Architecture technique

CHAPITRE 17

Sens et portée de l'étude	Compétences visées	Notions et contenus
Être capable d'identifier les principales architectures techniques	Caractériser l'architecture technique d'une organisation. Accompagner une démarche de choix et de déploiement d'une architecture technique.	Les principales architectures techniques (client-serveur, médiateur (middleware), transactionnel, intégration, portail)

REMARQUES PRÉALABLES

L'épreuve de MSI laisse une part belle aux réflexions managériales : stratégie SI à adopter pour une organisation, choix tactiques à mettre en œuvre face à une situation donnée, etc.

Même s'*il n'est pas question de conception des SI* (dans le sens de *construction* : modélisation, programmation, etc.), il demeure nécessaire de **maîtriser le vocabulaire technique** et tout ce qu'il englobe (avantages et risques liés à un choix d'infrastructure, par exemple).

I. Quelques concepts clés

1. L'architecture du SI

Qui dit architecture dit construction, plans (représentations simplifiées), ingénieurs, outils… et, bien entendu, architectes !

a. De la comparaison avec l'architecture d'un bâtiment

Lorsqu'un architecte élabore les plans d'un bâtiment, il s'intéresse à ses futures fondations (sa robustesse), sa disposition (qui impacte son côté fonctionnel), et réfléchit (entre autres) aux travaux ultérieurs menés par les différents corps de métiers (plomberie, électricité, etc.).

L'objectif est de participer à la construction d'une œuvre *pérenne*.

La conception du SI répond à des problématiques tout aussi complexes, si ce n'est que, dans l'écrasante majorité des cas, **cette réflexion entre en jeu alors que le SI est d'ores et déjà existant** (!).

b. Construction ou rénovation ?

On est donc plus proche, en termes de démarche, d'un chantier de rénovation que de la construction d'une bâtisse neuve : ainsi les applications furent ajoutées au fur et à mesure de l'apparition de besoins fonctionnels (un progiciel dédié à la fonction RH, un serveur mail interne face à l'augmentation des échanges, etc.), et il n'est pas rare que la partie informatisée du SI s'apparente à un amoncellement d'applications.

On parle alors de « refonte » ou de « réingénierie » (reverse engineering) dans le but d'étendre, d'urbaniser ou de faire évoluer le SI. C'est l'occasion de **créer** ou de **mettre à jour les plans (cartographie)**, de repérer les actuels **dysfonctionnements** et d'envisager les **évolutions** à mettre en œuvre[1]. Cette démarche peut être longue (recensement des données et applications, etc.) et il faut rester modeste car elle ne peut être que difficilement achevée de façon complète.

Par ailleurs, le SI est, par essence même, une structure mouvante (départ de salariés, développement d'une nouvelle technologie, adoption d'un progiciel) et il apparaît nécessaire de s'appuyer sur une architecture évolutive et sécurisée.

Même si la définition d'une d'architecture SI ne répond pas à des normes générales et imposées, il existe des *Design patterns* (patrons) qui correspondent à des **modèles génériques** tels que le *Client-Serveur,* le *N-Tiers*, *l'architecture orientée services*, etc.[2]

MAÎTRISER L'ARCHITECTURE DU SI, UN DÉFI !

« Force est de constater que la complexité des SI est proportionnelle à la complexité croissante des technologies et des organisations elles-mêmes. Le SI doit répondre aux enjeux stratégiques de l'entreprise, au développement du marché, supporter l'évolution des métiers et des fonctions au sein de l'organisation, et également supporter l'évolution du périmètre de l'organisation (fusion, intégration de différentes entreprises).(…) L'enjeu est alors le suivant : faire évoluer le système voire refaire certains morceaux, sans détruire l'existant, en intégrant des composants divers et en exploitant les avancées technologiques dans un souci de cohérence générale, de réactivité et de flexibilité. »[3]

b. En quoi l'architecture diffère de la démarche d'urbanisation

Au sein du chapitre 3 nous avions abordé les notions propres à l'urbanisation : vues, îlots, zone, quartiers… Il s'agissait d'une perception plutôt générale du SI, basée sur le vocabulaire propre à l'urbanisme.

1. Dès lors, ce type de questionnement concerne davantage des structures pour lesquelles la vision de la fonction SI témoigne d'une certaine maturité. Une petite entreprise relevant de nombreux défis au quotidien, avec l'objectif de pérenniser [voire « faire survivre »] son activité se sentirait moins concernée par cette réflexion.

2. Ces modèles fournissent des réponses aux problématiques soulevées par ce qu'on qualifie d'anti-patterns : la métaphore du *plat de spaghettis* ou de *l'usine à gaz !*

3. SERVIGNE Sylvie. *Conception, architecture et urbanisation des SI.* [En ligne]. Url : https://perso.liris.cnrs.fr/sylvie.servigne/Encyclopedia.pdf

L'architecture du SI, quant à elle, propose une **vision détaillée**[1] de l'organisation à travers ses composantes et les interactions existantes entre ces dernières.

