denko, Laurence Tianruo Yang, Yan Zhang. Autonomic Computing and Networking. *SPRINGER* 2009;

**[49]**: Mohammad Abdollahi Azgomi, Esmaeil Nourani, A Dependable Web Service Architecture Based on Design Diversity Techniques and WS-BPEL, *Iranian Journal of Electrical and Computer Engineering*, VOL. 11, NO. 1, WINTER-SPRING *2012*;

**[50]**: Mohamed Taha Bennani. Tolérance aux Fautes dans les Systèmes Répartis à base d'Intergiciels Réflexifs Standards. *Institut National des Sciences Appliquées de TOULOUSE*. Thèse pour l'obtention du grade de Doctorat. Juin 2005**;**

**[51]** : Nerea Arenaza. Composition semi-automatique de Services Web. Mémoire pour l'obtention du grade de Master. *Ecole polytechnique Fédérale de Lausanne.* Février 2006 ;

**[52]**: Nimbus Project. http://www.nimbusproject.org/; Consulté le : 04/06/2013 ;

**[53]**: Open Cloud Computing Interface. http://occi-wg.org/, Consulté le : 04/06/2013 ;

**[54]**: Open Grid Forum. http://www.gridforum.org/. Consulté le : 04/06/2013

**[55]**: OpenNebula. http://opennebula.org/; Consulté le : 04/06/2013 ;

[56] **[OPE]**: OpenStack. http://www.openstack.org/projects/compute/, Consulté le : 04/06/2013 ;

**[57]**: Peter Bartalos, Mária Bielikova. Automatic dynamic web service composition: a survey and problem Formalization. *Computing and Informatics, Vol. 30, pages 793–827.* 2011;

**[58]**: Princy Gupta, Sugandha Banga. Review of Cloud Computing in Fault Tolerant Environment With Efficient Energy Consumption. *International Journal of scientific research and management (IJSRM), Vol.1, pages 251-254. 2013;*

**[59]**: Qingqing Feng, Jizhong Han, Yun Gao, Dan Meng. Magicube: High Reliability and Low Redundancy Storage Architecture for Cloud Computing. *7th International Conference on Networking, Architecture, and Storage, pages 89-93.* 2012. IEEE Computer Society;

**[60]**: Rafael Moreno-Vozmediano, Ruben S. Montero, Ignacio M. Llorente. «Elastic management of cluster-based services in the cloud». *The 1st Workshop on Automated control for datacenters and clouds (ACDC)*, USA, pages 19-24. 2009. ACM Digital Library;

**[61]**: Rajkumar Buyya, James Broberg, Aandrjez Goscinski. CLOUD COMPUTING: Principles and Paradigms. *WILEY* 2011;

**[62]**: RamSanthosh Gopala Krishna Guptha. Analysis of Provisioning Dependable Spot Virtual Machines in Cloud Environments. 2011/2012 ;

**[63]**:Ricardo Paharsingh, Olivia Das. An Availability Model of a Virtual TMR System with Applications in Cloud-Cluster Computing. *13th International Symposium on High-Assurance Systems Engineering,pages 261- 268. 2011.* IEEE Computer Society;

**[64]**: Richard Hill, Peter Lake, Laurie Hirsch, Siavash Moshiri. Guide to Cloud Computing: Principles and Practice. *Springer* 2013;

**[65]**: Ravi Jhawar, Vincenzo Piuri. Fault Tolerance and Resilience in Cloud Computing Environments. *Computer and Information Security Handbook*,2013;

**[66] [JHA, 2012]**: Ravi Jhawar, Vincenzo Piuri, Marco Santambrogio. A Comprehensive Conceptual System-Level Approach to Fault Tolerance in Cloud Computing. *IEEE International System Conference, Pages 1-5.* 2012. IEEE Computer Society;

**[67]**: Riadh Ben Halima. Conception, implantation et expérimentation d'une architecture en bus pour l'auto-réparation des applications distribuées à base de services web. Université de TOULOUSE. Thèse pour l'obtention du grade de Doctorat. Mai 2009 ;

**[68]**: Roger Halbheer, Doug Cavit. livre-Blanc : Sécurité et Cloud computing. Janvier 2010;

**[69]**: Rolf P. Wurtz. Organic Computing. SPRINGER 2008;

**[70]**: Ronald L. Krutz, Russell Dean Vines. Cloud Security: A Comprehensive Guide to Secure Cloud Computing. *WILEY* 2010;

**[71]**: Sandeep K.Sood. A combined approach to ensure data security in cloud computing . *Journal of Network and Computer Applications, pages 1831-1838. 2012*. Elsevier;

**[72]**: Sheheryar Malik, Fabrice Huet. Adaptive Fault Tolerance in Real Time Cloud Computing. *World Congress on Services, pages 280-287.* 2011. IEEE Computer Society;

**[73]:** Sheikh Mahbub Habib, Sascha Hauke, Sebastian Ries, Max Muhlhauser. Trust as a facilitator in cloud computing: a survey. *Journal of Cloud Computing: Advances, Systems and Applications, 2012. SPRINGER;*

**[74]**: Slaheddine Maaref. Cloud en Afrique: Situation et perspectives. Avril 2012 ;

**[75]** : Stéphane Gill. Type d'attaques.2003 ;

**[76]**: Surya Nepal, Mukaddim Pathan. Security, Privacy and Trust in Cloud Systems. Springer 2014;

**[77]**: Sweta Patel, Ajay Shanker Singh. « Fault Tolerance Mechanisms and its Implementation in Cloud Computing – A Review». *International Journal of Advanced Research in Computer Science and Software Engineering*. Décembre 2013;

**[78]:** Thomas Pareaud. « Adaptation en ligne de mécanismes de tolérance aux fautes par une approche à composants ouverts ». *Université de TOULOUSE*. Thèse pour l'obtention du grade de Doctorat. Janvier 2009;

**[79]**: Tim Mather, Subra Kumaraswamy, Shahed Latif. Cloud Security And Privacy: An Entreprise Perspective On Risks And Compliance.O'REILLY 2009;

**[80]**: Timothy Grance, Peter Mell. «The NIST definition of cloud computing». *National Institute of Standards and Technology*. Septembre 2011;

**[81]**: Top 20 Platform as a Service Vendors: http://www.clouds360.com/paas.php; Consulté le: 21/03/2013 ;

**[82]** : Ulrich Duvent. Guillaume Ansel. Les Architectures Orientées Services (SOA). Mémoire pour l'obtention de Grade de Master en Informatique. *Université du Littoral Côte d'Opale*. 2009 -2010;

**[83]**: Urban Richter, Moez Mnif, Jurgen Branke, Christian Schloer, Hartmut Schmeck. Towards a generic observer/controller architecture for Organic Computing. 2007;

**[84]**: Vahid Ashktorab, Sayed Reza Taghizadeh. Security Threats and Countermeasures in Cloud Computing. *International Journal of Application or Innovation in Engineering & Management (IJAIEM), Vol 1, pages 234-245*. October 2012;

**[85]**: Valeria Cardellini, Emiliano Casalicchio, Júlio Cezar Estrella , Francisco José Monaco. Performance and Dependability in Service Computing: Concepts, Techniques and Research Directions. 2012;

**[86]**: Vic (J.R.) Winkler, Securing the Cloud: Cloud Computer Security-Techniques and Tactics, *Elsevier* 2011;

**[87]:** Waseem Roshen. SOA-Based Enterprise Integration: A Step-by-Step Guide to Services-Based Application Integration. 2009;

**[88]:** Wenbing Zhao, P.Melliar-Smith, L.Moser. Fault Tolerance Middleware for Cloud Computing. *3rd International Conference on Cloud Computing, pages 67-74*. 2010. IEEE Computer Society;

**[89] :**Wygwam TM : bureau d'expertise technologique. Le Cloud Computing : Réelle révolution ou simple évolution ?. 2010 ;

**[90]**: Xiaolong Yang, Huimin Zhang. Cloud Computing and SOA Convergence Research. *5th International Symposium on Computational Intelligence and Design, Vol 1 pages 330-335*. 2012. IEEE Computer Society;

**[91]**: Xavier Besseron. Tolérance aux fautes et reconfiguration dynamique pour les applications distribuées à grande échelle. *Université de GRENOBLE*. Thèse pour l'obtention du grade de Doctorat. Avril 2010;

**[92]**: Yousri Kouki . Approche dirigée par les contrats de niveaux de service pour la gestion de l'élasticité du "nuage". *Laboratoire d'informatique de Nantes-Atlantique (LINA)*. Mémoire présenté en vue de l'obtention du grade de Docteur. Décembre 2013 ;

**[93]**: Yassine Chabeb. Contributions à la Description et la Découverte de Services Web Sémantiques. Université d'Evry-Val d'Essonne. Thèse présentée pour l'obtention du diplôme de Doctorat. 2011 ;

**[94]:** Zibin Zheng and Michael Lyu. A QoS-Aware Fault Tolerant Middleware for Dependable Service Composition. *International Conference On Dependable Systems And Networs, pages 239-248. 2009*. IEEE Computer Society*;*

**[95]**: Zibin Zheng, Tom Chao Zhou, Michael R. Lyu, Irwin King. FTCloud: A Component Ranking Framework for Fault-Tolerant Cloud Applications. *21st International Symposium on Software Reliability Engineering, pages 398-407.* 2010. IEEE Computer Society;