Dès lors, on peut souligner qu'il existe différentes architectures, au même titre qu'il existe différentes vues pour l'urbanisation :

- L'architecture **fonctionnelle** présente les fonctions du SI. Celles-ci sont intimement liées aux besoins des métiers en traitements et informations (impératifs propres à la réalisation du business) ;
- L'architecture **applicative** (logicielle) présente les applications en soulignant la modularité de celles-ci : logiciels employés, interfaces présentes et échanges possibles ;
- L'architecture **technique** présente le mode de fonctionnement des applications : les logiciels. Elle est encore à distinguer de l'infrastructure, qui présente l'aspect purement matériel à travers les éléments du SI (postes, interconnexions, interfaces entre réseaux…).

2. Les applications du SI

L'emploi répandu des Smartphones[2] a généralisé l'usage du terme « application ». Une redéfinition de ce qu'englobe le terme (sous l'angle MSI) semble nécessaire.

a. Une application se compose d'un (ou plusieurs) programme(s)

L'application peut être un *ensemble* de programmes (ou logiciels). Elle est conçue pour apporter une *solution à une problématique d'utilisateur* : gestion de la relation client, gestion des stocks, définition du prix de conception d'une pièce.

EXEMPLE

Une application de gestion des stocks pourrait s'appuyer sur l'usage d'un formulaire Web (en HTML), sollicitant un script en langage dynamique (PHP) et attaquant une base de données POSTGRESQL via des requêtes en langage SQL. Cette application pourrait être accessible via l'Intranet de l'entreprise, sous réserve d'obtention des droits d'accès à cette dernière.

b. Une application peut être décomposée

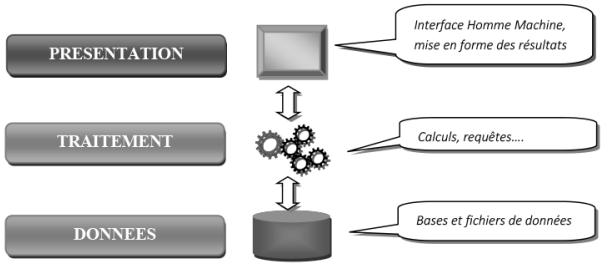
On distingue **trois composantes** dans une application.

La modularité de ces dernières permet d'assurer la maintenance (corrective et évolutive) d'une composante sans affecter les deux autres.

1. Cette vision demeure malgré tout suffisamment abstraite pour rester lisible : si on ajoute trop de détails les schémas, modèles, diagrammes sont surchargés et deviennent illisibles.

2. 5 milliards d'utilisateurs en 2019 (Source : Statista).

Ces trois composantes peuvent être regroupées dans un même logiciel : *dans ce cas, le logiciel est l'application*.



c. Acquisition ou conception d'un logiciel ?

Les programmes utilisés par les applications peuvent être **achetés** (progiciels commercialisés) ou **récupérés** (progiciels Open Source, c'est-à-dire libres d'exploitation). On dispose alors de solutions génériques, avec une différenciation moins forte vis-à-vis de la concurrence, mais bien souvent dotées d'un support client (ou à défaut de forums d'utilisateurs) plus fourni(s).

Un logiciel dit « maison » est quant à lui construit *sur mesure,* pour répondre à un besoin singulier, et pourra être utilisé dans un contexte donné (le système informatique au sein duquel il sera intégré).

On s'appuie dès lors sur la programmation à l'aide d'un langage plus ou moins évolué, portable, sécurisé : VBA, Java, C+, Python, etc.

Certains langages sont exécutables : ils sont interprétés ligne par ligne, c'est-à-dire que l'exécution du programme se fait directement à partir du code.

> Exemple : VBA

D'autres sont compilables : il faut dans un premier temps utiliser un compilateur pour transformer le code source en fichier exécutable.

> Exemple : C ++

d. La difficile gestion du parc applicatif

Un SI peut comporter un « parc applicatif » plus ou moins hétérogène. Il n'est pas rare que solutions commercialisées, solutions « maisons » et solutions libres se côtoient – le défi du DSI demeurant le **maintien de la cohérence et de l'unification** des données et traitements.

Pour faire communiquer deux programmes entre eux, on recourt à une « interface », i.e. un programme qui s'occupe de gérer les échanges de données entre des logiciels distincts. Ces **interfaces** peuvent être fournies par l'éditeur du logiciel, ou conçues selon les besoins spécifiques par des développeurs.

II. Perspective historique de l'évolution des architectures

« *Pour savoir où l'on va, il faut savoir d'où l'on vient* »[1]. Nous abordions au sein du chapitre 2 l'évolution de la place de l'informatique au sein de l'organisation, voyons à présent ce qui a caractérisé les architectures des dernières décades.

1. Les années 70 ou l'avènement de l'ordinateur central

a. Le mainframe, ou supercalculateur de la taille d'une [grande] pièce

Si vous avez vu « *Les figures de l'ombre* »[2], vous avez constaté la place (au sens littéral et figuré) que prirent les mainframes au sein de structures scientifiques telles que la NASA. Ces imposants supercalculateurs devaient remplacer plusieurs dizaines de « calculatrices » humaines, et il s'agissait bien d'un ordinateur central et non pas de réseaux informatiques.

b. Principes d'architecture de l'ordinateur central

Dans le principe, **toutes les composantes de l'application se retrouvent dans le mainframe**.

Ce dernier fonctionne assez souvent sur un OS Unix, et s'il s'agit d'*entrer* [au clavier] ou d'*afficher* [à l'écran] des données en mode caractères. Il faut recourir à un terminal passif. Le terminal n'est pas autonome – d'ailleurs il ne contient ni processeur ni mémoire, il joue uniquement le rôle de périphérique.

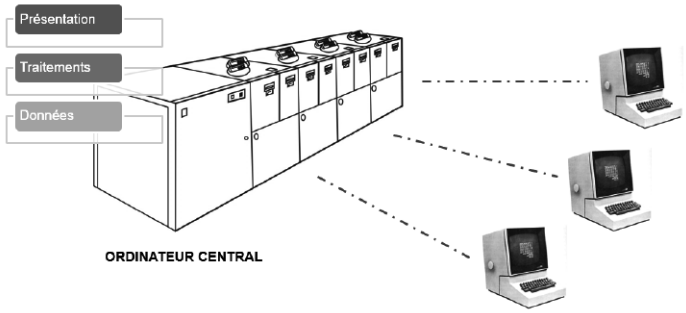
Certains mainframes « historiques » sont toujours en activité (IBM S/390, Bull DPS 7x…) au sein des banques et assurances. C'est parfois un véritable défi pour les sociétés hébergeant de telles solutions techniques d'assurer la maintenance de ces dernières dans des langages tels que l'assembleur, COBOL ou JCL.

1. Proverbe bien connu dont la paternité est difficile à déterminer.

2. Theodore MELFI. *Hidden Figures.* 2017

LA MAINTENANCE DES « SYSTÈMES HÉRITÉS STRATÉGIQUES »

« On aurait du mal à croire en la pénurie de compétences sur un secteur qui, selon le cabinet d'étude IDC, représente 13 % de l'économie pour un environnement qui supporte 65 % des applications critiques des grandes industries et entreprises bancaires. (…) Face au vieillissement de la population mainframe engendrant de nombreux départs à la retraite, l'abandon de l'apprentissage du langage COBOL dans les écoles d'ingénieurs ne permet pas de combler ces départs…



Le langage COBOL pâtit d'une réputation d'obsolescence, alimentée par de nombreux observateurs extérieurs. Un point de vue contredit par l'activité même d'IBM, qui fait évoluer le COBOL au fil des années afin de l'adapter au développement d'applications mainframe. Ainsi, COBOL est non seulement un langage évolué, mais il est aujourd'hui le langage de programmation le plus performant sur les System z. »[1]

2. Les années 80 et l'interconnexion des postes

Les années 80 ce n'est pas que les synthétiseurs et les pulls à rayures.

C'est aussi l'interconnexion des réseaux et le développement de l'usage du PC (Ordinateur Personnel) pour les entreprises et les particuliers.

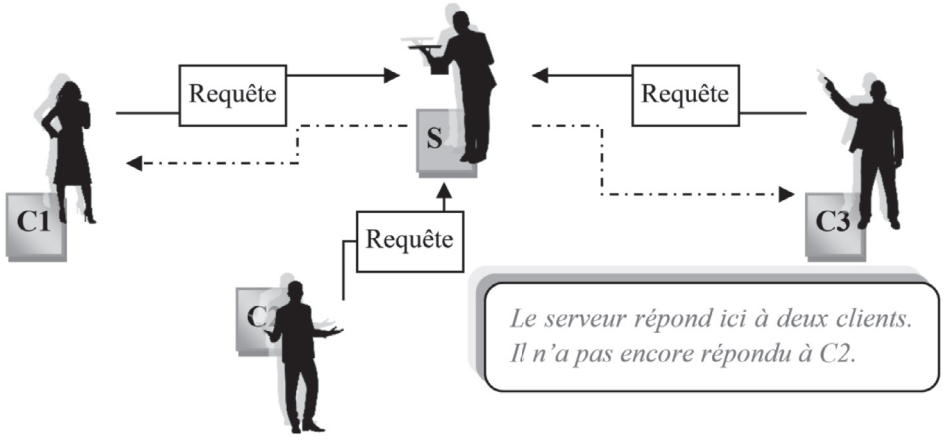
a. Client-Serveur, principes

L'analogie avec un service en terrasse est assez évidente : un client interpelle un serveur (*« Garçon, un café ! »*), qui s'empresse de répondre à sa requête s'il n'est pas déjà surchargé.

D'un point de vue informatique, Client et Serveur sont des **entités logicielles** (des programmes[2]). Le Serveur (ou « Service ») demeure dans l'attente afin de répondre à une sollicitation du client (renvoi d'un fichier, réponse à une requête, fourniture d'un traitement…).

1. *Mainframe is alive ! Insights of Virtel on mainframe modernization.* Avril 2015. [En ligne]. Url : http://www.virtelweb.com/blog/competences-mainframe/

2. Prenons le temps d'insister sur la sémantique : il ne s'agit pas, de façon systématique, de machines dédiées. Plusieurs clients peuvent côtoyer différents serveurs sur la même machine physique. La confusion puise ses sources dans maintes illustrations, où on voit apparaître au titre de « serveur » l'icône d'une unité centrale.



b. L'interconnexion des réseaux

Dans les années 70, les ordinateurs sont interconnectés pour des besoins militaires (Arpanet, 1972) puis universitaires. L'idée de pouvoir mutualiser des ressources et d'échanger à grande vitesse un flux d'informations ne va pas tarder à séduire le commun des mortels.

La difficulté reste la même : comment interconnecter des machines différentes tout en assurant la cohérence de l'échange ?

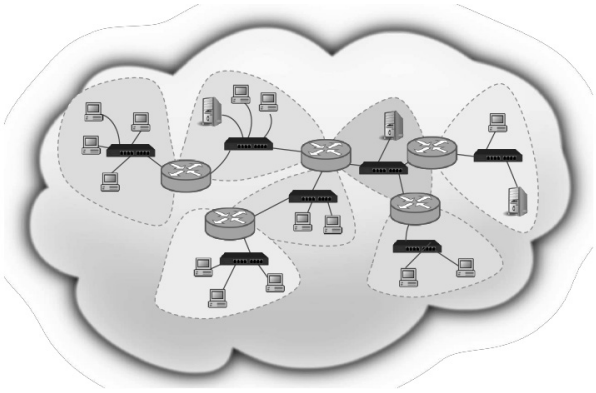
C'est ainsi, qu'au fur et à mesure des développements, les usages sont standardisés et donnent naissance à des protocoles (Transmission Control Protocol et Internet Protocol, i.e. TCP/IP) de communication.

3. Les années 90 et le développement du Web

a. La grande toile

Le World Wide Web (« la toile mondiale »), ou Internet (« le réseau des réseaux » sur lequel on navigue) est à vrai dire une interconnexion de réseaux via des routeurs[1].

1. Matériel permettant la jonction entre réseaux distincts. Au titre de rappel des notions fondamentales abordées dans le programme DCG (UE8), un poste ne peut communiquer de façon directe qu'avec les postes de son réseau. Au-delà, il faut qu'il s'adresse à une passerelle (un routeur) qui va lui permettre [ou pas] de s'adresser à un autre réseau.



b. Web et nouveaux usages

Ce gigantesque réseau va révolutionner les échanges : on assiste à une véritable remise en question de la dimension espace/temps, il est à présent possible de partager de grosses quantités d'informations en temps réel entre des utilisateurs qui sont séparés par des milliers de kilomètres.

Le Web c'est aussi la mise à disposition de services plus ou moins avancés : Serveurs mail, solutions SaaS[1], Cloud, etc. qui ont tendance à redéfinir (et à rendre plus floues) les limites du SI. Interconnecter son SI au « nuage », c'est accepter l'idée qu'un certain nombre de traitements et données échappent au contrôle total de la DSI.

III. Principales architectures techniques

1. Le concept du Client-Serveur

a. Une machine ou un logiciel ?

Pour rappel, dans une architecture client-serveur, client et serveur sont des entités logicielles : elles peuvent donc être placées sur le même poste physique.

> La confusion sémantique (un serveur = une machine) vient du fait qu'on place souvent au sein d'un même poste un service dédié (le serveur d'authentification, le serveur de fichier, etc.) afin de faciliter la maintenance et d'augmenter la performance du service.

b. La répartition de charges entre client et serveur

Un client « léger », par opposition à un client « lourd », est un applicatif qui n'utilisera que peu de ressources pour exécuter des opérations.

> Exemple : le navigateur Web (Chrome, Internet Explorer, Opera, etc.)

1. Software As A Service (Applications en ligne).